Universitatea “Politehnica” din București
Facultatea de Electronică, Telecomunicații și Tehnologia Informației

Sistem de manevrare a unui robot prin autentificare facială

Proiect de diplomă

prezentat ca cerință parțială pentru obținerea titlului de
Inginer în domeniul *Calculatoare şi Tehnologia Informaţiei*
programul de studii de licență *Ingineria Informaţiei*

Conducător(i) științific(i)
Ş. L. Dr. Ing. Anamaria RĂDOI
Prof. Dr. Ing. Corneliu BURILEANU

Absolvent
Alexandru-Marius Cristea

București
2019

Anexa 1

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

TEMA PROIECTULUI DE DIPLOMĂ
a studentului **CRISTEA L. Alexandru-Marius , 441A**

1. Titlul temei: Sistem de manevrare a unui robot prin autentificare facială

2. Descrierea contribuției originale a studentului (în afara părții de documentare) și specificații de proiectare:
Scopul lucrării de diplomă este dezvoltarea unui sistem de deplasare a unui robot KUKA prin urmărire în timp real, deplasarea realizându-se numai în cazul în care o persoană autorizată (administrator) este identificată. Acest sistem este util mai ales în situațiile în care robotul utilizat are o greutate considerabilă și mutarea acestuia este dificil de realizat, așa cum este cazul robotului KUKA (> 23 Kg).
Lucrarea va avea la bază două module și anume: (1) modulul de recunoaștere facială ce va integra concepte de prelucrări de imagini și clasificare, (2) modulul de deplasare a robotului ce va permite mișcarea acestuia în funcție de poziția curentă a persoanei autorizate. Modulul de recunoaștere facială presupune folosirea unor metode din domeniul inteligenței artificiale bazate pe analiza componentelor principale, descriptori Fisher sau construirea unor histograme de descriptori binari. Pe parcursul deplasării, robotul va trebui să efectueze ajustările necesare astfel încât persoana urmărită să fie permanent plasată în interiorul cadrului captat de camera video amplasată pe robotul KUKA. În momentul în care persoana părăsește cadrul sau se află la o distanță mai mare care să nu permită distingerea unor suficiente trăsături faciale, robotul se va opri din deplasare. În acest sens, studentul va evalua mai mulți algoritmi de recunoaștere facială în raport cu acuratețea acestora și timpul de procesare, urmând a determina care algoritm îndeplinește criteriile de funcționare în timp real. Algoritmii de recunoaștere și software-ul pentru deplasare vor fi implementați în Python și vor folosi librăria OpenCV.

3. Resurse folosite la dezvoltarea proiectului:
Limbaje de programare: Python; Biblioteci: NumPy, OpenCV, YouBotBase, YouBotManipulator; Dispozitive: robot KUKA, CPU

4. Proiectul se bazează pe cunoștințe dobîndite în principal la următoarele 3-4 discipline:
POO, DEPI, RFIA, TOP

5. Proprietatea intelectuală asupra proiectului aparține: U.P.B.

6. Data înregistrării temei: 2018-11-29 17:18:28

Conducător(i) lucrare,
Ș. L. Dr. Ing. Anamaria RĂDOI
semnătura: ..

Prof. Dr. Ing. Corneliu BURILEANU
semnătura:.....

Student,
semnătura:.....

Director departament,
Prof. dr. ing Sever PAȘCA
semnătura:.....

Decan,
Prof. dr. ing. Cristian NEGRESCU
semnătura:.....

Cod Validare: **0c667d352e**

Declarație de onestitate academică

Prin prezenta declar că lucrarea cu titlul " *Sistem de manevrare a unui robot prin autentificare facială*", 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 *Calculatoare și Tehnologia Informației*, programul de studii *Ingineria Informației (CTI – INF)* 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, *20.06.2019*

Absolvent *Alexandru-Marius Cristea*

(semnătură în original)

