Anexa 3

Universitatea “Politehnica” din București

Facultatea de Electronică, Telecomunicații și Tehnologia Informației

Sistem poligraf automat bazat pe analiza feței

Proiect de diplomă

prezentat ca cerință parțială pentru obținerea titlului de

Inginer în domeniul Electronică şi Telecomunicaţii

programul de studii de licență Microelectronică, Optoelectronică şi Nanotehnologii

Conducător științific

Prof. dr. ing. Bogdan Emanuel IONESCU

Absolvent

Dan-Cristian STANCIU

Anul 2019

Universitatea "Politehnica" din Bucureşti
Facultatea de Electronică, Telecomunicaţii şi Tehnologia Informaţiei
Departamentul **TEF**

Anexa 1

TEMA PROIECTULUI DE DIPLOMĂ
a studentului **STANCIU D.F. Dan-Cristian , 441E**

1. Titlul temei: Sistem poligraf automat bazat pe analiza feței

2. Descrierea contribuţiei originale a studentului (în afara părţii de documentare) și specificații de proiectare:
Contribuția originală principală va consta în dezvoltarea de tehnici de analiză și prelucrare video a informației faciale și de învățare automată cu scopul de a dezvolta algoritmi ce permit identificarea automată a emoțiilor unei persoane, asociate cu un comportament disimulat sau ascuns. Scopul final este obținerea unui "detector de minciuni" automat.

Cercetarea aferentă va include: (1) sinteza bibliografică a realizărilor actuale din domeniu și identificarea limitărilor sistemelor existente, (2) dezvoltarea de algoritmi de descriere a emoțiilor din imagini și video, (3) dezvoltarea de algoritmi de învățare automată ce permit clasificarea emoțiilor ca fiind specifice unui comportament disimulat, sau nu, (4) simularea și validarea experimentală a algoritmilor dezvoltați, (5) concluzionarea cercetării și a contribuției originale cât și enunțarea perspectivelor ulterioare de cercetare.

3. Resurse folosite la dezvoltarea proiectului:
Proiectul va necesita folosirea de medii de dezvoltare precum Matlab, Visual C++, Python, cât și accesul la camere video (se vor folosi resursele Centrului de Cercetare CAMPUS).

4. Proiectul se bazează pe cunoştinţe dobîndite în principal la următoarele 3-4 discipline:
Programarea Calculatoarelor, Structuri de Date și Algoritmi, Programare Obiect Orientată

5. Proprietatea intelectuală asupra proiectului aparţine: studentului

6. Data înregistrării temei: 2019-01-15 00:57:11

Conducător(i) lucrare,
Prof. dr. ing. Bogdan Emanuel IONESCU

semnătura: .....

Student,

semnătura: .....

Director departament,
Conf. dr. ing. Marian VLĂDESCU

semnătura: .....

Decan,
Prof. dr. ing. Cristian NEGRESCU

semnătura: .....

Cod Validare: **eeeb865851**

Declarație de onestitate academică

Prin prezenta declar că lucrarea cu titlul *" Sistem poligraf automat bazat pe analiza feței"*, 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 *Electronică și Telecomunicații*, programul de studii de licență *Microelectronică, Optoelectronică și Nanotehnologii* 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 măsurătorilor 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, 28/06/2019

Absolvent Dan-Cristian STANCIU

(semnătura în original)

Cuprins

Listă de figuri

Listă de tabele

Lista acronimelor

EKG = Electrocardiograf
RC = Ritm Cardiac =
BPM = Bătăi pe minut =
FPG = Fotopletismogramă =
ARS = Aritmia respiratorie sinusală
ACI = Analiza componentelor independente
EVM = Amplificarea Eulerianã a Video-ului
IR = Infraroșu
TFD = Transformata Fourier Discretă
FFT = Transformata Fourier Rapidă
RGB = Roșu, Verde, Albastru
FPS = Cadre pe secunda = Frames per second
BGR = Albastru, Verde, Roșu

Introducere

În perioada recentă, evoluții în domeniul procesării de imagini și vederii automate au dus la performanțe impresionante în detecția, analiza și măsurarea diverselor fenomene ce se petrec la nivelul feței umane, precum: detecția emoțiilor, microemoțiilor, mișcări ale feței sau ochilor ,observabile cu ochiul liber sau nu. Dacă aceste informații pot fi studiate atât de calculatoare și algoritmi de învățare automată cât și de un ochi uman antrenat, există și anumite fenomene ce sunt invizibile chiar și pentru experți în domeniul psihologiei bine antrenați. Această lucrare propune prezentarea unor algoritmi care pot extrage informații de la nivelul feței/bustului care sunt invizibile pentru ochiul uman: măsurarea ritmului cardiac(RC), ritmului respirator și detecția, măsurarea și afișarea mișcărilor subtile din fișiere video.

Pulsul, respirația și mișcarea sunt procese fundamentale întâlnite în rândul oamenilor și de multe ori necesită aparate specializate pentru măsurarea lor. Această lucrare va prezenta metode de măsurare non-contact ale acestora, urmând apoi explicarea metodelor de implementare personale ale algoritmilor și posibilele lor utilizări.

Scopul final al lucrării este extragerea cât mai precisă a informației enumerate mai sus, apoi urmând realizarea unui sistem ce are că intrări rezultatele măsurătorilor anterioare și are ca scop o detecție binară a unui comportament disimulat/nedisimulat sau a emoțiilor ce se pot asocia cu disimularea. Un ultim pas va fi testarea acestor algoritmi pe o baza de date concepută personal ce ar putea arăta un comportament disimulat. Menționez de asemenea faptul că algoritmii ce vor fi prezentați ulterior nu au o utilizare limitată la un sistem poligraf, ei având potențiale utilizări în medicină , în siguranță și nu numai.

În această lucrare se vor prezenta diverse metode consacrate de extragere a pulsului din secvențe video filmând la nivelul feței și aplicarea lor cu ajutorul a mai mulți senzori optici. De asemenea, se va realiza o comparație între metodele folosite în literatură, fiecare folosind diverse tipuri de senzori. Se va realiza o comparație între metodele actuale de extragere a pulsului ce necesită contact și metodele ce se pot regăsi în stadiul curent al tehnologiei și dezvoltării. Prin comparație, se va prezenta o implementare personală a mai multor variante ale acestor algoritmi și eventualele îmbunătățiri de performanță aduse. Pulsul este una dintre cele mai importante informații privind emoțiile ce afectează o persoană (sau schimbarea de emoții) în depistarea unui comportament disimulat sau ascuns, această lucrare încercând să deducă corelația dintre acestea.

O altă informație importantă ce va fi determinată cu ajutorul algoritmilor perezentați în această lucrare este ritmul respirator, care poate fi de asemenea corelat cu unele emoții.

În cele din urmă, se va prezenta o abordare diferită asupra măsurării ritmului cardiac, folosind algoritmul de amplificare Euleriană prezentat recent în cadrul institutului de tehnologie din

Massachusetts(MIT). Acest algoritm va permite o vizualizare a schimbărilor în timp în cadrul unei imagini, precum: culoare (cu diferite frecvențe) sau mișcări greu sau imposibil de observat cu ochiul uman.

Cele trei canale de informație (puls/schimbarea culorii pielii, ritmul respirator și mișcarea amplificată) vor crea un sistem ce permit unui expert uman și poate chiar și unui non-expert să vadă diverse semne ce indică posibile emoții/reacții ce ar putea să nu fie în concordanță cu comportamentul uman ce se vede cu ochiul liber, ajungând astfel la o posibilă concluzie privind nivelul de disimulare din imagine.

Deși semnele de disimulare se pot observa de cele mai multe ori din măsurătorile făcute mai sus, acestea țin de cele mai multe ori de o părere subiectivă a unei persoane. Având în majoritatea timpului nevoie de un algoritm obiectiv, lucrarea va prezenta în continuare și crearea unei baze de date ce conține diverse secvențe cu persoane având un comportament disimulat sau nu, și o primă încercare de creare a unui algoritm de învățare automată ce va clasifică comportamentul binar, că disimulat sau non-disimulat. În cele din urmă, se vor prezenta perspective de viitor privind acești algoritmi și îmbunătățirea lor, precum și motive pentru o eventuală precizie scăzută a acestora.

1.Context

Monitorizarea semnelor vitale este esențială atât în medicină, cât și în scopul detectării emoțiilor și disimulării. În general, dispozitive ce fac contact cu utilizatorul au precizie ridicată, dar necesită cooperare a subiectului. Mai mult decât atât, prezenta acestor dispozitive poate influența rezultatele măsurătorilor, deoarece faptul că pulsul este măsurat poate provoca anxietate în anumite situații, emoție ce poate schimba ratabătăilor inimii.

În cazul aplicațiilor din medicină, o măsurare fără contact a semnelor vitale poate fi folosită pentru măsurarea pulsului și ritmului respirator al copiilor, astfel neavând nevoie de o cooperare din partea acestora. În cazul adulților dar și al copiilor, o monitorizare prin camere de filmat a semnelor vitale poate fi folosită noaptea, atunci când mișcările involuntare din timpul somnului pot cauza dereglarea contactelor aparatelor tradiționale. În cele ce urmează vor fi prezentate metodele de măsurare a semnelor vitale fără contact și se va realiza o comparație cu cele tradiționale folosite în medicină, precum electrocardiograful (EKG).

În cazul detectării emoțiilor și disimulării, se va prezenta în capitolul următor efectul emoțiilor asupra ritmului cardiac și respirației. Se va arăta că unele emoții au un efect semnificativ asupra variațiilor numărului de bătăi ale inimii pe minut, astfel metodele de măsurare dovedindu-se o metodă bună pentru a încerca cuantificarea emoțiilor.

În final, se va prezenta o abordare ce presupune folosirea unui algoritm de învățare automată pentru a extrage informații cu privire la emoțiile umane ce pot conduce la dezvăluirea unui comportament disimulat ,din fracvența bătăilor inimii. Odată cu dezvoltările tehnologice, metodele de învățare automată bazate pe rețele neuronale au devenit tot mai viabile, puterea de calcul crescând tot mai mult. Algoritmul de învățare automată va folosi informațiile din baza de date de disimulare creată și adnotată personal, despre care se va discuta ulterior. Un rezultat final ideal al acestei lucrări ar fi un detector binar de disimulare, folosind tehnici de extragere a informației non-contact.

1.1. Ritmul cardiac

Ritmul cardiac este definit că și numărul de bătăi ale inimii realizate în perioada de 1 minut(beats per minute = bpm). Deci, știind frecvența bătăilor inimii, f, bpm = 60 * f. Frecvența bătăilor inimii poate varia în funcție de nevoile fiziologice ale corpului, deoarece un rol principal al inimii este oxigenarea sângelui și eliminarea dioxidului de carbon.

Frecvența cu care inima bate este o funcție ce ține de mulți factori, deci această poate varia mult în timp. Câțiva factori ce pot influență frecvența bătăilor inimi sunt: starea de oboseală, vârstă, sexul , exercițiul fizic, somnul, anxietatea, stresul, bolile (de la răceală obișnuită până la boli ce afectează exclusiv inima și funcționarea ei), somnul, consumul sau lipsa anumitor substanțe. Cunoscând aceste date, experții pot depista de multe ori motivele schimbărilor în evoluția în timp a RC.

BPM-ul măsurat la un om este așteptat să fie între niște valori standard din punct de vedere statistic, depinzând de vârstă, sex, condiția fizică. În Tabelul 1 și Tabelul 2 sunt prezentate intervalele în care ar trebui să se afle valoarea de bpm, în funcție de sex, vârstă și condiție fizică:

Vârsta	0-3 luni	Copii 1-10 ani	Adulți	60+ ani
Atlet	-	-	45-55	50-55
Condiție fizică medie	99-149 bpm	69-129	60-75	70-75
Condiție fizică slabă	-	-	85+	80+

Tabelul 1.1. BPM-ul așteptat pentru FEMEI, in funcție de vârsta, condiție fizică, in repaos

Vârsta	0-3 luni	Copii 1-10 ani	Adulți	60+ ani
Atlet	-	-	54-60	54-59
Condiție fizica medie	99-149 bpm	75-130	73-78	73-76
Condiție fizica slabă	-	-	85+	84+

Tabelul 1.2. BPM-ul așteptat pentru FEMEI, in funcție de vârsta, condiție fizică, in repaos

Din cele două tabele prezentate mai sus se poate observa că un bărbat va avea, în medie, bpm-ul puțin mai mic decât al unei femei. Se poate observa de asemenea că există o corelație strânsă între condiția fizică și ritmul cardiac mic. În cele din urmă, se poate observa cum numărul de bătăi ale inimii pe minut scade odată cu vârsta.

Pulsul reprezintă palpitații ale arterelor datorate sângelui pompat de către mușchii inimii în corp. În general, pulsul este echivalent cu ritmul cardiac. Sângele este pompat de către inimă în vene și artere la fiecare bătaie a inimii. În general, pulsul poate fi măsurat fără instrumente de măsură, folosind

doar degetele plasate pe o arteră(în zona gâtului, în zona încheieturii etc), pentru a simți compresia și decompresia arterelor datorită creșterii și scăderii presiunii sângelui din această, odată cu bătaia inimii. În Figura 1.1 se poate observa un grafic arătând creșterea și scăderea presiunii sângelui în artere, în timpul unui ciclu cardiac(o bătaie a inimii) [1].

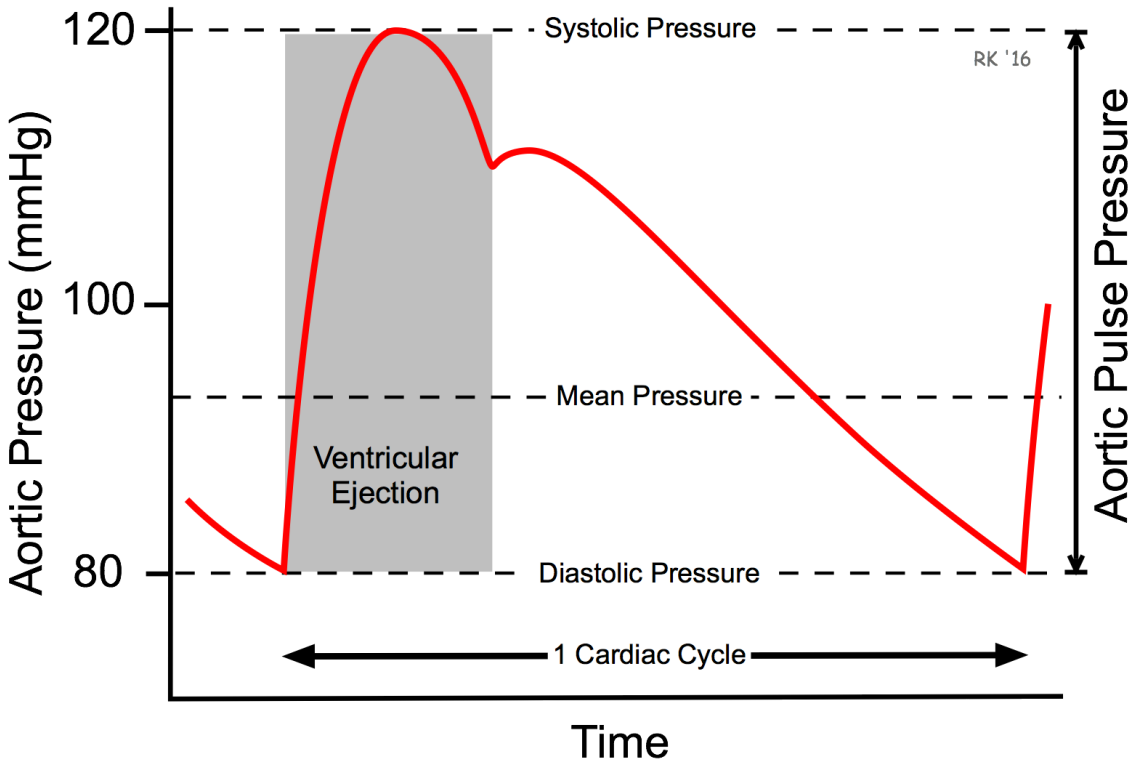

Figura 1.1. Presiunea sângelui in artere, in timpul unui ciclu cardiac [1]

Figura 1.1 prezintă presiunea sângelui în artere, în timpul unui bătăi a inimii. Pe axa x este reprezentată presiunea în cea mai mare arteră din corpul uman, numită Aorta, ce trece prin regiunea pieptului. Axa y reprezintă timpul, în care are loc un ciclu cardiac. Se poate observa că presiunea în artere crește odată cu lansarea sângelui din ventricule, apoi scăzând aproape constant. Această presiune poate fi simțită fără ajutorul aparatelor de specialitate și din numărul de pulsații pe minut ale arterei, poate rezulta bpm-ul.

1.1.1. Metode de măsură pentru ritmul cardiac

În acest capitol se vor enumera metodele de măsură ale ritmului cardiac, urmând că apoi metodele folosite în această lucrare să fie explicate în detaliu în cele ce urmează.

Cea mai simplă metodă, explicată și mai sus este numărarea numărului de contracții pe minut ale arterelor într-o perioada de timp determinată. Această metodă se poate realiza de către persoane fără experiență, în orice moment de timp.

O metodă de mare precizie pentru măsurarea ritmului cardiac este electrocardiogramă(EKG). Această metodă permite măsurarea și înregistrarea activității electrice a fibrelor musculare ale inimii, utilizând electrozi plasați în diferite zone ale pielii. Acești electrozi detectează încărcarea și descărcarea electrică(polarizarea și depolarizarea) mușchilor inimii. Acest dispozitiv este folosit în special pentru a detecta diferite afecțiuni ale inimii.

În ultimii ani, dispozitivele de tip smartwatch (sau "ceas inteligent") au devenit tot mai accesibile pentru populație. Aceste dispozitive de folosesc de metodă numită Fotopletismograma [2] (FPG) pentru a detecta schimbările de presiune și volum de pe suprafață pielii.

O pletismogramă este o măsurătoare a schimbărilor de volum într-un organ sau o suprafață a corpului, care rezultă datorită fluctuațiilor de cantitate de sânge sau aer regăsit în această zonă. FPG este obținută utilizând senzori optici pentru a măsură culoarea pielii , care este la rândul ei luminată în mai multe puncte. La fiecare bătaie a inimii, inima va pompa sânge către organe. Schimbarea de volum cauzată de presiunea pulsului fiind detectată de senzorul optic deoarece lumina se reflectă diferit înainte și după schimbarea volumului de sânge din zona măsurată. În general, sursele de lumina ce vor lumina pielea în cazul dispozitivelor de tip ceas inteligent sunt verzi, deoarece sângele fiind roșu, suprafață pielii va reflectă lumina roșie și va absorbi lumina verde. Deci, când sângele va trece prin încheietură, mai multă lumina verde este absorbită iar senzorul optic va detecta acest fenomen. Rezultatul este un semnal ce tinde spre a fi periodic, reprezentând pulsul persoanei. FPG poate fi folosită nu numai pentru a monitoriza pulsul, ci și pentru a monitoriza respirația, hipovolemia sau alte condiții ale sistemului circulator.

Figura 1.2 prezintă un ceas inteligent , având sistemul de măsurare a ritmului cardiac pe spate(astfel încât să fie în contact cu pielea). Se pot observa cele două surse de lumină verde care luminează zona de interes și senzorul optic care măsoară culoarea (deci, câtă lumina verde a fost absorbită) pielii.

Figura 1.2. Sistemul de detectie si măsurare a RC evidențiat pe un ceas inteligent

Această lucrare va prezenta metode non-contact care se bazează pe aplicarea FPG pe imagini înregistrate la nivelul feței, observând astfel schimbările de culoare aduse la nivelul pielii, datorate traversării sângelui prin zone ale feței(în principal, vasele de sânge aflate la nivelul frunții). Se va dovedi că acuratețea metodelor de măsură non-contact ce vor fi prezentate nu este departe de cea a dispozitivelor tip smartwatch(cele mai noi dintre acestea fiind, la rândul lor, la doar câteva bătăi pe minut: având variantă față de dispozitive care măsoară ritmul cardiac cu ajutorul semnalelor electrice între 0.67 bpm și 2.3 bpm[3]).

1.2. Ritmul respirator

Ritmul respirator este definit că numărul de respirații realizate de o persoană într-un minut. Ritmul respirator este ușor de măsurat, fiind necesare doar numărul de respirații în perioada de timp. O altă metodă de măsură este măsurarea ridicărilor și coborârilor la nivelul pieptului unei persoane.

Tabelul 3 prezintă numărul ritmul respirator mediu pentru persoane, în funcție de vârsta acestora [4][5].

Varsta	Numar de respirații pe minut
0-6 săptămâni	30-40
6 luni	25-40
3 ani	20-30
3-10 ani	17-25
Adulți	12-18
Bătrâni(65-80 ani)	12-28
Bătrâni(80+ ani)	10-30

Tabelul 1.3. Numărul de respirații pe minut in funcție de vârsta persoanei

Din informația din Tabelele 1,2,3 se poate observa o posibila corelație intre ritmul respirator si ritmul cardiac, ambele mărimi scăzând odată cu vârsta. Totuși, cele doua nu sunt mereu corelate in timp real.

In cele ce urmează, se vor prezenta două metode pentru determinarea ritmului respirator: una, folosind senzorul Kinect v2 , iar cea de-a doua folosind FPG.

1.3. Aritmia respiratorie sinusală(ARS)

Prin definiție, aritmia reprezintă o tulburare în ritmul bătăilor inimii.

Aritmia respiratorie sinusală este definită că variația ritmului cardiac care are loc cu fiecare ciclu de respirație. Ritmul cardiac crește când are loc inspirația și scade când are loc expirație[8]. Se crede că ARS este cauzată de proximitatea anatomică a inimii și plămânilor.

Deși ARS provoacă iregularități în ritmul respirator, nu este o afecțiune periculoasă. Majoritatea oamenilor au simptomele ARS, această fiind uneori mai proeminentă în rândul sportivilor.

Figura 1.3 prezintă rezultatele măsurătorilor bătăilor inimii unei persoane în cazul testelor făcute cu ajutorul unui pletismograf (pentru măsurarea ritmului cardiac) și a unui spirometru(pentru măsurarea volumului de aer inspirat și expirat) [9], pe durata a un minut:

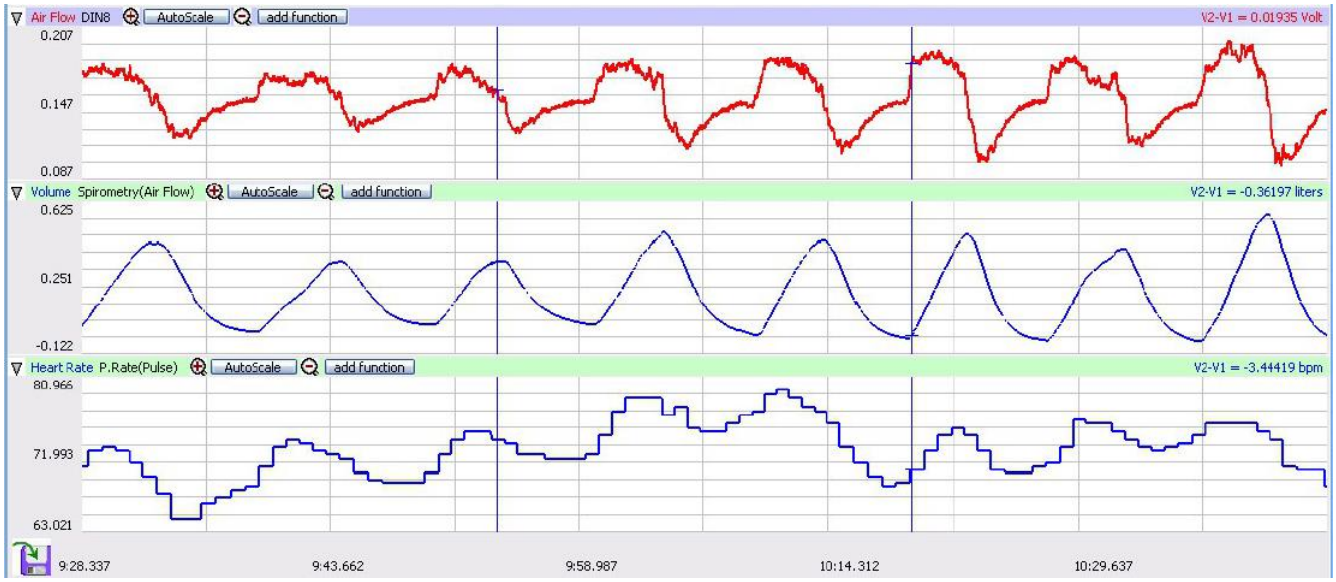

Figura 1.3. Rezultatele măsurătorii pulsului împreuna cu volumul de aer inspirat si expirat [9]

Figura 1.3 prezintă corelația între ritmul cardiac (în bpm) , prezentat în partea de jos a graficului, și volumul de aer aflat în plămâni, prezentat în partea se sus a graficului. Se poate observa o corelație mare între cele două.

Unele dispozitive de tip ceas inteligent țin cont de acest efect, pentru a arată un puls cât mai constant. Este de reținut că efectele respirației asupra ritmului cardiac nu trebuie confuzate cu efectele altor factori asupra respirației (cum ar fi emoțiile, anxietatea).

1.4. Amplificarea video

În anul 2012, cercetătorii MIT au publicat lucrarea "Eulerian Video Magnification for Revealing Subtle Changes în the World" [6], prezentând metode de a amplifica variații spațio-temporale pentru fișiere video. Revenind la FPG, semnalul rezultat din măsurarea nivelului de culoare din anumite regiuni de interes a fost dovedit a putea deduce mărimi de interes cum ar fi pulsul sau respirația. Totuși, aceste mărimi nu pot fi observate și cu ochiul liber de către omul de rând. Lucrarea prezentată în [6] prezintă o nouă tehnică de a amplifică variații în timp, rezultând în variații ale mărimilor vizibile chiar și pentru ochiul uman și cu acuratețe crescută pentru algoritmii precedenți.

