

Detecția pulsului folosind procesarea de imagini

Proiect de diplomă

prezentat ca cerință parțială pentru obținerea titlului
de Inginer în domeniul Electronică și
Telecomunicații programul de studii de licență
Electronică Aplicată

Conducători științifici

Conf. dr. ing. Daniela FAUR
Dr. ing. George SUCIU

Absolvent

Carmen-Violeta NĂDRAG

București, 2018

Universitatea "Politehnica" din București
Facultatea de Electronică, Telecomunicații și Tehnologia Informației
Departamentul **EAI**

TEMA PROIECTULUI DE DIPLOMĂ
a studentului **NĂDRAG V. Carmen-Violeta , 443B**

1. Titlul temei: Detecția pulsului folosind procesarea de imagini

2. Descrierea contribuției originale a studentului (în afara părții de documentare):

Scopul lucrării îl reprezintă măsurarea pulsului în timp real, prin procesarea imaginilor achiziționate cu ajutorul camerei web/smartphone. În vederea acestui lucru, procesarea imaginilor se va realiza folosind OpenCV și limbajul de programare Python. Spre deosebire de alte soluții existente, aceasta are ca scop măsurarea simultană a pulsului pentru mai multe persoane, folosind urmărirea de obiecte în conjuncție cu detecția de fețe. Acest lucru va reduce timpul de calcul necesar, astfel încât o implementare pe un terminal mobil de tip smartphone devine fezabilă. De asemenea, se vor studia particularitățile soluției mobile, în vederea comparației cu soluția dezvoltată pentru desktop/laptop.

3. Resurse folosite la dezvoltarea proiectului:

OpenCV, Python, Spyder IDE


4. Proiectul se bazează pe cunoștințe dobândite în principal la următoarele 3-4 discipline:

Imagistică Medicală, Semnale și Sisteme, Practică, Programarea Calculatoarelor


5. Proprietatea intelectuală asupra proiectului aparține: studentului

6. Data înregistrării temei: 2017-11-26 19:35:00

Conducător(i) lucrare,
Conf. dr. ing. FAUR Daniela

semnătura: 

Student,

semnătura: 

Dr. ing. George SUCIU, Beia Consult International

semnătura: 

Director departament,
Prof. dr. ing Sever PAȘCA

semnătura: 

Decan,
Prof. dr. ing. Cristian NEGRESCU

semnătura: 

Cod Validare: **3e7d35f88c**

Copyright © 2018, Nădrag Carmen-Violeta

Toate drepturile rezervate

Autorul acordă UPB dreptul de a reproduce și de a distribui public copii pe hârtie sau electronice ale acestei lucrări, în formă integrală sau parțială.

Declarație de onestitate academică

Prin prezenta declar că lucrarea cu titlul "*Detecția pulsului folosind procesarea de imagini*", prezentată în cadrul Facultății de Electronică, Telecomunicații și Tehnologia Informației a Universității "Politehnica" din București ca cerință parțială pentru obținerea titlului de *Inginer* în domeniul *Inginerie Electronice și Telecomunicații* programul de studii *Electronică aplicată* este scrisă de mine și nu a mai fost prezentată niciodată la o facultate sau instituție de învățământ superior din țară sau străinătate.

Declar că toate sursele utilizate, inclusiv cele de pe Internet, sunt indicate în lucrare, ca referințe bibliografice. Fragmentele de text din alte surse, reproduse exact, chiar și în traducere proprie din altă limbă, sunt scrise între ghilimele și fac referință la sursă. Reformularea în cuvinte proprii a textelor scrise de către alți autori face referință la sursă. Înțeleg că plagiatul constituie infracțiune și se sancționează conform legilor în vigoare.

Declar că toate rezultatele simulărilor, experimentelor și măsurărilor pe care le prezint ca fiind făcute de mine, precum și metodele prin care au fost obținute, sunt reale și provin din respectivele simulări, experimente și măsurători. Înțeleg că falsificarea datelor și rezultatelor constituie fraudă și se sancționează conform regulamentelor în vigoare.

București, 29.06.2018

Absolvent *Carmen-Violeta NĂDRAG*


(semnătura în original)

CUPRINS

Introducere	9
Capitolul 1. Stadiul Actual al Tehnologiei.....	11
1.1 Metode de măsurare a pulsului prin atingere.....	11
1.1.1 Brăţările inteligente.....	12
1.1.2 Inelul inteligent Oura	12
1.1.3 Centura pentru puls.....	13
1.1.4 Stetoscopul inteligent	14
1.1.5 Aplicaţii pentru smartphone.....	14
1.1.6 Avantaje şi dezavantaje.....	15
1.2 Metode de măsurare a pulsului fără atingere	15
Capitolul 2. Aspecte teoretice.....	19
2.1 Detecţia de feţe.....	19
2.1.1 Metoda Viola Jones.....	19
2.1.2 Utilizarea detecţiei de piele	23
2.1.3 Aplicaţii bazate pe detecţia de feţe.....	25
2.2 Urmărirea de obiecte.....	26
2.3 Selectarea regiunii de interes	29
Capitolul 3. Metoda experimentală.....	31
3.1 Descrierea algoritmului.....	31
3.2 Detecţia şi urmărirea feţelor	32
3.3 Alegerea regiunii de interes.....	34
3.4 Procesarea semnalului.....	35
Capitolul 4. Testarea programului şi validarea rezultatelor	37
Capitolul 5. Studiul particularităţilor soluţiei mobile	39
Concluzii.....	41
Bibliografie.....	43
ANEXA 1	45
ANEXA 2	51

Lista figurilor

Fig. 1.1 - Metode de transmitere a datelor de la dispozitive portabile în Cloud.....	11
Fig. 1.2 - Inele inteligente Oura.....	13
Fig. 1.3 - Vizualizarea grafică a măsurătorilor de puls realizate cu Oura.....	13
Fig. 1.4 - Centură pentru măsurarea pulsului.....	13
Fig. 1.5 - Stetoscopul inteligent Jabes.....	14
Fig. 1.6 - Rezultate obținute cu aplicația Heart Rate Plus, respectiv Heart Rate Monitor.....	15
Fig. 2.1 - Ajustarea ponderilor în funcție de eroare.....	20
Fig. 2.2 - Obținerea clasicatorului final ca sumă ponderată de clasicatori slabi.....	20
Fig. 2.3 - Caracteristicile Haar.....	21
Fig. 2.4 - Utilizarea caracteristicilor Haar.....	21
Fig. 2.5 - Exemplificarea calculului folosind imagini integrale.....	22
Fig. 2.6 - Modelul unei fețe.....	24
Fig. 2.7 - Rezultatele comparației algoritmilor de urmărire.....	29
Fig. 2.8 - Selecția regiunii de interes.....	30
Fig. 3.1 - Schema logică a algoritmului.....	32
Fig. 3.2 – Diagrama bloc a algoritmului.....	32
Fig. 3.3 - Experimentarea algoritmului KCF.....	33
Fig. 3.4 - Selecția regiunii de interes în timpul experimentării.....	34
Fig. 3.5 - Persoana 2 și Persoana 3 în timpul testelor.....	35
Fig. 3.6 (a) - Componente de magnitudini diferite obținute din cauza perturbațiilor.....	36
Fig. 3.6 (b) - Componenta ce indică pulsul.....	36
Fig. 3.7 (a) - Detecția pulsului pentru 1 persoană (lumină naturală).....	36
Fig. 3.7 (b) - Detecția pulsului pentru 2 persoane (lumină naturală).....	36
Fig. 4.1 - Camera web Logitech C615.....	37
Fig. 4.2 - Camera web Logitech BCC950.....	37
Fig. 4.3 - Fitbit Alta HR.....	38

Lista tabelelor

Tabelul 3.1 – Rezultatele experimentării algoritmilor de urmărire.....	34
Tabelul 3.2 – Medii din ROI pentru fiecare subiect participant.....	35
Tabelul 4.1 – Măsurători efectuate simultan.....	38

Lista acronimelor

2D/3D – Spațiu bidimensional/tridimensional
BLE – Bluetooth Low Energy
BPM - Bătăi Pe Minut
CPU - Central Processing Unit (Unitate Centrală de Prelucrare)
ECG - Electrocardiografie
FD – Face Detection (Detectie de fețe)
FFT - Fast Fourier Transform (Transformata Fourier Rapidă)
FT - Face Tracking (Urmărire de fețe)
GB – GigaByte
GPU – Graphics Processing Unit (Unitate Grafică de Procesare)
HD - High Definition (Rezoluție mare)
HRV - Heart Rate Variability (Variația Pulsului)
HR - Heart Rate (Puls)
Hz/GHz – Hertz/GigaHertz
ICA - Independent Component Analysis (Analiza Componentelor Independente)
IoT - Internet-of-Things
KCF – Kernelized Correlation Filters
KNN - K-Nearest Neighbour
MB - MegaByte
MIL – Multiple Instance Learning
MOSSE - Minimum Output Sum of Squared Error
NN – Neural Networks (Rețele Neurale)
PCA - Principal Component Analysis (Analiza Componentelor Principale)
PPG - Photoplethysmography (Fotoplethysmografie)
RAM - Random Access Memory
RGB – Red,Green,Blue
ROI – Region of Interest (Regiune de Interes)
SVM -Suport Vector Machine
TLD – Tracking-Learning-Detection
USB – Universal Serial Bus

Introducere

Scopul acestui proiect este dezvoltarea unui program software capabil să detecteze pulsul pentru mai multe persoane simultan, folosind procesarea de imagini. Această tehnică necesită ca subiectul să fie relaxat și să fie plasat în apropierea unei camere web. Distanța dintre cameră și pacient poate varia între 1 și 3 metri, iar condițiile de iluminare trebuie să fie constante în timpul măsurătorii. În vederea realizării proiectului, procesarea imaginilor s-a realizat folosind librăriile OpenCV și limbajul de programare Python. Pentru a măsura simultan pulsul mai multor persoane, au fost implementate atât detecția de fețe, cât și urmărirea de obiecte.

Rezultatele acestei lucrări au fost prezentate în cadrul conferinței COMM 2018, sesiunea „Image Processing” [1].

Punctul de pornire în realizarea celor menționate îl reprezintă faptul că în timpul bătăilor inimii, pomparea sângelui în corp cauzează variația culorii pielii. Această variație nu este sesizabilă cu ochiul liber, însă poate fi analizată în urma procesării imaginilor. Pentru fiecare cadru, se alege o regiune de interes (ROI) și se observă cum pixelii din zona selectată își schimbă intensitatea. Calculând frecvența la care se modifică nuanța pielii, se află frecvența bătăilor inimii.

Pulsul (Heart Rate- HR) unei persoane reprezintă numărul de bătăi ale inimii în decursul unui minut. De asemenea, acesta este un parametru fiziologic esențial, fiind o sursă de informare cu privire la activitatea întregului sistem cardiovascular. Pulsul are totodată o importanță deosebită în diagnosticare și evaluarea nivelului de stres la care este predispus pacientul. Spre exemplu, persoanele care sunt mai puțin active din punct de vedere fizic vor avea de obicei un puls mai ridicat, din cauza faptului că mușchiul miocardic va fi mai solicitat pentru menținerea ritmului cardiac.

Valoarea pulsului variază în funcție de factori precum vârstă, sex, greutate, însă intervalul normal pentru un adult este definit în articole [2] ca fiind 50-90 bătăi pe minut (BPM), în timp ce Asociația Americană a Inimii consideră intervalul normal ca fiind 60-100 BPM. Pulsul cardiac al unei persoane poate să difere și în funcție de alte aspecte, cum ar fi nevoia de oxigen a organismului.

Este binecunoscut faptul că trebuie investigată orice variație neobișnuită a pulsului cardiac, astfel că este necesară monitorizarea continuă a acestuia [3]. Există afecțiuni definite pe baza variației pulsului unei persoane, cum ar fi:

- Tahicardie – pulsul este ridicat, mai mare decât 100 BPM;
- Bradicardie – pulsul este scăzut, mai mic decât 40 BPM;
- Aritmia – există variații ale ritmului cardiac (bătăile inimii sunt neregulate).

În ultimele decade, societatea a devenit mai conștientă de importanța sănătății, iar o consecință a acestui lucru este multitudinea de platforme pentru monitorizarea de la distanță a sănătății pacienților [4], [5]. Printre cele mai frecvente cazuri de utilizare a acestora se numără monitorizarea stării de sănătate a bătrânilor și a pacienților cu probleme cronice.

Există însă situații în care este necesară monitorizarea continuă a pulsului unei persoane, iar contactul cu pielea pacientului este problematică din cauza anumitor afecțiuni de care acesta suferă. Astfel, conectarea pe o perioadă îndelungată la aparatele de monitorizare a pulsului devine inconfortabilă. În plus, un astfel de echipament poate monitoriza o singură persoană odată, ceea ce nu ajută în cazurile în care este nevoie de o monitorizare rapidă sau constantă a persoanelor dintr-o anumită locație - ca de exemplu nou-născuții dintr-un salon de spital, persoanele dintr-un birou sau o sală de așteptare.

În continuare, lucrarea este structurată în 5 capitole: Capitolul 1 descrie metode existente de măsurare a pulsului, Capitolul 2 prezintă aspecte teoretice ale detecției de puls folosind procesarea de imagini, Capitolul 3 descrie soluția dezvoltată în cadrul proiectului și etapele acesteia. De asemenea, Capitolul 4 prezintă testarea programului și metoda de validare a rezultatelor obținute, iar Capitolul 5 constituie un studiu al particularităților soluției mobile. În final, sunt prezentate Concluzii.

Capitolul 1. Stadiul Actual al Tehnologiei

În prezent există numeroase metode de măsurare a pulsului. Posibilitatea integrării acestora în diverse aplicații face ca activitatea de cercetare în această direcție să fie continuă și să aibă rezultate foarte bune. În cele ce urmează, am clasificat metodele de măsurare a pulsului în două mari categorii: metode de măsurare a pulsului prin atingere, respectiv fără atingere.

1.1 Metode de măsurare a pulsului prin atingere

În ultimii ani au devenit tot mai populare dispozitivele portabile, care permit utilizatorului să își monitorizeze permanent parametrii vitali. Acestea cuprind tehnologii electronice încorporate în articole de îmbrăcăminte și accesorii care pot fi purtate pe corp. Succesul acestora se datorează unei serii de facilități, cum ar fi sincronizarea rapidă a datelor în Cloud, posibilitatea stocării datelor pe dispozitiv pentru mai multe zile (sincronizarea fiind necesară de obicei săptămânal), durabilitatea bateriei. În general, tehnologia portabilă are o anumită capacitate de comunicație și permite accesul purtătorului la informații în timp real.

De asemenea, utilizatorul își poate îmbunătăți calitatea somnului și își poate organiza mult mai ușor activitățile fizice. După ce datele sunt înregistrate folosind senzorii integrați în dispozitive, se realizează transmiterea datelor în Cloud. Aceasta se poate face direct (prin conexiune Wi-Fi) sau prin sincronizarea cu aplicația mobilă folosind BLE (Bluetooth Low Energy), după care are loc transmiterea datelor în Cloud (Fig. 1.1).