Cuprins

Listă figuri

Listă de abrevieri

FTP – File Transfer Protocol
GB – GigaByte
HTTP – Hypetext Transfer Protocol
HTTPS – Hypetext Transfer Protocol Secure
IR – Infrarosu
LAN – Local Area Network
LBP – Local Binary Patterns
LBPH – Local Binary Patterns Histograms
LDA – Linear Discriminant Analysis
OpenCV – Open Source Computer Vision
OpenNI – Open Natural Interaction
PCA – Principal Component Analysis
POSIX – Portable Operating System Interface
RAM – Random Access Memory
RGB – Red – Green – Blue
ROS – Robot Operating System
SCP – Secure Copy
SDK – Software Development Kit
SSD – Solid State Drive
TFTP – Trivial File Transfer Protocol
URL – Uniform Resource Locator
USB – Universal Serial Bus
VGA – Video Graphics Array

Capitolul 1. Introducere

1.1 Motivația tezei

În ziua de azi, odată cu avansarea tehnologiei și extinderea domeniului său de aplicabilitate, a crescut și nivelul de integrare al acesteia în cadrul muncilor ce necesită deplasarea obiectelor cu masă ridicată, manevrabilitate sporită, executarea precisă a acțiunilor în condiții normale sau în condiții care ar necesita protecția amănunțită a forței de muncă umane. Pentru a realiza cu ușurință și cu cât mai puțin efort fizic aceste lucruri au fost concepute anumite platforme computerizate ce pot fi programate să desfășoare diverse activități, chiar și industriale, cu precizie și simplitate. Platformele sunt folosite în domenii precum cel al fabricării autovehiculelor, unde este necesară abilitatea de a manevra componente cu masă ridicată și de a efectua operații de precizie precum îmbinarea acestora, cel al electronicii unde sunt folosite la realizarea plăcilor cu componente electronice, cel al prelucrării de metale și chiar și in domeniul medical unde se folosesc și pentru executarea operațiilor cu precizie mai mare față de cea umană.

Nucleul acestui proiect este reprezentat de robotul KUKA youBot ce a fost conceput exclusiv pentru scopuri de cercetare și dezvoltarea aplicațiilor în cadrul roboticii mobile. Acest robot poate fi privit precum o reprezentare la scală mică a platformelor industriale ce sunt folosite în prezent. Deplasarea acestor platforme încă se dovedește a fi problematică necesitând o atenție sporită și cunoștințe propriu-zise de manevrare.

1.2 Obiectiv principal

Acest sistem are ca scop deplasarea facilă și fără efort fizic a platformelor masive. În acest context, lucrearea urmărește dezvoltarea unui sistem autonom capabil să se deplaseze în timp real luând decizii în funcție de stimulii primiți prin intermediul unei camere video.

Proiectul este construit pe baza a două module, și anume: (1) modulul de recunoaștere facială ce va integra concepte de prelucrări de imagini și clasificare, (2) modulul de deplasare a robotului ce va permite mișcarea acestuia în funcție de poziția curentă a persoanei autorizate. Primul modul va lua o decizie de mișcare în funcție de datele disponibile apoi le va transmite modulului secundar. Acesta poate fi privit ca o interfață cu platforma omnidirecțională și brațul robotului pe care este plasată camera video.

Pașii necesari pentru executarea deplasării sunt următorii:

1. Achiziția imaginii în format color
2. Procesarea imaginii (convertirea în spațiul de luminanță și redimensionare)
3. Detecția fețelor în imaginea procesată
4. Încercarea autentificării faciale pentru fiecare din fetele detectate
5. Efectuarea mișcării platformei și a brațului în raport cu poziția feței autentificate, dacă este cazul

1.3 Obiective specifice

- Analiza și evaluarea algoritmilor, ce vor fi prezentați în capitolele următoare, din punctul de vedere al acurateții și timpului necesar pentru luarea unei decizii.
- Dezvoltarea unui algoritm care să execute captura și procesarea imaginilor, detecția și recunoașterea fețelor, mișcarea platformei și a brațului.
- Implementarea unor soluții de optimizare pentru a obține o performanță în timp real cât mai ridicată, folosind cât mai eficient resursele limitate ale robotului.
- Implementarea unei soluții pentru adăugarea persoanelor noi în sistem și recalcularea modelului folosit de algoritmul ce a fost selectat ca fiind cel optim pentru rularea pe KUKA youBot.
- Dezvoltarea unei metode de a transfera automat modelul reantrenat pe robot la rularea aplicatiei de recunoastere.
- Implementarea unei interfețe grafice simpliste și intuitive.