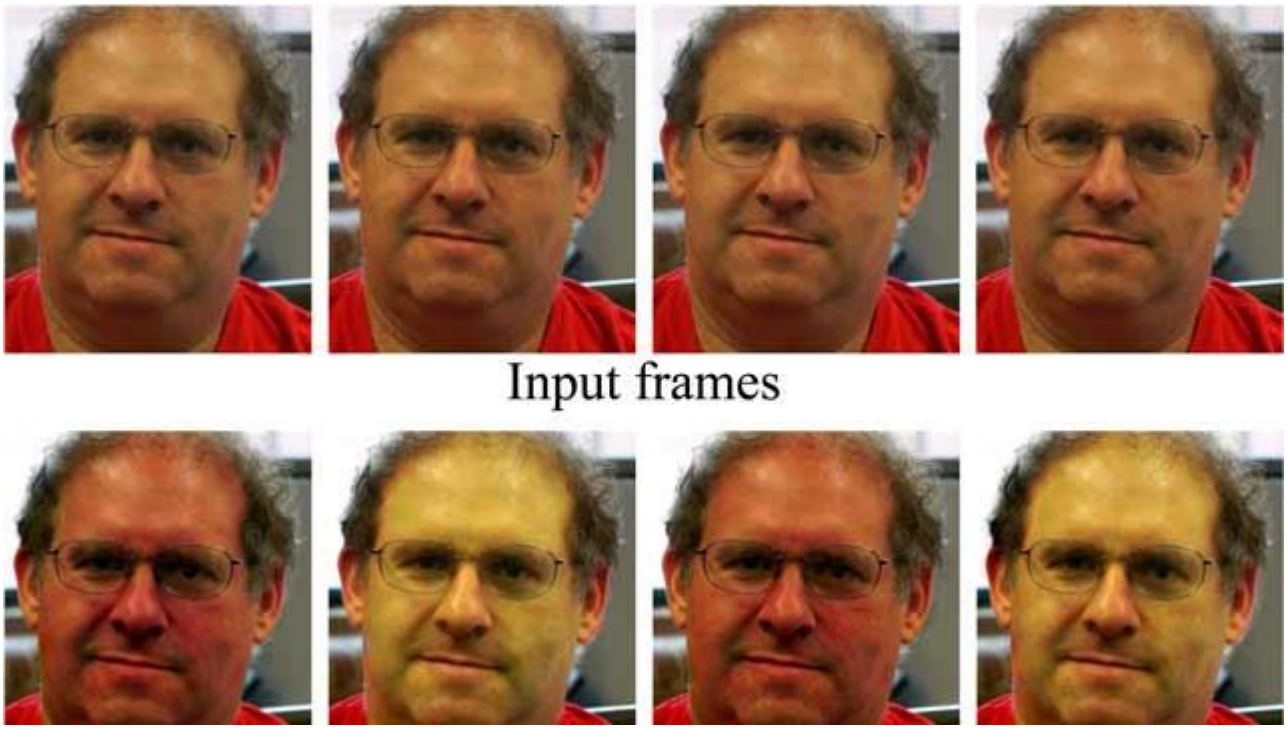

Figura 1.4. Rezultatul amplificării in video propusă in [6], evidențiind pulsul din culoarea pielii capului

În Figura 1.4 putem observa rezultatele algoritmului prezentat în [6]. Acesta este capabil de a evidenția și amplifica diferența dintre culoarea pielii pentru fiecare cadru.

Această lucrare propună o implementare realizată personal a algoritmului, în Python 3 și o comparație cu rezultatele prezentate în [6]. De asemenea, se vor evidenția și alte posibile utilizări ale algoritmului, în sepacial pentru evidențierea mișcărilor ce nu pot fi văzute cu ochiul liber , care pot aduce indicii spre comportamentul disimulat (de exemplu, o mâna care tremură datorită anxietății/fricii/adrenalinei etc).

De asemenea, în capitolele ce vor urma se va evidenția contribuția acestui algoritm asupra algoritmului de măsurare a pulsului bazat doar pe FPG, realizat de asemenea personal.

1.5. Semne ale comportamentului disimulat;Detecție si interpretare

Un comportament disimulat de definește ca un comportament care are ca scop ascunderea gândurilor, sentimentelor în favoarea unor gesturi false. Lucrarea are că scopul determinarea semnelor acestui comportament din elemenentele descrise mai sus, cu scopul de a avea informații suficiente pentru a determina obiectiv dacă un comportament este sau nu disimulat.

Deoarece comportamentul disimulat nu poate fi exprimat ca o funcție de un element obiectiv pentru toți oamenii, decizia binară între disimulare și onestitate este una foarte grea, chiar și pentru experți umani.

În prezent, cea mai bună metodă pentru detectarea disimulării este Poligraful sau așa-numitul “detector de minciuni”. Poligraful este un aparat care înregistrează indicatori psihologici precum presiunea sângelui, puls, respirație și conductivitatea pielii cu scopul de a detecta variații neobișnuite ale acestora în prezenta comportamentului deceptiv. Deși se spune că dispozitivul are o acuratețe de 90%, Consiliul Național al Cercetării din SUA nu a publicat dovezi cu privire la corectitudinea rezultatelor[7]. Aparatul nu este acceptat în momentul de față în instanțele de judecată și este adesea criticat deoarece se presupune că simplă conectare a unui subiect uman la acesta provoacă anxietate sau frică, aceste sentimente putând afecta rezultatele măsurătorilor.

Această lucrare propune utilizarea ritmului cardiac că și componentă obiectivă, pentru a realiza un detector binar de comportament disimulat, folosind rețele neuronale și învățare adâncă (deep learning). Deoarece comportarea disimulată și ascunderea acțiunilor sunt termeni foarte vagi și interpretabili, am realizat o bază de date ce încearcă punerea în evidență a reacției umane la stimuli vizuali. Baza de date va fi prezentată în următoarele capitole și va fi folosită pentru antrenarea unui algoritm de învățare automată.

Premiza experimentului este că subiecții umani își pot schimba ritmul cardiac sau ritmul respirator atunci când sunt în contact cu o informație șocantă, de interes, anxietate sau alte emoții. Baza

de date este unică și creată personal, putând fi subiect a multor erori de gândire sau măsurare. Rezultatele obținute prin antrenarea algoritmului binar vor fi prezentate în Capitolul 4.

2. Concepte teoretice aplicate

In acest capitol se vor prezenta detaliat conceptele folosite mai departe in realizarea parții practice, experimentelor, cat si conceptele ce stau la baza sistemelor asemănătoare existente in literatură. De asemenea, se vor introduce informații utile despre resursele software si hardware folosite.

2.1. Metode pentru determinarea non-contact a ritmului cardiac

In subcapitolele ce urmează se vor prezenta principalele metode de determinare a ritmului cardiac non-contact folosite in literatură:

- prin intermediul fotopletismogramiei si transformatei Fourier rapida (FFT)
- analiza componentelor independente (ACI)
- folosind Amplificarea Euleriană a video-ului (EVM)

2.1.1. Determinarea non-contact a ritmului cardiac prin intermediul fotopletismogramiei

Fotopletismogramia se bazează pe detecția variației volumului de sânge din anumite zone ale pielii, numite regiuni de interes cu scopul de a măsură schimbările nivelului de absorbție al luminii [2]. Algoritmul se bazează pe analiză în frecvența a semnalului temporal determinat de culoarea regiunii de interes aleasă.

Majoritatea algoritmilor non-contact se bazează pe alegerea unei regiuni de interes de pe suprafață feței, deoarece astfel este garantată o arie destul de mare de piele pentru o măsurare de acuratețe cât mai bună.

În lucrările recente, această metodă este îmbinată cu metodă ce se bazează pe EVM [10]. Totuși, numeroase lucrări au folosit și această metodă individual [11][12], deoarece precizia este suficientă în condiții bune [11][13][14]. Această metodă rămâne încă extrem de importantă datorită evoluției în domeniul amplificării culorilor din imagine [10], dar și datorită nivelului scăzut de resurse

folosite. Astfel, numeroase aplicații pentru telefoane inteligente au fost create folosind acest algoritm, regiunea de interes fiind atât regiunea feței cât și vârful degetului [15].

Determinarea regiunii de interes

Așa cum este menționat mai sus, aplicațiile pe terminalul mobil care folosesc doar metodă FPG tind să folosească regiunea din vârful degetului,plasat fix pe cameră[15], deoarece raportul semnal/zgomot este mult mai mic decât în cazul în care față ar fi filmată. Există numeroase aplicații atât Android cât și IOS care pot fi descărcate gratuit și care realizează funcția de citire a pulsului cu erori mici.

În cazul aplicațiilor ce folosesc sisteme mai performanțe[13][14][16], zone din regiunea feței sunt folosite deoarece față este ușor de detectat folosind algoritmi deja antrenați de învățare automată și se pot extrage ușor regiuni de interes conținând zone destul de mari de piele. De asemenea, algoritmii se pot folosi de vasele de sânge ce se află în diferite regiuni ale feței și care își schimbă presiunea, deci absorb lumina diferit, odată cu bătăile inimii.

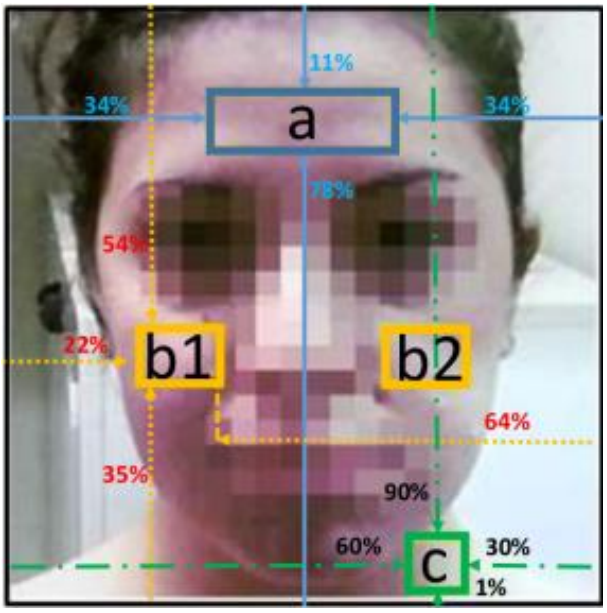

Figura 2.1. Regiunile de interes prezentate in [16]

În cazul lucrării prezentate în [16], multiple regiuni de interes au fost alese (Figura 2.1), acestea fiind așezate în diverse proporții față de pătratul ce definește regiunea facială. Au fost alese mai multe

regiuni pentru a avea redundanță , fiecare regiune producând un semnal de ritm cardiac diferit. Fiind proporționale cu dimensiunea feței, această alegere a zonelor implică faptul ca mișcările feței nu ar trebui să afecteze poziția relativă la față a zonelor de interes.

Un dezavantaj al alegerii a multe puncte de interes este faptul că mișcările feței pot introduce erori în unele puncte, având astfel valori diferite de puls pentru fiecare regiune de interes.

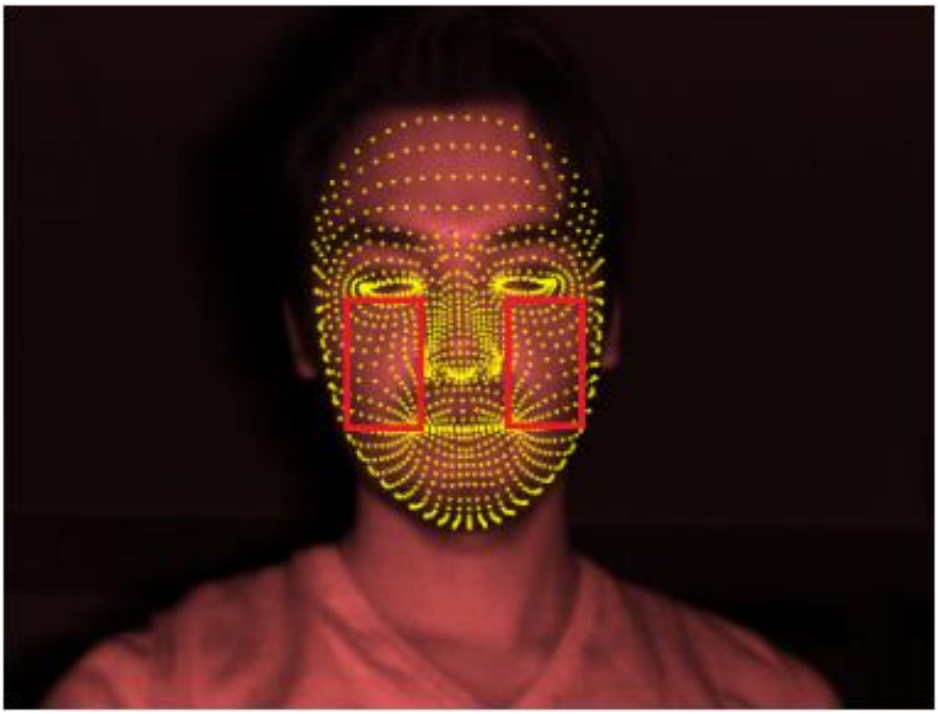

Figura 2.2. Alegerea regiunilor de interes in cazul [13]

În [13] este prezentată alegerea regiunii de interes folosită pentru măsurarea ritmului cardiac. În cazul acestei lucrări, un Microsoft Kinect v2 a fost folosit pentru realizarea filmărilor. Senzorul are capabilitate de redare de imagine 3D, această fiind folosită pentru a mapa multiple puncte staționare care reprezintă regiuni ale feței. Aceste puncte sunt realizate cu ajutorul Microsoft Kinect SDK 2.0 și sunt folosite pentru a delimita anumite regiuni ale feței. Două dreptunghiuri sunt trasate folosind punctele de la extremitățile regiunii gurii și punctele extremităților de jos ale regiunii ochilor.

Această metodă produce niște regiuni de interes foarte mari, dar are dezavantajul că aceste regiuni pot fi afectate mult mai mult de zgomotul ce provine din umbre. Mai mult decât atât, zonele de interes sunt sensibile la rotațiile capului și la inacuratețea senzorului Kinect. Se poate constata că pixelii care fac parte din regiunea de interes dar nu fac parte din regiunea feței adaugă un zgomot în plus măsurătorilor.

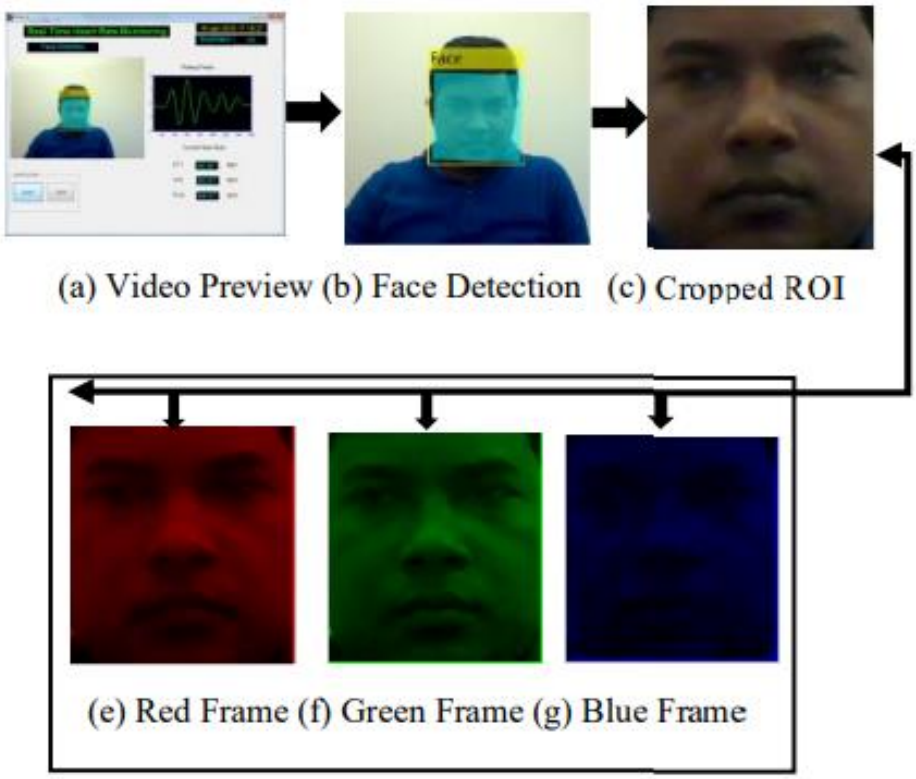

Figura 2.3. Regiunea de interes aleasă in cazul [14]

In cazul [14], regiunea de interes aleasă reprezintă chiar întreaga regiune a feței. Deoarece sub pielea de pe față exista o multitudine de vene , aceasta regiune de interes este una validă si poate aduce rezultate bune. In cazul [14], rezultatele experimentale au arătat existența coeficient de corelație de peste 0.9 cu valorile reale ale ritmului cardiac.

Detecția feței

Pentru detecția regiunii feței, majoritatea lucrărilor din literatură au folosit algoritmul de detecție al feței propus de Viola și Jones[17]. În funcție de parametrii primiți, algoritmul este capabil să funcționeze atât la viteze mici și precizie foarte bună, cât și la viteze mari, dar precizie mai mică. Algoritmul Viola-Jones folosind cascade Haar va fi prezentat în detaliu mai târziu în acest capitol.

Calcularea semnalului în timp

Odată ce regiunile de interes sunt obținute, calcularea semnalului în timp este realizată prin obținerea mediei valorilor pixelilor din acele zone.

Deși unele lucrări precum [14] au prezentat folosirea tuturor canalelor de culoare pentru obținerea valorii în timp a semnalului, majoritatea experimentelor au folosit un singur canal. Canalele de culoare roșie sau verde sunt preferate de către cercetători, dar informația dată de acestea este corelată așa că nu se poate găsi un argument definitiv pentru alegerea unui canal.

De asemenea, lucrările ce prezintă folosirea unui Microsoft Kinect arată și avantajele și dezavantajele folosirii unui semnal de frecvența din domeniul Infraroșu(IR).

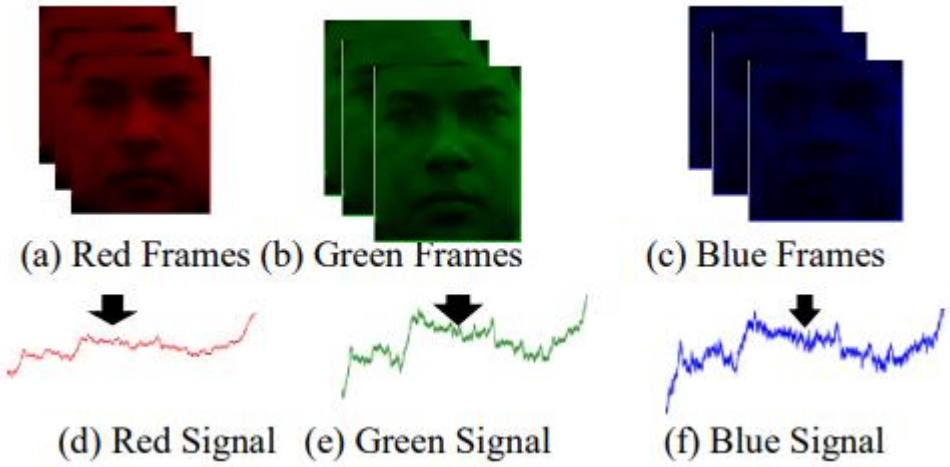

Figura 2.4 Semnalele in timp regiunilor de interes de culoare roșie, verde si albastră

Figura 2.4 prezintă exemple de valori ale evoluției in timp ale semnalelor de culoare roșie,verde si albastră in [14]. Se poate observa o corelație intre cele 3 canale reprezentând cele 3 culori.

Aceeași corelație se poate observa si in Figura 2.5:

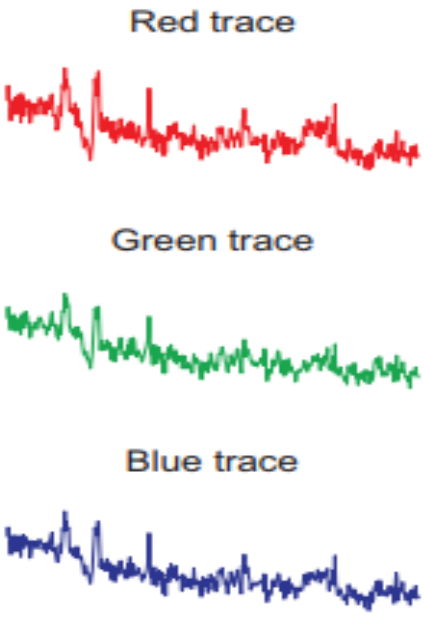

Figura 2.5. Semnale in timp reprezentând canalele
Rosu, Verde si Albastru exemplificate in [18]

Presupunând o distribuție Gaussiană a zgomotului, cu medie 0, medierea valorilor din interiorul regiunilor de interes duce la eliminarea zgomotului, eliminarea având mai mult succes cu cât numărul de pixeli este mai mare.

În majoritatea limbajelor de programare, semnalele canalelor de culoare sunt salvate pe un număr de biți (de obicei 8) și citite că numere întregi, de exemplu între 0 și 255 în cazul reprezentării pe 8 biți. Exemple pentru reprezentarea mediei pixelilor din regiunea de interes în cazul culorii verde se pot regăsi în Figura 2.6 si Figura 2.7. Exemplele sunt măsurate în [13] pentru subiecți umani reali, cu pulsul de 60bpm, respectiv 51 bpm.

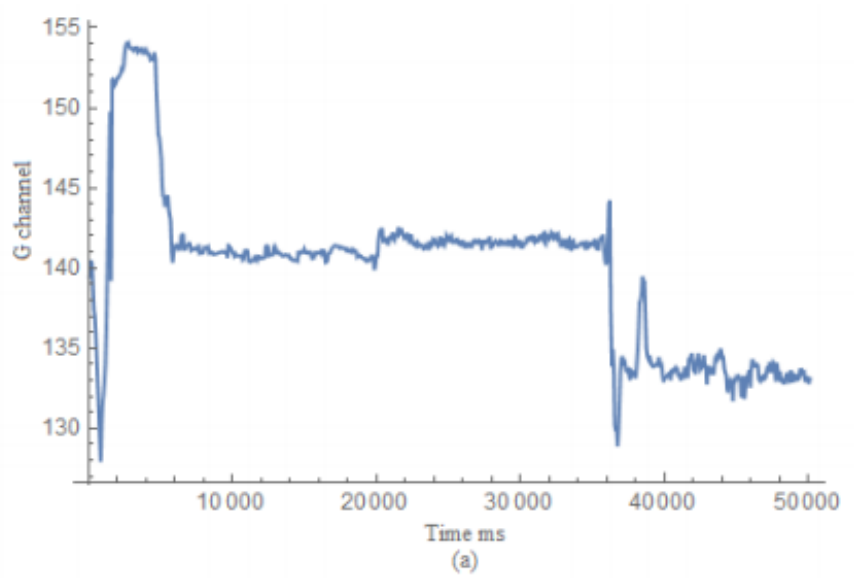

Figura 2.6. Evoluția in timp a canalului verde; Pe axa x este media valorilor de verde ale valorilor din regiunea de interes, pe axa y este timpul, pentru o persoană cu puls de 60 bpm

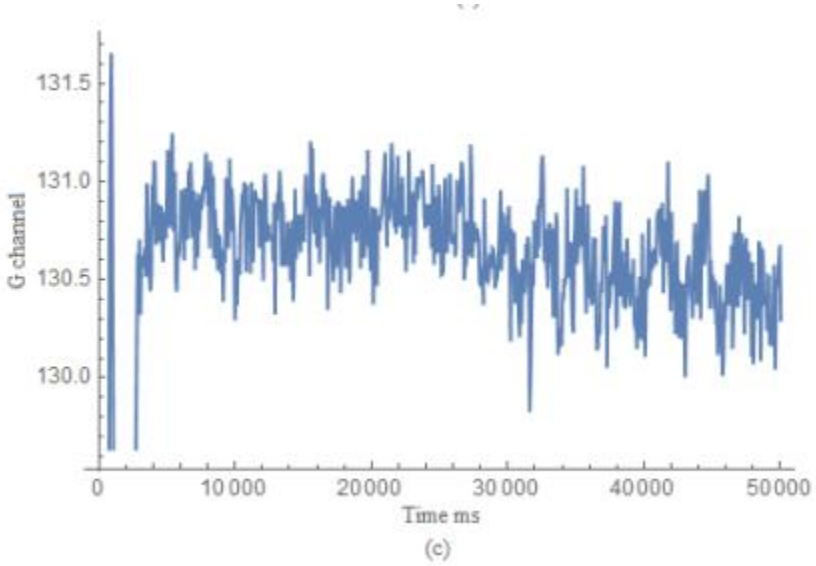

Figura 2.7. Evoluția in timp a canalului verde; Pe axa x este media valorilor de verde ale valorilor din regiunea de interes, pe axa y este timpul, pentru o persoană cu puls de 51 bpm

Pixelii a căror medie temporală este reprezentată în Figura 2.6 și Figura 2.7 au valori întregi, între 0 și 255. Se poate observa că semnalul ce reprezintă media pixelilor verzi din regiunile de interes este foarte zgomotos, fiind imposibilă distingerea unui semnal periodic din imagine.

De asemenea, se poate observa că deviația standard a mediei pixelilor în Figura 2.7 este mult mai mică decât cea din Figura 2.6. Unul dintre posibilele motive este poziția mai fixă a subiectului

uman în cadrul măsurătorilor descrise în Figura 2.7, spre deosebire de ușoarele schimbări de poziție, deci posibil și de nivel de lumină în cazul măsurătorilor ce se pot distinge în Figura 2.6.