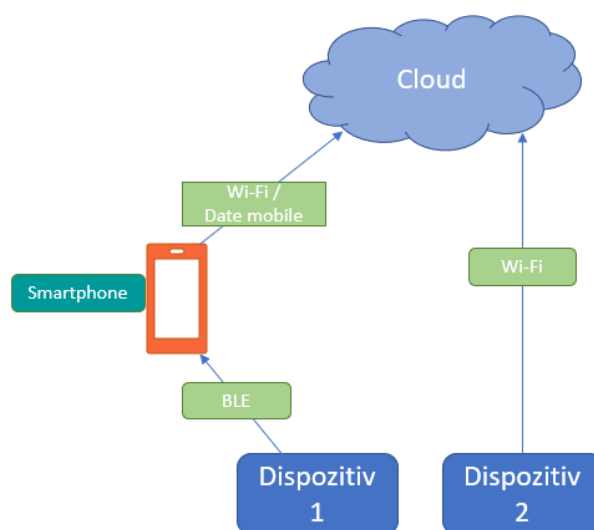


Fig. 1.1 - Metode de transmitere a datelor de la dispozitive portabile în Cloud

Primul dispozitiv portabil pentru măsurarea ritmului cardiac a fost creat în anul 1977, de către profesorul finlandez Seppo Säynäjäkangas, fapt ce a dus la înființarea companiei Polar Electro [6]. Dispozitivele clasice cuprindeau un transmițător și un receptor, și pentru o bună conductivitate era folosită apa sărată. De asemenea, în versiunile vechi ale monitoarelor pentru puls, de fiecare dată când era detectată o bătaie a inimii, era transmis un semnal radio. Cu ajutorul acestor semnale,

receptorul determina frecvența cardiacă a purtătorului dispozitivului. Exista însă riscul apariției unor interferențe, odată cu recepția de semnale de la alți utilizatori din zonă.

1.1.1 Brățelele inteligente

În prezent, printre dispozitivele portabile cu cel mai mare succes se numără brățelele inteligente. Acestea oferă o multitudine de opțiuni, inclusiv măsurarea pulsului. Datele obținute sunt post-procesate fie de către senzori, fie de către dispozitivele atașate senzorilor. Sincronizarea datelor este rapidă, astfel că dispozitivele pot fi utilizate pentru crearea unui istoric al activității cardiace.

Întrucât aceste dispozitive sunt purtate la încheietura mâinii, valorile măsurate sunt însă ușor influențabile în timpul activităților ce solicită această parte a corpului. Un exemplu poate fi utilizarea tastaturii: deși purtătorul este relaxat și are o valoare normală a pulsului, rezultatul obținut poate fi diferit din cauza activității intense a mâinilor și a modului în care este purtat dispozitivul (prea strâns, ori prea slab). Cu toate acestea, în condițiile în care utilizatorul este relaxat sau își solicită corpul într-un mod armonios, rezultatele obținute vor fi cât mai exacte.

Potrivit unor studii [7], majoritatea dispozitivelor portabile nu sunt însă destinate măsurării cu o precizie foarte bună a pulsului utilizatorului, deși au integrată această funcție, iar competiția între producători este foarte ridicată. Totuși, există dezvoltatori precum Apple [8] sau Fitbit [9], ce reprezintă vârful de gamă și care inovează cu fiecare produs lansat. Acest lucru se datorează și faptului că aceste companii investesc în cercetare și folosesc rezultatele lucrărilor științifice pentru îmbunătățirea propriilor produse [10].

Principiul de funcționare al acestor dispozitive este fotopletismografia (PPG) [11]. Dispozitivul este prevăzut cu o sursă de lumină, ce emite un fascicul către piele. O parte din lumină va fi absorbită de piele, iar o parte va fi supusă procesului de reflexie, în funcție de volumul de sânge care circulă prin vasele de sânge. Dacă țesutul este bine irigat, cantitatea de lumină absorbită va fi mai mare și va fi reflectată o cantitate mai mică de lumină. Prin măsurarea cantității de lumină ce se întoarce în urma reflexiei, se obține pulsul pacientului.

Într-un studiu [12] efectuat pe 12 persoane, autorii au comparat rezultatele obținute cu un astfel de dispozitiv, purtat pe încheietura mâinii, și cele ale unei electrocardiografii (ECG). Testele s-au efectuat în diferite condiții: stare de relaxare a subiecților sau după ce aceștia au efectuat exerciții fizice. Pulsul a fost monitorizat atât în timpul mișcării, cât și după, iar cea mai mare eroare întâlnită între măsurătorile efectuate cu cele două dispozitive a fost de 5 bătăi pe minut (BPM).

1.1.2 Inelul inteligent Oura

Oura [13] este un inel inteligent ce înglobează funcții precum înregistrarea ritmului zilnic de mișcare, intensitatea efortului fizic, durata și calitatea somnului, variația pulsului (HRV - Heart Rate Variability). Datele pot fi vizualizate cu ușurință atât pe Android cât și pe iOS, iar ritmul cardiac măsurat este asociat diferitelor activități realizate de utilizator (ex. exerciții fizice, somn) sau perioadelor de inactivitate. În Fig. 1.2 sunt prezentate inelele Oura, iar în Fig. 1.3 este prezentat modul de vizualizare în timp a pulsului măsurat cu un astfel de dispozitiv.



Fig. 1.2 - Inele inteligente Oura. Sursa: [12]



Fig. 1.3 - Vizualizarea grafică a măsurătorilor de puls realizate cu Oura

1.1.3 Centura pentru puls

Un alt tip de dispozitive portabile îl reprezintă centurile pentru piept. Acestea funcționează pe același principiu ca și electrocardiograma, măsurând activitatea electrică a inimii: în timpul activității cardiace, sunt generate semnale electrice care vor produce contracții ale mușchiului cardiac. Semnalul electric este înregistrat de electrozii cu care este prevăzută centura, și este apoi preluat de un transmițător, ce conține un microprocesor pentru analiza datelor.

Aceste centuri sunt considerate a avea cea mai mare precizie de măsurare, deoarece riscul de prindere inadecvată este mult mai mic [14], spre deosebire de cazul ceasurilor inteligente (care uneori sunt purtate prea strâns sau prea slab pe încheietura mâinii). De asemenea, centurile sunt purtate mult mai aproape de inimă. În Fig. 1.4 poate fi observată o centură pentru măsurarea pulsului.



Fig. 1.4 - Centură pentru măsurarea pulsului
Sursa: [15]

1.1.4 Stetoscopul inteligent

Stetoscopul este un instrument folosit în medicină, ce are rol de amplificare și percepute sunetele produse de organele interne ale pacientului în timpul funcționării acestora. Stetoscopul inteligent Jabes (Fig. 1.5) amplifică sunetele de până la 20 de ori, folosind filtre digitale [16]. Poate fi conectat la computer, iar software-ul conține o bază de date atât cu sunete de la pacienți sănătoși, cât și de la pacienți care suferă de afecțiuni ale ritmului cardiac. Astfel, persoana care îl utilizează poate realiza o comparație a rezultatului cu sunetele din baza de date.



Fig. 1.5 - Stetoscopul inteligent Jabes

Sursa: [17]

1.1.5 Aplicații pentru smartphone

Există aplicații mobile precum Heart Rate Plus - Pulse & Heart Rate Monitor [18] și Heart Rate Monitor [19], care măsoară pulsul unei persoane utilizând blițul smartphone-ului, având același principiu de funcționare descris anterior. De asemenea, Samsung a introdus această funcționalitate pe dispozitivele sale, astfel că smartphone-urile recente au un senzor dedicat pentru măsurarea pulsului. Aplicația necesară este pre-instalată, iar utilizatorul trebuie să acopere centrul senzorului cu degetul, fără să apese foarte puternic pentru că poate restricționa circulația sângelui. În timpul măsurătorii este necesar ca utilizatorul să fie relaxat și să nu vorbească. Am experimentat două dintre aplicații și am obținut rezultate similare, după cum este ilustrat în Fig. 1.6.

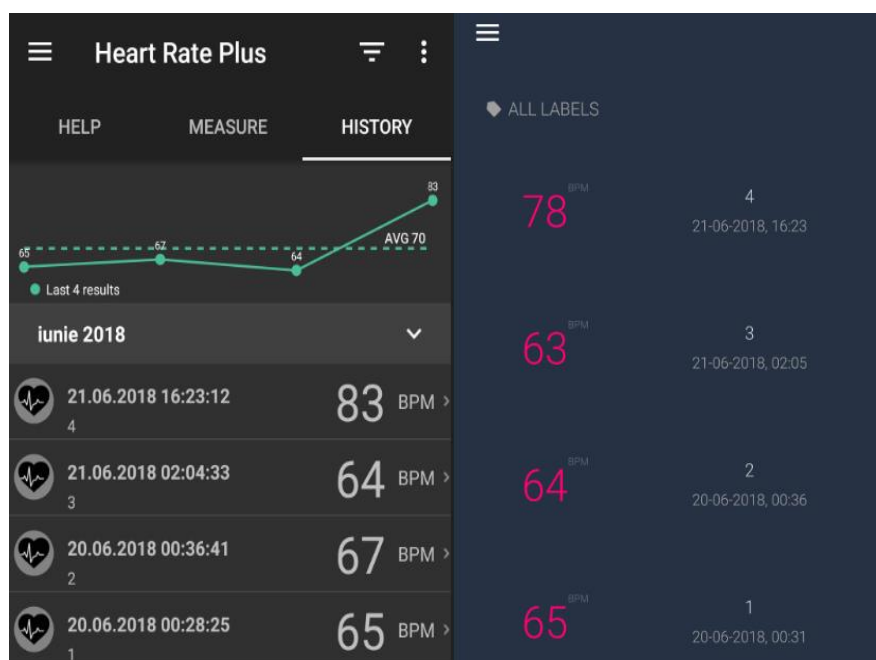


Fig. 1.6 - Rezultate obținute cu aplicația Heart Rate Plus, respectiv Heart Rate Monitor

1.1.6 Avantaje și dezavantaje

În ceea ce privește dispozitivele portabile, în decursul ultimilor ani a avut loc o creștere semnificativă a calității măsurărilor efectuate. Acest lucru se datorează în principal interesului ridicat al utilizatorilor, care duce la o competiție strânsă între dezvoltatori [20].

Principalele avantaje ale acestor tehnologii sunt accesul imediat și permanent la datele stocate, precum și integrarea în aplicații de SmartHealth a datelor obținute cu acestea. Măsurarea pulsului a devenit o acțiune extrem de rapidă și mult mai precisă în ultimii ani.

Totuși, aceste dispozitive au și o serie de dezavantaje ce îngreunează utilizarea lor, sau le fac inaccesibile pentru anumite categorii de persoane.

În primul rând, un dispozitiv cu o acuratețe mare de măsurare va avea și un preț ridicat. Prețurile pornesc de la câteva sute de lei, și ajung până la câteva mii. Un alt minus, în special pentru dispozitivele cu prețuri mai scăzute, este faptul că acestea pot atinge temperaturi destul de ridicate, din cauza anumitor componente sau a microprocesorului (dacă acesta este integrat). În plus, aceste dispozitive pot cauza iritații sau chiar arsuri la nivelul pielii utilizatorului [21].

1.2 Metode de măsurare a pulsului fără atingere

În decursul ultimilor ani a fost dezvoltată o serie de aplicații bazate pe Internetul Obiectelor (IoT - Internet-of-Things), ce integrează diverse soluții pentru urmărirea stării de sănătate. Se dorește monitorizarea pacienților de la distanță, prin transmiterea rapidă a datelor în Cloud, dar și eficientizarea metodelor de obținere a parametrilor necesari monitorizării. În ceea ce privește detecția de puls folosind procesarea de imagini, există două direcții ce permit dezvoltarea de aplicații științifice.

În primul rând, este analizată variația culorii pielii, cauzată de bătaile inimii. O altă metodă este reprezentată de măsurarea mișcărilor involuntare ale capului (și insesizabile cu ochiul liber), cauzate de activitatea cardiacă. Momentan, aceste metode sunt în special implementate în mediul științific, fiind subiectul a numeroase articole de cercetare și necesitând în primul rând implementarea algoritmului de detecție de fețe.

Există diverse lucrări științifice în care autorii prezintă o metodă fără contact pentru măsurarea pulsului, acestea fiind în special utile pentru oamenii care întâmpină diferite afecțiuni ale pielii. O astfel de aplicație [22] constă în utilizarea unor metode multiple de selecție a regiunii de interes (ROI), zonă pentru care se calculează apoi valorile medii ale pixelilor. În plus, metoda este testată în timp ce subiectul se mișcă, dar și atunci când apar tulburări de semnal. Erorile întâlnite sunt destul de scăzute: aproximativ $3,4 \pm 0,6$ BPM în cazurile în care subiectul stă nemișcat și $2,0 \pm 1,6$ BPM atunci când subiectul este în mișcare.

De asemenea, un alt sistem pentru măsurarea pulsului [23] este dezvoltat folosind o metodă de a înregistra modificările de ritm cardiac atât în cazul în care subiectul este nemișcat, cât și atunci când se află în mișcare. Regiunea de interes selectată este împărțită în 3 zone, iar pentru fiecare dintre acestea se calculează valoarea medie a pixelilor. Pentru a obține un semnal clar, are loc analiza componentelor independente (ICA), după care se utilizează detecția valorii de vârf a semnalului, pentru determinarea valorii ritmului cardiac. După cum menționează autorii, algoritmul nu oferă rezultatul în timp real, însă valorile obținute sunt cele așteptate, în comparație cu alte metode de măsurare a HR.

Într-un articol seminal [24], autorii prezintă o metodă ce are rezultate în timp real și se bazează pe implementarea a trei metode diferite de procesare a semnalului: ICA, Analiza Componentelor Principale (PCA) și transformarea rapidă Fourier (FFT). Obținerea de imagini este realizată utilizând o cameră de laptop, iar metoda de calcul a HR se bazează pe variația culorii pielii. Rezultatele obținute sunt similare cu alte metode.

În plus, există abordări [25] ce combină utilizarea camerelor web și a camerelor cu termoviziune. Experimentele au fost realizate în mijlocul zilei, astfel că singura sursă de iluminare este lumina soarelui. De asemenea, voluntarii (atât bărbați cât și femei) au stat la un metru de cameră și nu s-au mișcat în timpul măsurătorilor. Regiunea de interes a fost considerată pe frunte, deoarece această zonă are o temperatură constantă. Pentru reducerea complexității procesului, ROI selectată a fost mai mică, iar pe baza rezultatelor experimentului se concluzionează că ICA și PCA au precizie similară la extragerea pulsului.

De asemenea, în acest brevet [26] este descrisă estimarea variabilității pulsului (HR - Heart Rate Variability) prin procesarea video pentru obținerea unui semnal dependent de timp. Datele sunt apoi folosite pentru a extrage semnalul PPG și se calculează densitatea puterii spectrale pentru a detecta componentele de frecvență necesare pentru măsurarea HR.

În altă lucrare științifică [27] este propusă o metodă de măsurare a ritmului cardiac instantaneu, ce variază dinamic în cadrul unor secvențe video scurte. Provocarea în utilizarea acestui tip de secvențe video este aceea că sursele ICA ar putea să nu aibă o independență suficientă. Fără a determina independența surselor, există posibilitatea ca semnalul frecvenței cardiace să fie combinat cu alte semnale, ceea ce ar conduce la o citire inexactă. De aceea, există descrieri [28] pentru determinarea independenței surselor în vedea obținerii unei lecturi corecte.

O altă abordare [29] este detecția ritmului cardiac măsurând mișcarea capului subiectului. Aceasta este o urmare a fluxului de sânge aferent fiecărei bătaii a inimii. Metoda prezentată utilizează

analiza componentelor principale (PCA) pentru a descompune traiectoriile mișcărilor capului într-un set de mișcări de componente. Apoi se alege componenta ce corespunde cel mai bine bătăilor inimii, pe baza spectrului de frecvențe temporale.