Capitolul 2. Echipamentul folosit

2.1 KUKA youBot

Fondată în 1898, compania KUKA reprezintă unul dintre cei mai importanți producători de platforme și roboți industriali și, totodată, unul dintre cei mai prolifici dezvoltatori de soluții pentru automatizări în fabrici. Produsele acestei întreprinderi pot fi folosite în varii domenii precum în industria transporturilor unde se folosesci în special abilitățile lor de a manevra obiecte cu masă ridicată, în industria alimentelor, roboții fiind folosiți pentru împachetarea acestora, pentru stocare și controlul calității, în industria construcțiilor, în industria prelucrării metalelor, unde rezistența la temperaturi ridicate permite folosirea lor pentru manevrarea recipientelor încinse. [1]

Conceput de compania KUKA pentru scopuri educaționale, de cercetare, și pentru dezvoltarea aplicațiilor în special în domeniul logisticii și al navigației, youBot este, în esență, un ansamblu de cel puțin două componente, acestea fiind reprezentate, în cadrul acestui proiect, de platform de bază youBot și de brațul youBot.

2.1.1 Platforma youBot

Componenta fără de care robotul nu ar putea funcționa, platforma youBot, pune în valoare abilitatea de mișcare a robotului și, mai ales, abilitatea de a funcționa de unul singur, prin intermediul calculatorului intern de care dispune. Specificațiile generale ale platformei se regăsesc mai jos [3]:

- Masă aproximativă: 20 kg
- Lungime aproximativă: 58 cm
- Lațime aproximativă: 38 cm
- Viteză minimă de deplasare: 0.01 m/s
- Viteză maximă de deplasare: 0.8 m/s
- Tensiune de alimentare: 24 V

Imaginea de mai jos reprezintă o fotografie a platformei youBot.

Figura 2.1 Platforma bazei youBot

În figura următoare se poate observa geometria detaliată a platformei. Lațimea uneia dintre cele patru roți este notată cu A și este de 74.87 mm, iar diametrul acesteia este notat cu B și are valoarea de 100 mm, în timp ce distanța dintre centrele a două roți situate pe aceeași parte a platformei este de 471 mm și este notată cu C. Distanța dintre două roți situate simetric la unul din capetele robotului este notată cu literă D și are valoarea de 300mm. Fiecare roată este compusă din șase segmente cilindrice cu formă ovală, cu diametrul de 28 mm, notat cu litera E, și a căror poziționare pe diagonală față de suprafața roții permite mișcarea omnidirecțională a robotului. Platforma are o gardă față de sol de 20 mm. [2]