Deci, deși semnalul este variabil în zona frecventelor mai mari că 0, acesta are o componentă continuă care poate fi fixă sau variabilă, depinzând de mișcările feței subiectului uman.

Canalul de culoare ales

Autorii lucrării [20] au demonstrat faptul că canalul cu cel mai puternic semnal al inimii, prin comparație, este cel verde. Acest lucru se întâmplă deoarece semnalul este determinat de variațiile în volumul sângelui, deoarece hemoglobina absoarbe lumina din spectrul verde mult mai bine decât lumina din spectrele de albastru și roșu.

Totuși, autorii diverselor lucrări au ales să folosească diferite abordări, de la a folosi toate canalele până la a folosi doar canalul roșu sau cel verde. Este posibilă și folosirea de combinații folosind aceste 3 canale, de exemplu imaginea alb-negru/gri. De asemenea, există autori ce au folosit lumina din spectrul IR pentru a măsura ritmul cardiac.

Metode de filtrare folosite in literatură

Așa cum a fost descris mai sus, semnalele ce rezultă din media culorii regiunii de interes sunt extrem de zgomotoase. În plus, se cunoaște faptul că majoritatea oamenilor vor avea frecvența bătăilor inimii într-un anumit interval. De aceea, informațiile date de celelalte frecvente devin inutile pentru această aplicație.

Filtrele digitale folosite în aplicațiile din literatură diferă în funcție de descrierile și cerințele aplicațiilor. Spre exemplu, în [19], se folosește un filtru FIR trece-bandă de ordin M mare =40 a fost folosit pentru a extrage frecventele începând de la 0.2Hz până la 2Hz, ce includ atât frecventele posibile pentru domeniul de rată respiratorie (aprox. 0.2-0.5 Hz), cât și domeniul frecvențelor bătăilor

inimii (aprox. 0.6-2 Hz). Apoi, un nou filtru trece-bandă Butterworth a fost folosit pentru a extrage doar semnalul corespunzător bătăilor inimii (în cazul [19], s-au ales frecvențe în intervalul 0.6-1.8 Hz).

Banda de trecere în cazul obținerii semnalului de puls a fost în general în intervalul 0.6-2 Hz, corespunzând unor valori ale ritmului cardiac de 36-120 bpm, interval ce cuprinde toate valorile tipice pentru adulți în toate formele fizice.

Autorii care au decis să folosească și semnale de adâncime sau infraroșu date de senzorul Kinect au ales o bandă de frecvente începând de la 2 Hz, pentru a putea include și semnalul reprezentând numărul de respirații pe minut.

În figurile de mai jos vor fi prezentate rezultatele filtrărilor publicate de câțiva dintre autorii menționați:

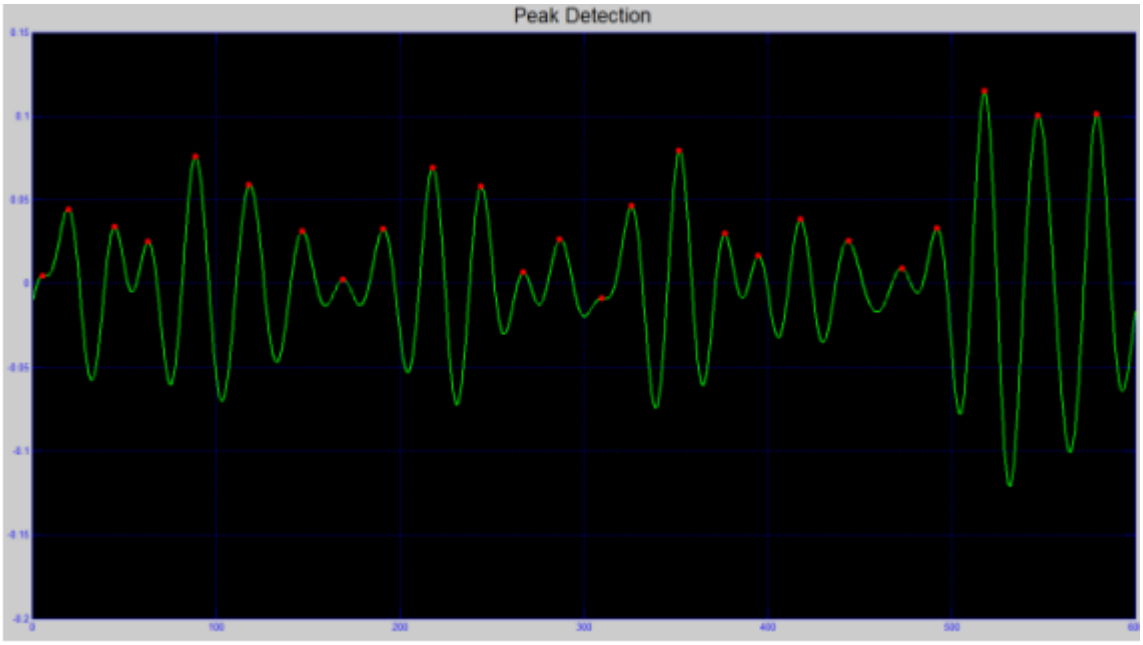

Figura 2.8. Semnalul ce reprezintă ritmul cardiac, după filtrare[14]

Figura 2.8 reprezintă un exemplu de semnal în timp obținut de autorii [14]. Pe axa x se regăsește unitatea de timp, măsurată în cadre, iar pe axa y este amplitudinea semnalului. Vârfurile semnalului au fost detectate, putându-se astfel măsură frecvența bătăilor inimii în timp, că număr de vârfuri/număr de cadre.

Se poate observa o diferența între vârfuri cu amplitudine mai mare și vârfuri cu amplitudine mică. Apariția acestei discrepanțe de amplitudine este rezultatul influenței respirației asupra bătăilor

inimii. Deci, dacă am pune pe un grafic amplitudinea vârfurilor versus timp, am obține semnalul ce reprezintă ritmul respirator.

Transformata Fourier;Obținerea frecvenței semnalului filtrat

Tranformata Fourier reprezintă o funcție care rescrie un semnal în timp că o sumă de sinusoide. Funcțiile în domeniul timp au în echivalent în domeniul frecvența. Transformată Fourier este folosită pentru a trece din reprezentarea unui semnal în domeniul timp în reprezentarea semnalului în domeniul frecvență. În cazul prelucrării de semnale digitale, autorii din literatură lucrează cu semnale discretizate, reprezentate cu ajutorul biților. În cazul evoluției în timp a unui semnal discret, Transformată Fourier Discretă(TFD) este folosită pentru a obține semnalul respectiv reprezentat în frecvență.

Transformata Fourier Rapidă(FFT) este un algoritm ce reduce complexitatea unei TFD de la $O(n^2)$ la $O(n \log n)$, unde n este lungimea semnalului. Există o multitudine de algoritmi FFT, mulți dintre ei fiind implementați în librăriile limbajelor de programare.

Având semnalul în timp ce reprezintă media filtrată a canalului sau canalelor de culoare din regiunea de interes, este esențială obținerea frecvenței principale a acelui semnal, această fiind frecvența ritmului cardiac.

Prin aplicarea FFT asupra semnalului în timp, se va obține o reprezentare în frecvența a acestuia. Pentru a defini frecvența pulsului, se măsoară frecvența la care valoarea FFT este cea mai mare, aceea fiind frecvența ritmului cardiac. Fiecare dintre autorii lucrărilor de mai sus au aplicat această metodă, urmând că în figurile de mai jos să fie descrise rezultatele.

În [16] este prezentată trecerea din domeniul timp în domeniul frecvență a unui exemplu de semnal filtrat, rezultatele fiind prezentate în Figura 2.9 și Figura 2.10.

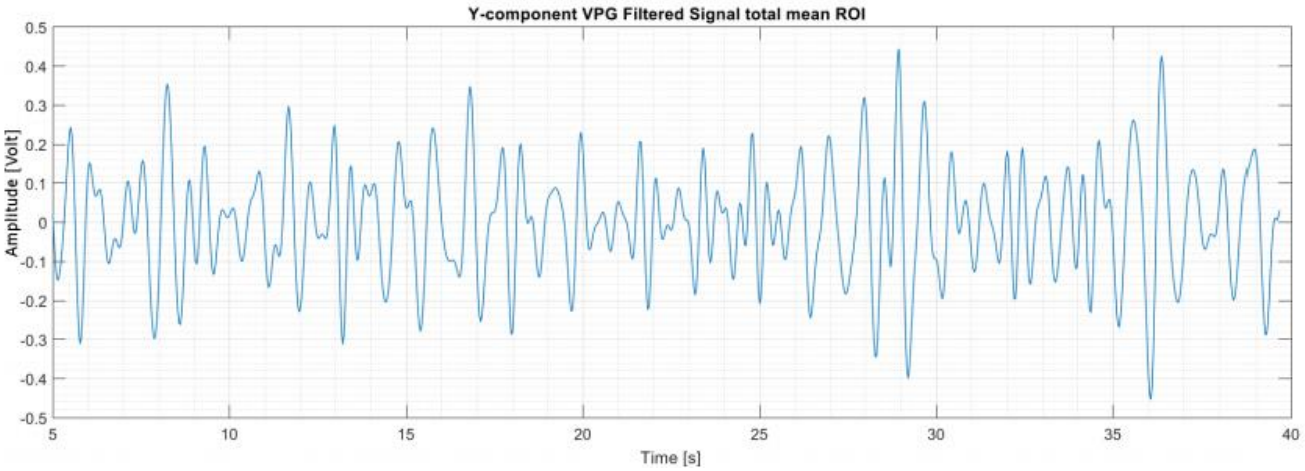

Figura 2.9. Un exemplu de semnal in timp reprezentând media intensității culorii in regiunea de interes in [16]

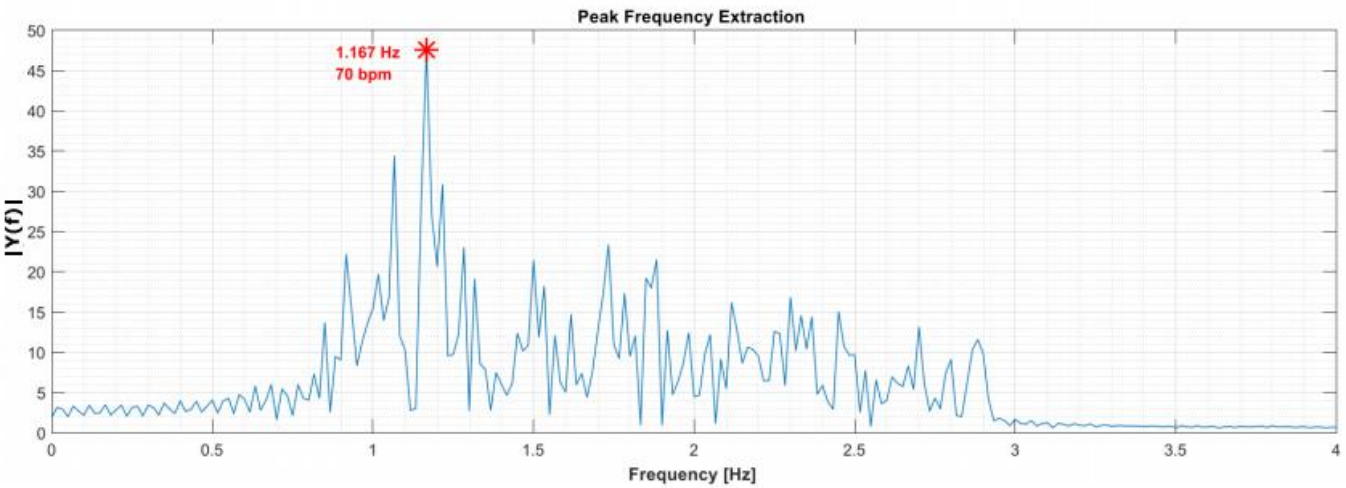

Figura 2.10. Valoarea absolută a reprezentării semnalului fin Figura 13 in frecvență, cu ajutorul FFT [16]

Figura 13 reprezintă un exemplu de evoluție în timp a semnalului extras din regiunea de interes. Pe axa x putem observa valoarea timpului, în secunde , iar pe axa y putem observa valoarea amplitudinii semnalului.

Figura 14 este reprezentarea în frecvența a semnalului din Figura 13. Pe axa x este frecvența în Hz, doar în domeniul de interes unde filtrul IIR trece-bandă a permis trecerea frecvențelor din bandă de trecere.

În final, se poate observa că vârful de cea mai mare amplitudine în reprezentarea în frecvența este la frecvența de 1,167 Hz, reprezentând 1,176 * 60 = aproximativ 70bpm.

2.1.2. Analiza componentelor independente (ACI)

Analiza componentelor independente este o metodă folosită cu preponderență în procesarea semnalelor și în învățarea automată ce are ca scop separarea semnalelor dependențe de multiple variabile în sub-componente aditive. Metodă funcționează doar cu presupunerea că semnalele nu sunt Gaussiene și sunt independente statistic unul de altul [21].

În cazul concret al lucrărilor publicate ce folosesc această metodă, semnalul sursă de interes este pulsul ce se propagă de la inimă prin tot corpul, care va modifica volumul de sânge din venele din zona feței, astfel încât schimbările în lumina reflectată de către zona de interes indică schimbările în timp ale evenimentelor de tip cardiovascular. Înregistrând pe cele 3 canale de culoare, roșu, verde și albastru, senzorii de culoare vor primi la intrare un semnal compus din componentă pletismografică și din alte semnale zgomotoase și fluctuații ale luminii. Fiecare dintre cele 3 canale de culoare ale senzorului va recepționa o parte din semnalul util și o parte din zgomotul luminos.

Semnalele observate din canalele roșu, verde și albastru sunt notate cu $X_1(t)$, $X_2(t)$, $X_3(t)$, fiecare reprezentând evoluția amplitudinii în timp a semnalului canalului respectiv.

În analiză componentelor independente, teoria spune că numărul de surse ce pot fi recuperate din semnal nu poate depăși numărul semnalelor. Notăm $s_1(t)$, $s_2(t)$, $s_3(t)$ cele 3 semnale sursa ce trebuie aflate. Legătura între semnalele de intrare și semnalele sursă este dată de ecuația: $X_i(t) = \sum_{j=1}^{3} a_{ij} * s_j(t)$, presupunându-se că există o relație liniară între semnalele observate și semnalele sursă.

Ecuația pentru cele 3 semnale se rezolva în mod matriceal: $x(t) = A * s(t)$, unde

A este o matrice 3x3 ce conține coeficienții a_{ij}.

Sistemul se va rezolva iterativ, încercând minimizarea unei funcții cost care măsoară cât de mult semnalele sursă sunt non-Gaussiene [18][22].

2.1.3. Amplificarea Eureliană Video (EVM)

În anul 2012, o echipa de cercetători de la universitatea MIT din Statele Unite ale Americii au publicat lucrarea intitulată “Eulerian Video Magnification for Revealing Subtle Changes in the World” [10], ce prezintă o metodă de a dezvălui variații temporale în video-uri, care deși sunt imposibil de văzut cu ochiul liber, pot fi amplificate astfel încât să poată fi percepute de persoana de rând. Semnalul rezultat poate de multe ori să redea informație ascunsă în secvența video.

Unul dintre cele mai faimoase experimente realizate cu această metodă este prezentat în Figura 2.11:

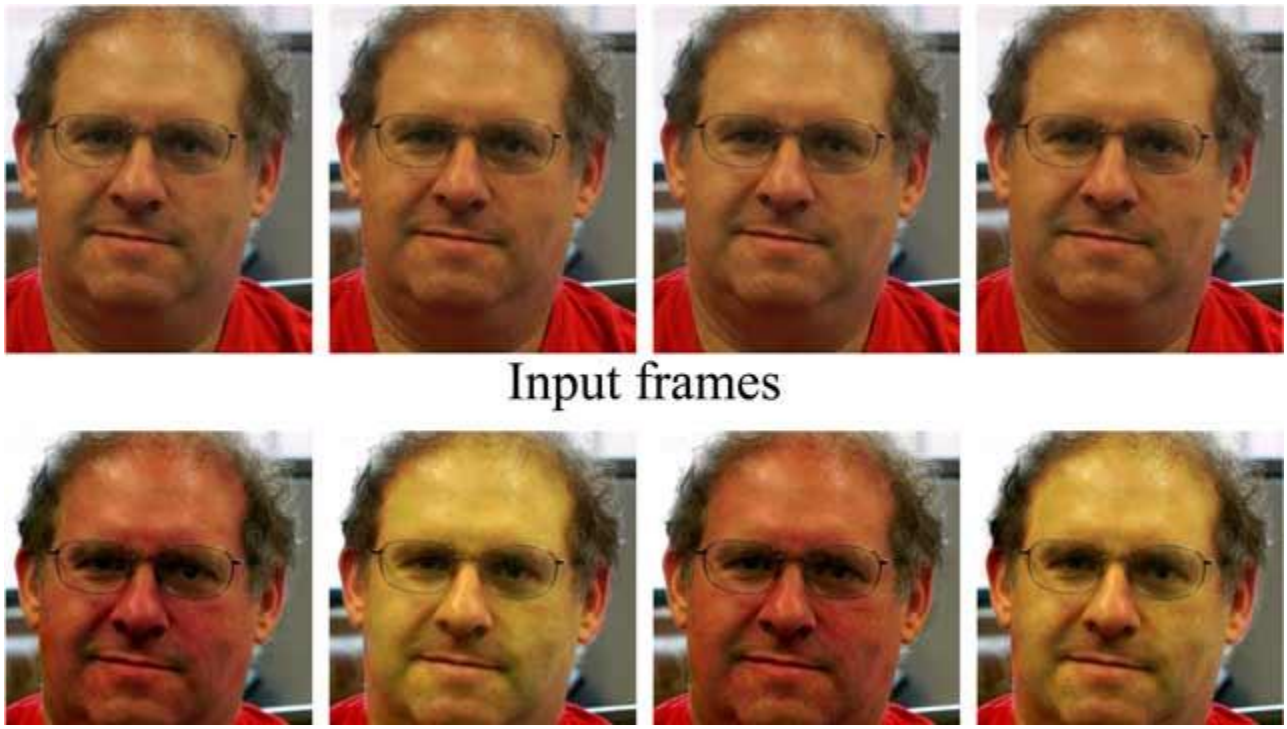

Figura 2.11. Rezultatele Amplificării Euleriene a unui video, arătând modificarea culorii pielii ca rezultat al bătăilor inimii

Figura 15 prezintă cel mai faimos experiment realizat în [10]. În partea de sus sunt 4 cadre video ce, pentru omul de rând, nu reprezintă niciun indiciu asupra ritmului cardiac al persoanei din imagine. În schimb, cadrele prezentate în partea de jos a figurii prezintă amplificarea diferenței de culoare adusă de trecerea sângelui cu presiune prin venele de pe suprafață feței. Diferențele sunt notabile, algoritmul nu dezvăluind numai schimbări ce se pot observa cu ochiul liber, dar și schimbări ce pot fi măsurate mult mai exact cu algoritmul prezentat în capitolul 2.1.1.

Algoritmul și-a găsit numeroase aplicații în multe domenii, dar în special în medicină. Una dintre cele mai importante aplicații ale algoritmului este verificarea asimetriei circulației sângelui în părți ale corpului, aceasta fiind un semn al problemelor arteriale. În scopul acestei lucrări, ritmul cardiac detectat cu ajutorul algoritmului și metodei FPG va fi folosit pentru identificarea de iregularități în pulsul uman, acestea fiind o dovadă a unor gânduri/emoții ascunse, ce nu sunt lăsate la vedere.

În afară de amplificarea culorii, autorii [10] au folosit algoritmul pentru a amplifica mișcările de amplitudine mică ce sunt greu sau imposibil de văzut cu ochiul uman. Pentru acest topic au existat și alte încercări de realizare de algoritmi, cercetătorii din [23] încercând abordarea Lagrangiană, concept asemănător cu urmărirea și localizarea particulelor în dinamică fluidelor. Metoda s-a dovedit a fi o abordare bună, dar de o complexitate mult prea mare pentru a putea fi calculată. În schimb, abordarea Euleriană este inspirată din proprietățile unui voxel (analogie cu pixel, dar îl loc de unitatea de imagine/culoare, fiind unitate de volum), presiunea și viteza au evoluția măsurată constant în timp. Astfel, exagerarea mișcării este realizată amplificând schimbările temporale de culoare ale fiecărui pixel la un moment de timp.

Figura 2.12 prezintă abordarea pentru amplificarea Euleriană folosita in [10]:

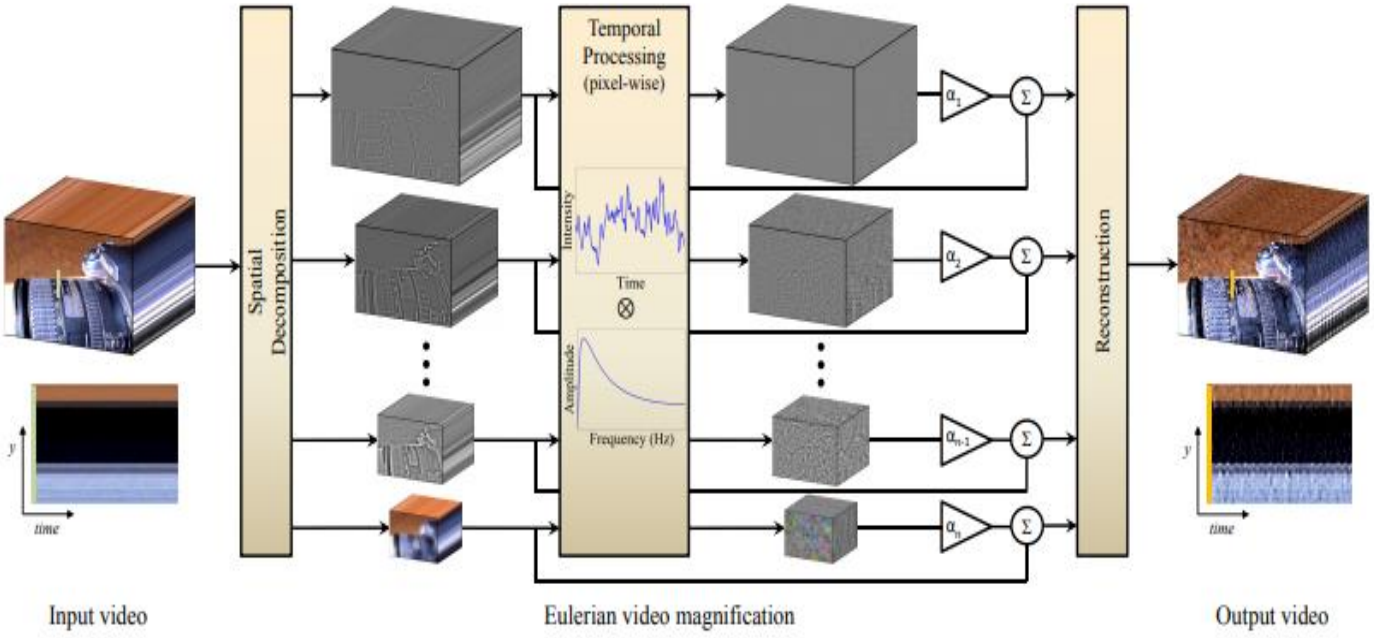

Figura 2.12. Abordarea din [10] pentru amplificarea Euleriană in video

Metoda prezentata in Figura [16] combina tehnici de procesare spațială si temporală pentru a obține rezultatele dorite. In prima faza, imaginile din video sunt descompuse in diferite benzi de frecventa spațială. In Figura 16 este prezentata folosire unei piramide Laplaciana. Un al doilea pas este

filtrarea, pixel cu pixel, a imaginilor. Apoi, imaginile sunt amplificate si adunate la imaginea inițială, construindu-se imaginea amplificata.

Piramida Gausiană

Piramida Gausiană este obținută prin convoluția fiecărei imagini cu o matrice nucleu Gausiană. Pentru a crea aceasta matrice, denumita si kernel, se folosește formula:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

, unde G(x,y) este elementul de pe linia x si coloana y, iar sigma este deviația standard a distribuției Gaussiene si este o variabilă de care depind valorile matricii.

Kernel-ul este practic un filtru care este folosit pentru încețoșarea imaginii. Un exemplu de încețoșare a imaginii cu un filtru Gausian se poate observa in Figura 2.13:

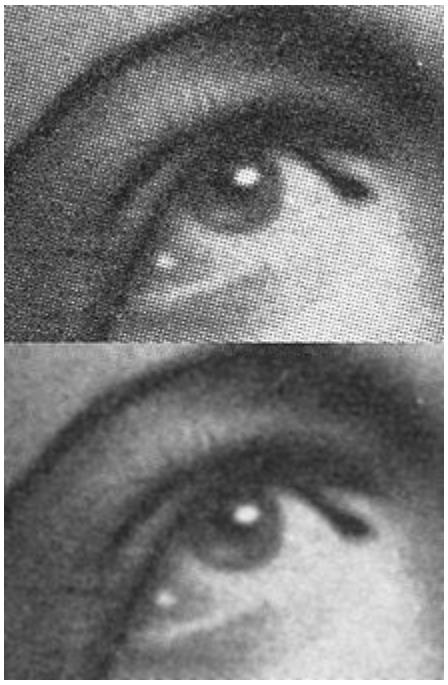

Figura 2.13. Încețoșarea imaginii cu ajutorul unui filtru Gausian

, unde imaginea de jos este cea originală și imaginea de sus este cea încețoșată.

Filtrul Gausian se aplică fiecărui pixel, iar valoarea fiecărui pixel este o medie ponderată a valorilor pixelilor vecini.

Imaginea căreia i s-a aplicat convoluția cu filtrul Gausian este apoi subeșantionata cu un factor de 2. Procedeul se repetă de mai multe ori, în funcție de numărul de nivele din algoritm.