Capitolul 2. Aspecte teoretice

2.1 Detecția de fețe

Primele descoperiri semnificative în ceea ce privește detecția de fețe au fost realizate în anii '60, de către matematicianul Woodrow Wilson Bledsoe. Printre publicațiile acestuia se numără două studii cunoscute ca fiind primele încercări de implementare a unui sistem computerizat de recunoaștere facială. În prima lucrare [30] cercetătorul analizează dacă tehnologiile existente la acel moment sunt suficiente pentru crearea unui sistem de recunoaștere facială bazat pe imaginirealizate din același unghi. Cea de-a doua lucrare [31] abordează o problemă mai complexă, și anume recunoașterea identității unei persoane folosind imagini noi ale acesteia, făcute din unghiuri diferite.

De-a lungul ultimelor decade, s-au făcut progrese semnificative în această direcție, iar în zilele noastre detecția și recunoașterea facială sunt implementate la scară largă, fiind utilizate în special de instituțiile guvernamentale și în cadrul platformelor de Social Media. Printre cazurile de utilizare ale acestora sunt recunoașterea unor persoane (infractori, persoane dispărute) prin comparația imaginilor obținute de la camerele de supraveghere cu cele din bazele de date, aplicații de securitate sau chiar alegerea ochelarilor de vedere potriviți cumpărătorului [32].

2.1.1 Metoda Viola Jones

Algoritmul de detecție de fețe dezvoltat de Paul Viola și Michael Jones în lucrarea lor [33] este bazat pe conceptul de învățare automatizată (machine learning) și antrenat pe seturi formate atât din imagini pozitive (imagini cu fețe) cât și imagini negative (imagini care nu conțin fețe). Deși este o metodă a cărei antrenare este considerată lentă, detecția este rapidă, cautându-se cele mai relevante trăsături, precum ochii, nasul, sprâncenele. Detectorul de fețe Viola-Jones utilizează mai multe caracteristici cheie, descrise în continuare.

2.1.1.1 AdaBoost

Algoritmul de învățare AdaBoost a fost introdus de Z. Freund și R. Schapire cu scopul de a combina trăsături slabe în clasificatori puternici (de ansamblu), și este unul dintre cei mai folosiți și studiați algoritmi. Conform studiului celor doi cercetători, un clasificator bine antrenat trebuie să îndeplinească 3 condiții [34]:

- Antrenarea trebuie să fie realizată pe un număr suficient de exemple, astfel încât să fie construit un model. Există însă și riscul de supra-antrenare (overfitting) [35], ce constă în memorarea datelor utilizate în cadrul etapei de învățare, pe care clasificatorul le va recunoaște foarte bine, însă orice alt exemplu (fie el pozitiv sau negativ), nu va fi recunoscut de acesta.

- Eroarea de antrenare trebuie să fie cât mai mică. Aceasta este definită ca fiind eroarea obținută la rularea modelului antrenat pe setul de date de antrenare. Spre deosebire de aceasta, eroarea de test apare la rularea modelului pe un set de date de test pentru măsurarea acurateții.

- Complexitatea clasificatorului nu trebuie să fie foarte ridicată. Pentru argumentarea acestei condiții, este deseori menționat „Briciul lui Occam”. Conform acestuia, pentru un fenomen este de preferat cea mai simplă explicație a acestuia. Din punct de vedere matematic, afirmația este susținută

de Teoria Probabilităților: orice ipoteză introduce posibilitatea existenței unei erori. Un exemplu practic este faptul că se recomandă folosirea unor clasificatori slabi drept clasificatori de bază, în ceea ce privește evitarea fenomenului de supra-antrenare, deoarece complexitatea modelului poate fi controlată mult mai ușor. De asemenea, există o condiție de învățare pentru clasificatorii slabi [33]: de exemplu, pentru o problemă de clasificare cu două clase, eroarea fiecăreia dintre acestea trebuie să fie mai mică decât $\frac{1}{2}$ (eroare = $\frac{1}{2} - x$, unde $x > 0$). Pe baza acestei condiții, se poate demonstra că eroarea finală de antrenare pentru AdaBoost scade foarte rapid către zero.

Exemple de clasificatori slabi (imperfecți): KNN (K-Nearest Neighbour), SVM (Suport Vector Machine), NN (Rețele Neurale). Aceștia pot fi combinați pentru a forma un clasificator de ansamblu (agregat). Vom considera exemplul unui clasificator agregat boolean, ce va returna 1 sau -1, având:

- un set de clasificatori slabi;
- „n” iterații;
- α - aceeași pondere inițială pentru toți clasificatorii;
- h_i - cel mai bun clasificator slab pentru iterația i, unde i ia valori de la 1 la n;
- H - clasificatorul de ansamblu;

Fiecare clasificator slab va avea o eroare cuprinsă între 0 (clasifică totul perfect) și 1 (clasifică totul greșit):

- în cazul în care eroarea este 0 (sau cât mai mică) ponderea se dorește a fi mai mare;
- în cazul în care eroarea este 1 (sau cât mai mare) ponderea se dorește a fi mai mică;

Fig. 2.1 ilustrează variația ponderii în funcție de eroarea obținută:

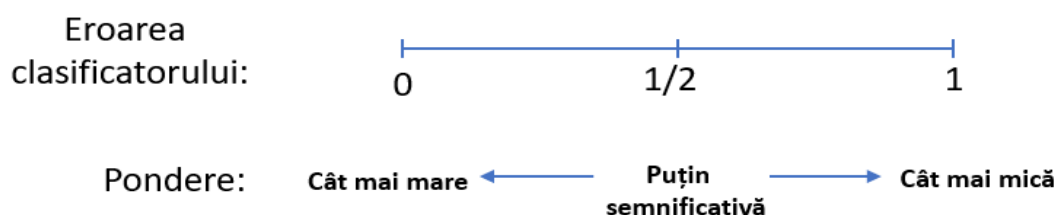


Fig. 2.1 - Ajustarea ponderilor în funcție de eroare

După fiecare iterație se alege clasificatorul cu cel mai bun rezultat, iar termenul corespunzător acestuia se adaugă în suma pentru clasificatorul final (Fig. 2.2).

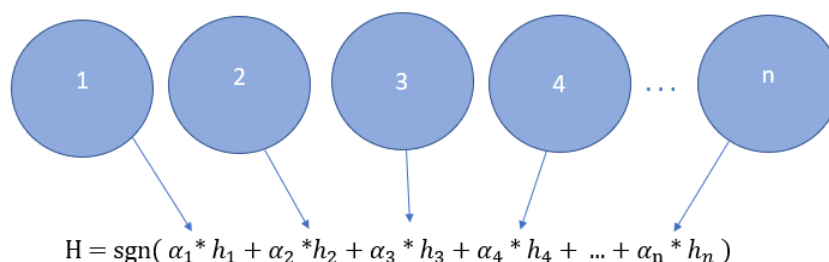


Fig. 2.2 - Obținerea clasificatorului final ca sumă ponderată de clasificatori slabi

2.1.1.2 Caracteristici Haar

Caracteristicile Haar prezentate în Fig. 2.3 sunt imagini dreptunghiulare ce pot fi definite ca diferența dintre media pixelilor din zona neagră și media pixelilor din zona albă. Există mai multe

tipuri de caracteristici, ce pot indica prezența sau absența unor elemente în imagine, cum ar fi muchii sau diferențe de luminanță. Acestea sunt deplasate de mai multe ori de-a lungul imaginii, clasificându-le ca fiind pozitivă sau negativă. Există mai multe tipuri de caracteristici:

- muchie - exemplu în Fig. 2.3 (a)
- linie - exemplu Fig. 2.3 (b)
- caracteristici cu 4 pătrate - exemplu Fig. 2.3 (c)

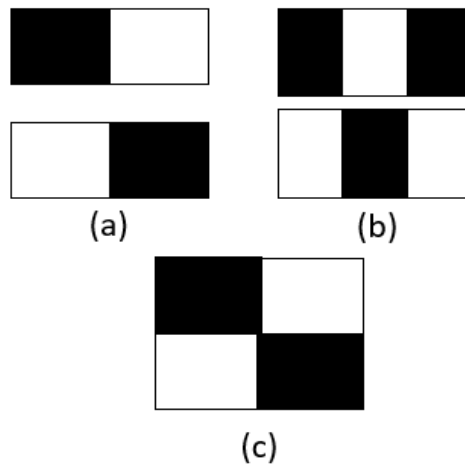


Fig. 2.3 - Caracteristicile Haar

Fiecare caracteristică este formată din zone de aceeași formă și dimensiune, dispuse vertical, orizontal sau pe diagonală. Această limitare a dispunerilor este însă compensată prin avantajele aduse de combinarea cu noțiunea de imagine integrală. Fig. 2.4 exemplifică implementarea algoritmului. Prima caracteristică este corespunzătoare zonei ochilor și calculează diferența de intensitate dintre sprâncene și ochi. Cea de-a doua caracteristică va compara intensitatea pixelilor din zona alăturată nasului cu cea a pixelilor nasului.



Fig. 2.4 - Utilizarea caracteristicilor Haar

2.1.1.3 Imagini integrale

Conversia valorilor intensității pixelilor într-o imagine integrală face posibil calculul rapid al sumei pixelilor dintr-o imagine sau dintr-un dreptunghi. Imaginea integrală într-un punct de coordonate (x,y) va conține suma tuturor pixelilor aflați deasupra și la stânga punctului considerat.

Pentru exemplificare, vom considera Fig. 2.5. Folosind imaginea integrală, valoarea dreptunghiului albastru poate fi calculată cu Formula 1 [36], unde fiecare termen reprezintă valoarea unui dreptunghi:

$$ii = ii(x_2, y_2) - ii(x_1, y_2) - ii(x_2, y_1) + ii(x_1, y_1)$$

Formula 1 - Calculul imaginii integrale

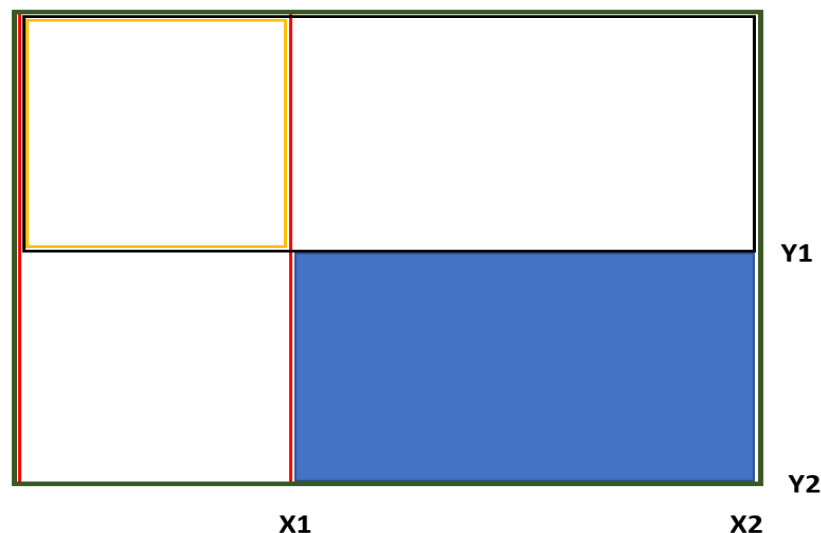


Fig. 2.5 - Exemplificarea calculului folosind imagini integrale

Astfel, scăzând valoarea dreptunghiului roșu și a celui negru din valoarea dreptunghiului verde și adăugând apoi valoarea dreptunghiului galben, se obține valoarea dreptunghiului colorat cu albastru. În acest mod, caracteristicile sunt evaluate mult mai rapid în procesul de detecție de fețe.

2.1.1.4 Clasificatori în cascadă

Pentru a evita aplicarea trăsăturilor în aceleași regiuni de mai multe ori, în condițiile în care un clasificator anterior a returnat un rezultat negativ, următorii clasificatori nu vor mai fi aplicați pentru acea zonă a imaginii. Astfel, operațiunile mai complexe vor fi aplicate numai în regiunile cu probabilitate ridicată de a reprezenta o față. Această structură a detecției în cascadă este comparată cu structura unui arbore de decizie degenerat [37], [38]. La final, zona ce are un rezultat pozitiv pentru toți clasificatorii va fi considerată a fi o față. Implementarea acestei metode face ca numărul zonelor pe care este aplicat clasificatorul final să fie redus la jumătate. De cele mai multe ori, o cascadă ce conține un număr mare de clasificatori va oferi rezultate mai satisfăcătoare, însă va necesita mai mult timp de procesare. Deși un detector de fețe rapid are multe aplicații, în situațiile în care viteza de procesare nu este esențială, algoritmul permite și implementarea unor procesări suplimentare.

Un clasificator este un arbore de decizie, iar prin selecția adâncimii maxime a acestuia, pot fi controlate 3 aspecte importante:

1. Bias-ul modelului: se dorește ca acesta să fie cât mai mic, adică rezultatul predicției să fie cât mai aproape de rezultatul corect.

2. Timpul de antrenare: pentru clasificatorii slabi se dorește un timp redus de antrenare a acestora, deoarece va fi antrenat un număr mare de clasificatori.

3. Timpul de predicție al clasificatorului trebuie să fie cât mai redus, pentru a construi un model cu o rată de predicție ridicată.

În articolul lor, Viola și Jones descriu o cascadă ce conține mai bine de 6000 de trăsături. Considerând un set dificil de date, cu 507 fețe și 75 milioane de ferestre de analiză, implementarea clasificatorului în cascadă a necesitat utilizarea a aproximativ 10 caracteristici pentru fiecare sub-ferastră a imaginii. În comparație cu soluțiile precedente, această abordare reprezintă un salt semnificativ - de exemplu, este de aproximativ 15 ori mai rapidă decât sistemul descris de Rowley et al [39].

Autorii au antrenat detectorul de fețe atât pe imagini pozitive, cât și negative. Fiind format din imagini descărcate aleator de pe internet, setul de antrenare a cuprins 4916 fețe etichetate manual. Numărul total de caracteristici ale detectorului a fost de 6061 (dispuse pe straturi) și numărul de trăsături a crescut odată cu numărul stratului, după cum reiese din primele 5 exemple:

- stratul 1: 1 caracteristică;
- stratul 2: 10 caracteristici;
- stratul 3: 25 caracteristici;
- stratul 4: 25 caracteristici;
- stratul 5: 50 caracteristici.

Fiecare clasificator a fost antrenat pe 9832 de fețe (cele 4916 menționate, și proiecțiile în oglindă ale acestora) și pe 10.000 de sub-ferestre ce nu conțineau fețe. Viteza detectorului final a depins de numărul de caracteristici evaluate pe fiecare sub-ferastră: pe un procesor Pentium III de 700 MHz, detectorul de fețe procesa o imagine de 384x288 pixeli în aproximativ 0.067 secunde. Rezultatul demonstrează saltul semnificativ realizat de cei doi cercetători, metoda prezentată de aceștia fiind de aproximativ 600 de ori mai rapidă decât detectorul Schneiderman-Kanade [40].

2.1.2 Utilizarea detecției de piele

Există algoritmi de detecție de fețe bazați pe segmentarea imaginii color [41] pentru a obține regiunile de interes, adică pixelii ce reprezintă piele. Ulterior regiunile pielii sunt împărțite în două clase: față sau non-față. Rezultatele experimentale arată că metoda menționată în această lucrare poate realiza o detecție cu o acuratețe ridicată. Pentru a evalua viteza de detecție a algoritmului propus, în cadrul articolului se realizează o comparație cu alte metode cunoscute. Algoritmul este format din 4 etape:

1. Clasificarea pixelilor din imaginea color în două mari clase (piele/non-piele), astfel că imaginea rezultată va fi o imagine binară.
2. Segmentarea imaginii binare în regiuni interconectate și consistente.
3. Eliminarea regiunilor ce nu sunt semnificative.
4. Localizarea zonelor semnificative din imagine.