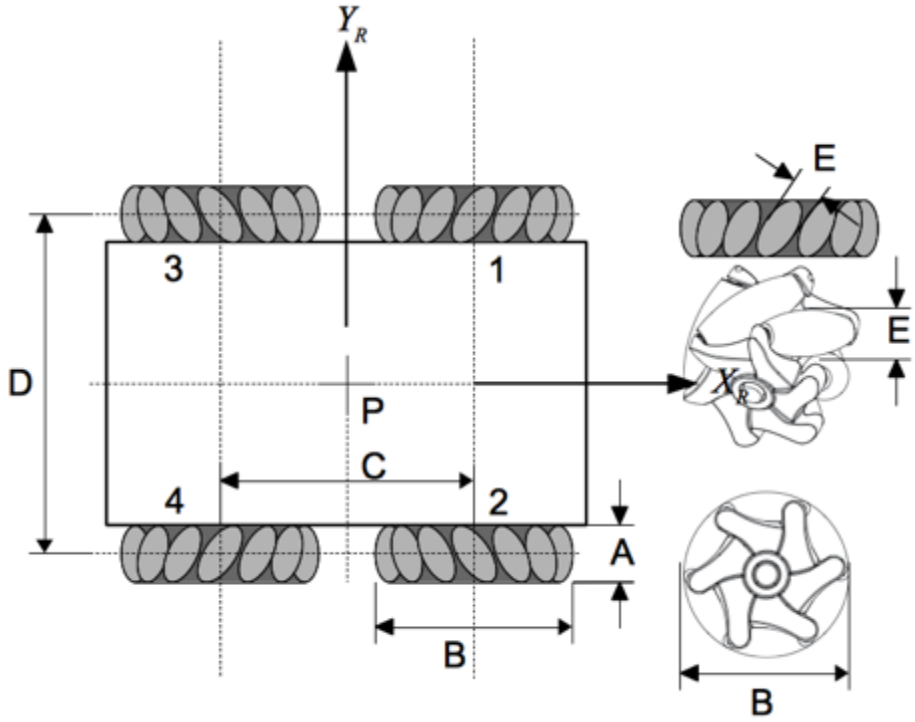

Figura 2.2 Geometria bazei youBot

Robotul poate efectua mișcări de înaintare pe direcțiile marcate de axele notate XR și YR, dar și o mișcare de rotație sub un unghi, Θ, între cele două direcții, marcând, astfel, 3 grade de

libertate. Platforma permite efectuarea mișcărilor pe rând sau simultan, acestea făcându-se în timp real.

Întregul robot este o reprezentare la scală mică a unui robot manipulator industrial, ceea ce face ca proiectele dezvoltate pe el să aibă o semnificație culeasă din lumea reală. Scala acestuia permite lucrul într-un mediu sigur, fiind totodata și suficient de portabil.

2.1.2 Brațul youBot

Venind ca o componentă auxiliară, brațul robotic este o unealtă ce poate fi folosită pentru a simula comportamentul, la scară redusă, a unui manipulator industrial.

Brațul dispune de 5 grade de libertate. Fiecare dintre acestea este reprezentat de o articulație a brațului, ce vor fi notate de la A1 până la A5. În figură se pot observa dimensiunile brațului robotic:

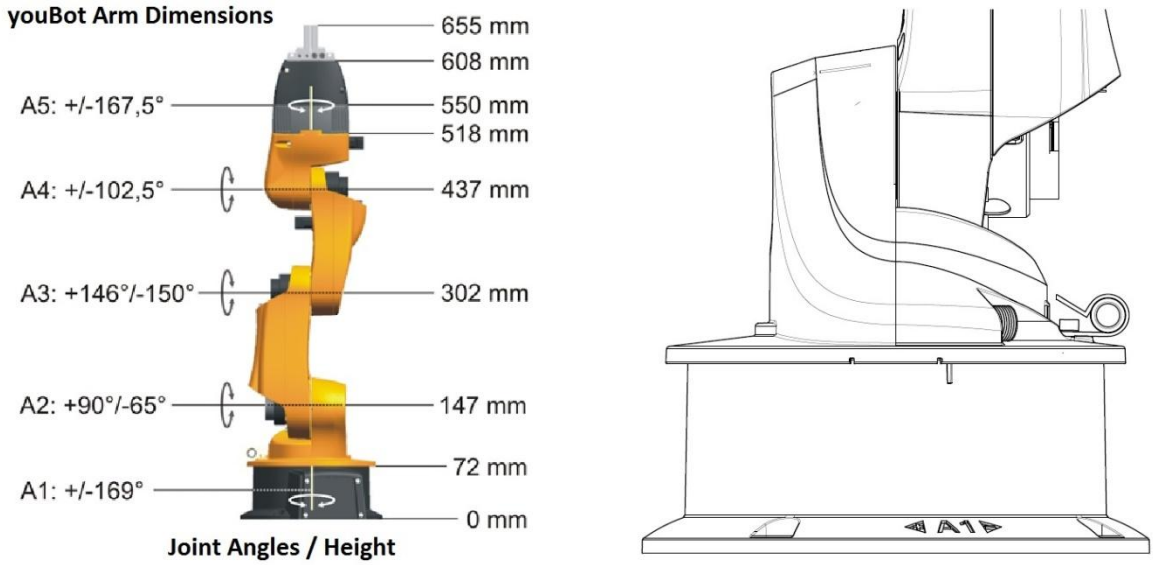

Figura 2.3 Dimensiunile manipulatorului

Manipulatorul are următoarele specificații generale:

- Numărul articulațiilor disponibile: 5
- Înălțime maximă: 655 mm
- Masă aproximativă: 5.3 kg
- Sarcina maximă suportată: 0.5 kg
- Tensiune de alimentare: 24V

Brațul robotic este montat pe platforma youBot, deși el poate funcționa și separat de aceasta dacă este alimentat la o tensiune continuă corespunzătoare, iar comunicația acestuia se face prin intermediul unei interfețe EtherCAT, aceasta fiind bazată pe protocolul Ethernet, dar diferind față

de comunicația de internet sau comunicația de rețea. Această interfață este optimizată în special pentru control automatizat industrial. [2]

Pentru fiecare articulație în parte, brațul dispune de o gamă largă de valori, în grade, ce pot fi setate, motoarele ce îl controlează realizând o mișcare lină și continuă, deși brațul nu își poate roti complet, la 360 grade, nicio articulație. Valorile minime, respectiv maxime, în grade, ce pot fi setate pentru fiecare în parte, se regăsesc mai jos, în tabel:

	Valoare minimă [°]	Valoare maximă [°]
Articulație 1 (A_1)	-167.5	+167.5
Articulație 2 (A_2)	-102.5	+102.5
Articulație 3 (A_3)	-150	+146
Articulație 4 (A_4)	-65	+90
Articulație 5 (A_5)	-169	+169

Tabel 2.1 Limite orientare articulații

În următoarele figuri se observă spațiul de lucru al manipulatorului, privit din lateral, respectiv de deasupra, cu detaliile tehnice aferente fiecărei articulții și traiectoriile marcate pentru fiecare articulație în parte, pentru cazul extinderii maxime a brațului.

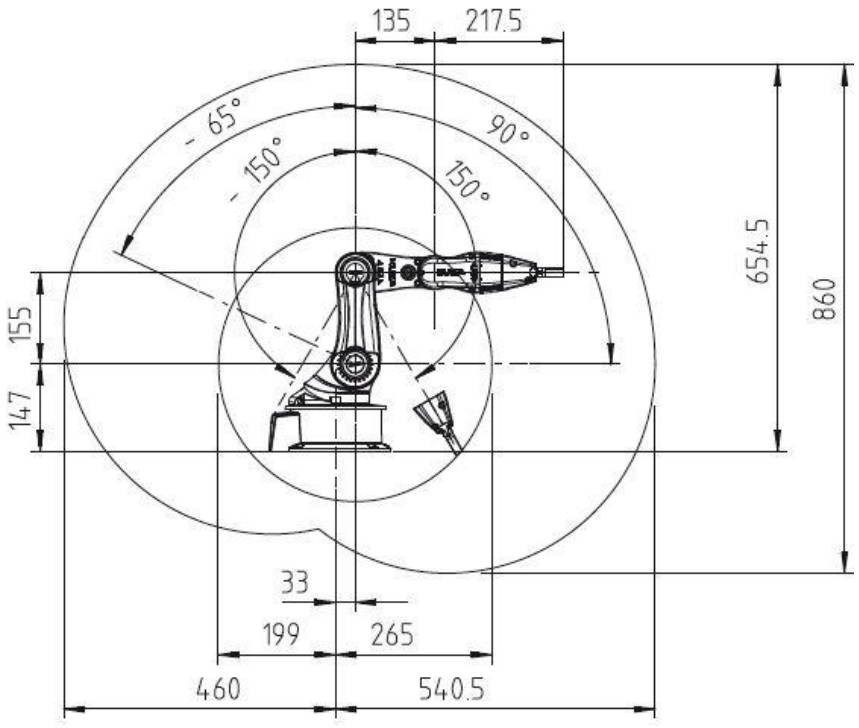

Figura 2.4 Traiectorii braț

Prima articulație, notată cu A1, este responsabilă cu orientarea brațului în planul orizontal. Acest lucru permite folosirea brațului fără a necesita poziționarea minuțioasă a platformei mobile. Articulațiile A2, A3 și A4 au rolul de a poziționa manipulatorul în plan vertical, fapt ce oferă posibilitatea de a-l orienta în aproape orice direcție, vârful brațului putând atinge orice punct din aria de acțiune. Cea de-a 5-a articulație, A5, se rotește într-un plan perpendicular pe planul format de A4 și ea însăși, pentru a permite mecanismului din vârful brațului, ce se aseamăna cu un clește (eng. "gripper"), să poată manipula obiecte de dimensiuni mici.