Pentru a obține o piramidă Laplaciană din cea Gausiană, înainte de subeșantionare, din nivelul mai mic al piramidei se scade nivelul nou obținut. Piramida Laplaciană este folosită cel mai des în compresia imaginilor și în detecția marginilor.

Filtrarea pixel cu pixel

Partea a două a algoritmului prezentai în Figura 16 este filtrarea pixel cu pixel al vectorului temporal de imagini, pentru fiecare nivel al piramidei Gaussiene. Această se realizează prim mai multe metode de către cercetătorii din [10]:

- Filtrarea ideală în frecvența, obținută prin realizarea transformatei Fourier Rapide, inițializarea cu 0 a valorilor componentelor frecventelor nedorite și realizarea transformatei Fourier inverse pentru a obține semnalul filtrat

- Realizând un filtru IIR trece bandă în timp, care va fi diferența dintre 2 filtre IIR trece jos cu frecvente de tăiere diferite. Filtrele sunt create după formulă :

$$L_n = L_{n-1} * (1-\omega) + \omega * M$$

, unde M este cadrul curent, L este acumulatorul în care valoarea filtrului trece jos este înregistrată iar ω este procentajul din frecvența de eșantionare care este frecvența de tăiere a filtrului (între 0 și 1).

- Realizând un filtru IIR în timp la fel că și cel prezentat mai sus, dar aplicându-l pe elementele unei piramide Laplaciene.

Rezultatele semnificative ale amplificării Euleriene

După ce fiecare nivel din piramidă Gausiană este obținut și filtrat, nivelele se vor scala la dimensiunea imaginii inițiale prin interpolarea cu zerouri și convoluție pentru netezire. Imaginile obținute din fiecare nivel vor fi amplificate cu un factor de amplificare alpha ales de utilizator în funcție de aplicație și vor fi adunate imaginii inițiale.

Figurile 2.14 și 2.15 vor prezenta alte aplicații celebre pentru amplificare Eulerianǎ Video, respectiv frecventele la care evenimentele acestea se produc și pot fi amplificate.

Figura 2.14. Exemple din [10] pentru folosirea algoritmului de amplificare a mișcării/culorii

Figura 18 prezintă aplicații celebre, ce se pot regăsi în video-ul de la [24]. Exemple de amplificarea mișcărilor din video se pot regăsi în imaginile cu etichetele “baby”, “guitar”, “subway” sau “shadow”. Așa cum este prezentat în Figura 19, diferite mișcări au frecvente fundamentale distincte, astfel că frecvența filtrului IIR prezentat mai sus trebuie schimbată că atare. Spre exemplu, frecvența de vibrație a unei coarde de chitară ar putea să fie în intervalul 175-225 Hz, definită în imaginea (b) din Figura 19. Frecvența cu care un metrou se mișcă pe sine ar putea să fie în intervalul prezentat în graficul (c) din Figura 19. În schimb, pentru detectarea pulsului trebuie setata o frecvența

din banda de trecere mult mai îngustă si joasă, de ordinul 0.6-2 Hz sau chiar mai îngust, depinzând de aplicație.

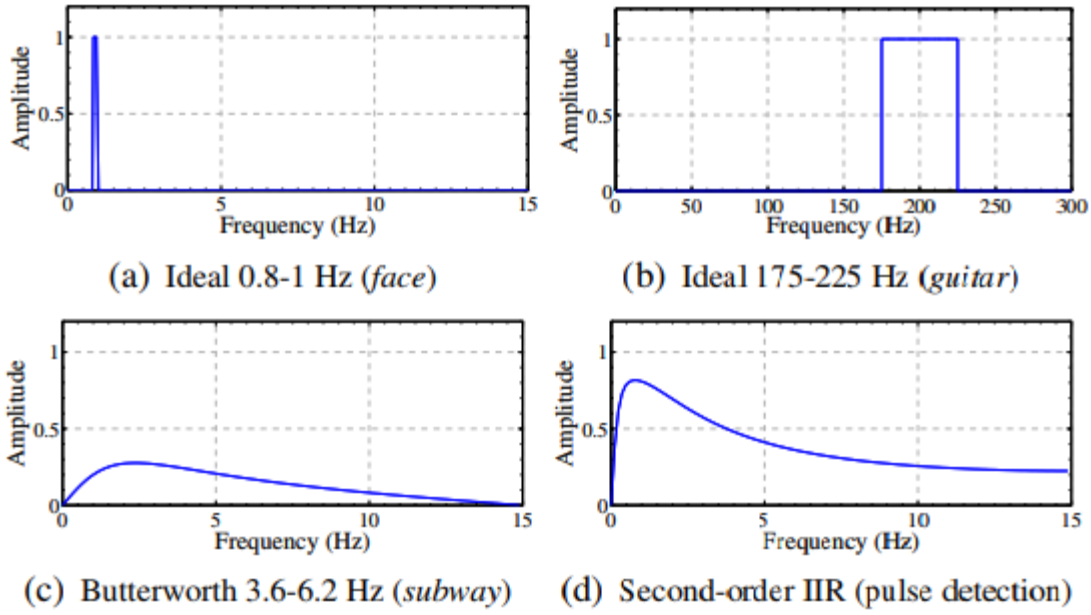

Figura 2.15. Frecvențele mișcărilor ce trebuie observate, pentru cazurile din Figura 2.14

Combinația dintre metoda prezentată in capitolul 2.1.1 care implică FPG si metoda curentă rezultă intr-o citire a pulsului cu acuratețe mărita, deoarece raportul semnal-zgomot este mărit de către amplificarea Eulerianã.

2.2. Metode pentru determinarea non-contact a ritmului respirator

În acest subcapitol vor fi prezentate două abordări pentru determinarea ritmului respirator:

- Utilizarea proprietăților aritmiei respiratorie sinusală pentru a determina ritmul respirator cu ajutorul pulsului detectat cu ajutorul principiului fotopletismogramiei

- Utilizarea senzorului de adâncime Microsoft Kinect pentru a determina ritmul respirator folosind informația de adâncime din imagine dată de senzorul de adâncime.

2.2.1. Utilizarea proprietăților ARS pentru determinarea ritmului respirator

Aritmia respiratorie sinusală[30] de definește că modificarea ritmului cardiac proporțional cu volumul de aer inspirat sau expirat în acel moment. Ritmul cardiac crește în timpul inspirației și descrește cu expirația.

Dispozitivele ce măsoară ritmul cardiac pot decide să elimine această variație, compensând pulsul invers proporțional cu ritmul respirator. De asemenea, lucrări publicate precum[31] prezintă un algoritm de determinare a ritmului respirator și ritmului cardiac și compensează pulsul în funcție de ritmul respirației. În algoritmul prezentat de [31] se folosește canalul infraroșu al unui senzor Microsoft Kinect pentru a observa temperatura la suprafață pielii. Temperatura pielii variază în mod constant datorită volumului de sânge ce trece prin această, astfel putând detecta ritmul cardiac din micile variații de temperatura.

Pentru detectarea ritmului respirator, senzorul de adâncime al Microsoft Kinect este folosit pentru a extrage mișcările cutiei toracice, astfel obținând un semnal semiperiodic ce este e echivalent cu ritmul respirator. Folosind minimele și maximele din semnalul reprezentând ritmul respirator, se poate compensa variația ritmului cardiac.

Pe de altă parte, corelația dintre cele două mărimi are că și consecință faptul că folosind variațiile de ritm cardiac, se pot obține variațiile ritmului respirator. O metodă pentru obținerea ritmului respirator din variațiile ritmului cardiac este prezentată în Capitolul 3, algoritmul fiind contribuție personală.

2.2.2. Determinarea ritmului respirator folosind senzorul de adâncime al Microsoft Kinect

Lucrări precum [13],[19] sau [31] prezintă folosirea senzorului de adâncime al Microsoft Kinect pentru a detecta variațiile de distanță față de camera al unor regiuni de interes alese. Senzorul de adâncime al Microsoft Kinect are o precizie limitată, atât la distanță între 0.6m și 4m cât și la faptul că distanță este măsurată că și timpul de întoarcere al luminii de spectru IR proiectată pe obiect și întoarsă

la senzor. În același timp, distanța este salvată cu precizie de 13 biți, putând observa în mod teoretic schimbări mici în adâncime.

Regiunea de interes propusă de autorii menționați este în general regiunea pieptului, deoarece inspirația și expirația modifică constant volumul cutiei toracice, această apropiindu-se sau depărtându-se de senzor. Figura 2.16 și Figura 2.17 prezintă alegerea regiunilor de interes în zona pieptului, în [13], respectiv [31]:

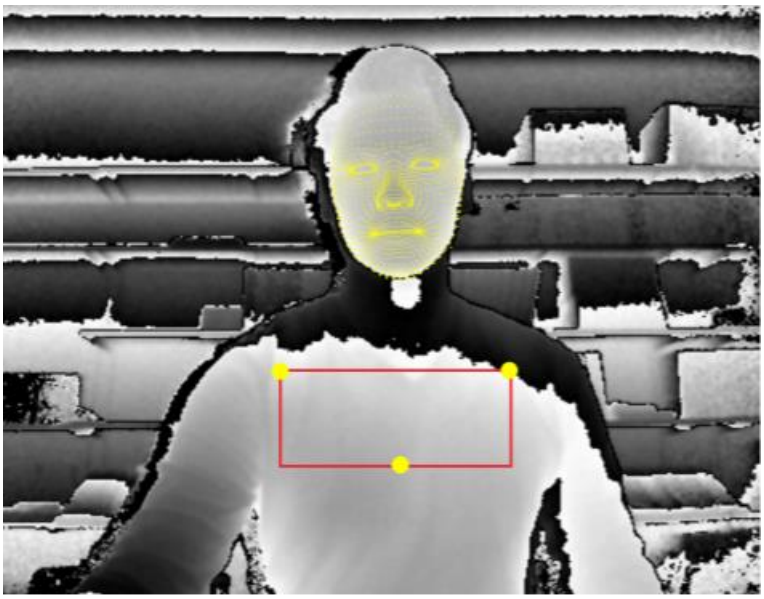

Figura 2.16. Alegerea regiunii de interes in zona pieptului in cazul [13]

Figura 2.17. Alegerea regiunii de interes in cazul [31]

Figurile 20 și 21 prezintă alegerea unei zone de interes pentru măsurarea adâncimii. Se poate observa că zona pieptului este aleasă în ambele lucrări, mișcarea cutiei toracice fiind cea mai proeminentă în timpul respirației. Deoarece senzorul folosit este Microsoft Kinect, există opțiunea de "joints tracking" sau detectarea articulațiilor. Kinect are abilitatea de a genera un întreg schelet bazându-se pe imagine și de a urmări majoritatea articulațiilor în corpul uman. Punctele galbene din Figura 20 reprezintă puncte de interes generate de algoritmul de "joints trancking" al Microsoft Kinect, în zona umerilor și în zona plexului. Zona de interes este un dreptunghi compus folosind aceste 3 puncte. Valorile de adâncime ale senzorului fiind citite doar în interiorul acestei zone.

Figura 22 prezintă rezultate în cazul unui exemplu de măsurare a ritmului respirator, pe perioada de 60 de secunde, în cazul unui subiect de respire constant și păstrează o poziție fixă.

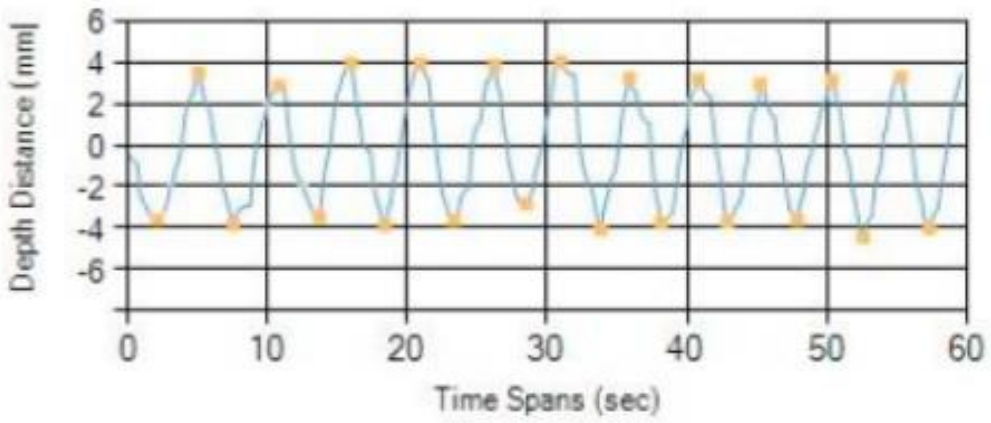

Figura 2.18. Grafic al evoluției ritmului respirator in [31]

Figura 2.18 prezintă evoluția în timp a diferenței față de medie a distanței față de senzorul de adâncime Kinect. Astfel, se poate observa ca axa x este axa timpului iar axa y este distanta în mm față de medie. Se poate observa periodicitatea semnalului si faptul ca in imagine exista 11-12 perioade in 60 de secunde, așa cum ar fi așteptat de la o respirație calma.

2.3.Resurse software

Acest capitol va prezenta atât resursele software utilizate in lucrările științifice citate, ca si cele ce va urma a fi utilizate in Capitolul 3.Metode propuse.

2.3.1. Limbaje de programare: Matlab, Python

Matlab

MATLAB[25] (venind de la Matrix Laboratory) este un mediu de dezvoltare specializat pe calcul numeric si analiza statistica si conține limbajul de programare cu același nume, Mathworks.

Matlab este specializat in manipularea matricelor, fiecare obiect numeric fiind definit ca o matrice in program.

Matlab a fost folosit de către numeroși autori precum [10][14][16] pentru a implementa algoritmii descriși mai sus, programul având la dispoziție numeroase unelte si biblioteci pentru calcul matriceal si prelucrări de semnale sau de imagini.

Python

Python este un limbaj de programare dinamic, orientat obiect, ce conține o serie vasta de biblioteci ce ajuta la prelucrarea semnalelor, imagistica digitala, învățare automată. Python pune accentul pe simplitatea si curățenia codului si are o sintaxa ce permite utilizatorilor sa programeze intr-o maniera mult mai simpla decât alte programe precum C.

Python este un limbaj flexibil fiind folosit in aplicații diverse cum ar fi proiectare web, statistica, matematica, scripting sau algoritmi de învățare automată, datorită pachetelor vaste incluse.

Deși niciuna dintre lucrările de mai sus nu a fost realizata in Python, aceasta lucrare va prezenta o abordare in acest limbaj a algoritmilor descriși in Cap. 2.1.

Printre cele mai importante biblioteci utilizate in implementare se enumera:

-Numpy[26]: Biblioteca ce conține unelte pentru calcul vectorial, algebra liniara etc

-Scipy[27]: Conține diverse modalități de realizare a graficelor, calcul statistic, filtre etc

-OpenCV[28]: Biblioteca specializata in viziune computerizata, prelucrare de imagini, algoritmi de învățare automată aplicată in special pe imagini etc

-PyKinect2 si pygame: biblioteci folosite in mânuirea kinect-ului si extragerea informației de la senzori

2.3.2. Algoritmul Viola-Jones pentru detecția feței

În anul 2001, Viola și Jones au propus un algoritm de clasificare în lucrarea [19], acesta devenind unul dintre cei mai populari algoritmi de clasificare atât pentru fete umane, cât și pentru o mulțime de alte obiecte. Această tehnică de clasificare are performanțe mai slabe decât cele ale rețelelor neuronale, dar algoritmii sunt ușor de implementat și funcționează proporțional cu rețelele în

rularea pe procesor. Antrenarea unui algoritm Viola-Jones este un proces de mare durata, dar detecția cu un algoritm preantrenat este rapidă și suficient de precisă.

Pentru a putea diferenția o față de restul obiectelor, algoritmul Viola-Jones folosește așa numitele caracteristici Haar dreptunghiulare. Aceste caracteristici se folosesc în general de suma valorii pixelilor dintr-o diferite regiuni dreptunghiulare pentru a formă clasificatori ce definesc caracteristici specific feței umane. Exemple de forme ale acestor clasificatori de pot observa în Figura 2.19:

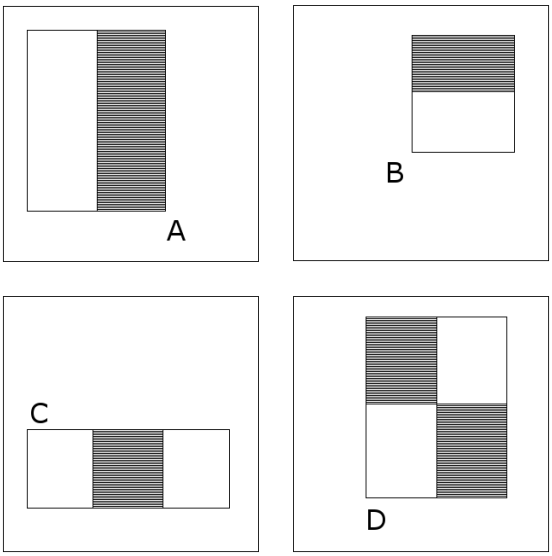

Figura 2.19. Exemple de forme pentru diferite caracteristici dreptunghiulare aplicate la nivelul feței

Se știe ca majoritatea fețelor umane împart câteva seturi de proprietăți. Câteva exemple de aplicare a unor caracteristici Haar din Figura 23 sunt exemplificate in Figura 2.20 si Figura 2.21:

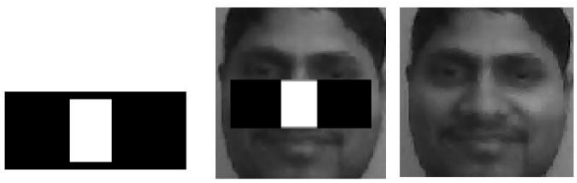

Figura 2.20. Aplicarea caracteristicii Haar de la punctul C din Figura 23

Figura 2.21. Aplicarea caracteristicii Haar de la punctul A din Figura 2.16

Figura 2.20 prezintă aplicarea caracteristicii Haar de la punctul C din Figura 2.19 pe o imagine. Una dintre proprietățile definitorii a feței umane este că regiunea nasului este mai întunecată decât regiunea din dreptul ochilor. Caracteristică Haar din Figura 24 este aplicată și va face media valorii pixelilor din fiecare dreptunghi pentru a dovedi acest lucru.

În Figura 2.21 prezintă aplicarea caracteristicii Haar de la punctul A al Figurii 2.19 pe o imagine continuând o față umană. Se poate observa că o altă proprietate definitore a feței umane este că zona pomeților este mai deschisă la culoare decât zona ochilor.

Fiecare caracteristică Haar dreptunghiulară poate fi considerată că fiind un clasificator slab. Acești clasificatori slabi pot obține o rată de detecție de aproape 100% a caracteristicilor feței, dar pot avea și un număr mare de detecții false. Algoritmul Viola-Jones folosește acești clasificatori slabi cascadați, putând elimina astfel detecțiile false , astfel garantând precizie mare pentru detecția unei fețe, chiar și în timp real.

2.4. Resurse Hardware

Pentru extragerea imaginilor, diverși senzori optici au fost folosiți pentru a asigura diversitatea, dar si pentru a realiza comparații intre diverși factori cum ar fi calitatea imaginii sau numărul de cadre pe secundă.

2.4.1. Camere utilizate

Prima categorie de senzori optici folosiți sunt camerele RGB, de diverse tipuri. Mai multe tipuri de camere au fost folosite pentru a evidenția diferențele de calitate, imagini pe secundă, culoare etc. Printre camerele foto folosite pentru a realiza filmări se enumeră:

- Camere video integrate in laptop-uri: Prima camera, integrata intr-ul laptop model HP Elitebook 8570w, având rezoluția video maxima de 1280x720 (0.9MP) si putând filma in 16 FPS. Referințele viitoare la aceasta camera vor fi făcute prin prescurtarea **CAM1**.
  Cea de-a doua camera, integrata in laptopul Lenovo T580, având rezoluția 960x540 si putând filma in 30 FPS. Referințele viitoare la această camera vor fi făcute prin prescurtarea **CAM2**.
- Camere integrate in telefoane mobile: O prima camera, filmând la rezoluția 1280x720 in 30 FPS. Referințele viitoare la aceasta camera vor fi făcute prin prescurtarea **MOB1**.
  O a doua camera, filmând in rezoluția 4k sau 3840 x 2160 in 30 FPS. Referințele viitoare la aceasta camera vor fi făcute prin prescurtarea **MOB2**.
  Camera foto performanta: O camera Canon 70D, care filmează la rezoluția 1280x720, in 60FPS. Referințele viitoare la aceasta camera vor fi făcute prin prescurtarea **CAMHD.**

2.4.2. Senzorul optic Microsoft Kinect pentru Windows v2

Kinect v2 este un senzor optic fabricat de Microsoft ce a fost folosit tot mai mult pentru proiecte de cercetare, având funcții în plus față de camerele obișnuite. Lucrări precum [16],[19] folosesc un senzor Kinect v2 pentru a capta atât imagini color de calitate superioare, cât și rezultatele produse de senzorii de adâncime, sau senzorul de frecvența infraroșu.

În continuare se vor prezenta specificațiile Microsoft Kinect for Windows v2 [27], împreună cu modul de funcționare a anumiți senzori prezenți pe dispozitiv și modul lor de folosire în lucrările citate [16][19]:

- Senzor de adâncime ce folosește timpul călătorie al luminii în spectru IR pentru a calcula adâncimea obiectelor din imagine. Senzorul poate măsură o adâncime între 0.8 și 7 m. Adâncimea este folosite în lucrările ce folosesc Microsoft Kinect pentru a extrage semnalul din care se determina rată de respirație[13][16][19]. Senzorul de adâncime este folosit cu ajutorul senzorului IR și poate filma în rezoluția 512x424, stocând informația pe 13 biți
- Senzorul infraroșu și emițătoarele de lumina IR sunt capabile să genereze imagini IR în rezoluție 512x424, cu acuratețe de 11 biți
- Camera RGB filmează la rezoluția nativă de 1920x1080, la o rată de 30 imagini pe secundă
- Datele sunt transmise prin USB 3.0 și au latentă de 60ms
- Câmpul vizual al camerei este de 70 grade pe orizontal și 60 grade vertical

3. Metode propuse si implementarea lor

In acest capitol se vor prezenta metodele propuse si fiecare algoritm implementat, fiind explicat modul de implementare folosind o diagrama a codului. Se vor prezenta eventualele îmbunătățiri aduse față de algoritmii prezentați in Capitolul 2 si eventualele probleme întâmpinate.

3.1. Algoritmul de măsurare a ritmului cardiac folosind fotopletismogramia

Algoritmul propus inițial se bazează doar pe principul FPG și este implementat folosind Python[A1]. Principiul de funcționare al algoritmului este asemănător cu cel descris în Capitolul 2.1.1.

În Figura 26 este prezentată diagramă ce descrie funcționalitatea programului, urmând că apoi să fie explicat fiecare bloc funcțional. Programul poate funcționa atât cu video-uri încărcate de pe mașină cât și în timp real. Pentru a alege dintre cele două opțiuni, se setează parametrul “use_live” că având valoarea True pentru folosire în timp real și False pentru folosirea unui video ce este încărcat de la adresa descrisă în parametrul “video_adress”. Explicarea fiecărui parametru de intrare al programului se poate regăsi în Capitolul 4.

Algoritmul folosește o buclă “while” pentru a itera prin toate cadrele din videoclip, un cadru fiind citit o dată pe iterație cu ajutorul bibliotecii cv2 din OpenCV.