Alegerea unui model de culoare pentru piele este complexă, iar detecția pielii este unul din pașii de început în algoritmul de detecție a feței. Metoda aleasă trebuie să fie rapidă și totodată să aibă o rată de succes semnificativă: un număr ridicat de rezultate de tip true positive (un pixel clasificat ca fiind piele este într-adevăr piele) și un număr cât mai redus de false positives (un pixel clasificat ca fiind piele nu este de fapt piele).

Următorul pas implementat în lucrare este eliminarea zonelor de piele ce au o dimensiune mai mică decât un anumit prag (mai exact 15x15 pixeli), astfel că zonele rămase sunt considerate potențiale fețe. Sunt verificate 4 condiții ce trebuie îndeplinite simultan:

- Imaginea binară rezultată anterior conține cel puțin trei zone care unite formează un triunghi.
- Se trasează o axă verticală ce trece prin centrul de greutate al zonei de piele selectate. Această axă trebuie să treacă și prin cea mai de jos zonă detectată la punctul 1.
- Celelalte două zone identificate la punctul 1 trebuie să aparțină celor două cadrane Q1 și Q2 exemplificate în Fig. 2.6.
- Distanțele dintre zona de jos identificată la punctul 2 și celelalte două zone trebuie să fie aproximativ egale.

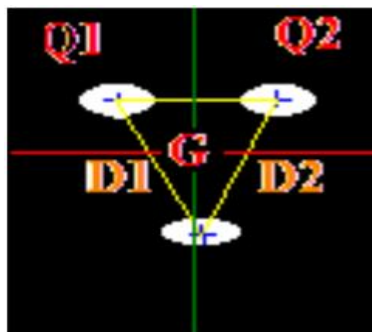


Fig. 2.6 - Modelul unei fețe. Sursa: [40]

În cadrul unei alte lucrări științifice [42], autorii compară algoritmi de detecție de fețe bazați pe culoarea pielii, implementați în 3 spații de culoare diferite. Rezultatele obținute sunt utilizate pentru a obține o nouă culoare pentru pixelii de piele, iar metoda prezentată are o acuratețe de 95,18%.

Utilizarea unui detector de piele are câteva avantaje, printre care și un timp mai redus de procesare. Acest lucru se datorează faptului că procesarea culorilor este mai rapidă decât procesarea trăsăturilor faciale. Detecția și urmărirea de fețe folosind culoarea pixelilor ce reprezintă o față este complexă, din cauza faptului că rezultatul poate fi influențat de numeroși factori, cum ar fi lumina ambientală sau poziția și mișcarea subiectului. De asemenea, reprezentarea unei persoane diferă în spațiul culorilor, de la o cameră web la alta.

Prin combinarea rezultatelor acestor algoritmi, sunt extrase regiunile ce reprezintă piele. Următorul pas este extragerea trăsăturilor faciale și construirea unui dreptunghi în jurul regiunii ce a fost identificată ca fiind față. În cazul în care algoritmi au rezultate diferite, însă cel puțin unul dintre ei a clasificat o zonă drept față, în final regiunea respectivă va fi considerată o față.

Există numeroase lucrări în care autorii prezintă metode de detecție a fețelor în video, în timp real, folosind modelul detecției de piele. De exemplu, în această lucrare [43] se realizează o diferență între cadre, după care se aplică algoritmul AdaBoost. De asemenea, sunt prezentate 3 caracteristici care îngreunează detecția de fețe:

1. Schimbările de expresie facială.
2. Existența ochelarilor, a bărbii sau a altor elemente pe față.
3. Vulnerabilitatea la schimbările de iluminare.

Algoritmul prezentat de autori constă în:

- separarea țintei de fundalul imaginii: fundalul unei imagini are întotdeauna un impact puternic asupra procesului de detecție de fețe, fapt pentru care se dorește eliminarea acestuia. Acest lucru se realizează prin calculul diferenței dintre două cadre succesive, pentru a se stabili dacă în video există

un obiect ce se află în mișcare. O problemă poate să apară însă în cazul în care obiectul are o mișcare lentă în video, care nu este identificată.

Astfel, distanța rezultată între cei doi pixeli având aceleași coordonate (x,y) în cele 2 imagini succesive se calculează aflând diferența pentru fiecare componentă R (canalul de roșu), G (canalul de verde), B (canalul de albastru), dintre cadrul anterior și cel curent. Dacă această diferență este mai mică decât un anumit prag, pixelul din imaginea diferențială va avea aceeași valoare din imaginea anterioară, altfel valoarea lui va fi 0.

2.1.3 Aplicații bazate pe detecția de fețe

În prezent, detecția, recunoașterea și urmărirea de fețe sunt operațiuni integrate în platformele de socializare sub diferite aspecte. Recunoașterea și urmărirea de fețe sunt acțiuni pentru care este necesară mai întâi realizarea detecției de fețe.

Spre exemplu, etichetarea persoanelor în imagini a devenit o acțiune tot mai simplă în ultimii ani, datorită sugestiilor de etichetare. Facebook a dezvoltat un program de recunoaștere facială [44], numit „DeepFace”, bazat pe algoritmi de detecție și recunoaștere dezvoltați de compania Face.com [45]. Acesta funcționează pe baza unui algoritm de calcul ce generează un model pentru fiecare utilizator. Modelul este realizat pe baza trăsăturilor faciale ale unei persoane, calculându-se diferite distanțe dintre ochi, nas și urechi. Programul identifică dacă două fețe din două imagini diferite sunt de fapt ale aceleiași persoane, cu o acuratețe de 97,25%, comparabilă cu cea a unui om, de 97,53%. Identificarea nu este influențată de diferențe de iluminare a fețelor sau de unghiul din care este realizată imaginea. Baza de date folosită pentru procesul de învățare a fost constituită dintr-un număr mare de imagini încărcate pe Facebook și un set de date de pe YouTube.

Pentru dezvoltarea aplicației au fost folosite modelarea 3D a fețelor și rețele neurale cu o adâncime de 9 straturi. Rețelele neurale sunt optime pentru lucrul cu seturi foarte mari de date de antrenare, și în particular, arhitectura acestei rețele este bazată pe fixarea locației fiecărei fețe, pixel cu pixel. Concret, contribuțiile pentru dezvoltarea acestei aplicații sunt:

1. Dezvoltarea arhitecturii unei rețele neurale și a unei metode de învățare ce folosește un set mare de date pentru reprezentarea de fețe.
2. Dezvoltarea unui sistem eficient de aliniere facială, bazat pe modelarea 3D explicită.
3. Obținerea unui sistem ce întrece performanțele sistemelor existente și are o rată de succes comparabilă cu performanța umană.

Snapchat este o aplicație foarte populară, ce permite modificarea imaginii în diferite modalități, cum ar fi aplicarea unor măști, a unor efecte, sau chiar „schimbarea” fețelor între persoane. Implementarea acestor opțiuni pentru utilizatori implică:

- detecția de fețe: pentru cadrele achiziționate se realizează detecția de fețe, în urma căreia este returnat dreptunghiul de detecție - coordonatele x , y (coordoanatele unui colț al dreptunghiului pentru față), h și w (înălțimea, respectiv lățimea dreptunghiului ce va delimita fața identificată).
- identificarea trăsăturilor faciale: vor fi returnate și coordonatele pentru ochi, nas, gură;
- procesarea imaginii.

Antrenarea sistemului [46] s-a făcut folosind câteva zeci de mii de fețe pe care s-au marcat manual punctele ce delimitează trăsăturile faciale. Pe baza acestora, s-a constituit apoi o mască (o rețea obținută prin unirea punctelor marcate) folosită pentru inserarea efectelor vizuale.

Serviciile cognitive Microsoft dispun de o aplicație pentru detecție a fețelor în imagini [47]. Acesta este punctul de start pentru alte aplicații ce pot fi dezvoltate și returnează un dreptunghi corespunzător feței detectate, și opțional, marcarea trăsăturilor faciale. Pentru experimentarea serviciului este recomandată folosirea unor imagini frontale de o calitate ridicată, ce pot avea o mărime de până la 6 MB. Dimensiunea minimă a fețelor ce pot fi detectate este de 36x36 pixeli, iar într-o imagine pot fi detectate cel mult 64 de fețe ce vor fi ordonate crescător, după dimensiune. De asemenea, sunt evaluate diferite atribute, precum zâmbetul (0 - inexistent, 1 - zâmbet larg) sau vârsta.

2.2 Urmărirea de obiecte

Urmărirea de obiecte (object tracking) are multiple aplicații practice, și este un obiect de studiu curent în procesarea de imagini. Pentru implementare, este necesară mai întâi localizarea trăsăturilor de interes (poziția de start), după care are loc urmărirea locației zonei respective, în cadrul clipului video. Necesitatea utilizării algoritmilor de urmărire provine din timpul mare de procesare al detectoarelor de obiecte sau de fețe.

Denumirea de „online tracking” se datorează faptului că implementarea algoritmului se face pe măsură ce se primesc noi cadre de la camera web, adică procesarea se face într-un singur sens: înainte. Există un singur cadru din care se preia informația necesară, după care urmărirea trebuie făcută corespunzător. Complexitatea problemei de urmărire este ridicată, din cauza numeroșilor factori ce pot interveni, precum:

- apariția unor surse multiple de iluminare sau o iluminare prea slabă a obiectului - acestea pot schimba aspectul zonei de interes;
- textura obiectului (mat ori strălucitor) poate afecta aspectul obiectului;
- reflexia speculară - în cazul obiectelor lucioase, are loc reflexia unei cantități semnificative de lumină, într-un cadru restrâns de unghiuri. În ceea ce privește obiectele mate, în cazul acestora are loc reflexia difuză, în care lumina este reflectată uniform;
- transparența obiectului;
- forma obiectului;
- confuzia obiectelor - de exemplu, un grup de pinguini;
- modelul de mișcare al obiectului;
- contrastul imaginii;
- mișcarea camerei web sau zoom-ul.

Un algoritm de urmărire este descris ca având 5 componente [48]:

- ținta - poate fi reprezentată de un obiect, un dreptunghi, un contur, o arie, un grup de dreptunghiuri, etc.;
- tipul caracteristicilor - poate fi un model geometric 2D sau 3D, etc.;
- modelul de mișcare - uniform, Gaussian, Implicit, etc.;
- optimizarea;
- actualizarea.

Autorii descriu un algoritm bun de urmărire ca fiind un algoritm implementat cu succes într-un număr mare de videoclipuri în care au loc schimbări precum cele menționate anterior. De asemenea, s-a realizat testarea a 19 algoritmi de urmărire, pe un set de 315 videoclipuri cu durata medie de 9,2 secunde. Setul a fost ulterior suplimentat cu 10 videoclipuri cu durata de 1-2 minute.

Pentru fiecare dintre acești algoritmi se dă un dreptunghi de urmărire inițial, după care se compară locația subiectului ce trebuie urmărit cu dreptunghiul procesat de algoritm. Singura informație necesară pentru urmărirea obiectului este locația inițială a acestuia, astfel încât rezultatul depinde numai de acuratețea algoritmului. Nu se realizează o antrenare pe obiecte sau persoane, pentru că ar fi introdusă dependența de setul de învățare.

Evaluarea performanțelor trackerilor este obiectivă, bazată pe teste statistice și s-a realizat folosind curba de supraviețuire Kaplan Meier [49]. Dintre cei 19 algoritmi testați, TLD (Tracking-Learning-Detection) se află în primii 4, având unul dintre cele mai bune rezultate. Este unul dintre trackerii disponibili în OpenCV, motiv pentru care a fost testat și în cadrul acestei lucrări.

Deși urmărirea de obiecte este o temă de actualitate, iar OpenCV este o librărie de cercetare în Computer Vision intens utilizată, aceasta oferă posibilitatea de experimentare a unui număr redus de algoritmi de urmărire a obiectelor. Algoritmii disponibili pot fi împărțiți în două categorii [50]:

- Algoritmi dezvoltați să urmărească ținta prin urmărirea traiectoriei acesteia și care folosesc predicția viitoarei locații, fapt ce presupune corecția unei erori între cadre. Unul dintre dezavantajele acestora este pierderea acurateții atunci când camera web se mișcă prea repede sau are loc o schimbare bruscă a direcției de deplasare a țintei, urmată de identificarea altui obiect în locația prezisă de algoritm. Exemple de algoritmi: MIL (Multiple Instance Learning), Boosting, MedianFlow.

- Algoritmi dezvoltați pentru a fi implementați în probleme complexe de urmărire, adaptându-se la noi condiții și corectând toate erorile semnificative existente. Dezavantajul acestora este necesitatea unei memorii mai mari și a unui consum de putere de procesare mai mare. Un astfel de algoritm disponibil în OpenCV este TLD.

În continuare, vor fi descriși algoritmi de urmărire de obiecte disponibili în librăria OpenCV [51].

Algoritmul TLD împarte activitatea de urmărire a țintei în 3 sub-activități: urmărire, învățare și detecție [52]:

- Obiectul țintă este definit prin locația sa din primul cadru, după care, pentru fiecare cadru care urmează, se dorește urmărirea țintei sau să se indice că obiectul nu mai este prezent (după caz). Estimarea mișcării obiectului de la un cadru la altul presupune faptul că acesta este vizibil, iar mișcarea este limitată. Există probabilitatea ca algoritmul să eșueze, dacă obiectul iese din cadru și revine apoi.

- Detectorul localizează toate aparițiile obiectului și corectează urmărirea, dacă este necesar. Fiecare cadru este procesat independent și este scanat complet, în vederea identificării aparițiilor observate anterior și învățate. La fel ca în cazul oricărui alt detector, și aici erorile întâlnite pot fi de două tipuri: false positives și false negatives.

- Etapa de învățare constă în aproximarea erorilor de detecție și actualizarea detectorului astfel încât să evite aceste erori în cadrele ce urmează. Se ține cont că ambele etape anterioare pot da greș.

MIL Tracker (Multiple Instance Learning) [53] este un algoritm de învățare supervizată, folosit în problemele în care nu există suficiente informații cu privire la etichetele datelor de învățare.

În învățarea supervizată instanțele sunt etichetate individual, în timp ce MIL utilizează grupuri etichetate de instanțe. Algoritmul MIL de urmărire de obiecte este bazat pe conceptul de „boosting”, adică pe combinarea unui număr mare de clasificatori slabi într-un clasificator puternic. Antrenarea acestor clasificatori se realizează secvențial. Acest algoritm nu consideră numai locația curentă a obiectului (etichete pozitive), ci ia în considerare și vecinătatea acestuia (potențiale etichete pozitive).

Algoritmul BOOSTING [54] este bazat pe o versiune online de AdaBoost și necesită să fie antrenat atât cu imagini pozitive, cât și negative ale obiectului țintă. Dreptunghiul de inițializare (ales de utilizator sau obținut prin detecție) este considerat un exemplu pozitiv al obiectului, iar pentru cadrele următoare, clasificatorul va fi rulat pentru fiecare pixel din vecinătatea locației inițiale. Se obține un scor al clasificatorului, iar centrul noii locații va corespunde valorii maxime.

Algoritmul de urmărire KCF (Kernelized Correlation Filters) [55] utilizează faptul că pentru etichetele pozitive folosite în algoritmul MIL există multe regiuni în care acestea se suprapun. Datele care se suprapun conduc la anumite proprietăți matematice care sunt folosite de acest algoritm pentru a face procesul de urmărire mai rapid și mai precis.

MedianFlow Tracker [56]: dându-se un punct în imagine, acest algoritm va încerca localizarea aceluiași punct în cadrele următoare. Pentru utilizarea algoritmului în aplicații, se va selecta de fapt o zonă de interes (desenarea unui dreptunghi). Pentru reactualizarea regiunii de interes, se va considera media punctelor pentru fiecare axă, în timpul mișcării.