În cazul extinderii maxime, distanța dintre cele două degete este de 70 mm. Degetele sunt controlate separat prin intermediul unor motorașe ce se deplasează pas cu pas. Imaginea următoare reprezintă un braț youBot cu cleștele desfăcut.

Figura 2.5 Gripper braț

2.1.3 Calculatorul intern

2.1.3.1 Specificații hardware

Pentru a avea o funcționalitate de sine stătătoare, platforma youBot vine echipată cu un calculator intern pe care rulează un sistem de operare personalizat. Specificațiile hardware ale acestuia sunt următoarele:

- Procesor: Intel Atom D510 Dual Core
- RAM: 2 GB
- Stocare: SSD cu capacitate de 32 GB
- 6 porturi USB 2.0
- 1 port VGA
- 2 interfețe LAN

Procesorul Intel Atom D510 Dual Core este construit pe baza arhitecturii i386, pe 32 de biți. El rulează la o frecvență de 1.66 GHz și dispune de 2 nuclee, care, la rândul lor, pot gestiona câte 2 fire de execuție în parte. Astfel, aplicațiile dezvoltate pe youBot pot profita de cele 4 fire de execuție puse la dispoziție de procesor.

Memoria RAM este de tip SO-DDR2, cu 200 de pini, și rulează la frecvență de 667 Mhz. SSD-ul folosit pentru stocare aduce un plus de viteză sistemului, fiind capabil să execute citiri și scrieri cu viteze semnificativ ridicate față de un Hard Disk. Cele 6 porturi USB permit conectarea perifericelor, precum mouse, tastatură, cameră video, dar și conectarea modulelor WiFi, deoarece platforma dispune doar de interfața Ethernet integrată. Astfel se pot dezvolta aplicații de la distanță fără a fi nevoie de conexiunea fizică a platformei la rețeaua locală, prin Ethernet. [2]

2.1.3.2 Specificații software

Sistemul de operare instalat pe calculator este Ubuntu, varianta 12.04, una dintre cele mai populare distribuții de sisteme de operare bazate pe Linux. Deoarece arhitectura pe care este construit sistemul de i386, pe acesta se pot instala chiar și sistemele de operare Linux de actualitate, și chiar și Windows, însă specificațiile sale hardware limitate nu ar face față rulării sistemelor de operare curente. De aceea, producătorii robotului au realizat o versiune personalizată a lui Ubuntu 12.04. Din aceasta au fost eliminate componente care ar fi îngreunat rularea sistemului cu interfața grafică, dar au fost adăugate driver-ele corespunzătoare platformei și manipulatorului youBot, precompilate și gata de folosit încă de la prima pornire a sistemului. Totodată, se găsește preinstalata și variantă driver-elor ce folosește framework-ul ROȘ și o versiune a librăriei OpenCV, iar pe lângă acestea youBot oferă și surse demonstrative ce pot fi folosite că punct de plecare în dezvoltarea aplicațiilor.

Resursele limitate ale calculatorului determină ca acesta să fie considerat un sistem embedded, comparabil cu puterea de procesare actuală a calculatoarelor și chiar a telefoanelor mobile inteligente. Acest fapt determină nevoia de a folosi cât mai mult din puterea de calcul a procesorului și de a folosi tehnici de paralelizare a codului pentru a spori performanțele aplicațiilor dezvoltate.

2.2 Orbbec Astra Pro

Fiind un proiect axat pe prelucrarea imaginilor, detecție și recunoaștere facială, acest lucru necesită prezența unui echipament adecvat pentru a prelua imagini în timp real.