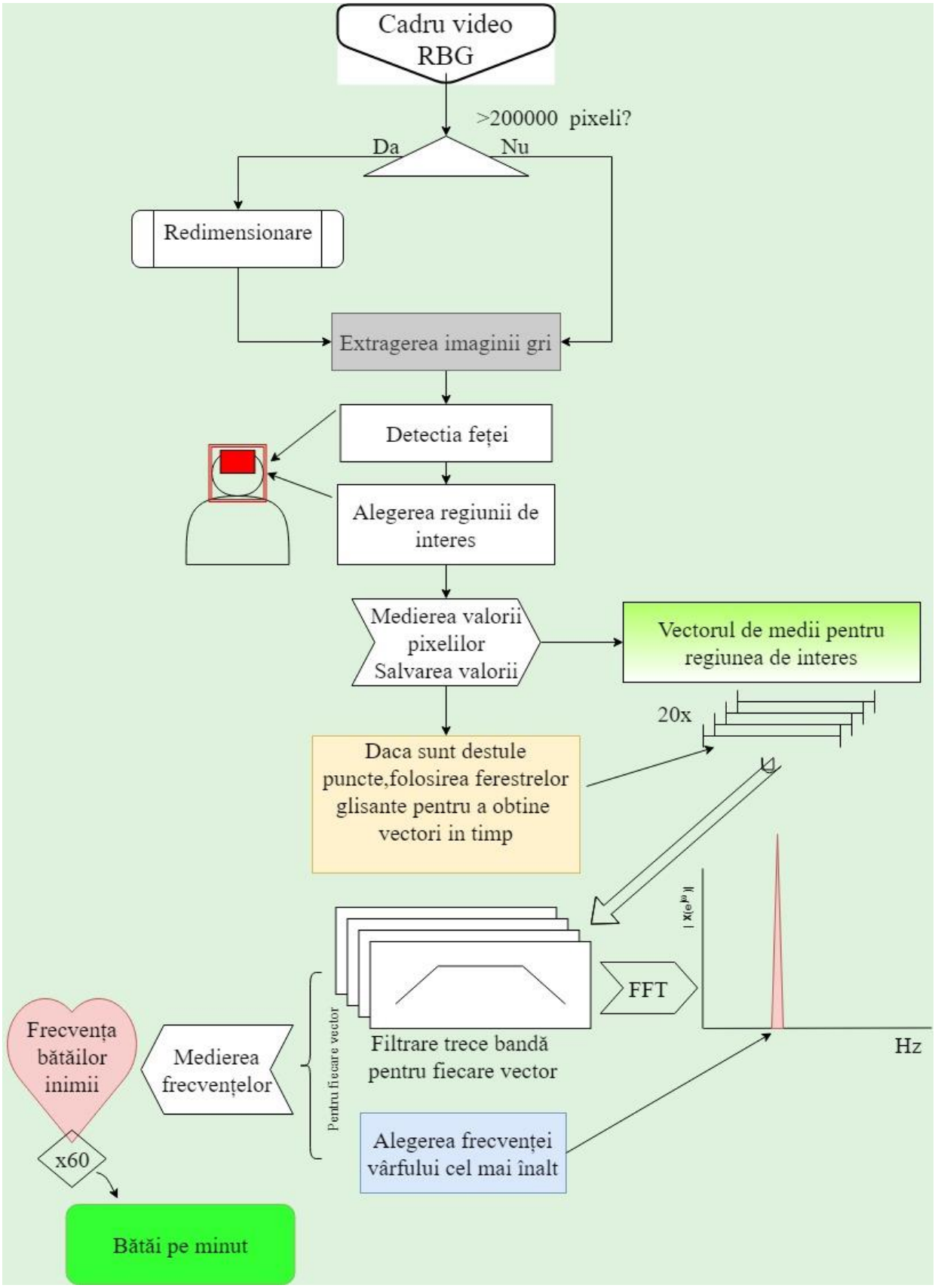

Figura 3.1. Diagrama codului ce folosește FPG pentru a extrage ritmul respirator din video

Figura 3.1 prezintă diagramă codului [A1]. Înainte de rulare, este necesară modificarea unor parametrii ce vor determina modul în care programul rulează:

- Parametrul “use_live”, ce determină dacă programul va funcționa în timp real sau va folosi un video indicat la adresa stocată în “video_adress”
- Parametrul “signal_window”, care reprezintă durata în secunde pe care o vor avea ferestrele glisante cu ajutorul cărora se va calcula ritmul cardiac.
- Parametrul read_rate, care reprezintă numărul de cadre la care valoarea ritmului cardiac va fi calculată

Restul parametrilor de intrare ai programului sunt variabili și determina performanță, de aceea studiindu-se în Capitolul 4.

În continuare, algoritmul din Figura 3.1 va fi explicat, descriind funcționalitatea fiecărui bloc:

- Blocul “Cadru Video RBG” reprezintă citirea unui cadru cu ajutorul bibliotecii cv2 din OpenCV. Setarea implicită pentru OpenCV este să citească imaginea în format BGR = Albastru, Verde, Roșu, reprezentând în ordine cele 3 canale de culoare. Imaginea este citită în rezoluție nativă.
- Deoarece performanță programului este invers proporțională cu numărul de pixeli din imagine, se dorește reducerea numărului de pixeli pentru a crește performanță și a facilita executarea unor calcule în timp real. De aceea, următorul bloc după cel de citire a imaginii compară numărul de pixeli din imagine cu 200000, iar în caz că imaginea are o rezoluție mai mare se va realiza o operație de redimensionare a imaginii la o rezoluție ce conține sub 200000 de pixeli. Acest lucru se face cu ajutorul librăriei cv2, funcția de redimensionare primind că și parametru un factor de redimensionare calculat. În cazul în care imaginea nu depășește 200000 de pixeli, această nu va fi redimensionată, păstrându-se dimensiunile inițiale.
- A treia operație este extragerea imaginii de culoare gri(sau alb-negru) din cea BGR, deoarece algoritmul de detecție a feței funcționează mult mai rapid pe imagini având un singur canal de culoare decât 3 canale. Imaginea gri sau Greyscale se extrage cu o funcție implicită a bibliotecii cv2, care obține canalul gri din cele 3 canale de culoare folosind contribuții procentuale pentru fiecare canal, realizând următoarea operație pixel cu pixel:

Canal_Gri = ((0.3 * Canal_Roșu) + (0.59 * Canal_Verde) + (0.11 * Canal_albastru)

Figura 3.2. Exemplu de Imagine color

Figura 3.3. Exemplu de imagine alb-negru

- Detecția feței se realizează cu algoritmul Viola-Jones, prezentat in capitolul 2.3.2., folosind un clasificator preantrenat pentru detecția feței ce se găsește in biblioteca OpenCV, numit “haarcascade_frontalface_default2”. Clasificatorul a fost ales prin comparație cu celelalte clasificatoare ce se găsesc in biblioteca OpenCV, acesta având cele mai bune rezultate. In Figura 3.4 (a),(b),(c) sunt prezentate exemple de detectare a regiunii faciale folosite in maturatori:

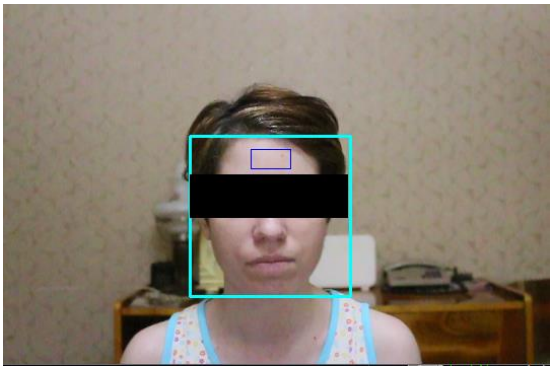

Figura 3.4 (a)

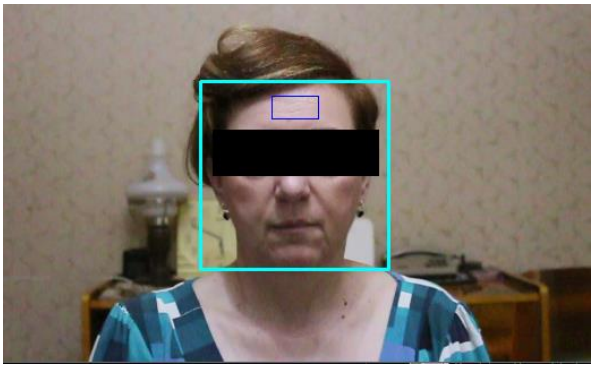

Figura 3.4 (b)

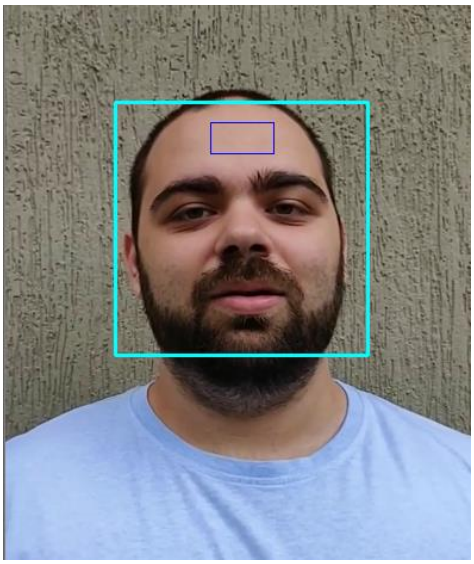

Figura 3.4 (c)

Figura 3.4.(a)(b)(c) Exemple concrete de utilizare a algoritmului Viola-Jones pentru detectarea feței si alegerea regiunii de interes

- Regiunea de interes este aleasă bazându-se pe proporționalitatea feței, astfel încât această ar trebui să fie pe mijlocul frunții pentru majoritatea oamenilor. Ea este marcată printr-un dreptunghi de culoare albastru închis, având centrul amplasat la jumătatea lățimii dreptunghiului care delimitează față și latura de sus la 1/12 din înălțimea framei față de porțiunea de sus. Regiunea de sus a fost aleasă că fiind fruntea deoarece este zona cu cea

mai mare arie de piele nedescoperită, având posibilitatea să aibă cel mai mic raport semnal/zgomot.

- Blocul de mediere a valorii pixelilor are funcția de a realiza o medie a valorii de pe canalul verde (deoarece semnalul de pe canalul verde este cel mai puternic, conform [20]). Totuși, canalul de culoare dorit pentru funcția de mediere poate fi selectat folosind variabilă "choose_colour". Valoarea obținută după mediere este apoi salvată într-un vector.
- Numărul de puncte necesare pentru că fereastră glisantă să fie destul de mare pentru a începe operația de extragere a frecvenței este durata în secundă a ferestrei * numărul de cadre pe secundă înregistrat de camera(fs = frecvența de eșantionare).Pentru a evita rezultate eronate, 20 de ferestre glisante de aceeași dimensiune , dar mutate la stânga cu un cadru sunt folosite pentru calcul.
- Fiecare vector de valori din fereastră glisantă va fi filtrat cu ajutorul unui filtru trece-bandă Butterworth , pentru a elimina frecventele în intervalele ce nu pot fi frecvente ale inimii. O filtrare cât mai largă care să includă frecventele bătăilor inimii ar putea fi între 0.6 și 3 Hz, corespunzând valorilor de 36 și 180 bpm. Un exemplu de fereastră mai îngustă de poate aplică dacă se știe vârstă sau condiția fizică a participantului, folosind Tabelul 1 sau Tabelul 2 prezentate în capitolul 1. Astfel, un exemplu de filtru mai îngust ar putea să fie între 1 Hz și 1.7 Hz, corespunzând valorilor între 60 și 100 bpm.

  Figura 3.5 prezintă evoluția în timp a semnalului extras din medierea valorilor canalului verde în regiunea de interes:

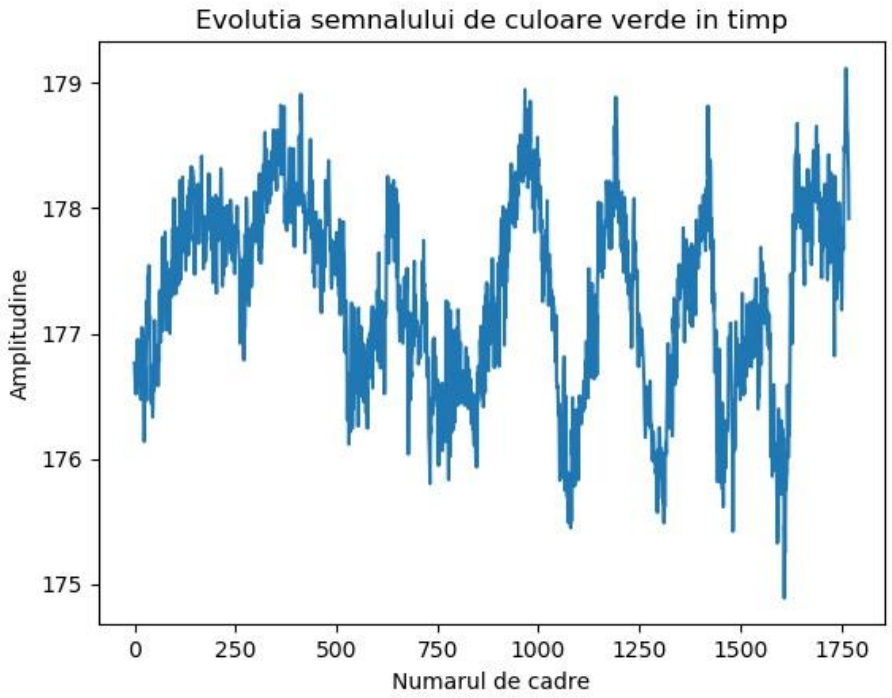

Figura 3.5. Evoluția in timp a semnalului verde din regiunea de interes

Se poate observa o oarecare periodicitate in cazul Figurii 30, aceasta fiind data de ritmul respirator. Acesta va fi extras in următorul subcapitol.

Semnalul după filtrarea cu un filtru Butterworth trece-banda de ordinul 4 este reprezentat in Figura 3.6:

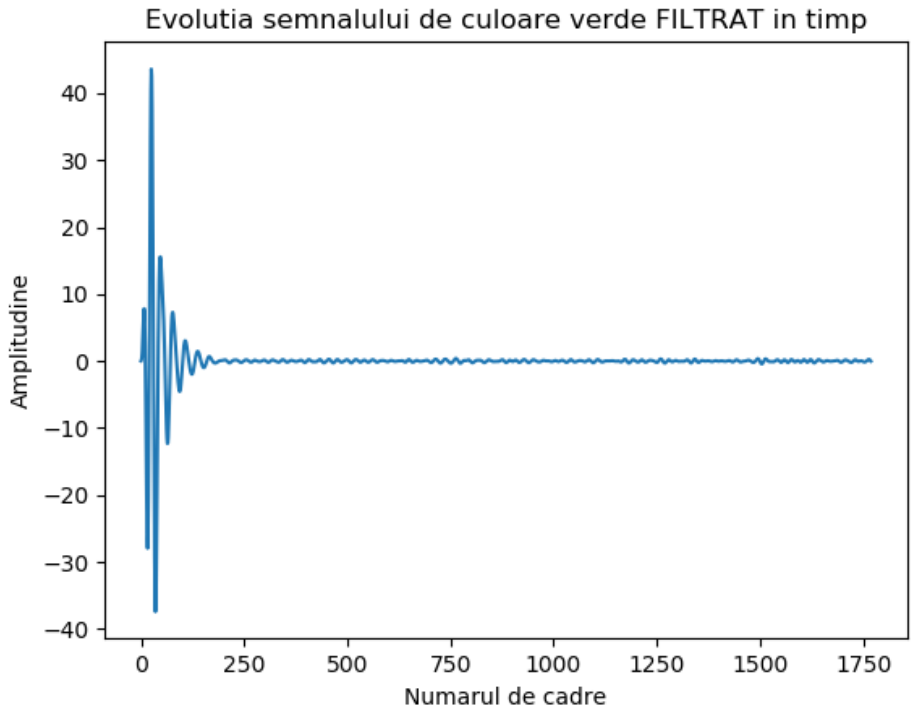

Figura 3.6. Semnalul de interes in timp , după filtrare cu filtru neinițializat

Se poate observa o oarecare periodicitate în semnalul filtrat. Totuși, se observa că semnalul are o amplitudine foarte mică la început, fiind filtrat după. Această este cauza faptului că filtrul este neinițializat. Pentru a remedia această problema, la semnalul de interes se va adaugă la început în semnal ce are o valoare continuă egală cu media unei fracțiuni de la început a semnalului. Acest semnal a primit denumirea de “dummy” , deoarece este un semnal ce nu conține informație, el fiind utilizat doar pentru inițializarea filtrului. Figura 32 prezintă semnalul în timp, după filtrare, începând de la al 200-lea eșantion:

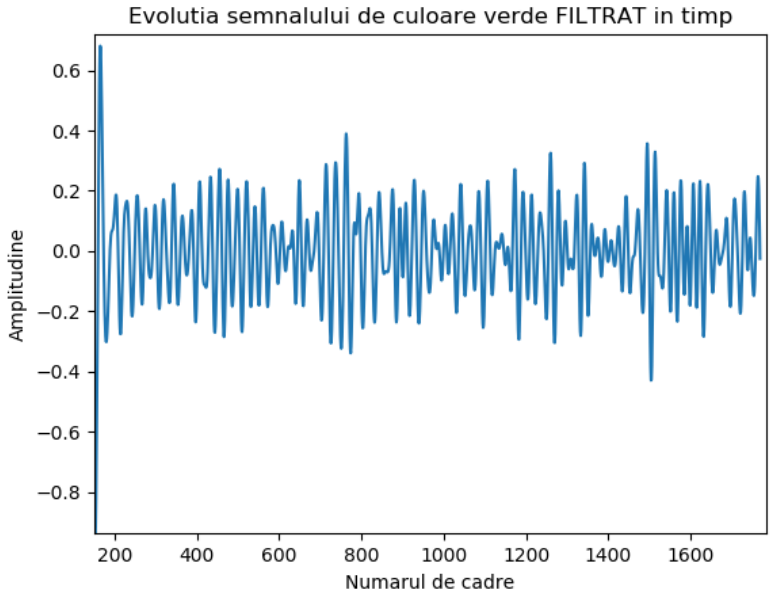

Figura 3.7. Semnalul din Figura 31, fără primele 200 de eșantioane

Se poate observa o periodicitate in semnalul din Figura 32, care reprezintă chiar variația bătăilor inimii in timp.

Figura 3.7 prezintă semnalul in timp, la care a fost adăugat semnalul "dummy" pentru inițializarea filtrului. In Figura 3.8 se poate observa rezultatul inițializării filtrului, după eliminarea semnalului "dummy" filtrat:

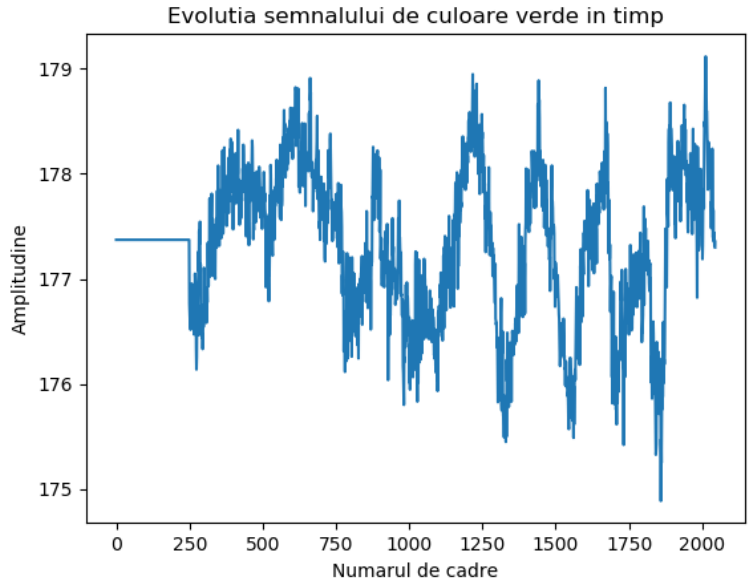

Figura 3.8. Semnalul de interes in timp, la care a fost adăugat semnalul "dummy"

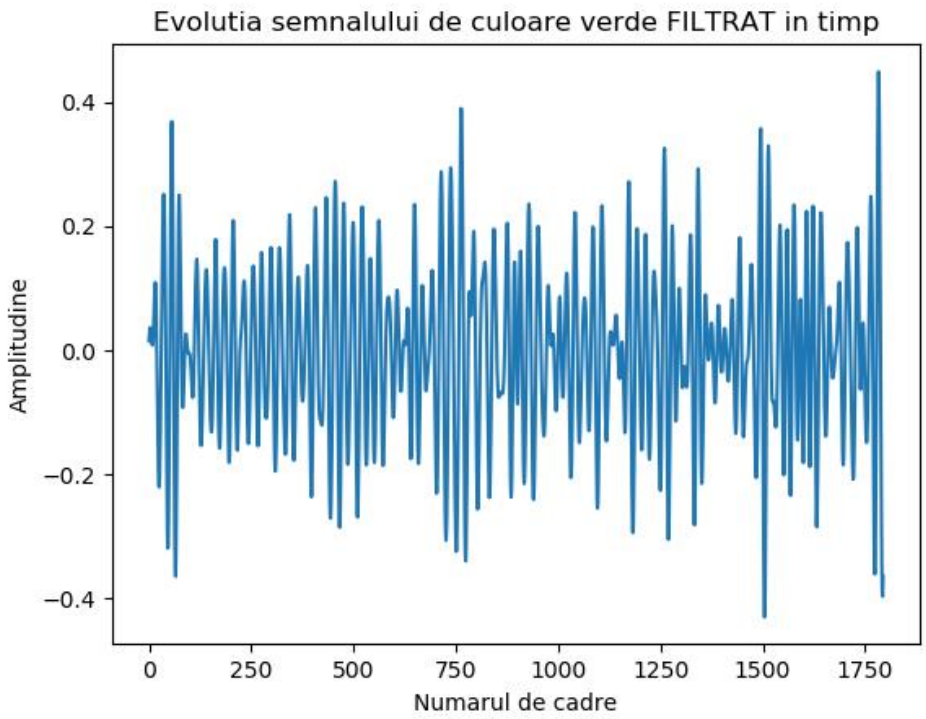

Figura 3.9. Semnalul de interes după filtrare si eliminarea semnalului "dummy": evoluția ritmului cardiac in timp

După filtrarea inițializată de semnalul "dummy", primele eșantioane ale semnalului filtrat sunt asemănătoare cu restul semnalului. Introducerea acestui semnal de inițializare auxiliar este o contribuție personală unică, celelalte lucrări științifice prezentate preferând să elimine cu totul porțiuni de la începutul semnalului.

- Ultimul pas prezentat în organigramă codului este realizarea transformatei Fourier rapide pentru a trece din domeniul timp în domeniul frecvența, și alegerea vârfului cu cea mai mare amplitudine că fiind frecvența principala reprezentând frecvența bătăilor inimii. Rezultatul se înmulțește cu 60, obținând valoarea numărului de bătăi ale inimii într-un minut. Figura 35 prezintă formă în frecvența a semnalului fără adăugarea semnalului "dummy", iar Figura 36 prezintă reprezentarea in frecvența a semnalului filtrat cu filtrul inițializat. Persoana din semnalul exemplu a avut un puls variabil între valoarea de 81 bpm până la 86 bpm, conform unui dispozitiv Smartwatch.

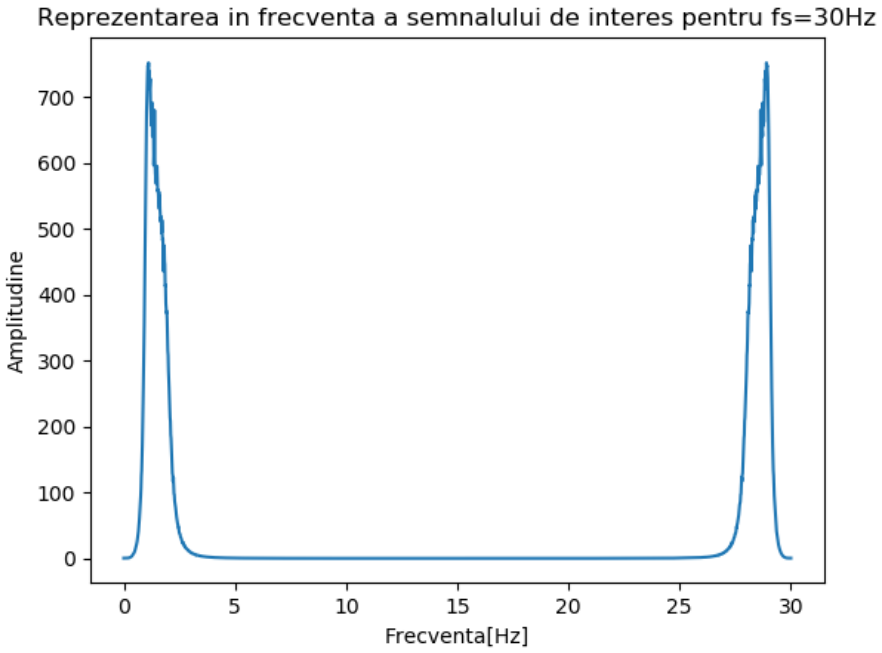

Figura 3.10. Reprezentarea in frecvența a semnalului de interes fără utilizarea "dummy"

In cazul Figurii 3.10, frecventa componentei de valoare maxima este de 66.92bpm, pentru un semnal intre 81 si 86 bpm.

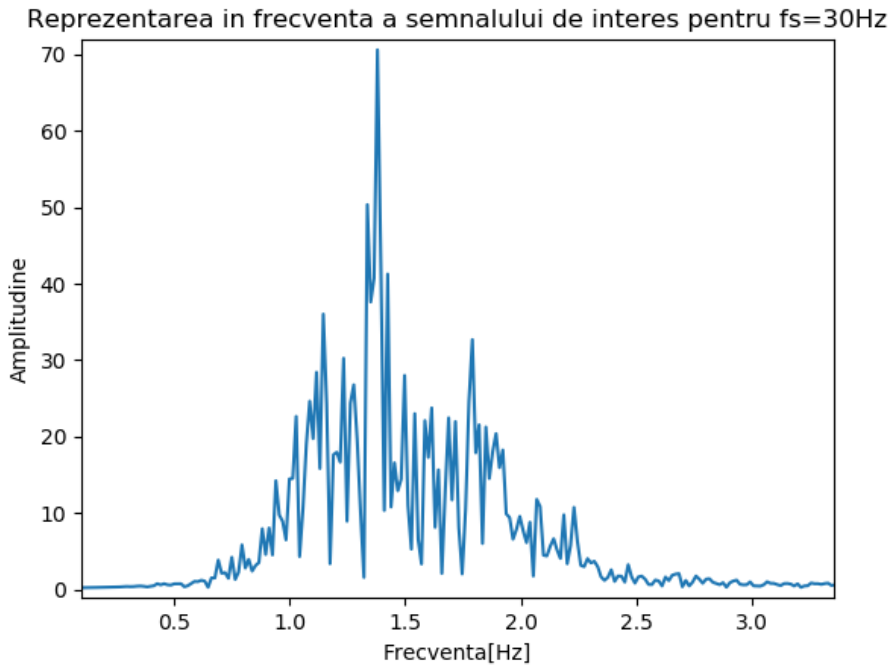

Figura 3.11. Reprezentarea in frecvență a semnalului de interes cu utilizarea semnalului de inițializare "dummy"

Daca semnalul din Figura 3.10 are un vârf chiar la început, deoarece frecvența semnalului de amplitudine mare este mica față de restul semnalului, putem observa un vârf principal in cazul Figurii 3.11, datorita implementării semnalului Dummy.

Codul pentru acest algoritm [A1] prezintă atât abordarea cu fereastră glisantă, cat si o abordare ulterioară de analiză a întregului semnal.