Algoritmul de urmărire GoTurn [57] este bazat pe Rețele Neuronale Convoluționale. De asemenea, folosește un model pre-antrenat, fapt pentru care este mai rapid în comparație cu ceilalți algoritmi.

Algoritmul MOSSE [58] este bazat pe filtre MOSSE (Minimum Output Sum of Squared Error), care produce filtre stabile de corelație când este inițializat cu un singur cadru. Acesta este un algoritm puțin sensibil la variațiile de iluminare, scală, poziție, și poate procesa până la aproximativ 700 cadre pe secunde.

Este dificil de realizat o comparație complexă și obiectivă a algoritmilor prezentați anterior, din cauza diferitelor situații în care aceștia pot fi utilizați [59]. De asemenea, algoritmii de urmărire sunt dezvoltați pentru a depăși diferite probleme întâlnite: unii funcționează mai bine în situația în care obiectul țintă se rotește, alții fac față mai bine schimbărilor condițiilor de iluminare, sau mișcărilor bruște ale camerei web. Uneori, inițializarea trackerului eșuează, ori ținta este pierdută în timpul rulării algoritmului. Acest lucru face ca pentru o comparație cât mai realistă, să fie nevoie de un set foarte mare de date.

Fig. 2.7 prezintă rezultatele unui studiu comparativ al acestor algoritmi. Algoritmii MIL și BOOSTING au avut cele mai bune rezultate, aceștia fiind reactualizați pe parcursul videoclipului cu imaginea obiectului țintă.

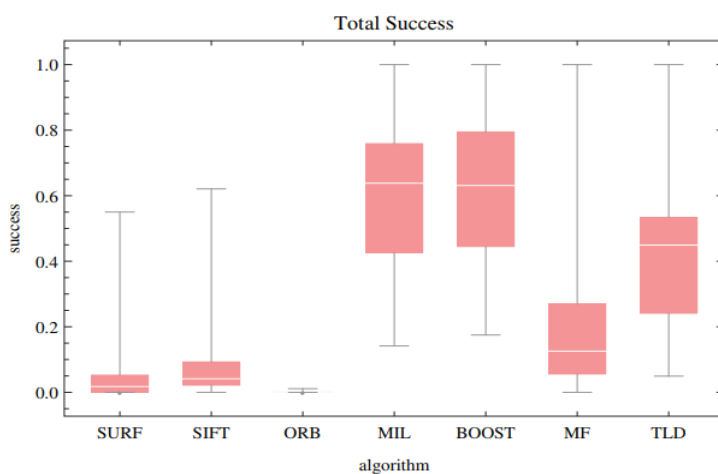


Fig. 2.7 - Rezultatele comparației algoritmilor de urmărire
Sursa: [59]

2.3 Selectarea regiunii de interes

Algoritmii de procesare a imaginilor sunt destinați detecției și localizării unor trăsături specifice dintr-o imagine digitală. Multe aplicații de supraveghere și recunoaștere funcționează pe baza detecției unor potențiale obiective, denumite și regiuni de interes (ROI - Region of Interest) în imaginile digitale. Aceste regiuni pot fi apoi utilizate pentru a solicita automat acțiuni suplimentare sau intervenții. De asemenea, poate fi necesară o analiză ulterioară a acestora, pentru identificarea și recunoașterea obiectului țintă.

A fost studiată problema utilizării regiunilor de interes pentru a crește eficiența transmisiei de imagini/video pe canale ce au lățime de bandă limitată. De exemplu, imaginile prin satelit pot fi utilizate pentru a investiga și supraveghea zboruri. Există și situații în care, din motive de eficiență, se dorește transmiterea unei imagini ce conține numai zona de interes, și nu întregul cadru. Un astfel de exemplu este prezentat în Fig. 2.8. Diferite aplicații pot beneficia puternic de distribuția inegală de resurse [60] (biți, protecția la eroare, rezoluție etc.) pe diferite regiuni de imagine. De exemplu, imaginile pot fi transmise prin atribuirea unei priorități mai mari pachetelor care acoperă o regiune considerată a necesita o rezistență mai mare la erorile de transmisiune cauzate de o rețea cu zgomot. De asemenea, în cazul codării cu o rată de compresie scăzută (ex. videoconferințe pe telefoanele mobile), este importantă codificarea anumitor zone de imagine (de exemplu fețe) cu fidelitate mai mare decât altele.

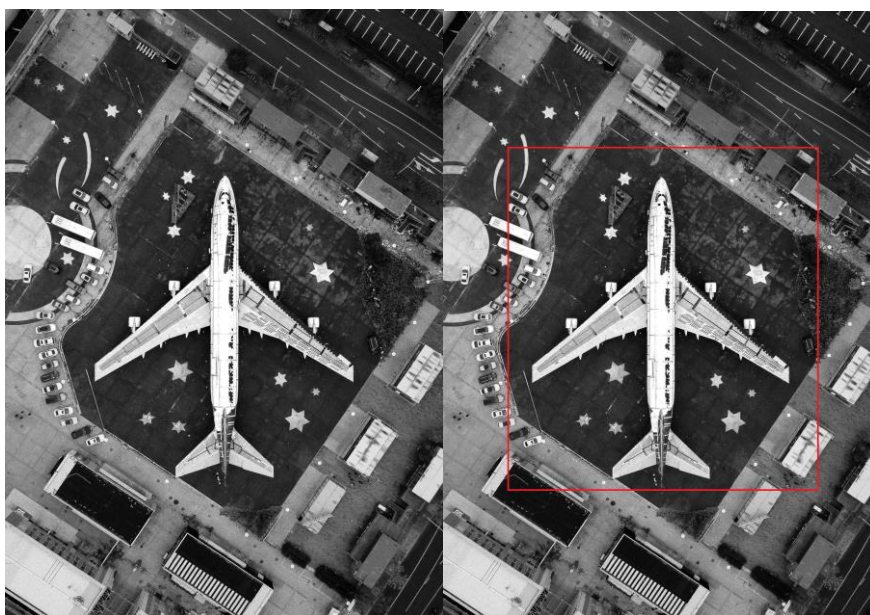


Fig. 2.8 - Selecția regiunii de interes, Sursa: [61]

În lucrarea [62] selecția regiunii de interes pe baza algoritmilor de procesare a imaginilor este comparată cu selecția naturală a acesteia, de către observator. Studiile psihologice au arătat că oamenii detectează regiuni de interes înainte de a face orice prelucrare a informației vizuale. Prin mișcările globului ocular, atenția vizuală este focusată asupra anumitor părți ale unei imagini, ce trebuie fixate și mai apoi analizate cu o atenție sporită. Autorii au arătat că, în general, oamenii privesc aceleași regiuni ale unei imagini. De obicei, la vizualizarea unui cadru nou, au loc aproximativ trei fixări diferite pe secundă, intercalate cu mișcări ale ochilor. Pentru recunoașterea unui cadru vizual

complex, creierul are nevoie de câteva astfel de fixări, regiunile respective fiind ulterior procesate în detaliu. Cercetătorii au arătat, de asemenea, că algoritmi computerizați ar putea fi concepuți pentru a găsi regiuni de interes similare cu cele determinate de oameni.

Conținutul informațional al unei imagini este înglobat în valorile anumitor parametri, ce sunt stabilite în urma aplicării algoritmilor de procesare a imaginilor. Aplicarea unui algoritm de procesare a imaginilor înseamnă reprezentarea pixelilor imaginii într-un anumit interval de valori, într-un mod caracteristic algoritmului aplicat. Astfel, se generează maxime locale specifice, iar după aplicarea unei proceduri de grupare, setul de maxime locale este redus la un subset de regiuni de interes generate după procesarea imaginii.

De-a lungul anilor, a fost studiată identificarea regiunii de interes atât pe baza caracteristicilor cât și folosind obiecte. În ceea ce privește utilizarea unor caracteristici, această metodă constă în identificarea pixelilor ce au trăsături similare cu ținta stabilită, și agregarea acestor pixeli pentru a defini regiunea de interes. Însă din cauza faptului că nu toți pixelii au caracteristici optice semnificative, uneori regiunile de interes nu sunt identificate corespunzător. Intervine și inabilitatea de a distinge țintele, ceea ce poate cauza confuzie în anumite stagii de procesare. În ceea ce privește abordarea bazată pe obiecte, aceasta utilizează informații precum forma și dimensiunea țintei, astfel că detecția regiunii de interes se face la un nivel mai performant decât prin metoda prelucrării fiecărui pixel în parte. Ambele metode sunt însă limitate atunci când sunt implementate pe imagini de o calitate scăzută.

Capitolul 3. Metoda experimentală

În timpul bătăilor inimii, sângele este pompat în întregul corp, provocând variații ale culorii pielii. Aceste schimbări nu pot fi observate cu ochiul liber, însă pot fi detectate pe baza prelucrării videoclipurilor. Pentru a implementa algoritmi de procesare a imaginilor, este necesară alegerea unei regiuni de interes, relevantă pentru observarea modului în care pixelii din zona selectată își modifică intensitatea. Prin calculul valorii medii a intensității acestor pixeli pentru fiecare cadru, prelucrarea datelor obținute și extragerea frecvenței la care apare valoarea maximă în urma procesării mediilor, se poate afla frecvența bătăilor inimii.

3.1 Descrierea algoritmului

Schema logică a algoritmului este ilustrată în Fig. 3.1.

Pentru fiecare cadru preluat de la camera web, se va decide în primul rând dacă se aplică detecția de fețe (FD – Face Detection) sau urmărirea de fețe (FT – Face Tracking):

- se aplică FD dacă se procesează primul cadru preluat de la camera web sau dacă a trecut un anumit timp de la ultimul FD aplicat. Apoi se identifică toate fețele din imagine și se va hotărî, pentru fiecare dintre acestea, dacă a fost întâlnită și la detecția aplicată anterior. Dacă o față a mai fost întâlnită, va avea loc actualizarea parametrilor acesteia (ex. momentul de timp la care fața a fost găsită ultima dată, dreptunghiul corespunzător). Dacă fața este una nouă, se va crea o persoană nouă. De asemenea, odată cu aplicarea FD are loc și inițializarea algoritmului de urmărire, care va fi aplicat începând cu următorul cadru.

- dacă trebuie aplicat FT, are loc actualizarea algoritmului de urmărire (tracker) și a parametrilor corespunzători, cum ar fi momentul de timp la care fața a fost reactualizată.

Următorul pas este eliminarea persoanelor care între timp au ieșit din cadru, numite în această lucrare „persoane istorice”. O persoană este considerată istorică dacă în urma aplicării algoritmului de urmărire sau detecție, actualizarea feței nu a mai avut loc cu succes de un anumit timp. De aceea, după fiecare FD sau FT încheiat cu succes, se salvează momentul de timp corespunzător, și se face apoi diferența de timp pentru fiecare cadru ce urmează. Dacă următoarele cadre se încheie cu o detecție/urmărire, se actualizează variabila cu noul moment de timp. Altfel, dacă persoana nu mai este găsită iar diferența de timp dintre cadrul curent și ultimul cadru este mai mare decât un prag prestabilit, persoana este eliminată. Pentru persoanele din cadru, se va afișa valoarea pulsului.

Pentru a ușura modul de lucru în timpul dezvoltării programului [ANEXA 1], am folosit un fișier de configurare [ANEXA 2]. Astfel, se stabilesc mult mai ușor diferite setări, precum: numărul de secunde după care o față este considerată „istorică”, ce algoritm de urmărire se folosește, dacă o funcție trebuie sau nu executată în timpul experimentărilor, etc.

Datele de configurare, scrise în format JSON, sunt utilizate ca setări ale programului. Folosind modulul json, setările din fișierul de configurare sunt importate folosind metoda *json.Load*. Aceasta citește datele din fișier și populează dicționarul Python, care este o colecție de date.

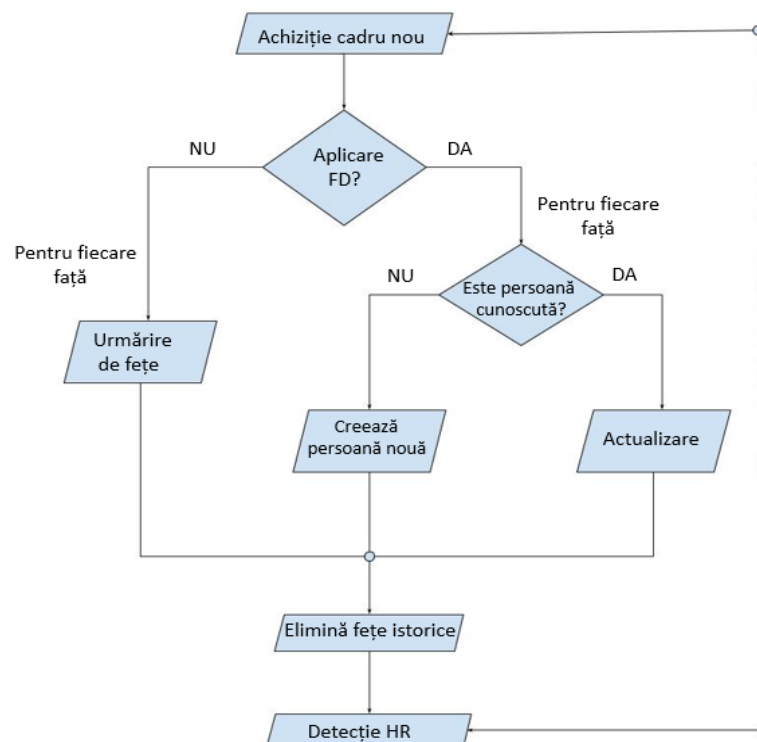


Fig. 3.1 - Schema logică a algoritmului

3.2 Detecția și urmărirea fețelor

Programul dezvoltat procesează imaginile primite de la camera web, din fiecare cadru fiind extrase informațiile de interes. Diagrama bloc a algoritmului este prezentată în Fig. 3.2.

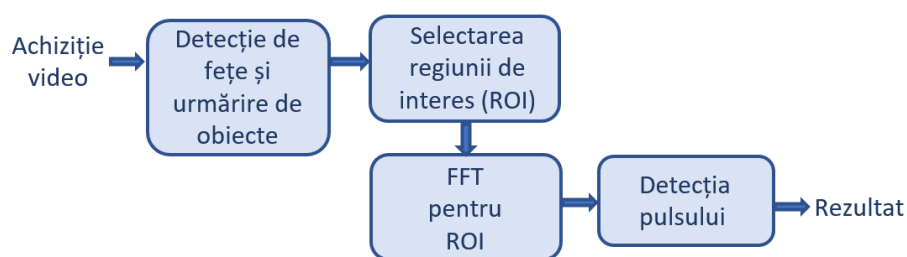


Fig. 3.2 – Diagrama bloc a algoritmului

În această lucrare, scopul principal nu este implementarea detectorului de fețe. Din acest motiv, nu am ales cea mai bună metodă curentă, ci una care este ușor accesibilă și cunoscută ca fiind deja implementată și pe alte platforme [63]. Detecția de fețe (FD - Face Detection) a fost implementată folosind Cascadele Haar. Am experimentat mai multe cascade, însă nu am observat diferențe semnificative nici la utilizarea separată a detectorului, nici în combinație cu algoritmul de urmărire.

În continuare, pentru obținerea zonei de interes, pot fi implementate două variante:

- Aplicarea algoritmului pentru detecție de fețe pentru fiecare cadru.
- Aplicarea algoritmului pentru detecție de fețe pentru primul cadru, iar în continuare (pentru următoarele „N-1” cadre), urmărirea feței detectate.