Compania Orbbec a fost înființată în 2013 și se ocupă cu producerea de camere video 3D, cu dezvoltarea software-ului și cu implementarea acestora în domenii precum cel al roboticii, al automatizărilor, al realității virtuale, al jocurilor video, dar și în proiecte menite să folosească tehnologia lor în calculatoare și telefoane mobile inteligente. [3]

În tabelul următor se găsesc specificațiile tehnice ale camerei folosite:

Raza de acțiune	0.6m – 8m
Câmp vizual	60°O x 49.5°V x 73°A
Rezoluția maximă a unei imagini RGB	1280 x 720 @30fps
Rezoluția maximă a unei imagini în adâncime	640 x 480 @30fps
Microfon	2
Dimensiuni	165mm x 30mm x 40mm
Conector pentru alimentare	USB2.0
Consumul de putere	< 2.4 W
Sisteme de operare compatibile	Android/Linux/Windows 7/8/10
Software Compatibil	Astra SDK / OpenNI

Tabel 2.2 Specificatii camera

Figura următoare reprezintă o imagine frontală a camerei Orbbec Astra Pro. Aceasta dispune de cele 2 microfoane așezate la extremități, de un senzor RGB folosit pentru captarea imaginilor color, un proiector infraroșu cu rolul de a emite unde laser, un senzor IR folosit în aplicații pentru preluarea imaginilor pentru detectarea mișcării și proximității prin analiza imaginilor ce conțin obiecte peste care au fost proiectate undele laser IR și de un senzor folosit pentru protecția ochiului uman, ce are rolul de a opri proiectorul de unde laser atunci când este detectat un obiect plasat în fața camerei la mai puțin de 20 cm de aceasta.

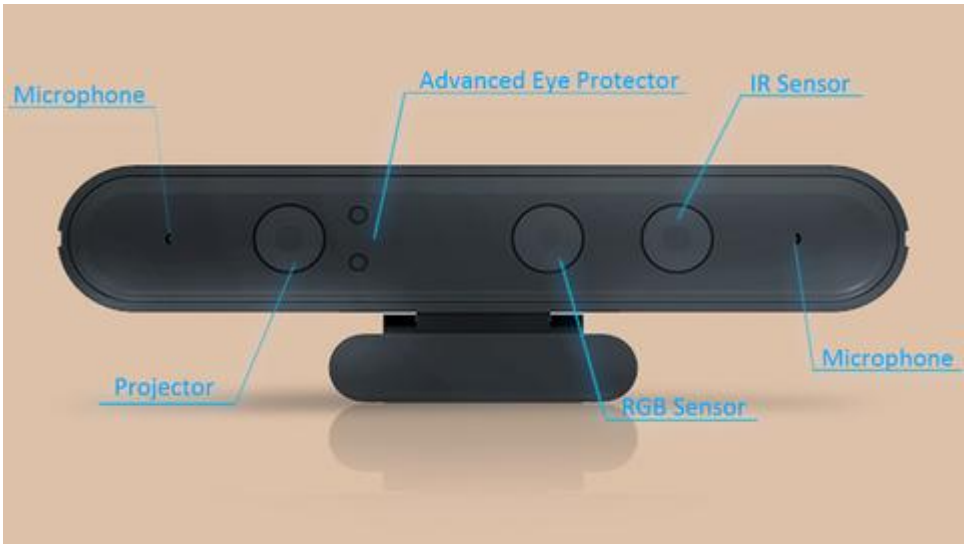

Figura 2.6 Camera video

Capitolul 3. Algoritmi utilizați și componente software

3.1 Detecția facială

Detecția facială este componenta fără de care elementul principal al tezei, recunoașterea facială, nu ar putea funcționa corect. Acest pas este necesar deoarece imaginea captată de camera video poate conține și obiecte care ar face ca algoritmul de recunoaștere să execute o clasificare incorectă.

Prin detecția facială se asigură faptul că sunt trimise către intrarea algoritmului de recunoaștere doar părți din imaginea inițială, părți care sunt rezultatul algoritmului de detecție și care garantează faptul că în conținutul lor se regăsește o singură față a unei persoane. Fața persoanei va ocupa un procentaj foarte mare din numărul de pixeli ai imaginii rezultate.

Detecția facială este folosită, în momentul actual, pentru a ajuta la îmbunătățirea fotografiilor făcute de telefoanele mobile, ajustând proprietățile pentru mediul ambiental în funcție de poziționarea feței/fețelor pentru a le spori claritatea. Detecția este folosită și pentru deblocarea facială a telefoanelor mobile și în domeniul securității de anumite companii.