3.2. Algoritmul de măsurare a ritmului respirator folosind fotopletismogramia

Acest algoritm este prezent de asemenea în [A1], utilizând același semnal și aceeași suprafață de interes că în cazul algoritmului pentru determinarea ritmului cardiac. Algoritmul este explicat în Capitolul 2.2.1.

Principiul algoritmului este același, exceptând faptul că filtrarea se face la frecvente mai mici , un exemplu de interval fiind 0.2Hz-0.4Hz, care este echivalent cu 12 respirații pe minut – 24 respirații pe minut. Spre deosebire de algoritmul pentru determinarea ritmului cardiac, unde sursele de zgomot principale erau introduse de mici schimbări ale unghiului cu care lumina lovește regiunea de interes, în acest caz o sursă posibila de zgomot este chiar mișcarea subiectului uman, ce se poate întâmplă să fie în banda semnalului de interes.

Algoritmul este de asemenea greu de realizat în timp real fără a avea un video de mare durata, deoarece semnalul filtrat va avea un număr mult mai mic de perioade decât în cazul bătăilor inimii. Totuși, în caz de nevoie se poate folosi acest algoritm împreună cu ferestre mai mari, de cel puțin 10 secunde. O principala aplicație a detectării ritmului respirator este compensarea ritmului cardiac, ce se schimbă cu respirația datorită ARS.

În cazul [A1], algoritmul de determinare al ritmului respirator este aproape identic cu cel de determinare a ritmului cardiac, exceptând frecvențele de tăiere ale filtrului. Exemple de rezultate în timp, ale algoritmului, care conține și metodă semnalului “dummy” este prezentată în Figura 3.12. Filtrarea semnalului se realizează cu un filtru de ordin 1, Butterworth, preinițializat cu semnalul “dummy”, între frecvențele de 0.2 și 0.4Hz.

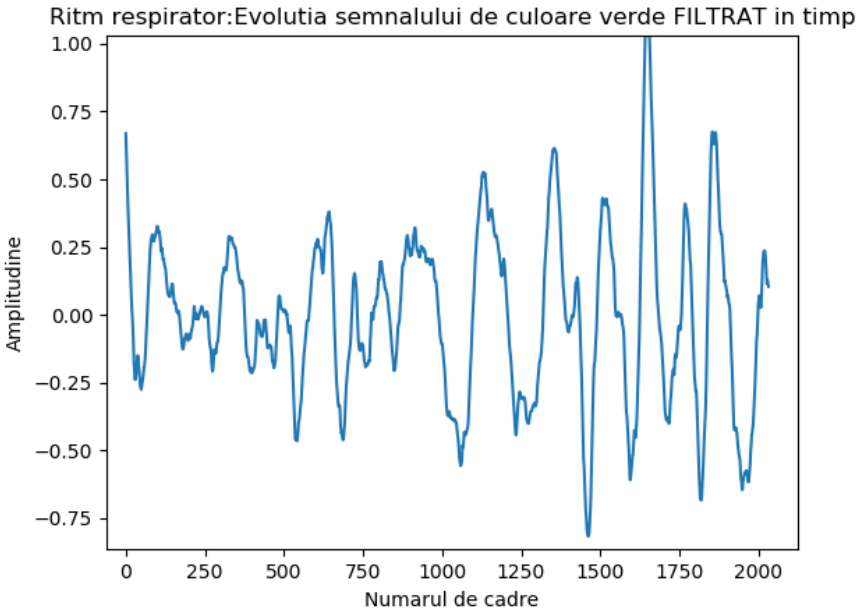

Figura 3.12. Semnalul de respirație, filtrat cu un filtru Butterworth de ordinul 1

Se poate observa ca filtrarea cu filtru de ordin 1 este suficienta pentru i identifica vârfurile semnalului de respirație. Ritmul respirator obținut in acest exemplu este de aprox. 13 respirații pe minut, care se potrivește cu rezultatele din imagine care conțin cam 15 respirații in 70 de secunde.

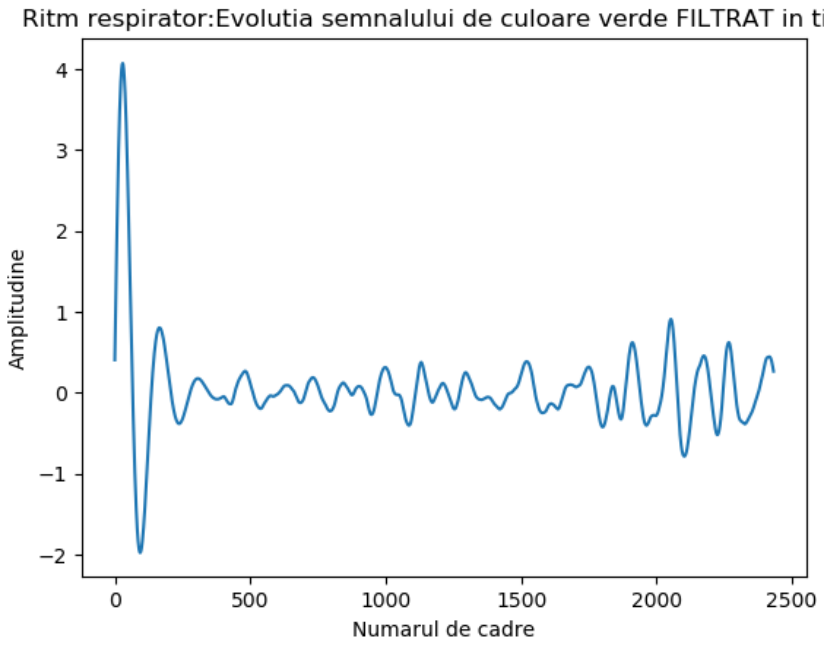

Figura 3.13. Filtrarea semnalului de respirație cu filtru de ordinul 2

Filtrarea semnalului cu un filtru de ordin mai mare elimina mare parte din zgomot, dar are de asemenea probleme din punctul de vedere al inițializării filtrului.

3.3. Algoritmul de amplificare Eulerianã a video-ului

In acest subcapitol se va prezenta algoritmul implementat pentru Amplificarea Eulerianã a unui video[A2][A3],fiind asemănător cu cel descris din punct de vedere teoretic in Capitolul 2.1.3. Metoda propusa se bazează implementarea si îmbunătățirea algoritmului prezentat in [32]. Folosind toate nivelele piramidei Gausianã pentru a realiza amplificarea video.

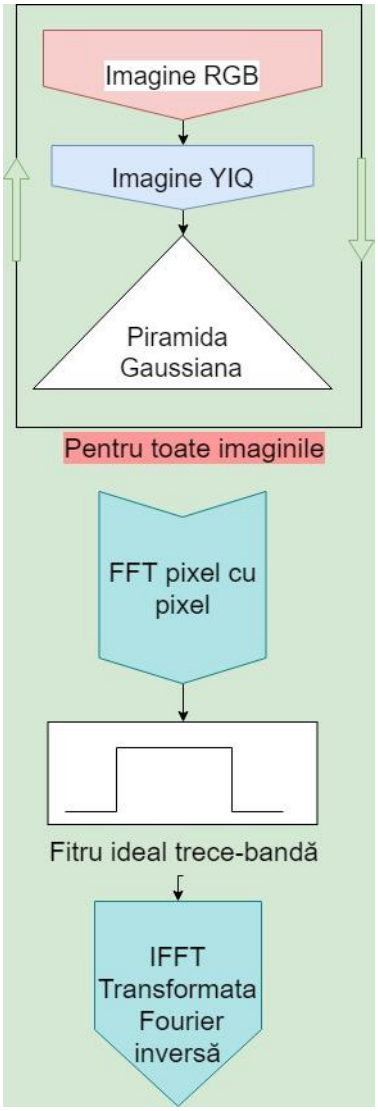

Figura 3.14. Filtrarea ideala a secvenței video

Figura 3.14 prezintă filtrarea ideala a secvenței video. Deoarece singurele variații de interes sunt in banda de trecere a filtrului ideal (depinzând însa de aplicație), eliminarea celorlalte variații este importantă deoarece ele sunt o sursa de zgomot pentru secvența video.

Imaginea RGB de la intrare va fi transformată in imagine YIQ,ce reprezintă: Y = luminanța imaginii, fiind practic reprezentarea imaginii alb-negru, I si Q se numesc fază si cuadratură, reprezentând componenta de albastru-roșu , respectiv mov-gri. I si Q dau nivelul de culoare din imagine, ce va fi atenuat sau amplificat mai departe. Aceasta trecere este realizata pentru a putea apoi amplifica sau atenua nivelul de culoare din imagine, după nevoie. După filtrare se va realiza amplificarea sau atenuarea culorii imaginilor si se va trece imaginea înapoi in reprezentarea RGB.

Construirea piramidei Gausiene se realizează ulterior, folosind algoritmii prezentați în capitolul 2.1.3. Pentru fiecare nivel al piramidei și pentru fiecare imagine în timp cu acel nivel, se va realiza transformata Fourier pixel cu pixel. Apoi, se vor egala cu 0 toate frecvențele ce nu sunt în bandă de frecvența dorită, aplicând astfel un filtru trece-bandă ideal. În cele din urmă, se va realiza transformată Fourier Inversă pentru a obține semnalele imagini ce au variații doar în banda de frecvența dorită.

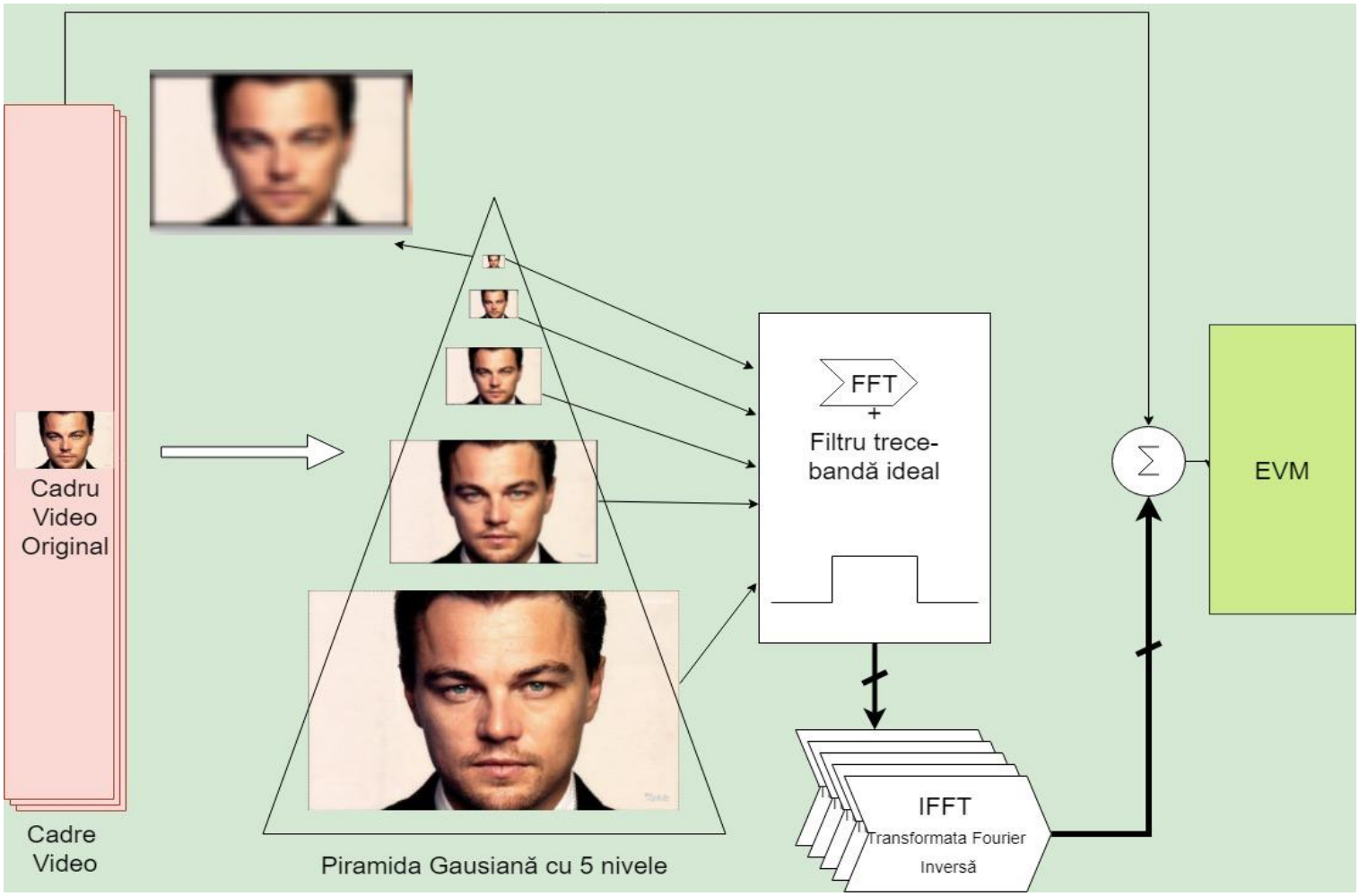

Figura 3.15. Algoritmul pentru obținerea Amplificaării Euleriană a unui video din format cadrele primite

Algoritmul prezentat în Figura 3.15 are că scop realizarea amplificării Euleriană a unui video primit la intrare. Pentru fiecare dintre cadre, se realizează o piramidă Gausiană conform algoritmului explicat în Capitolul 2.1.3. Apoi, având evoluția în timp a fiecărui nivel al piramidei, se va realiza trecerea în frecvența a semnalului imagine, pixel cu pixel și apoi filtrarea ideală trece-bandă, eliminând componentele de frecvența din afară domeniului de interes și realizând transformata Fourier inversă pentru a ajunge înapoi la imagini.

Imaginile filtrate , pentru fiecare nivel, se vor aduna cu imaginea inițială pentru a obține EVM (Amplificarea Euleriană a video-ului).

În Figura 41 se pot observa rezultatele amplificării video cu ajutorul algoritmului, folosind 6 nivele ale piramidei Gausiene și o amplificare alpha = 60, filtrând frecvențele în afară intervalului 0.8Hz și 1Hz, deoarece se cunoaște pulsul individului că fiind 54bpm.

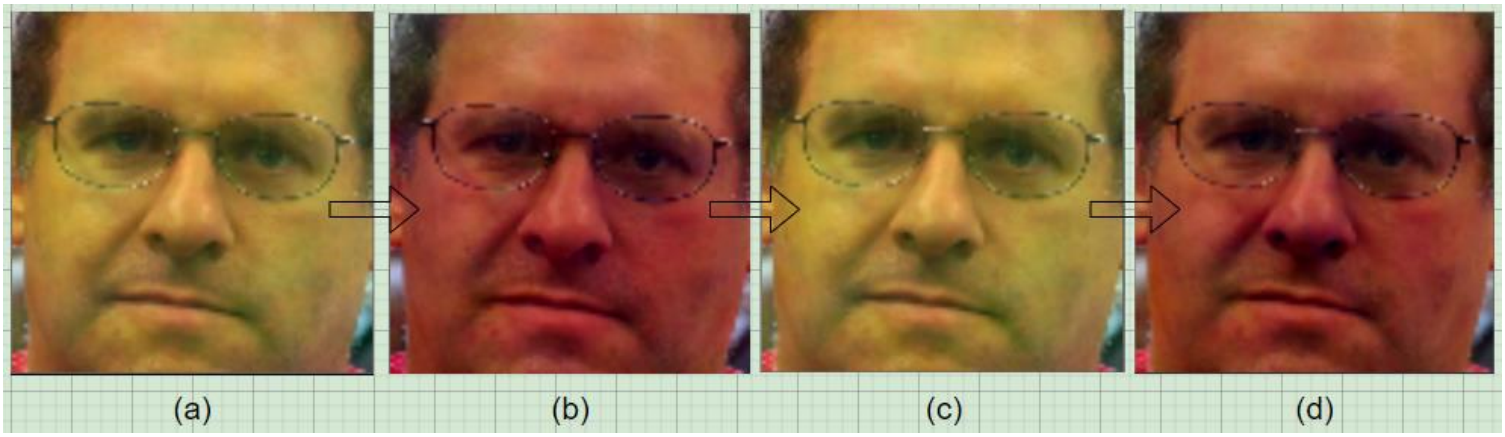

Figura 3.16. Rezultatele Amplificării Euleriene Video, putându-se observa pulsul cu ochiul liber

Figura 3.16 arată rezultatele EVM asupra imaginii: se poate vedea clar cu ochiul liber evoluția în timp a bătăilor inimii, schimbării presiunii în zona venelor feței.

În [A3] sunt prezente diverse funcții pentru a ușura calculul EVM, cum ar fi funcții de calculare a matricii de pixeli Gausiană ce va fi folosită pentru realizarea piramidei, sau funcția Load_video care folosește la salvarea imaginilor dintr-un video și a zonei de interes să feței într-un vector.

Un avantaj major al EVM este îmbunătățirea raportului semnal-zgomot al imaginilor, astfel putându-se folosi algoritmul prezentat în Capitolul 3.1 (impreună cu cel din 3.2, dar folosind altă bandă de frecvențe specifică respirațiilor) mult mai eficient.

Dacă banda de frecvente este aleasă în alt domeniu, se poate observa amplificarea mișcărilor , văzându-se cu ochiul liber mișcări milimetrice neobservabile înainte.

În acest moment, programul prezentat în [A2] nu este folosibil în timp real. Totuși, o abordare spre folosirea în timp real a EVM folosind filtrare ideală este utilizarea unei ferestre glisante pentru a alege punctele în timp pentru care se realizează filtrarea. Totuși, această metodă este foarte costisitoare din punct de vedere Hardware. În capitolul 2.1.3., în dreptul subcapitolului Filtrarea în timp real este prezentată o metodă de filtrare care nu este costisitoare din punct de vedere al hardware-ului, dar este mult mai zgomotoasă decât filtrarea ideală. De aceea, am ales să folosesc în algoritmul implementat metoda filtrării ideale descrisă până acum.

3.4. Baza de date pentru detecția disimulării

Scopul lucrării este crearea unui sistem cât mai aproape de un sistem poligraf. Până în acest moment, s-au prezentat implementările algoritmilor ce ajută la detecția ritmului cardiac și ritmului respirator. Un scop final este crearea unui algoritm ce realizează o distingere între un comportament normal și unul disimulat.

În acest scop , am creat o bază de date ce se bazează pe reacțiile involuntare ale subiecților umani la diferite imagini. Baza de date conține 16 videoclipuri de lungime aproximativ un minut în care subiecții umani vizualizează o serie de obiecte, având că scop obținerea unei reacții involuntare la observarea unui obiect cunoscut prealabil. Participanții au primit un obiect înaintea filmării videoclipului și au fost instruiți să nu reacționeze la imaginile cu obiecte ce le-au fost arătate. Presupunerea este că atunci când subiecții vor vedea obiectul ce li s-a prezentat în prealabil, vor avea o reacție inconștientă care se va translata în o schimbare a ritmului respirator. Videoclipurile au fost filmate cu o cameră ce funcționează la 60 FPS și are o rezoluție de 1280x720.Camera este prezentată în Capitolul 2.4.1. sub denumirea de CAMHD.

În scopul prezicerii a unei valori cât mai bune a ritmului respirator, pulsul subiecților umani a fost măsurat înainte și după vizionarea videoclipului. Imagini din videoclip se pot observa în Figura 3.17, obiectul arătat prealabil subiecților fiind încercuit cu culoarea roșie.

Figura 3.17. Obiectele arătate subiecților in videoclipuri

Deoarece dimensiunea bazei de date este mică, neavând destule măsurători, nu se poate crea o rețea neuronală pentru a învăța caracteristicile comportamentului disimulat, deoarece cantitatea datelor pentru antrenare cât și datelor pentru validare pentru a ne asigura rețeaua neuronală nu va supraînvăța toate datele, fără a învață să detecteze disimularea în cazul general, nu este suficientă

Totuși, in imaginea de mai jos este prezentată o rețea neuronală simplă ce în viitor ar putea învață tendințele comportamentului disimulat, având o baza de date extinsă, folosind valorile ritmului cardiac extrase din fișierele video. O abordare asemănătoare se poate regăsi în lucrările ce încearcă să dezvolte sisteme de pot determina dacă pacientul prezintă o afecțiune cardiacă, precum [33][34][35]. Aceste lucrări au folosit abordări simple, având doar un strat ascuns. O prezentare a conceptului unei rețele neuronale ce ar putea folosită pentru o detecție binară a disimulării este prezentată în Figura 3.18:

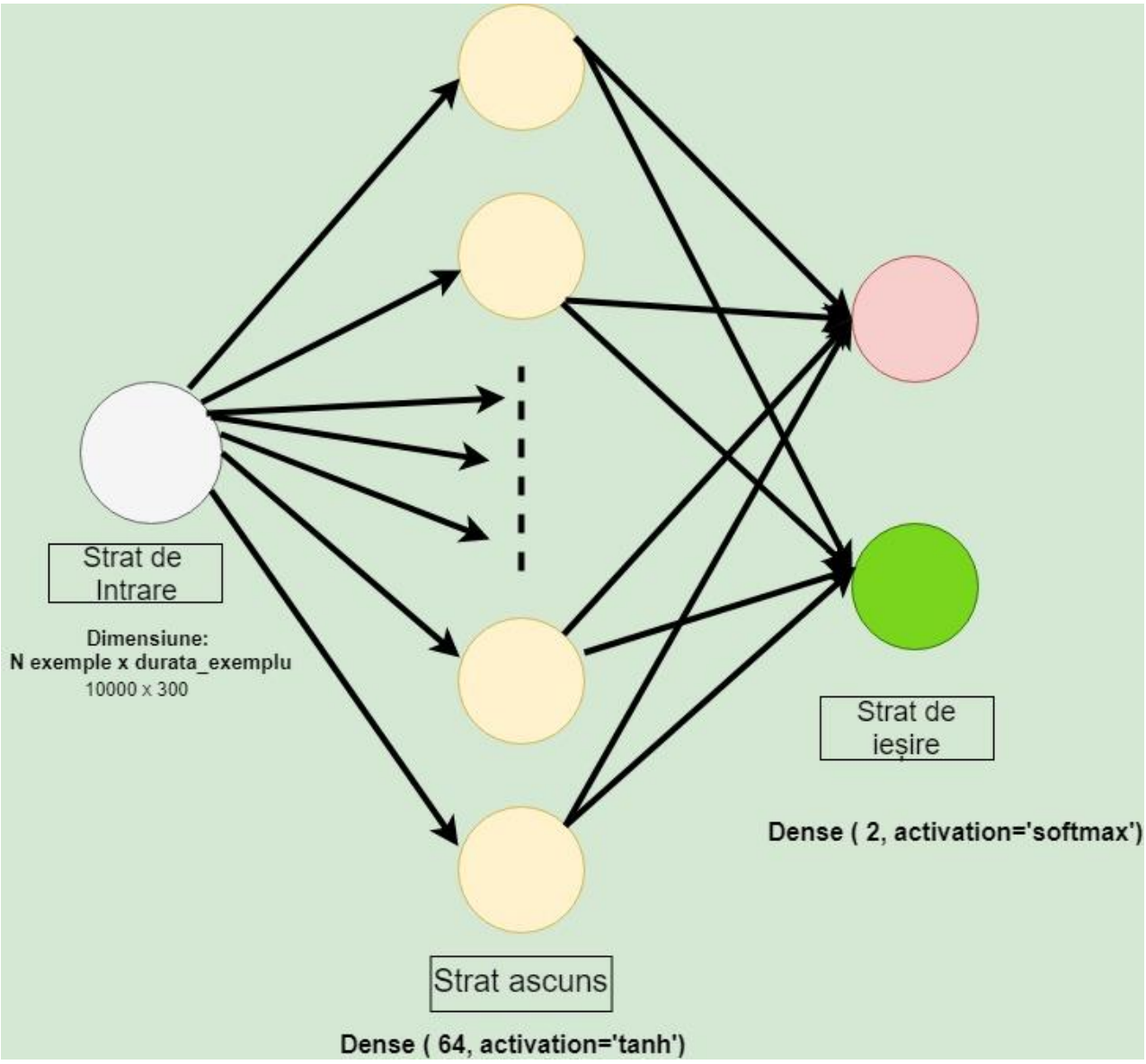

Figura 3.18. Exemplu de posibilă structură a rețelei folosite pentru identificarea disimulării, in Keras

Semnalele ce sunt preluate de stratul de intrare sunt semnale ce reprezintă valoarea ritmului cardiac in timp, citita o data pe cadru si normalizată. Stratul de intrare va primi videoclipuri de 10 secunde(=300 cadre in cazul unei camere cu 30fps). Acestea vor fi adnotate, având eticheta 1 dacă subiectul se uita la imaginea cu obiectul pe care l-a primit inițial si 0 in cazul in care acesta se uita la una din celelalte imagini.

Stratul ascuns este creat in biblioteca Keras, Python si este un strat Dense, utilizând o funcție de activare de complexitate scăzută.

Stratul de ieșire este un strat Dense, cu activare ‘softmax’ deoarece ieșirea rețelei trebuie sa ducă intr-una din valorile de 0 sau 1(reprezentând comportament normal sau disimulat).

O perspectivă de viitor pentru această lucrare este expandarea bazei de date de disimulare si antrenarea rețelei in perspectiva de a obține un detector binar de disimulare.

3.5. Folosirea Microsoft Kinect v2 prin intermediul Python

Folosirea senzorului Microsoft Kinect este importantă pentru obținerea atât a unei imagini color de mare calitate, cât și a unor imagini de spectru IR sau de adâncime pentru obținerea ratei bătăilor inimii și ratei respiratorii.

În [A4] și [A5] sunt prezentate programe ce folosesc senzorul Microsoft Kinect pentru a înregistra cadre în lumina IR, respectiv cadre RGB. Totuși, resursele Hardware nu permit folosirea programului [A5] în mai mult de 15 fps. Atât cadrele color cât și cele IR vor fi salvate că și imagini .JPG, în fișierul de lucru, pentru folosire ulterioară.