Cea de-a doua variantă este mai puțin costisitoare și mai rapidă [50]. Când este urmărit un obiect detectat în cadrul anterior, se știu detalii cu privire la aspectul obiectului. De asemenea, se știu locația din cadrul anterior, direcția și viteza mișcării sale. Cu toate acestea, se pot acumula erori și în algoritmul de urmărire, iar dreptunghiul de urmărire să se îndepărteze de obiect, cu fiecare cadru. De asemenea, dacă obiectul este acoperit de un obstacol pentru o perioadă mai îndelungată, sau dacă se mișcă prea repede, se poate pierde locația acestuia.

Am ales implementarea celei de-a doua variante, din două motive:

- Urmărirea unei fețe este mult mai rapidă în comparație cu detecția de fețe. Realizarea detecției o dată la câteva cadre, și utilizarea unui algoritm de urmărire pentru restul cadrelor înseamnă o reducere semnificativă a timpului de procesare.

- Implementarea urmăririi de fețe este necesară pentru a putea calcula pulsul pentru mai multe persoane simultan.

Am experimentat 6 algoritmi de urmărire disponibili în OpenCV:

- BOOSTING;
- MIL (Multiple Instance Learning);
- KCF (Kernelized Correlation Filters);
- TLD (Tracking-Learning-Detection);
- MEDIANFLOW;
- GOTURN;
- MOSSE;

În Fig. 3.3 se observă experimentarea algoritmului KCF.

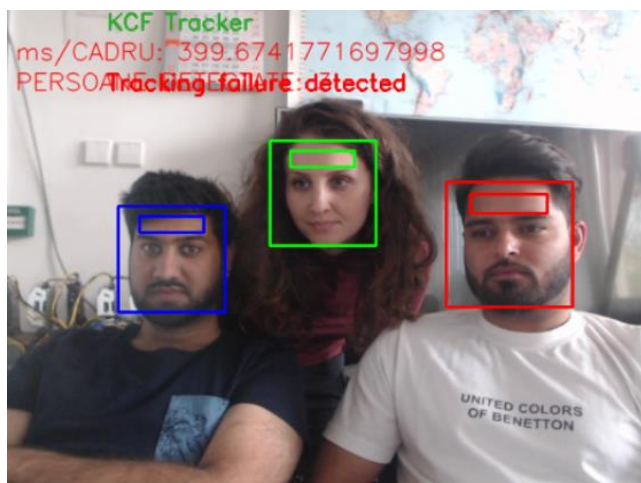


Fig. 3.3 - Experimentarea algoritmului KCF

Am realizat o comparație subiectivă a acestor algoritmi, prezentată în Tabelul 3.1, pentru care am rulat fiecare algoritm de urmărire pentru 30 de secunde. La experiment au participat 3 persoane, iar pentru fiecare dintre algoritmi am apreciat stabilitatea și cât de bine este urmărită fiecare față. Pe baza experimentării am acordat calificative de la 1 (foarte slab) la 5 (foarte bine) pentru fiecare secțiune. Atât aprecierea stabilității, cât și a capacității de urmărire a unui astfel de algoritm depind de viteza de procesare a fiecărui cadru. Un algoritm cu o viteză mare de procesare este perceput ca fiind mai stabil, din cauza faptului că dreptunghiul se va mișca mult mai repede, și nu vor fi percepute toate deplasările dintre cadre.

Nr.	Algoritm	Medie durată (ms/cadru)	Urmărire	Stabilitate
1	MIL	107.0822	4	3
2	BOOSTING	106.4241	3	3
3	KCF	19.7777	5	4
4	TLD	42.3148	3	3
5	MEDIANFLOW	10.6709	4	3
6	GOTURN	10.3720	5	4
7	MOSSE	9.2340	3	4

Tabelul 3.1 – Rezultatele experimentării algoritmilor de urmărire

Algoritmii cu rezultatele cele mai bune, atât în condiții de iluminare naturală cât și artificială, au fost KCF și MOSSE.

3.3 Alegerea regiunii de interes

Regiunea de interes este o zonă a imaginii selectată pe criterii specifice, ce urmează să fie utilizată în timpul procesărilor. Din motive ce țin de simplitatea și corectitudinea aplicării algoritmului, regiunea de interes va fi considerată în zona frunții. Pentru a observa variația culorii pielii, fruntea este cea mai indicată zonă deoarece oferă detaliat modificările întâlnite. Dimensiunile dreptunghiului ce delimitează această zonă sunt în raport cu dreptunghiul de detectare a feței, astfel încât dimensiunea ROI se schimbă în funcție de distanța dintre subiect și cameră web. Selectarea ROI poate fi observată în Fig. 3.4.

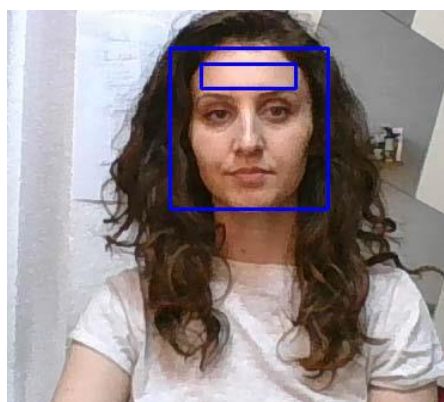


Fig. 3.4 - Selecția regiunii de interes în timpul experimentării

Următorul pas este calculul mediei pixelilor din regiunea de interes, pentru fiecare cadru. Fiecare valoare calculată este salvată într-un vector de dimensiune constantă, prestabilită. Am considerat lungimea vectorului ca fiind 250.

După rularea programului, funcția de măsurare a pulsului va fi apelată numai după ce au fost procesate minim 10 cadre, adică vectorul de medii are cel puțin 10 elemente. După câteva secunde, când vectorul ajunge să fie compus din 250 de elemente, se adaugă media corespunzătoare la sfârșitul vectorului pentru fiecare cadru nou, după care vectorul este reconsiderat ca fiind format numai din ultimele 250 elemente (primul element nu se mai ia în considerare).

După calculul valorii medii a pixelilor din ROI, am comparat rezultatele obținute pentru persoanele care au o diferență vizibilă a culorii pielii, așa cum este prezentat în Tabelul 3.2. Testele au fost realizate în aceleași condiții pentru fiecare dintre persoanele implicate.

Persoana	Vârsta	Nuanța pielii	Media pixelilor
1	23	Mediu	152.74
2	26	Mediu	127.03
3	23	Deschis	201.25

Tabelul 3.2 – Medii din ROI pentru fiecare subiect participant

În Fig. 3.5 sunt prezentate Persoana 2 și Persoana 3 în timpul efectuării testelor. În timpul experimentului, s-a încercat pe cât posibil ca persoanele să stea nemișcate, la aceeași distanță de camera web și să dispună de aceleași condiții de iluminare. Toate experimentele au fost realizate ziua, singura sursă de lumină fiind lumina soarelui. Subiecții nu au purtat machiaj și nici unul dintre ei nu suferă de vreo afecțiune a pielii.

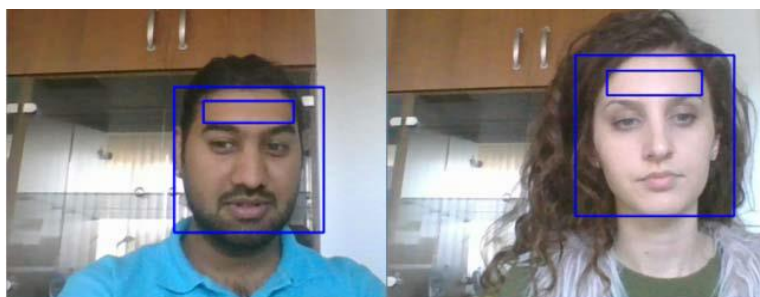


Fig. 3.5 - Persoana 2 și Persoana 3 în timpul testelor

3.4 Procesarea semnalului

Am început procesarea semnalului prin egalizarea de histogramă a cadrului primit de la camera web. Histograma unei imagini este distribuția nivelelor de gri ale acesteia (reprezentarea grafică a numărului de pixeli pentru fiecare intensitate considerată). Egalizarea histogrammei este o modalitate de îmbunătățire a contrastului unei imagini, având ca scop obținerea unei histogramme uniforme.

Următorul pas a fost aplicarea Transformatei Fourier (FFT - Fast Fourier Transform) pe vectorul format din ultimele „N” (aici maxim 250) medii obținute la punctul anterior. Pentru calculul corect al transformatei Fourier, a fost nevoie mai întâi de aplicarea unei ferestre Hamming. Atunci când se aplică transformata Fourier finită, semnalul este „tăiat” la cele două capete, ceea ce înseamnă multiplicarea acestuia cu o fereastră dreptunghiulară. Dacă cele două capete ale semnalului nu corespund, în urma aplicării transformatei vor fi introduse zgomote de frecvență înaltă. Aplicarea ferestrei Hamming menține un semnal „neted”.

Am considerat frecvențele cardiace posibile ca fiind între 35 și 195 bătăi pe minut, astfel că pentru evitarea citirilor eronate se poate aplica filtrarea frecvențelor. Acest interval de puls corespunde unor frecvențe între 0,5 Hz și 3 Hz. Această gamă de frecvențe este departe de frecvența de 50 Hz sau 60 Hz a rețelei de alimentare, ceea ce înseamnă foarte puține șanse de interferență. Pe de altă parte, componenta continuă poate influența spectrul, dată fiind apropierea frecvenței cardiace de cea de 0 Hz. De asemenea, în timpul procesului, frecvența de eșantionare are efect numai asupra densității spectrale, deoarece algoritmul va funcționa la frecvența camerei web. Mai întâi este detectat maximul, evitând valoarea componentei de 0 Hz.

Se selectează banda spectrală corespunzătoare frecvențelor cardiace între 35 și 195 bătăi pe minut. În acest fel, se rezolvă problema prezenței componente continue, care este destul de puternică. În acest semnal, în cazul ideal, ar trebui să apară un singur vârf, puternic evidențiat față de restul spectrului, care corespunde frecvenței pulsului. În realitate, în jurul frecvenței pulsului apar mai multe componente, de magnitudini diferite, din cauza mișcării subiectului, variației ferestrei de interes, etc. (Fig. 3.6 - (a)). În majoritatea cazurilor însă, semnalul de puls este mult mai mare decât aceste componente (Fig. 3.6 - (b)), așa că în algoritm am ales să selectez doar maximum din spectru, pentru care (prin proiecție pe abscisă) se află frecvența ce corespunde pulsului. Detecția de puls este prezentată în Fig. 3.7.

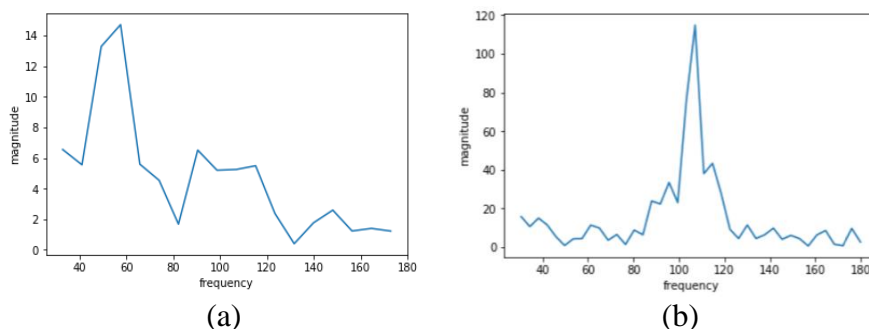


Fig. 3.6— (a) Componente de magnitudini diferite obținute din cauza perturbațiilor
(b) Componenta ce indică pulsul

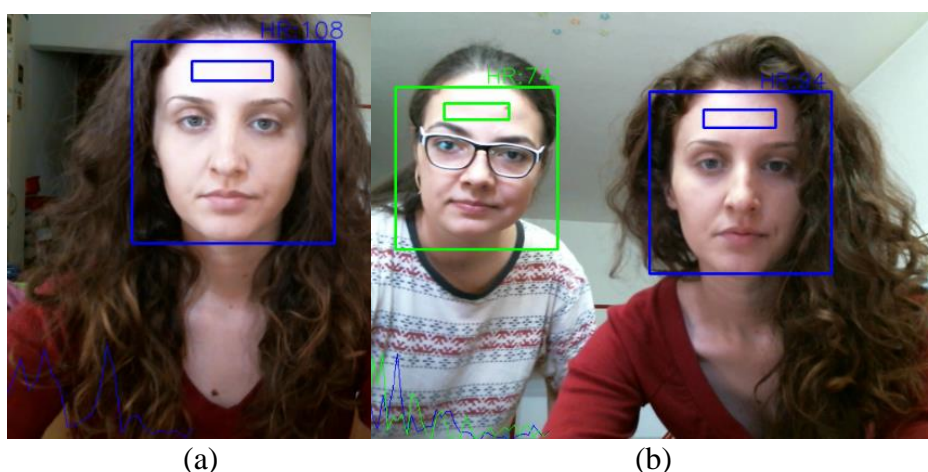


Fig. 3.7 – (a) Detecția pulsului pentru 1 persoană (lumină naturală)
(b) Detecția pulsului pentru 2 persoane (lumină naturală)

Rezultatele depind foarte mult de condițiile de iluminare. Seturile de teste au fost realizate în condiții diferite, în perioade diferite ale zilei, folosind atât lumina naturală, cât și lumină artificială. De asemenea, subiecții au participat la teste înainte și după efectuarea de exerciții fizice. Rezultatele au variat vizibil.

Capitolul 4. Testarea programului și validarea rezultatelor

În timpul dezvoltării aplicației, s-au efectuat teste folosind laptop HP ProBook 470 G4, ce dispune de:

- Procesor: Intel (R) Core™ i5-7200U, CPU @ 2.50 GHz.
- RAM: 4 GB.
- Cameră Web HD de 720p.

De asemenea, codul a fost rulat folosind și alte camere web, observându-se o diferență semnificativă a calității semnalului obținut.

- Logitech C615 (Fig 4.1) – camera permite înregistrarea de videoclipuri de rezoluție full HD, având autofocus și pentru cadre apropiate. Rezoluția este de 1920x1080 pixeli, și dispune de interfață USB 2.0.



Fig. 4.1- Camera web Logitech C615

Sursa: [64]

- Logitech BCC950 (Fig 4.2) – camera are funcție de rotire video de până la 180 de grade și permite înregistrare video HD 1080 pixeli. De asemenea, rezoluția este de 1920x1080 pixeli, iar camera dispune de interfață USB 2.0.



Fig. 4.2 - Camera web Logitech BCC950

Sursa: [64]

Diferențele apar în principal în iluminarea subiectului, zgomot de luminanță și cromatică, și în rata de imagini pe care camera le poate captura în fiecare secundă. Rezoluția mai mare nu este de ajutor, pentru că detecția de fețe se realizează pe o imagine de maxim 640x480 pixeli. Pentru imagini mai mari, s-a considerat o încetinire considerabilă a detecției, astfel că am decis limitarea rezoluției pentru a putea realiza procesarea în timp real.

Pentru evaluarea corectitudinii detecției, s-a realizat o comparație între valoarea pulsului așa cum este calculată de algoritm și pulsul pe care îl măsoară un dispozitiv Fitbit Alta HR [65], în paralel. Fitbit măsoară pulsul cu o mică întârziere, iar valorile sunt înregistrate doar o dată la câteva secunde, comparat cu metoda prezentată în această lucrare, care furnizează valori în mod continuu. Fitbit Alta HR este prezentat în Fig. 4.3.



Fig. 4.3 - Fitbit Alta HR

Considerând rezultatele măsurate prin dispozitivul Fitbit ca fiind reale, se observă că metoda implementată în lucrare obține valori ale pulsului cu erori relative între 5% și 12%. Tabelul 4.1 prezintă intervalele de valori măsurate și eroarea aproximativă pentru fiecare dintre acestea. Eroarea s-a calculat în raport cu rezultatele obținute cu dispozitivul Fitbit.

Interval	Rezultate Experimentale	
	Interval [BPM]	Eroare [%]
1	55 - 85	~ 8
2	85 - 95	~ 5
3	95 - 125	~ 10

Tabelul 4.1 – Erori obținute pentru intervale de valori măsurate

Testele au fost efectuate mai târziu în paralel cu mai multe dispozitive de măsurare a pulsului, pentru a verifica precizia măsurătorilor. Este însă destul de greu de calculat eroarea reală, din cauza faptului că și aceste dispozitive prezintă anumite erori de măsurare a pulsului. Media erorii, calculată pe întregul experiment, se situează la 8%.

De asemenea, testele au fost făcute pe nuanțe diferite de piele, pe subiecți de origine europeană și două persoane ce provin din subcontinentul indian. Nu au fost descoperite diferențe semnificative în ceea ce privește precizia măsurătorilor.

Capitolul 5. Studiul particularităților soluției mobile

În prezent, există algoritmi de măsurare a pulsului pentru care a fost implementată o soluție mobilă.

O astfel de metodă a fost dezvoltată de compania FotoNation Limited [66]. A fost propusă o abordare ce îmbunătățește performanțele procesării imaginilor pe dispozitivele mobile, precum și acuratețea detecției și urmăririi de fețe.

Autorii descriu implementarea detecției de fețe ținându-se cont de nivelul de încredere acumulat de-a lungul procesării cadrelor, ceea ce determină probabilitatea ca o față să fie prezentă într-o anumită sub-fereastră a imaginii. În situația în care valoarea probabilității depășește o valoare de prag prestabilită, este confirmată prezența unei fețe în zona procesată a imaginii. De asemenea, se descrie obținerea unor locații cu șanse mari să reprezinte o față, prin aplicarea algoritmului de detecție pe o hartă a pielii din imagine. Pentru aceasta, este mai întâi necesară implementarea unui detector de piele.

Alte metode existente deja în dispozitivele mobile, deși nu au publice datele de implementare, folosesc metode bazate pe Viola-Jones, implementate ori în hardware dedicat, ori folosind GPU.

Principalele probleme care apar în realizarea unei implementări mobile sunt reprezentate de puterea limitată de procesare și de calitatea mai scăzută a senzorului și a sistemului optic, care va produce și o imagine de calitate mai proastă.

Problemele legate de puterea de procesare de pe mobil s-ar putea compensa prin folosirea unui algoritm implementat într-un limbaj de nivel mai jos decât Python, și prin rularea a unei părți mai mari a codului în GPU, în loc de CPU.

Alte modalități prin care algoritmul se poate optimiza pentru mobil ar fi folosirea unei implementări mai puțin generice, diferită de OpenCV, în care particularitățile programului prezentat ar putea fi exploatate. De exemplu, cascada ar putea fi simplificată deoarece nu avem nevoie de o detecție perfectă a fețelor, în schimb este necesar un răspuns rapid. Variante în acest sens ar fi folosirea unei cascade supra-antrenate pentru fețe frontale, care sunt singurele de interes pentru acest caz.

Pentru a verifica fezabilitatea unei implementări mobile, am experimentat detecția de puls folosind un video înregistrat cu un dispozitiv mobil, algoritmul de detecție a pulsului fiind rulat apoi pe un laptop. Rezultatele au fost promițătoare. Detecția de fețe și urmărirea de obiecte au funcționat la fel ca în cazul camerei web din laptop. Detecția de puls s-a realizat, însă cu erori mai mari.

Concluzii

Monitorizarea valorii pulsului este esențială, deoarece variațiile neobișnuite ale acestuia ajută la stabilirea unui diagnostic și pot reprezenta o alertă înainte de apariția unor probleme mai grave, de tipul infarct miocardic. Dintre toate metodele de măsurare a pulsului, cele ce nu necesită contactul cu pielea pacientului sunt considerate de viitor atât în viața de zi cu zi, cât și în ceea ce privește integrarea acestora în aplicații de IoT. Pentru cazul unei competiții sau eveniment sportiv de masă, în care pot apărea probleme pentru unul din participanți, organizatorii pot încerca monitorizarea de la distanță a persoanelor, folosind o metodă de tipul celei prezentate în lucrare. Avantajul față de o metodă care necesită atingere este că metoda prezentată poate fi scalată ușor cu numărul de participanți.

Programul pe care l-am prezentat în această lucrare rezolvă problema monitorizării pulsului pentru mai multe persoane simultan și face ca implementarea pe dispozitivele mobile să devină fezabilă. Folosind aceasta metodă, a fost măsurat pulsul pentru maxim 4 persoane. De asemenea, algoritmul necesită un timp redus de procesare, datorită implementării detecției de fețe în conjuncție cu urmărirea de obiecte.

În ceea ce privește posibilitățile viitoare de dezvoltare a acestei lucrări, voi considera studiul influenței calității camerei web asupra rezultatelor finale și asupra numărului de fețe ce pot fi detectate și asupra cărora se poate realiza procesarea. De asemenea, voi studia variațiile ce apar în privința distanței maxime dintre camera web și subiect. Pentru evitarea erorilor de detecție de fețe, voi studia implementarea unui detector de piele, care să ajute la o detecție corectă.

O posibilă experimentare constă în monitorizarea pulsului mai multor persoane dintr-o încăpere, folosind mai multe camere web. Astfel, pentru fiecare persoană pot fi comparate rezultatele obținute de la fiecare cameră, ce pot fi corelate cu variații ale iluminării regiunii de interes.

Se poate imagina un sistem care să transmită în timp real datele de puls de la diverse persoane, împreună cu o „amprentă” a persoanei la un serviciu Cloud, unde mai apoi să se poată face o analiză mai complexă.

Bibliografie

- [1] C. Nădrag, V. Poenaru, G. Suci, "Heart Rate Measurement Using Face Detection in Video," 12th International Conference in Communications (COMM), București, 2018.
- [2] A. Hjalmarson et al., "Influence of heart rate on mortality after acute myocardial infarction," *The American Journal of Cardiology*, 1990.
- [3] Why it is important to know your heart rate," <http://blog.zensorium.com/why-it-is-important-to-know-your-heart-rate/>, accesat la data 12.04.2018.
- [4] J. Evans, A. Papadopoulos, CT. Silvers et al., "Remote health monitoring for older adults and those with heart failure: adherence and system usability," *Telemedicine Journal and e-Health.*, vol. 22, pp. 480-488, 2016.
- [5] S. Majumder, T. Mondal, MJ. Deen, "Wearable sensors for remote health monitoring," *Sensors (Basel, Switzerland)*, vol. 17, pp. 130, 2017.
- [6] Polar Electro, https://www.polar.com/us-en/about_polar/press_room/press_releases/30, accesat la data 29.04.2018.
- [7] M. E. M. Cayamcela, W. Lim, D. Kwon, "Using body-measurement indices and wrist-type photoplethysmography signals to categorize consumer electronic users' health state through a smartwatch application," *International Conference on Advances in Computing, Communication, & Automation (ICACCA)*, pp. 1-4, 2016.
- [8] Apple.com, <https://www.apple.com/ro/apple-watch-series-3/>, accesat la data 29.04.2018.
- [9] Fitbit.com, <https://www.fitbit.com/eu/home>, accesat la data 29.04.2018.
- [10] M. Zambotti, A. Goldstone, S. Claudatos, I. M. C., F. C. Baker, "A validation study of Fitbit Charge 2™ compared with polysomnography in adults," *Chronobiology International*, Vol. 35, pp. 465-476.
- [11] J. Allen, "Photoplethysmography and its application in clinical physiological measurement," *Physiological measurement*, 2007.
- [12] J. Karvonen, J. Chwalbinska-Moneta, S. Saynajakangas, "Comparison of Heart Rates Measured by ECG and Microcomputer," *The Physician and Sportsmedicine*, Vol. 12, pp. 65-69.
- [13] Oura Ring, <https://blog.ouraring.com/blog/how-to-measure-heart-rate-variability/>, accesat la data 18.05.2018.
- [14] "How wearable heart-rate monitors work, and which is best for you", <https://arstechnica.com/gadgets/2017/04/how-wearable-heart-rate-monitors-work-and-which-is-best-for-you/>, accesat la data 18.05.2018.
- [15] MedicalExpo, <http://www.medicalexpo.com>, accesat la data 18.05.2018.
- [16] "The world best Stethoscope, JABES", GS Technology Co., Ltd., <http://almasonic.com/wp-content/uploads/2013/08/JABES-Presentation-main2.pdf>, 24.05.2018.
- [17] Allheart.com, <https://www.allheart.com/jabes-electronic-stethoscope/p/jsjabes3/>, accesat la data 25.05.2018.
- [18] Heart Rate Plus - Pulse & Heart Rate Monitor, <https://play.google.com/store/apps/details?id=com.dungelin.heartrate&hl=ro>, accesat la data 25.05.2018.
- [19] Heart Rate Monitor, <https://play.google.com/store/apps/details?id=com.repsi.heartrate>, accesat la data 25.05.2018
- [20] European Commission, "Smart Wearables Reflection and Orientation Paper, Brussels, 2016.
- [21] Techcrunch.com, <https://techcrunch.com/2015/02/11/fitbit-acknowledges-latest-devices-are-causing-rashes-advises-to-give-your-wrist-a-rest/>, accesat la data 25.05.2018.
- [22] I. Bush, "Measuring heart rate from video," *Stanford Computer Science*, in press, 2016.
- [23] T. Pursche, J. Krajewski, and R. Moeller, "Video-based heart rate measurement from human faces," *International Conference on Consumer Electronics*, 2012.
- [24] H. Rahman, M. Ahmed, S. Begum, P. Funk: "Real time heart rate monitoring from facial RGB color video using webcam," 9th Annual Workshop of the Swedish Artificial Intelligence Society, 2016.
- [25] M. Lewandowska, J. Rumiński, T. Kocejko and J. Nowak, "Measuring pulse rate with a webcam — A non-contact method for evaluating cardiac activity," *Federated Conference on Computer Science and Information Systems*, Szczecin, pp. 405-410, 2011.
- [26] L.K. Mestha, S. Kyal, B. Xu, and H.J. Madhu, "Video-based estimation of heart rate variability", 2015.
- [27] Yong-Poh Yu, P. Raveendran, and Chern-Loon Lim, "Dynamic heart rate measurements from video sequences," *Biomed. Opt. Express* 6, pp. 2466-2480, 2015.
- [28] T. M. Cover and J. A. Thomas, in *Elements of information theory*, (John Wiley & Sons), 2012.
- [29] G. Balakrishnan, F. Durand and J. Guttag, "Detecting Pulse from Head Motions in Video," 2013 IEEE Conference on Computer Vision and Pattern Recognition, Portland, pp. 3430-3437, 2013.
- [30] W. W. Bledsoe, "A PROPOSAL FOR A STUDY TO DETERMINE THE FEASIBILITY OF A SIMPLIFIED FACE RECOGNITION MACHINE," 1963.
- [31] W. W. Bledsoe, "Facial Recognition Project Report", Panoramic Research Inc., Palo Alto, California, 1964.
- [32] Kairos.com, <https://www.kairos.com/blog/inspiring-uses-of-facial-recognition-in-the-real-world>, accesat la data 04.06.2018.
- [33] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol.1, pp. 511-518, 2001.
- [34] E. Schapire, Robert., "Explaining AdaBoost", 2013.
- [35] D. W. J. Meijer, "Regularizing AdaBoost to prevent overfitting on label noise," *Master Thesis*, 2016.

- [36] Computer Vision – The Integral Image, <https://computersciencesource.wordpress.com/2010/09/03/computer-vision-the-integral-image/>, accesat la data 04.06.2018.
- [37] Y. Amit, D. Geman, K. Wilder., “Joint induction of shape features and tree classifiers,” 1997.
- [38] F. Fleuret, D. Geman., “Coarse-to-fine face detection. Int. J. Computer Vision,” 2001.
- [39] R. E. Schapire, Y. Freund, P. Bartlett, W. S. Lee., “Boosting the margin: a new explanation for the effectiveness of voting methods,” Ann. Stat., vol. 26, pp. 1651–1686, 1998.
- [40] H. Schneiderman and T. Kanade., “A statistical method for 3D object detection applied to faces and cars,” International Conference on Computer Vision, 2000.
- [41] A. Hajraoui, et al. “Face Detection Algorithm based on Skin Detection, Watershed Method and Gabor Filters,” 2014.
- [42] S.Kr. Singh, D.S. Chauhan, M. Vatsa, R. Singh,J., “A robust skin color based face detection algorithm,” vol. 6, pp. 227-234, 2010.
- [43] Cui-huan DU, Hong ZHU, Li-ming LUO, Jie LIU, Xiang-yang HUANG, “Face detection in video based on AdaBoost algorithm and skin model,” The Journal of China Universities of Posts and Telecommunications, vol. 20, pp. 6-24, 2013.
- [44] DeepFace: Closing the Gap to Human-Level Performance in Face Verification, <https://research.fb.com/publications/deepface-closing-the-gap-to-human-level-performance-in-face-verification/>, accesat la data 06.06.2018.
- [45] Techcrunch.com, <https://techcrunch.com/2012/06/18/facebook-scoops-up-face-com-for-100m-to-bolster-its-facial-recognition-tech/>, accesat la data 06.06.2018.
- [46] “A Look at How Snapchat’s Powerful Facial Recognition Tech Works”, <https://petapixel.com/2016/06/30/snapchats-powerful-facial-recognition-technology-works/>, accesat la data 06.06.2018.
- [47] Microsoft Cognitive Services, <https://westus.dev.cognitive.microsoft.com/docs/services/563879b61984550e40cbbe8d/operations/563879b61984550f30395236>, accesat la data 06.06.2018.
- [48] A. Smeulders, D. M. Chu, R. Cucchiara, S. Calderara, A. Dehghan, Afshin, M. Shah, “Visual Tracking: An Experimental Survey,” IEEE Transactions on Pattern Analysis and Machine Intelligence, 2013.
- [49] MK.Goel, P. Khanna, J. Kishore, “Understanding survival analysis: Kaplan-Meier estimate,” International Journal of Ayurveda Research, 2010.
- [50] Matec Conferences, https://www.matec-conferences.org/articles/mateconf/pdf/2016/39/mateconf_csc2016_04031.pdf, accesat la data 15.06.2018.
- [51] “Object Tracking using OpenCV”, <https://www.learnopencv.com/object-tracking-using-opencv-cpp-python/>, accesat la data 15.06.2018.
- [52] Z. Kalal, K. Mikolajczyk, J. Matas, “Tracking-Learning-Detection,” in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 34, no. 7, pp. 1409-1422, 2012.
- [53] B. Babenko, M. H. Yang and S. Belongie, “Visual tracking with online Multiple Instance Learning,” IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, pp. 983-990, 2009.
- [54] P. Buhlmann, T. Hothorn, “REJOINDER: BOOSTING ALGORITHMS: REGULARIZATION, PREDICTION AND MODEL FITTING,” Statistical Science, Vol.22, pp. 477-505, 2007.
- [55] Y. Zhang, C. Zeng, H. Liang, J. Luo, F. Xu, “A visual target tracking algorithm based on improved Kernelized Correlation Filters,” IEEE International Conference on Mechatronics and Automation, Harbin, pp. 199-204, 2016.
- [56] Z. Kalal, K. Mikolajczyk and J. Matas, “Forward-Backward Error: Automatic Detection of Tracking Failures,” 20th International Conference on Pattern Recognition, Istanbul, pp. 2756-2759, 2010.
- [57] Tracker GOTURN Class Reference, https://docs.opencv.org/master/d7/d4c/classcv_1_1TrackerGOTURN.html, accesat la data 15.06.2018
- [58] D. S. Bolme, J. R. Beveridge, B. A. Draper and Y. M. Lui, “Visual object tracking using adaptive correlation filters,” IEEE Computer Society Conference on Computer Vision and Pattern Recognition, San Francisco, CA, pp. 2544-2550, 2010.
- [59] P. Janku, K. Koplik, T. Dulík, and I. Szabo, “Comparison of tracking algorithms implemented in OpenCV,” MATEC Web of Conferences. vol. 76, no. 04031, 2016.
- [60] S. Han, N. Vasconcelos, “Image Compression using Object-Based Regions of Interest,” International Conference on Image Processing, Atlanta, GA, 2006, pp. 3097-3100, 2006.
- [61] Yiran Ding, Unsplash, <https://unsplash.com/photos/Oi5kG7CPKj8>, accesat la data 27.06.2018.
- [62] C. M. Privitera, L. W. Stark, “Algorithms for defining visual regions-of-interest: comparison with eye fixations,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, no. 9, pp. 970-982, 2000.
- [63] Fotonation Limited, “Digital image processing using face detection and skin tone information,” <https://patents.justia.com/patent/9516217>, 2014.
- [64] Logitech.com, <https://www.logitech.com>, accesat la data 27.06.2018.
- [65] Fitbit.com, <https://www.fitbit.com/eu/purepulse>, accesat la data 27.06.2018.
- [66] FotoNation, “Face searching and detection in a digital image acquisition device”, <https://patents.justia.com/patent/20160104032>, accesat la data 27.06.2018.