Figura 3.19. Exemplu de imagine IR,salvată sub forma alb-negru

4.Rezultate Experimentale

Acest capitol va prezenta rezultatele experimentale produse de algoritmul de măsurare a pulsului , cât și a ratei respirației, comparând rezultatele cu cele din lucrările prezentate in Capitolul 2. In plus, se va observa eficienta algoritmilor când EVM a fost folosita in prealabil.

4.1. Algoritmul de determinare a ritmului cardiac si al ritmului respirator

Pentru testarea algoritmilor, s-au folosit 10 fișiere video cu lungime între 22 și 240 de secunde conținând imagini cu persoane diferite, de sex masculin și feminin, cu vârste cuprinse între 17 și 52 de ani. Pulsul persoanelor a fost măsurat continuu cu ajutorul unui ceas inteligent, iar valori de puls au fost înregistrate la interval de 10 secunde. Față de ceasul inteligent, algoritmul folosit nu ține cont de compensarea ARS. În Tabelul 4.1 se află rezultatele măsurătorilor de puls , deviația standard și măsurătoarea pulsului pe toată durata filmării. Filtrul folosit este un filtru Butterworth de ordinul 4, iar frecvențele din bandă de trecere sunt între 1Hz și 2Hz, reprezentând valori ale pulsului între 60 și 120 bpm. Fereastră glisantă folosită are lungime de 10 secunde.

Videoclipurile 1,2,3 sunt filmate cu o camera integrată într-un laptop, cu 30FPS.

Videoclipurile 4,5 sunt filmate cu ajutorul unei camere de telefon de rezoluție 4k, care filmează în 30FPS.

Videoclipurile 6,7,8,9,10 sunt filmate cu ajutorul unui aparat foto, având rezoluția 1280x720 și putând filma în 60FPS.

Videoclip #	Intervalul de valori ale pulsului măsurate de ceasul inteligent	Intervalul de valori ale pulsului măsurate de algoritm	Media pulsului	Deviația standard	Valoarea frecventei dominante in filmare	% valori in intervalul corect ±10%
1	80-88	68.9-104	81.36	8.7	83.03	74.4
2	85-90	68-101	74.21	7.46	78	62
3	89-93	87-96	91.36	3.21	98	100
4	95-102	83-106	91	5.83	99.51	92.14
5	95-110	81-111	92	7.63	83	84.2
6	76-79	71-87	78.04	4.47	78.16	97.41
7	68-72	66-94	70.48	4.31	69.91	96.6
8	66-74	68-95	71.86	2.17	73.06	99.18
9	57-63	60-78	61.2	4.46	61.03	89.9
10	73-80	69-112	79.91	9.94	79.27	85.7

Tabelul 4.1. Comparație intre valorile înregistrate cu ceas inteligent si valorile obținute de algoritm

Observații: Participanții din videoclipurile #1 și #2 s-au mișcat foarte mult în timpul filmărilor. În schimb, participanții înregistrărilor cu numărul 6,7,8,9,10 au fost instruiți să se miște cat mai puțin.

Videoclipurile cu numerele 4 și 5 conțin imagini cu subiecți ce înainte de filmare s-au angajat în a face activitate fizică, dorind în mod intenționat să își crească ritmul cardiac.

Videoclipul #3 a avut o durată de doar 22 de secunde.

În ceea ce privește ritmul respirator, subiecții din fiecare videoclip au ținut cont de numărul de inspirații de pe durata videoclipului. Acestea s-au comparat cu semnalul din regiunea de interes, filtrat la frecvențe cuprinse între 0.2 Hz și 0.4 Hz, numărând vârfurile semnalului.

Media ritmului respirator măsurat de către subiecții filmărilor s-a încadrat între 12 respirații pe minut și 22 respirații pe minut. Măsurând semnalul filtrat ce indică ritmul respirator, s-a constatat că aceste are acuratețe de ± 1,5 respirații pe minut. Ca exemplu, Figura 45 prezintă ritmul respirator al unei persoane ce a numărat, în medie, 20 de respirații pe minut.

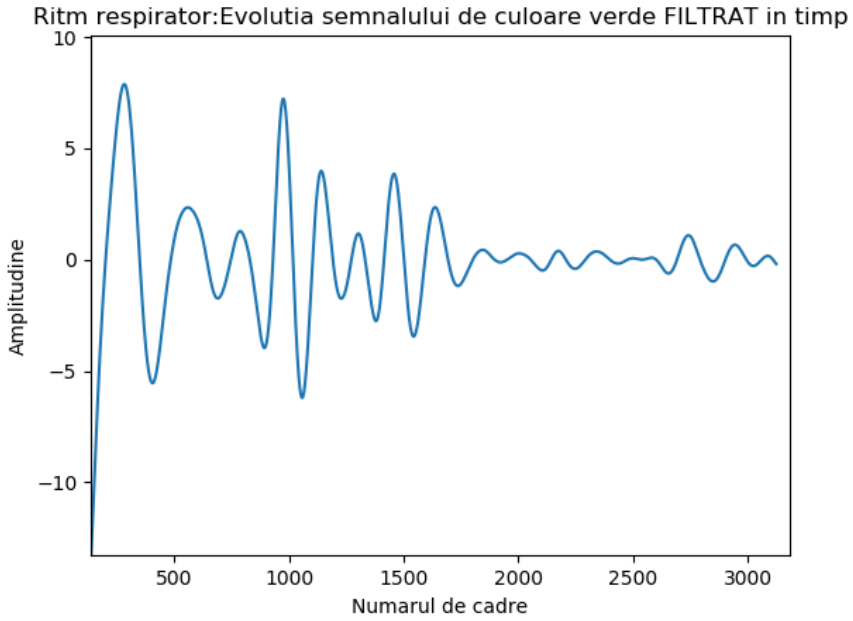

Figura 4.1.Evoluția semnalului ce reprezintă respirația, in timp, pentru fs=60Hz

In Figura 4.1 se pot observa 16 vârfuri ale semnalului in perioada de 3000 cadre = 50 secunde. Deci, 19.2 bătăi ale inimii pe minut(Valoare apropiată de 30).

Comparativ cu rezultatele din Tabelul 4, lucrările publicate au avut metode diferite de măsură pentru ritmul cardiac. Multe dintre acestea au măsurat doar frecvența dominantă a întregului semnal, comparând-o cu rezultatele date de alte dispozitive de măsură.

Lucrarea [13] prezintă comparația algoritmului cu un oximetru, in cazul măsurării frecvenței dominante pentru 3 subiecți. Rezultatele arată rata de eroare între 5% și 8 %.

În [14] este prezentată eroarea în cazul măsurării a 10 subiecți, pentru algoritmul în timp real și pentru algoritmul offline. Pentru această lucrare, eroarea pentru media pulsului este între 24% și 2.8%, o eroare medie fiind în jur de 8%. Autorii [14] prezintă și deviația standard pentru cele 10 măsurători, această fiind între 3 și 7 bpm pentru cele 10 măsurători. Prin comparație, deviația standard a algoritmului implementat în această lucrare este între 9 și 2.14, valoarea medie a deviației standard fiind în jur de 5bpm.

4.2. Algoritmul de determinare a ritmului cardiac ajutat de Amplificarea Euleriană Video

In cazul acestor măsurători,folosindu-se codul prezentat in [A6], s-au ales 3 videoclipuri de durată medie din Tabelul 4, urmând ca apoi algoritmul de detecție a ritmului cardiac sa fie aplicat pe imaginile nou create. Rezultatele sunt prezentate in Tabelul 4.2:

Video	Ritm cardiac mediu înainte de aplicarea EVM	Ritm cardiac mediu după aplicarea EVM
Face.mp4	51.73	51.73
6	78.04	76.46
2	74.21	75.6

Tabelul 4.2. Comparație între măsurătorile înainte si după aplicarea EVM

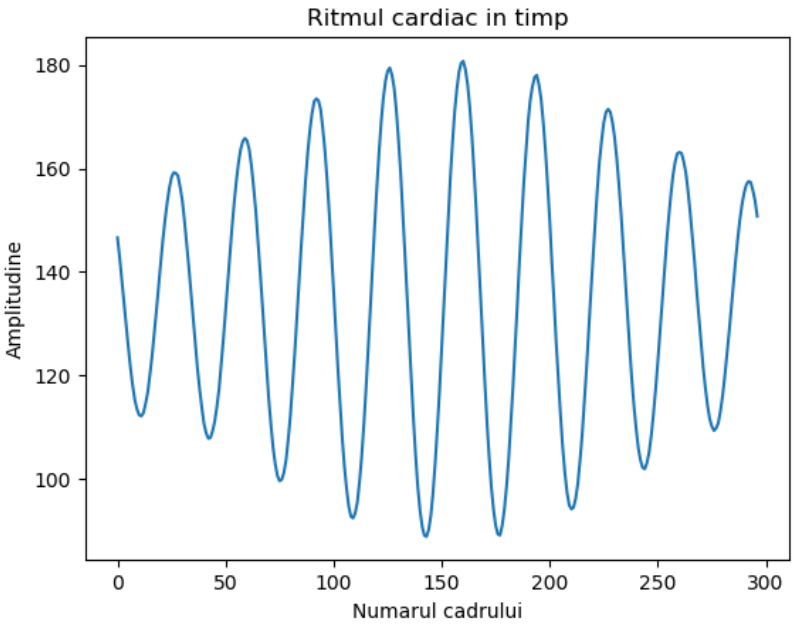

Figura 4.2. Evoluția amplitudinii in timp in cazul aplicării EVM pe filmul face.mp4

Se poate observa ca deși nu exista o imbunătățire in cazul aplicării EVM pe faimosul video folosit de [6], semnalul rezultat din EVM pare a fi curat, lipsit de zgomot, putându-se citi frecventa cu ușurința.

In cazul Figurii 4.3,se poate vedea de asemenea îmbunătățirea raportului semnal/zgomot:

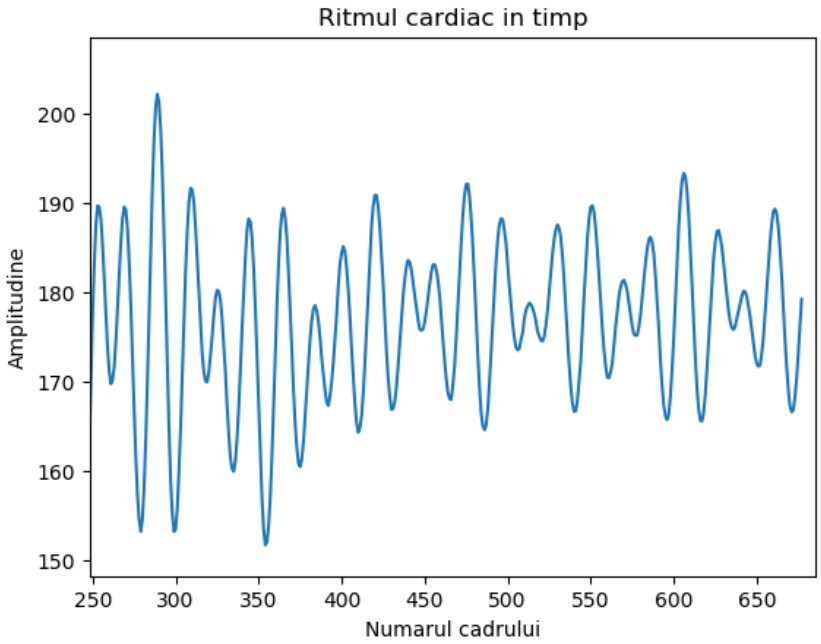

Figura 4.3. Evoluția amplitudinii înainte de filtrarea semnalului imbunătățit prin aplicarea EVM

În cazul video-ului cu numărul 2, se poate observa cea mai mică precizie pentru metodă testată în Capitolul 4.1. După aplicarea EVM, se poate observa o creștere a preciziei de 1%. Acest lucru dovedește că deși metoda EVM aduce îmbunătățiri, această nu poate repara alte condiții de eroare cum ar fi lumina variabilă sau schimbarea deasă a poziției zonei de interes.

Lucrarea [16] prezintă folosirea EVM în plus față de metodele tradiționale. Eroarea în cazul [16] este între 12.07% și 0.49%, cu o medie în jurul valorii de 3.13%.

Se poate observa că folosirea EVM pentru extragerea pulsului aduce un plus de acuratețe, deoarece raportul semnal/zgomot este mărit.

În final, deoarece amplificarea mișcării nu poate fi cuantificată, nu se pot prezenta rezultate pentru această metodă de folosire a EVM. Însă, o demonstrație video arată importantă amplificării mișcării pentru a vedea detalii aparent invizibile pentru ochiul uman.

Metoda EVM obține rezultate fascinante care pot ajută în detectarea unor schimbări în videoclipuri, dar nu este o ce poate fi aplicată de sine-stătătoare, această bazându-se foarte mult pe alegerea parametrilor de amplificare (alpha) și numărului de nivele ale piramidei Gausiene construite.

4.3. Folosirea senzorului IR al Microsoft Kinect pentru citirea pulsului

Pentru a testa daca folosirea senzorului de lumină IR al Microsoft Kinect aduce îmbunătățiri algoritmului de detectare a pulsului, videoclipurile #7 si #8 au fost filmate și cu ajutorul senzorului. Rezultatele algoritmului sunt prezente in Tabelul 4.3:

# Video	BPM mediu pentru semnalul RGB	BPM mediu pentru semnalul IR	Deviație standard pentru BPM al semnalului RGB	Deviație standard pentru BPM al semnalului IR
7	70.48	64.28	4.31	10.22
8	71.86	62.61	2.17	9.35

Tabelul 4.3.Comparație semnal RGB vs IR

Rezultatele din Tabelul 4.3 dovedesc ca deși ritmul cardiac poate fi citit de un senzor IR, această citire este mult mai imprecisă decât folosirea imaginilor RGB. Un avantaj, însa, pentru folosirea senzorului IR al Kinect v2 este capabilitatea de a fi folosit in întuneric.

Concluzii

Pulsul și respirația sunt procese fundamentale întâlnite în rândul tuturor oamenilor. Un aparat poligraf este capabil să extragă măsurători pentru puls și respirație, acestea putând fi indicatori pentru un posibil comportament disimulat, emoții, sau chiar și starea de sănătate a unei persoane. Deoarece metoda poligrafului implică contact fizic cu aparatul și cunoașterea prezenței acestuia, încrederea în rezultate nu poate fi niciodată deplină deoarece nu se poate știi dacă prezenta în sine a aparatului provoacă schimbări în rândul manifestărilor corpului uman.

Această lucrare a prezentat metode diferite de a obține valori pentru ritmul bătăilor inimii sau pentru ritmul respirator, și chiar și metode de a le vedea cu ochiul liber folosind EVM. Metodele folosite în alte lucrări științifice sunt comparate, iar o implementare pentru un algoritm de extragere a pulsului și a respirației folosind imagini video non-contact a fost creată în scopul detectării disimulării. Lucrarea propune și o propunere de abordare folosind învățarea automată pentru a detecta disimularea având că mărimi de intrare pulsul și respirația. O baza de date care definește un proces de disimulare prin intermediul filmărilor a fost creată, iar odată cu extinderea acesteia în viitor va putea face posibilă implementarea de învățare automată propusă.

Rezultatele algoritmilor ce detectează pulsul și respirația sunt comparabile cu cele din literatură, dovedind astfel eficiența algoritmilor. Implementarea algoritmului de amplificare Euleriană ar putea fi doar primul pas într-o serie de dezvoltări științifice care ar putea, în viitor, să dezvăluie nenumărate lucruri ce nu pot fi observate în acest moment de algoritmii de viziune computerizată.

În final, această lucrare a dovedit posibilitatea de extragere a semnelor vitale de la distanță, fără a fi nevoie de contact, din care experții si non-experții umani ar putea să extragă foarte ușor semnele disimulării. Lucrarea construiește o fundație pentru muncă în viitor și crearea unui algoritm binar de detectare a disimulării non-contact, folosind învățarea automată.

Bibliografie

[1] https://www.cvphysiology.com/Blood%20Pressure/BP002 – accesat ultima data 25.06.2019

[2] K. Shelley and S. Shelley, Pulse Oximeter Waveform: Photoelectric Plethysmography, in Clinical Monitoring, Carol Lake, R. Hines, and C. Blitt, Eds.: W.B. Saunders Company, 2001, pp. 420-428.

[3] https://www.tomsguide.com/us/heart-rate-monitor,review-2885.html – accesat ultima date 25.06.2019

[4] Lindh WQ, Pooler M, Tamparo C, Dahl BM (9 March 2009). Delmar's Comprehensive Medical Assisting: Administrative and Clinical Competencies. Cengage Learning. p. 573. ISBN 978-1-4354-1914-8.

[5] Rodríguez-Molinero A, Narvaiza L, Ruiz J, Gálvez-Barrón C (December 2013). "Normal respiratory rate and peripheral blood oxygen saturation in the elderly population". Journal of the American Geriatrics Society. 61 (12): 2238–40. doi:10.1111/jgs.12580. PMID 24329828.

[6] Wu, Hao-Yu, Michael Rubinstein, Eugene Shih, John Guttag, Frédo Durand, and William Freeman. "Eulerian Video Magnification for Revealing Subtle Changes in the World." ACM Transactions on Graphics 31, no. 4 (July 1, 2012): 1–8

[7] "Monitor on Psychology – The polygraph in doubt". American Psychological Association. July 2004. Retrieved 2008-02-29.

[8] Berntson GG, Cacioppo JT, Quigley KS (March 1993). "Respiratory sinus arrhythmia: autonomic origins, physiological mechanisms, and psychophysiological implications". Psychophysiology. 30 (2): 183–96. doi:10.1111/j.1469-8986.1993.tb01731.x. PMID 8434081.

[9] http://iworx.com/documents/IX-ELVIS_labexercises/Breathing-HeartRate-LS2_ELVIS.pdf – accesat ultima data 26.06.2019

[10] Hao-Yu Wu, Michael Rubinstein, Eugene Shih, John Guttag, Frédo Durand, and William T. Freeman. Eulerian video magnification for revealing subtle changes in the world. ACM Trans. Graph. (Proceedings SIGGRAPH 2012),2012.

[11] G. Tabak and A.C. Singer. Non-contact heart rate detection via periodic signal detection methods. 2015 49th Asilomar Conference on Signals, Systems and Computers, 2015.

[12] W. Verkruysse, L.O. Svaasand and J.S. Nelson. Remote plethysmographic imaging using ambient light. Opt Express, vol. 16, no. 26, 2008.

[13] https://esc.fnwi.uva.nl/thesis/centraal/files/f306804614.pdf : Heart rate and respiratory rate detection algorithm based on the Kinect for Windows v2 – Paul Hofland,2016, accesat ultima data 26.06.2019

[14] H. Rahman, M.U. Ahmed, S. Begum, P. Funk - Real Time Heart Rate Monitoring From Facial RGB Color Video Using Webcam, June 2016

[15] Domenico Grimaldi, Francesco Lamonaca, Yuriy Kurylyak - Smartphone-Based Photoplethysmogram Measurement

[16] Ennio Gambi , Angela Agostinelli, Alberto Belli, Laura Burattini , Enea Cippitelli, Sandro Fioretti, Paola Pierleoni, Manola Ricciuti, Agnese Sbrollini ID and Susanna Spinsante - Heart Rate Detection Using Microsoft Kinect: Validation and Comparison to Wearable Devices, 2017

[17] P. Viola and M. Jones, "Robust real-time face detection," in Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on, 2001, pp. 747-747

[18] Poh, Ming-Zher, Daniel J. McDuff, and Rosalind W. Picard. “Noncontact, automated cardiac pulse measurements using video imaging and blind source separation.” Optics Express 18 (2010): 10762. ©2011 Optical Society of America.

[19] Aleš Procházka , Martin Schätz , Oldˇrich Vyšata and Martin Vališ - Microsoft Kinect Visual and Depth Sensors for Breathing and Heart Rate Analysis

[20] Wim Verkruysse, Lars O Svaasand, and J Stuart Nelson. Remote plethysmographic imaging using ambient light. Optics express, 16(26):21434–21445, 2008.

[21] P. Comon. Independent component analysis, a new concept? Signal processing, 36(3):287–314, 1994.

[22] M.Z. Poh, D.J. McDuff, and R.W. Picard. Advancements in noncontact, multiparameter physiological measurements using a webcam. Biomedical Engineering, IEEE Transactions on, 58(1):7–11, 2011.

[23] C. Liu, A. Torralba, W.T. Freeman, F. Durand, and E.H. Adelson. Motion magnification. In ACM Transactions on Graphics (TOG), volume 24, pages 519–526. ACM, 2005.

[24] https://www.youtube.com/watch?v=ONZcjs1Pjmk , accesat ultima data pe 27.06.2019

[25] https://www.mathworks.com/products/matlab.html, accesat ultima data 27.06.2019

[26] https://www.numpy.org/, accesat ultima data pe 27.06.2019

[27] https://www.scipy.org/, accesat ultima data pe 27.06.2019

[28] https://pypi.org/project/opencv-python/, accesat ultima data pe 27.06.2019

[29] https://www.mcvuk.com/next-xbox-leak-reveals-kinect-2-specs/, accesat ultima data pe 27.06.2019

[30] Saykrs BM. Analysis of Heart Rate Variability. Ergonomics 1973;16(1):17-32.

[31] Kaveh Bakhtiyari, Nils Beckmann, Jürgen Ziegler. Contactless heart rate variability measurement by IR and 3D depth sensors with respiratory sinus arrhythmia

[32] https://www.cg.tuwien.ac.at/courses/Visualisierung2/HallOfFame/2018/Wu2012/html/index.html, accesat ultima data 28.06.2019

[33]Jyoti Soni, Ujma Ansari ,Dipesh Sharma - Predictive Data Mining for Medical Diagnosis: An Overview of Heart Disease Prediction, Martie 2011

[34] Tülay KarayÕlan, Özkan KÕlÕç - Prediction of Heart Disease Using Neural Network

[35] R karthik, Dhruv Tyagi, Amogh Raut, Soumya Saxena, Rajesh Kumar M- Implementation of Neural Network and feature extraction to classify ECG signals, 2014