ANEXA 1

```
from __future__ import print_function
import sys
import json
import os.path
import math
import cv2
import time
import numpy as np
from matplotlib import pyplot as plt

COLORS = [(255, 0, 0), (0, 255, 0), (0, 0, 255), (255, 128, 0), (128, 255, 0), (0, 128, 255)]
#algoritmi de urmarire
TRACKER_TYPES = {
    'BOOSTING': cv2.TrackerBoosting,
    'MIL': cv2.TrackerMIL,
    'KCF': cv2.TrackerKCF,
    'TLD': cv2.TrackerTLD,
    'MEDIANFLOW': cv2.TrackerMedianFlow,
    'GOTURN': cv2.TrackerGOTURN,
    'MOSSE': cv2.TrackerMOSSE,
}

class LoggedObject:
    DEBUG = False
    def debug(self, *args):
        if self.DEBUG:
            print(*args)

class Settings: #setari preluare date din fisier de configurare
    def __init__(self, config_file=None):
        try:
            with open(config_file) as f:
                self.config = json.load(f)
        except json.JSONDecodeError as e:
            print("Error loading config file:", str(e)) #eroare la incarcare
            sys.exit()
        except Exception as e:
            self.config = {}

    @property
    def ALWAYS_FACE_DETECTION(self):
        #detecția se face sau nu
        return not bool(self.config.get("do_tracking", False))

    @property
    def MAX_TRACK_SECONDS(self):
        #câte secunde se face urmarire, fara detectie
        return self.config.get("max_track_seconds", 0.3)

    @property
    def KEEP_PERSON_SECONDS(self):
        #dupa cate secunde o fata devine istorica
        return self.config.get("keep_person_seconds", 5)

    @property
    def CASCADE_FILENAME(self):
        #calea catre fisierul cascada utilizat
        program_dir = os.path.dirname(os.path.abspath(__file__))
        default_cascade = os.path.join(program_dir, 'haarcascade_frontalface_default.xml')
        return self.config.get("cascade_filename", default_cascade)

    @property
    def SCALE_FACTOR(self):
        #parametru detector - factor de scalare
        return self.config.get("scale_factor", 1.3)
```

```

@property
def MIN_NEIGHBORS(self):
    #parametru detector - nr minim vecini
    return self.config.get("min_neighbors", 4)

@property
def MIN_SIZE_X(self):
    #parametru detector - x minim
    return self.config.get("min_size_x", 50)

@property
def MIN_SIZE_Y(self):
    #parametru detector - y minim
    return self.config.get("min_size_y", 50)

@property
def TRACKER_TYPE(self):
    #algoritm urmarire prestabilit
    tracker_type = self.config.get("tracker_type", "MIL")
    if tracker_type not in TRACKER_TYPES:
        print("[ERROR]: Invalid tracker_type: %s" % tracker_type)
        tracker_type = "MIL"
    return tracker_type

@property
def DETECT_HEARTRATE(self):
    #calculeaza puls sau nu
    return self.config.get("detect_heartrate", False)

@property
def FONT(self):
    return cv2.FONT_HERSHEY_SIMPLEX

class Person(LoggedObject):
    COLOR_INDEX = 0

    MIN_HR = 30
    MAX_HR = 180

    def __init__(self, cntx, settings):
        self.display_fft = True
        self.cntx = cntx
        self.N = 250
        self.t0 = time.time() #moment de start
        self.means = []
        self.times = []
        self.magnitude = np.array([])
        self.freqs = np.array([])
        self.color = COLORS[Person.COLOR_INDEX]
        Person.COLOR_INDEX = (Person.COLOR_INDEX + 1) % len(COLORS)
        self.tracker_type = settings.TRACKER_TYPE
        self.font = settings.FONT
        self.last_fd_frame_time = cntx.last_fd_frame_time

    def roi(self, rectangle):
        x, y, w, h = rectangle
        return int(x+0.3*w), int(y+0.1*h), int(0.4*w), int(0.1*h)

    def roi_mean(self):
        x, y, w, h = self.roi(self.rectangle)
        rect_roi = self.cntx.g[y:y+h, x:x+w]
        return rect_roi.mean()

    def update_face(self, rectangle=None): #pe fiecare frame de fd recreez tracker si il initializez
        if rectangle is not None:
            self.last_detection_time = time.time()
            self.rectangle = rectangle #save rect pt a sti ca este al pers resp
            self.tracker = TRACKER_TYPES[self.tracker_type].create()

```

```

        self.tracker.init(self.cntx.g, tuple(self.rectangle))
        self.add_timestamp()

    def track_face(self):
        ok, rectangle = self.tracker.update(self.cntx.g)
        if ok: #daca tracking se face cu succes
            self.last_detection_time = time.time() #salveaza momentul de timp
            self.rectangle = tuple(map(int, rectangle))
            self.add_timestamp()

    def add_timestamp(self):
        self.times.append(time.time() - self.t0)
        self.times = self.times[-self.N:]
        self.debug("times: %d" % len(self.times))

    def calculate_means(self):
        self.means.append(self.roi_mean()) #formez un sir din ultima parte din vector si adaug un
        element
        self.means = self.means[-self.N:]
        self.debug("means: %d" % len(self.means))

    def calculate_hr(self):
        self.calculate_means()
        if len(self.means) < 10:
            return
        y = np.array(self.means, dtype=float)
        n = len(y) # lunigmea semnalului
        fps = float(n) / (self.times[-1] - self.times[0])
        even_times = np.linspace(self.times[0], self.times[-1], n)
        y = np.interp(even_times, self.times, y)
        y = np.hamming(n) * y #corelatie
        y = y - np.mean(y)
        raw = np.fft.rfft(y*2)
        fft = np.abs(raw)
        freqs = float(fps) / n * np.arange(n / 2 + 1)
        freqs = 60. * freqs
        idx = np.where((freqs > Person.MIN_HR) & (freqs < Person.MAX_HR))
        self.freqs = freqs[idx]
        self.magnitude = fft[idx]

    @property
    def heart_rate(self):
        if len(self.magnitude) < 10:
            return None
        max_idx = np.argmax(self.magnitude)
        return self.freqs[max_idx]

    def draw_widgets(self):
        x, y, w, h = self.rectangle #dreptunghiul de fata
        cv2.rectangle(self.cntx.frame, (x, y), (x + w, y + h), self.color, 2)
        x1, y1, w1, h1 = self.roi(self.rectangle) # dreptunghiul regiunii de interes
        cv2.rectangle(self.cntx.frame, (x1, y1), (x1 + w1, y1 + h1), self.color, 2)
        if self.heart_rate:
            cv2.putText(self.cntx.frame, "HR:" + str(int(self.heart_rate)), (x + w - 80, y - 4),
            self.font, 0.8, self.color, 1, cv2.LINE_AA)
        if self.display_fft:
            freqs = (self.freqs - min(self.freqs)) * 200. / (max(self.freqs) - min(self.freqs))
            mag = self.cntx.video_height - self.magnitude * 100 / max(self.magnitude)
            pts = np.vstack((freqs, mag)).astype(np.int32).T
            cv2.polylines(self.cntx.frame, [pts], isClosed=False, color=self.color)

    def is_my_face(self, rect):
        l, t, w, h = self.rectangle
        r, b = l + w, t + h
        l1, t1, w1, h1 = rect
        r1, b1 = l1 + w1, t1 + h1
        is_mine = l < r1 and l1 < r and t < b1 and t1 < b
        if not is_mine:
            print(rect, "is not mine", self.rectangle)
        return is_mine

```

```

@property
def display_fft(self):
    return self._display_fft and len(self.freqs) > 10

@property
def display_fft.setter
def display_fft(self, value):
    self._display_fft = bool(value)

class Program(LoggedObject):
    #constructorul programului
    def __init__(self, settings):
        #in instanta programului salvez setarile
        self.settings = settings
        self.font = settings.FONT
        self.face_detector = FaceDetector(self, settings) #creez detector de fete
        self.skin_detector = SkinDetector(self)
        self.cap = cv2.VideoCapture(0) #se incepe captura
        self.current_frame_time = None
        self.last_fd_frame_time = None
        self.last_frame_time = None
        self.persons = []
        self.timings_trk=[]
        self.timings_fd=[]

    def remove_persons(self):
        to_keep = []
        for person in self.persons:
            if self.current_frame_time - person.last_detection_time <
self.settings.KEEP_PERSON_SECONDS:
                to_keep.append(person)
            else:
                print("Deleting", id(person))
        self.persons = to_keep

    def is_facedetection_frame(self):
        #se face detectie daca:
        if self.settings.ALWAYS_FACE_DETECTION:
            return True #este specificat in setari ca se face mereu
        if len(self.persons) == 0 or self.last_fd_frame_time is None:
            return True #sau nu e nicio persoana deja detectata
            #sau daca a trecut un anumit timp de la ultima detectie
        return self.current_frame_time - self.last_fd_frame_time > self.settings.MAX_TRACK_SECONDS

@property
def video_height(self):
    return self.cap.get(cv2.CAP_PROP_FRAME_HEIGHT)

@property
def video_width(self):
    return self.cap.get(cv2.CAP_PROP_FRAME_WIDTH)

    def run(self):
        while(True):
            start = self.current_frame_time = time.time() #moment de timp la care se salveaza cadrul
            ret, frame = self.cap.read() #preluare cadru, caz in care ret = true
            frame = cv2.flip(frame, 1)
            if not ret: #daca nu s-a preluat cadru de la camera web
                continue #ruleaza din nou bucla while

            self.frame = frame
            self.gray = cv2.equalizeHist(cv2.cvtColor(self.frame, cv2.COLOR_BGR2GRAY))
            self.b, self.g, self.r = cv2.split(self.frame) #canalele de culoare
            self.skin_map = self.skin_detector.get_skin()
            #verific daca frame curent este de detectie sau tracking
            face_detection = self.is_facedetection_frame()
            self.debug(len(self.persons), "FaceDetection" if face_detection else "FaceTracking")

            if face_detection:
                faces = self.face_detector.get_faces() #iau toate fetele detectate
                self.last_fd_frame_time = self.current_frame_time

```



```

        is_already_a_person = [False] * len(faces) #vector initializat cu False
        for person in self.persons: #pentru fiecare persoana detectata
            found_my_face = False
            for i, face in enumerate(faces): #verific ce fata se potriveste
                if not is_already_a_person[i] and person.is_my_face(face): #pentru fiecare
persoana iau toate fetele
                    is_already_a_person[i] = True #notez căf faÈa aparÈine deja unei
persoane
                    person.update_face(face) #update cu noul dreptunghi de fata
                    found_my_face = True #notez ca pt persoana respectiva am gasit fata
                    break #caz in care ma opresc din cautat
            if not found_my_face: #daca nu am gasit fata (detectorul are erori)
                self.debug("Updating my face with the previous frame", id(person)) #iau
ultima fata pe care o stiam
                person.update_face() #adauga inca un element in vectorul de medii si in cel
de timp
                #este necesar ca subiectul sa nu se fi miscat prea mult
                #daca am un id de persoana pentru care nu am gasit fata, utilizez ultima fata
detectata
                #in cazul in care persoana nu mai este in cadru, va deveni persoana istorica
            for face, is_person in zip(faces, is_already_a_person):
                if not is_person: #pentru fetele care nu sunt persoane
                    person = Person(self, self.settings) #creez o persoana
                    print("Creating a new person", id(person))
                    person.update_face(face) #update
                    self.persons.append(person) #se adauga persoana la lista de persoane
        else:
            # frame-ul current este un frame pe care se face object tracking
            for person in self.persons:
                person.track_face()
            self.remove_persons()
            # Afiseaza frame-ul rezultat
            for person in self.persons: #pentru fiecare persoana
                if self.settings.DETECT_HEARTRATE:
                    person.calculate_hr()
                person.draw_widgets() #desenez dreptunghiul de fata
            end = time.time() #masoara cât a durat procesarea cadrului
            if face_detection:
                self.timings_fd.append((end-start)*1000) #adaug ori în timings de detectie
            else:
                self.timings_trk.append((end-start)*1000) #ori în timings de urmarire
                #cv2.putText(frame, str(int(1000*(end - self.current_frame_time))) + "ms/frame", (10,50),
self.font, 0.8, (0,0,255), 1, cv2.LINE_AA)
                # cv2.putText(frame, "PERSOANE DETECTATE: " + str(len(self.persons)), (10,80), self.font,
0.8, (0,0,255), 1, cv2.LINE_AA)
                cv2.imshow('frame', frame)
                if cv2.waitKey(1) & 0xFF == ord('q'):
                    break
            self.cap.release()
            cv2.destroyAllWindows()
            print("Mean time/frame with FD:", sum(self.timings_fd)/len(self.timings_fd))
            print("Mean time/frame with Tracking:", sum(self.timings_trk)/max(1, len(self.timings_trk)))

class SkinDetector(LoggedObject):
    def __init__(self, cntx):
        self.cntx = cntx

    def get_skin(self):
        # TODO: detectie de piele
        return self.cntx.frame

class FaceDetector(LoggedObject):
    def __init__(self, cntx, settings):
        self.cntx = cntx
        self.detector = cv2.CascadeClassifier(settings.CASCADE_FILENAME)
        self.scaleFactor = settings.SCALE_FACTOR
        self.minNeighbors = settings.MIN_NEIGHBORS
        self.minSize = (settings.MIN_SIZE_X, settings.MIN_SIZE_Y)

```

```

def get_faces(self):#pentru implementare detectie de piele
    faces = []
    for face in self.detector.detectMultiScale(self.cntx.frame, scaleFactor=self.scaleFactor,
        minNeighbors=self.minNeighbors,
        minSize=self.minSize,
        flags=cv2.CASCADE_SCALE_IMAGE):
        if self.check_skin(face, self.cntx.skin_map):
            faces.append(face)
    return faces

def check_skin(self, face, skin_map):#verific daca zona contine suficienta piele
    # TODO: daca in dreptunghi suprafata cu piele> 60%
    return True

if __name__ == "__main__":
    app = Program(Settings("hr_config.json"))
    app.run()

```

ANEXA 2

```
{  
  "do_tracking": true,  
  "max_track_seconds": 0.5,  
  "tracker_type": "MOSSE",  
  "detect_hearttrate": true,  
  "keep_person_seconds": 5,  
  "display_fft": false  
}
```