Anexa

[A1] Implementarea algoritmului de detectie a pulsului si ritmului respirator: Pulse_from_video.py

```
# -*- coding: utf-8 -*-
"""
Created on Wed May 22 11:58:26 2019

@author: Cristi
"""

import numpy as np
import cv2
import matplotlib.pyplot as plt
from scipy.signal import butter, lfilter, cheby1, freqz, iirfilter, find_peaks, ellip




# Bandpass filter
def butter_bandpass(lowcut, highcut, fs, order=5):
    nyq = 0.5 * fs
    low = lowcut / nyq
    high = highcut / nyq
    b, a = butter(order, [low, high], btype='band')
    return b, a


def butter_bandpass_filter(data, lowcut, highcut, fs, order=5):
    b, a = butter_bandpass(lowcut, highcut, fs, order=order)
    y = lfilter(b, a, data)
    return y

 # Eliptic bandpass filter
# def ellip_bandpass(fs, lowcut, highcut, rp, rs, order=1):
#    b, a = ellip(order, rp, rs, [lowcut/(fs/2), highcut/(fs/2)], btype='bandpass')
#    return b, a
#
#
# def ellip_bandpass_filter(data, fs, lowcut, highcut, rp, rs,  order=1):
#    b, a = ellip_bandpass(fs, lowcut, highcut, rp, rs, order)
#    y = lfilter(b, a, data)
#    return y



def get_frequency(array, count_peaks=False ):

    #Windowing the signal for better fft accuracy
    #array = array * np.blackman(len(array))
    # Preinitialising the filter
    array = np.array(list(np.ones(200)*np.mean(array[0:int(len(array)/3)])) + list(array)  )


    filtered_array = butter_bandpass_filter(array, lowcut_freq, highcut_freq, fs=fs, order=4)
```

```
    #filtered_array = butter_bandpass_filter(filtered_array, lowcut_freq, highcut_freq, fs=fs, order=1)


    #Zero padding enables you to obtain more accurate amplitude estimates of resolvable signal
components
    #adding as many zeros as len of the array
    filtered_array = list(filtered_array[200::]) + list(np.zeros(len(filtered_array)-200))



    array_fft = abs(np.fft.fft(filtered_array))
    #max index, but only from the first half
    max_index = np.argmax(abs(array_fft[0:int((len(filtered_array) - 1 +
len(np.zeros(len(filtered_array)-200))) / 2)]))

    # T = 1/fs
    xf = np.linspace(0.0, 1.0 / (T), len(array_fft))


#    plt.figure()
#    plt.plot(filtered_array)
#    plt.show()
#    plt.title('ord2+1')



    if not count_peaks:

        return xf[max_index]

    if count_peaks:

        peaks_locations = find_peaks(filtered_array)[0]
        before = -10000
        correct_peaks = 0
        for i in peaks_locations:
            if i - before >= 12:
                correct_peaks += 1
                before = i
            else:
                before=before

        return fs / ((peaks_locations[-1]-peaks_locations[0]) / correct_peaks)

if __name__=="__main__":

################################################################################################################
    "Changeable parameters"
    #choosing the face cascade
    face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default2.xml')
    use_live = False
    video_adress = r'C:\Users\Cristi\Desktop\Facultate\Licenta\Cod\videos\MVI_0072.mov'
    lowcut_freq = 1.15
    highcut_freq = 2
    choose_colour = 1  # BGR image, so 1=Green
    signal_window = 10 # seconds - Heart rate signal moving window
    read_rate = 1 # frames - How often will the bulse be read

    # For accuracy of face detection
    scaleFactor = 1.15
    minNeighbors = 10
```

```
##########################################################################################################



    if use_live:

        cap = cv2.VideoCapture(0)
        cap.set(cv2.CAP_PROP_FPS, 16)

    else:

        cap = cv2.VideoCapture(video_adress)


    fs = int(cap.get(5))
    print(fs)
    T = 1/fs
    f_square_width_vector = []
    frame_nr = 0
    first_10 = True
    ret = True
    colour_vector = []
    total_frames = 0
    all_bpm = []
    nr_of_window_samples = int(signal_window * fs)
    first = True
    max_nr_of_pixels = 200000
    #face_frames = []
    while ret:

        ret, img_init = cap.read()
        if ret==False:
            break

        if first:
            nr_of_pixels = img_init.shape[0]*img_init.shape[1]
            print(nr_of_pixels)
            if nr_of_pixels > max_nr_of_pixels:
                resize_factor = max_nr_of_pixels/nr_of_pixels
            else:
                resize_factor = 1
            first=False

        img = cv2.resize(img_init, None, fx=resize_factor, fy=resize_factor)

        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        faces = face_cascade.detectMultiScale(gray, scaleFactor, minNeighbors)



        if len(faces) == 0:
            cv2.imshow('img', img)
            k = cv2.waitKey(30) & 0xff
            if k == 27:
                break
            continue

        if not first_10:
            if len(faces) == 1:
                total_frames += 1
```

```
        # Using the first 10 for rectangle.
        if len(faces) == 1:
            frame_nr += 1
            f_square_width = int(faces[0][2] / 4)
            #f_square_width = faces[0][2] / 4
            if frame_nr == 10:
                first_10 = False


            for (x, y, w, h) in faces:
                roi_gray = cv2.resize(gray[y:y + h, x:x + w], (96,96), interpolation = cv2.INTER_CUBIC)
                roi_color = cv2.resize(img[y:y + h, x:x + w], (96,96), interpolation = cv2.INTER_CUBIC)
                #face_frames.append(roi_color)
                cv2.rectangle(img, (x, y), (x + w, y + h), (255, 255, 0), 2)


            if not first_10:
                forehead = [
                    [x + int(w / 2 - int(f_square_width / 2)), y + int(h / 12), f_square_width,
int(f_square_width / 2)]]

                forehead_square_pixels = 1
                for (a, b, c, d) in forehead:
                    cv2.rectangle(img, (a, b), (a + c, b + d), (255, 0, 0), forehead_square_pixels)

                colour_values = img[forehead[0][1] + forehead_square_pixels:forehead[0][1] +
forehead[0][
                    3] - forehead_square_pixels,
                                forehead[0][0] + forehead_square_pixels:forehead[0][0] + forehead[0][
                                    2] - forehead_square_pixels,
                                choose_colour]

                f_colour = np.mean(colour_values)
                colour_vector.append(f_colour)

            if total_frames > nr_of_window_samples + 20 and total_frames % read_rate == 0:
                #heart_beat_frequency = get_frequency(colour_vector[-nr_of_window_samples:])

                total_freqs = 0
                for i in range(1, 21):
                    total_freqs += get_frequency(colour_vector[-(nr_of_window_samples + i):-abs(i)])

                print('BPM:', end='')
                print(total_freqs / 20 * 60,end=' Frame:')
                print(total_frames)

                all_bpm.append(total_freqs / 20 * 60)


        cv2.imshow('img', img)
        k = cv2.waitKey(30) & 0xff
        if k == 27:
            break

    cv2.destroyAllWindows()
    cap.release()


    print('Finished live heart rate analysis')
```

```
    ###############################################################################
    # Further analysis for the full signal

    colour_vector = np.array(list(np.ones(250) * np.mean(colour_vector[0:int(len(colour_vector)/10)]))
+ list(colour_vector))

    plt.figure()
    plt.plot(colour_vector)
    plt.show()
    plt.title('Evolutia semnalului de culoare verde in timp')
    plt.xlabel('Numarul de cadre')
    plt.ylabel('Amplitudine')

    lowcut_freq = 1
    highcut_freq = 2


    xf = np.linspace(0.0, 1.0 / (T), len(colour_vector)-250)



    filtered_colour_vector = butter_bandpass_filter(colour_vector, lowcut_freq, highcut_freq, fs=fs,
order=4)
    #filtered_colour_vector = butter_bandpass_filter(filtered_colour_vector, lowcut_freq, highcut_freq,
fs=fs, order=1)

    filtered_colour_vector = filtered_colour_vector[250::]

    colour_vector_fft = abs(np.fft.fft(filtered_colour_vector))

    max_index = np.argmax(abs(colour_vector_fft[0:int((total_frames - 1) / 2)]))


    print('Freq:', end='')
    print(xf[max_index])

    print('RPM: ', end='')
    print(xf[max_index] * 60)

    plt.figure()
    plt.plot(xf, colour_vector_fft)
    plt.show()
    plt.xlabel('Frecventa[Hz]')
    plt.ylabel('Amplitudine')
    plt.title('Reprezentarea in frecventa a semnalului de interes ')

    plt.figure()
    plt.plot(filtered_colour_vector)
    plt.show()
    plt.title('Evolutia semnalului de culoare verde FILTRAT in timp')
    plt.xlabel('Numarul de cadre')
    plt.ylabel('Amplitudine')




    ###############################################################################
    # For breathing measurement:

    #colour_vector = colour_vector[250::]
    # adaugarea a unui semnal dummy mai mare
    colour_vector = np.array(list(np.ones(400) * np.mean(colour_vector[250:int(len(colour_vector)/5)]))
+ list(colour_vector))
```

```
    plt.figure()
    plt.plot(colour_vector)
    plt.show()
    plt.title('Ritm respirator:Evolutia semnalului de culoare verde in timp')
    plt.xlabel('Numarul de cadre')
    plt.ylabel('Amplitudine')

    lowcut_freq = 0.2
    highcut_freq = 0.4


    xf = np.linspace(0.0, 1.0 / (T), len(colour_vector)-250)



    filtered_colour_vector = butter_bandpass_filter(colour_vector, lowcut_freq, highcut_freq, fs=fs,
order=3)
    #filtered_colour_vector = butter_bandpass_filter(filtered_colour_vector, lowcut_freq, highcut_freq,
fs=fs, order=1)

    filtered_colour_vector = filtered_colour_vector[250::]

    colour_vector_fft = abs(np.fft.fft(filtered_colour_vector))

    max_index = np.argmax(abs(colour_vector_fft[0:int((total_frames - 1) / 2)]))


    print('Freq:', end='')
    print(xf[max_index])

    print('RPM: ', end='')
    print(xf[max_index] * 60)

    plt.figure()
    plt.plot(xf, colour_vector_fft)
    plt.show()
    plt.xlabel('Frecventa[Hz]')
    plt.ylabel('Amplitudine')
    plt.title('Ritm respirator:Reprezentarea in frecventa a semnalului de interes ')

    plt.figure()
    plt.plot(filtered_colour_vector)
    plt.show()
    plt.title('Ritm respirator:Evolutia semnalului de culoare verde FILTRAT in timp')
    plt.xlabel('Numarul de cadre')
    plt.ylabel('Amplitudine')
```

[A2] Implementarea codului ce folosire a algoritmului EVM: evm4video2

```
# -*- coding: utf-8 -*-
"""
Created on Mon Jun 24 14:13:56 2019

@author: Cristi
"""

# -*- coding: utf-8 -*-
"""
Created on Mon Jun 24 02:16:34 2019

@author: Cristi
"""

from video_functions2 import *
import scipy.fftpack as fftpack
import numpy as np
import cv2


def evm4video(video_frames, alpha,  lowcut_freq, highcut_freq, chrom_attenuation=None , fs=30,
levels=4):



    " Returns colour magnified video after creating a fft from the whole movie"
    "(pixel by pixel), applying an ideal filter to select only the frequencies of inetrest"
    "and adding the last level gaussian pyramid element"



    #  from rgb to yiq
    video_frames = rgb2yiq(video_frames) #luminance (Y) and chrominance (I and Q)


    # build Gaussian pyramid and use the highest level
    high_gauss_vid = gaussian_video(video_frames, levels)

    for level_vid in high_gauss_vid:


        # apply fft
        fft = fftpack.rfft(level_vid, axis=0)
        frequencies = fftpack.rfftfreq(fft.shape[0], d=1.0 / fs)  # sample frequencies
        bp_filter = np.logical_and(frequencies > lowcut_freq, frequencies < highcut_freq)  # logical
array if values between low and high frequencies

        fft[~bp_filter] = 0                # cutoff values outside the bandpass

        filtered = fftpack.irfft(fft, axis=0)   # inverse fourier transformation

        filtered *= alpha  # magnification
```

```
        # chromatic attenuation
        if chrom_attenuation is not None:
            filtered[:,:,:,1] = filtered[:,:,:,1] * chrom_attenuation
            filtered[:,:,:,2] = filtered[:,:,:,2] * chrom_attenuation

        # resize last gaussian level to the frames size
        filtered_video_list = np.zeros(video_frames.shape)  ###!!!!!!!!!!!!!!!!!!!!!!!!E bine aici?
        for i in range(len(video_frames)):
            f = filtered[i]
            filtered_video_list[i] = cv2.resize(f, (video_frames.shape[2], video_frames.shape[1]))

        final = filtered_video_list/len(high_gauss_vid)

        # Add to original
        video_frames = final + video_frames



    # from yiq to rgb
    video_frames = yiq2rgb(video_frames)

    # Cutoff wrong values
    video_frames[video_frames < 0] = 0
    video_frames[video_frames > 255] = 255

    return video_frames




if __name__=="__main__":

    video_path = r'C:\Users\Cristi\Desktop\Facultate\Licenta\Cod\90-87-87-85-87-89-86-85-86.mp4'
    face_frames , fs = load_video(video_path, save_face=True, only_face_frames = True)
    #full_video = video_resize(full_video, 4)

    evm_video = evm4video(face_frames, alpha=40,  lowcut_freq=1.25, highcut_freq=1.6, fs=fs, levels=6)


    #import time

    for frame in evm_video:

        frame = bgr2rgb(frame)
        frame = cv2.resize(frame,(400,400))

        cv2.imshow('img', frame/255)
        k = cv2.waitKey(30) & 0xff
        if k == 27:
            break


    for frame in full_video:

        frame = frame[:,:,[2,1,0]]
        frame = cv2.resize(frame,(400,400))
        cv2.imshow('img', frame)
        k = cv2.waitKey(30) & 0xff
        if k == 27:
            break
```

[A3] Implementarea unor functii ce ajuta la manuirea fisierelor video si realizarea EVM: video_functions2.py

```
# -*- coding: utf-8 -*-
"""
Created on Mon Jun 24 01:25:22 2019

@author: Cristi
"""

import numpy as np
import cv2
from copy import copy
from scipy.signal import convolve2d


def load_video(video_path, save_face=False, roi_dim = (96,96) , only_face_frames = False):


    cap = cv2.VideoCapture(video_path)
    fps = cap.get(cv2.CAP_PROP_FPS)

    if save_face:
        face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default2.xml')



    full_video = []
    face_frames = []
    ret = True
    while ret:


        ret, frame_bgr = cap.read()



        if not ret:
            break

        frame = cv2.cvtColor(frame_bgr, cv2.COLOR_BGR2RGB)

#        frame = frame_bgr.copy()
#        temp = None
#        temp = frame[:,:,0]
#        frame[:,:,0] = frame[:,:,2]
#        frame[:,:,2] = temp

        if save_face:
            gray = cv2.cvtColor(frame_bgr, cv2.COLOR_RGB2GRAY)
            faces = face_cascade.detectMultiScale(gray, 1.05, 7)
```

```
            if len(faces)==1:
                for (x, y, w, h) in faces:
                   #roi_color = frame[y:y + h, x:x + w]
                    roi_color = cv2.resize(frame[y:y + h, x:x + w], roi_dim, interpolation =
cv2.INTER_CUBIC)
                    face_frames.append(roi_color)

        full_video.append(frame)

    print('Video loaded')

    if only_face_frames:
        return np.array(face_frames) , fps
    else:
        if not save_face:
            return np.array(full_video) , fps
        if save_face:
            return np.array(full_video) , np.array(face_frames) , fps


def video_resize(video, factor):

    return video[:,::factor,::factor,:]



def get_kernel(dim=None, sigma=None):

    if dim is not None and sigma is not None:

        kernel = np.zeros((dim, dim))

        for y in range(dim):
            for x in range(dim):

                kernel[x][y] = np.exp(-(abs(y-int(dim/2))**2+abs(x-
int(dim/2))**2)/(2*sigma**2))/(2*np.pi*sigma**2)

        return  kernel

    else:
        # Default kernel
        kernel = 1/273 * np.array([[1,4,7,4,1] ,
                                   [4,16,26,16,4] ,
                                   [7,26,41,26,7] ,
                                   [4,16,26,16,4] ,
                                   [1,4,7,4,1]])

    return kernel



def gaussian_pyramid(image, levels, kernel=None):

    if kernel == None:
        kernel = get_kernel()


    gauss = image.copy()
    gauss_pyr = [gauss]

    for level in range(1, levels):
        for channel in range(3):
            gauss[:,:,channel] = convolve2d(gauss[:,:,channel], kernel, 'same')
```

```
        gauss = gauss[::2,::2,:]
        gauss_pyr.append(gauss)

    return gauss_pyr



def gaussian_video(video, levels, kernel=None):



    levels_vid_data = []
    for level in range(1,levels):
        for frame_nr in range(0, len(video)):
            frame = video[frame_nr]
            pyr = gaussian_pyramid(frame, levels, kernel)
            gaussian_frame = pyr[level]  # use only highest gaussian level is used
            if frame_nr == 0:                # initialize for the first time
                vid_gauss_data = np.zeros((len(video), gaussian_frame.shape[0],
gaussian_frame.shape[1], 3))

            vid_gauss_data[frame_nr] = gaussian_frame

        levels_vid_data.append(vid_gauss_data)

    return levels_vid_data


def rgb2yiq(video):
    '''
    Converts the video color from RGB to YIQ (NTSC)

    '''
    yiq_from_rgb = np.array([[0.299, 0.587, 0.114],
                             [0.596, -0.274, -0.322],
                             [0.211, -0.523, 0.312]])
    t = np.dot(video, yiq_from_rgb.T)
    return t


def yiq2rgb(video):

    "Converts the video color from YIQ (NTSC) to RGB"


    rgb_from_yiq = np.array([[1, 0.956, 0.621],
                             [1, -0.272, -0.647],
                             [1, -1.106, 1.703]])
    t = np.dot(video, rgb_from_yiq.T)
    return t


def bgr2rgb(frame):

    return frame[:,:,[2,1,0]]


if __name__ == "__main__":

    img = cv2.imread(r'C:\Users\Cristi\Desktop\Facultate\Licenta\Cod\test.jpg', cv2.IMREAD_COLOR)

    pyr = gaussian_pyramid(img,levels=5)
```

[A4] Implementarea/adaptarea codului pentru filmarea RGB si salvarea cadrelor cu ajutorul senzorului Kinect v2:
Kinect_film_colour_frames.py

```
from pykinect2 import PyKinectV2
from pykinect2.PyKinectV2 import *
from pykinect2 import PyKinectRuntime

import ctypes
import _ctypes
import pygame
import sys



class BodyGameRuntime(object):
    def __init__(self):
        pygame.init()

        # Used to manage how fast the screen updates
        self._clock = pygame.time.Clock()

        # Set the width and height of the screen [width, height]
        self._infoObject = pygame.display.Info()
        self._screen = pygame.display.set_mode((self._infoObject.current_w >> 1,
self._infoObject.current_h >> 1),
                                               pygame.HWSURFACE|pygame.DOUBLEBUF, 32)

        pygame.display.set_caption("Kinect for Windows v2 Color Frames")

        # Loop until the user clicks the close button.
        self._done = False

        # Kinect runtime object, we want only color and body frames
        self._kinect = PyKinectRuntime.PyKinectRuntime(PyKinectV2.FrameSourceTypes_Color )

        # back buffer surface for getting Kinect color frames, 32bit color, width and height equal to
the Kinect color frame size
        self._frame_surface = pygame.Surface((1920, 1080), 0, 32)


        # here we will store the color frames
        self._frames = []



#    def draw_color_frame(self, frame, target_surface):
#        target_surface.lock()
#        address = self._kinect.surface_as_array(target_surface.get_buffer())
#        ctypes.memmove(address, frame.ctypes.data, frame.size)
#        del address
```

```
#        target_surface.unlock()

    def run(self):
        # -------- Main Program Loop -----------
        while not self._done:
            # --- Main event loop
            for event in pygame.event.get(): # User did something
                if event.type == pygame.QUIT: # If user clicked close
                    self._done = True # Flag that we are done so we exit this loop



            # --- Woohoo! We've got a color frame! Let's fill out back buffer surface with frame's data
            if self._kinect.has_new_color_frame():
                frame = self._kinect.get_last_color_frame()
                #self.draw_color_frame(frame, self._frame_surface)
                self._frames.append(frame)
                frame = None



            # --- copy back buffer surface pixels to the screen, resize it if needed and keep aspect
ratio
            # --- (screen size may be different from Kinect's color frame size)
            h_to_w = float(self._frame_surface.get_height()) / self._frame_surface.get_width()
            target_height = int(h_to_w * self._screen.get_width())
            surface_to_draw = pygame.transform.scale(self._frame_surface, (2000, 2500));
            self._screen.blit(surface_to_draw, (0,0))
            surface_to_draw = None
            pygame.display.update()

            # --- Go ahead and update the screen with what we've drawn.
            pygame.display.flip()

            # --- Limit to 15 frames per second
            self._clock.tick(15)

        # Close our Kinect sensor, close the window and quit.
        self._kinect.close()
        pygame.quit()



__main__ = "Kinect v2 Body Game"
game = BodyGameRuntime();
game.run();
#
#print(len(game._frames))
```

[A5] Implementarea/ Adaptarea codului pentru a realiza filmari IR cu senzorul Kinect V2: kinect_film_ir_frames.py

```
from pykinect2 import PyKinectV2
```

```
from pykinect2.PyKinectV2 import *
from pykinect2 import PyKinectRuntime

import matplotlib.pyplot as plt
import matplotlib.image as mpimg

import ctypes
import _ctypes
import pygame
import sys
import numpy as np
import os
import cv2


class InfraRedRuntime(object):
    def __init__(self):
        pygame.init()

        # Used to manage how fast the screen updates
        self._clock = pygame.time.Clock()

        # Loop until the user clicks the close button.
        self._done = False

        # Kinect runtime object, we want infrared frames and colour frames
        self._kinect = PyKinectRuntime.PyKinectRuntime(PyKinectV2.FrameSourceTypes_Infrared)
        #self._kinect1 = PyKinectRuntime.PyKinectRuntime(PyKinectV2.FrameSourceTypes_Color)  #?????????

        # back buffer surface for getting Kinect infrared frames, 8bit grey, width and height equal to
the Kinect ir frame size
        # width, height, flags, depth
        self._frame_surface = pygame.Surface((self._kinect.infrared_frame_desc.Width,
self._kinect.infrared_frame_desc.Height), 0, 24)

        # Set the width and height of the screen [width, height]
        self._infoObject = pygame.display.Info()
        self._screen = pygame.display.set_mode((self._kinect.infrared_frame_desc.Width,
self._kinect.infrared_frame_desc.Height),
                                               pygame.HWSURFACE, 32)


        pygame.display.set_caption("Kinect for Windows v2 Infrared")

        # Here I will save all the frames
        self._frames =[]

        self.save_dir = r'C:\Users\Cristi\Desktop\Facultate\Licenta\Cod\IR_frames'




    def draw_infrared_frame(self, frame, target_surface):
        if frame is None:  # some usb hub do not provide the infrared image. it works with Kinect
studio though
            return
        target_surface.lock()
        f8=np.uint8(frame.clip(1,13000)/80.)
        frame8bit=np.dstack((f8,f8,f8))
        address = self._kinect.surface_as_array(target_surface.get_buffer())
        ctypes.memmove(address, frame8bit.ctypes.data, frame8bit.size)
        del address
        target_surface.unlock()
```

```
    def run(self):
        # -------- Main Program Loop -----------
        while not self._done:
            # --- Main event loop
            for event in pygame.event.get(): # User did something
                if event.type == pygame.QUIT: # If user clicked close
                    self._done = True # Flag that we are done so we exit this loop



            # --- Getting frames and drawing
            if self._kinect.has_new_infrared_frame():
                frame = self._kinect.get_last_infrared_frame()
                self.draw_infrared_frame(frame, self._frame_surface)
                self._frames.append(frame)
                frame = None

            self._screen.blit(self._frame_surface, (0,0))
            pygame.display.update()

            # --- Go ahead and update the screen with what we've drawn.
            pygame.display.flip()

            # --- Limit to 60 frames per second
            self._clock.tick(30)

        # Close our Kinect sensor, close the window and quit.
        self._kinect.close()
        pygame.quit()


    def save_frames(self, dir_name):

        os.chdir(self.save_dir)
        os.mkdir(dir_name)
        os.chdir(self.save_dir + '\\' + str(dir_name) )
        i=-1
        for frame in self._frames:
            i+=1
            frame = (frame.reshape(424,512)/(2**16)*255).astype(int)
            cv2.imwrite(str(i) + '.jpg', frame)

        np.savetxt("raw_ir_data.csv", self._frames, delimiter=",")




__main__ = "Kinect v2 InfraRed"
game =InfraRedRuntime();
game.run();
game.save_frames('test4_ir')
#
#print(game._frames)
##resolution = 512*424
#
#plt.imshow(game._frames[0].reshape(424,512)/np.max(game._frames[0]))
```

[A6] Implementarea codului folosit pentru extragerea pulsului din cadrele pe care a fost aplicat EV care prezinta deja fata:

```
# -*- coding: utf-8 -*-
"""
Created on Fri Jun 28 18:56:30 2019

@author: Cristi
"""
import cv2
import numpy as np
import matplotlib.pyplot as plt
from scipy.signal import butter, lfilter, cheby1, freqz, iirfilter, find_peaks, ellip


# Bandpass filter
def butter_bandpass(lowcut, highcut, fs, order=5):
    nyq = 0.5 * fs
    low = lowcut / nyq
    high = highcut / nyq
    b, a = butter(order, [low, high], btype='band')
    return b, a


def butter_bandpass_filter(data, lowcut, highcut, fs, order=5):
    b, a = butter_bandpass(lowcut, highcut, fs, order=order)
    y = lfilter(b, a, data)
    return y

 # Eliptic bandpass filter
# def ellip_bandpass(fs, lowcut, highcut, rp, rs, order=1):
#    b, a = ellip(order, rp, rs, [lowcut/(fs/2), highcut/(fs/2)], btype='bandpass')
#    return b, a
#
#
# def ellip_bandpass_filter(data, fs, lowcut, highcut, rp, rs,  order=1):
#    b, a = ellip_bandpass(fs, lowcut, highcut, rp, rs, order)
#    y = lfilter(b, a, data)
#    return y



def get_frequency(array):

    #Windowing the signal for better fft accuracy
    #array = array * np.blackman(len(array))
    # Preinitialising the filter
    global lowcut_freq
    global highcut_freq
    global fs
    array = np.array(list(np.ones(200)*np.mean(array[0:int(len(array)/3)])) + list(array)  )


    filtered_array = butter_bandpass_filter(array, lowcut_freq, highcut_freq, fs=fs, order=4)
    #filtered_array = butter_bandpass_filter(filtered_array, lowcut_freq, highcut_freq, fs=fs, order=1)


    #Zero padding enables you to obtain more accurate amplitude estimates of resolvable signal
components
    #adding as many zeros as len of the array
```

```
    filtered_array = list(filtered_array[200::]) + list(np.zeros(len(filtered_array)-200))



    array_fft = abs(np.fft.fft(filtered_array))
    #max index, but only from the first half
    max_index = np.argmax(abs(array_fft[0:int((len(filtered_array) - 1 +
len(np.zeros(len(filtered_array)-200))) / 2)]))

    # T = 1/fs
    xf = np.linspace(0.0, 1.0 / (T), len(array_fft))


    return xf[max_index]




def print_pulse(video_array, fs=30, lowcut_freq = 1, highcut_freq = 2):

    choose_colour = 1
    colour_vector = []
    total_frames=0
    signal_window = 8
    nr_of_window_samples = int(signal_window * fs)
    all_bpm = []
    T = 1/fs
    for face_frame in video_array:


        total_frames+=1
        w = face_frame.shape[0]
        h = face_frame.shape[0]
        x=0
        y=0
        f_square_width = int(w/4)
        forehead = [[x + int(w / 2 - int(f_square_width / 2)),
                     y + int(h / 12), f_square_width,
                     int(f_square_width / 2)]]

        forehead_square_pixels = 1
        for (a, b, c, d) in forehead:
            cv2.rectangle(face_frame, (a, b), (a + c, b + d), (255, 0, 0), forehead_square_pixels)


        colour_values = face_frame[forehead[0][1] + forehead_square_pixels:forehead[0][1] +
forehead[0][3] - forehead_square_pixels,
                                   forehead[0][0] + forehead_square_pixels:forehead[0][0] + forehead[0][2] -
forehead_square_pixels,
                                   choose_colour]

        f_colour = np.mean(colour_values)
        colour_vector.append(f_colour)

        if total_frames > nr_of_window_samples + 20 :


            total_freqs = 0
            for i in range(1, 21):
                total_freqs += get_frequency(colour_vector[-(nr_of_window_samples + i):-abs(i)])

            print('BPM:', end='')
            print(total_freqs / 20 * 60,end=' Frame:')
            print(total_frames)
```

```
            all_bpm.append(total_freqs / 20 * 60)


        cv2.imshow('img', face_frame/255)
        k = cv2.waitKey(30) & 0xff
        if k == 27:
            break

    return all_bpm, colour_vector


if __name__=="__main__":
    lowcut_freq = 1.1
    highcut_freq = 1.9
    fs=60
    all_bpm, colour_vector = print_pulse(evm_video)
```