În implementarea proiectului a fost realizată detecția folosind conceptul de clasificatori cascadati, ce sunt folosiți de cele mai multe ori pentru detecția fețelor, însă pot fi folosiți pentru detecția obiectelor, în general.

Termenul "cascadat" se referă la ideea de a folosi informațiile de la ieșirea clasificatorului curent ca informație suplimentară pentru următorul clasificator. Fiecare clasificator analizează trăsături mai complicate față de cele analizate de clasificatorul anterior. Clasificatorii cascadati sunt antrenați cu câteva sute de imagini pozitive ale obiectului ce se dorește a fi detectat, sub diverse condiții de iluminare și diverse poziții, și cu imagini negative, aleatoare, care nu conțin obiectul și nu se aseamănă cu acesta. Este absolut necesar ca toate imaginile folosite pentru antrenare, pozitive și negative, să aibă aceeași dimensiune.

În lucrare au fost folosiți 2 tipuri de clasificatori cascadati, și anume: clasificatori cascadati bazați pe trăsături Haar și clasificatori cascadati bazați pe caracteristici LBP.

3.1.1 Clasificatori cascadați bazați pe caracteristici Haar

Această metodă se bazează pe faptul că trebuie extrase anumite trăsături dintr-o imagine, astfel nefiind nevoie de toate valorile pixelilor în analiză. Pentru a determina aceste trăsături se folosesc caracteristicile Haar din imaginea următoare. Fiecare trăsătură obținută reprezintă, de fapt, o singură valoare obținută prin diferența dintre suma valorilor pixelilor din dreptunghiul negru și suma valorilor pixelilor din dreptunghiul alb:

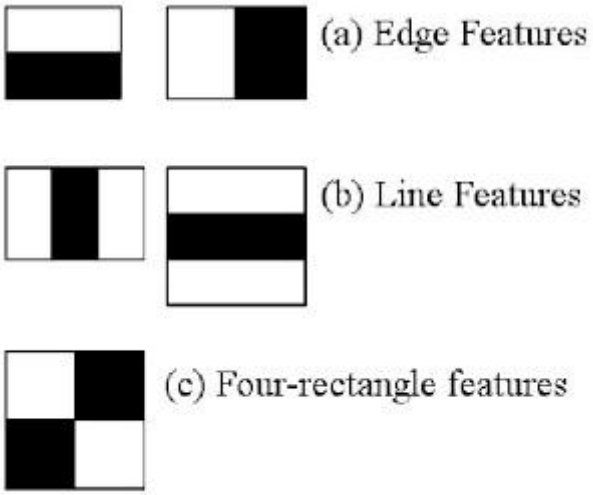

Figura 3.1 Trăsături Haar

La antrenare, umărul de trăsături care ar trebui calculate pentru extragere dintr-o imagine de 24x24 ar fi de peste 160000, deoarece trebuie calculate trăsăturile folosind caracteristicile Haar prezentate mai sus pentru fiecare dimensiune și locație a nucleului. Pentru a micșora semnificativ numărul de calcule necesare pentru calculul unei trăsături a fost introdus conceptul de imagine integrală. Conceptul a fost introdus de Paul Viola și Michael Jones în anul 2001, în lucrarea "Rapid object detection using a boosted cascade of simple features". [4] Astfel, valoarea din imaginea integrală a pixelului de pe poziția (x, y) este egală cu suma valorilor pixelilor din regiunea rectangulară formată de coordonatele (0, 0) și (x, y) din imaginea inițială. Folosind imaginea integrală, calculul unei caracteristici Haar pentru un pixel dat este redus la o operație în care sunt folosiți doar 4 pixeli. Din cele 160000 de trăsături trebuie păstrate doar cele care au o semnificație mare, acest lucru fiind realizat folosind tehnica Adaboost. După eliminarea trăsăturilor neimportante au rămas în jur de 6000 de trăsături. Acestea vor fi folosite în procedeul de testare.

Pentru a testa o imagine și pentru a vedea dacă aceasta conține o față sau nu ar trebui testate cele 6000 de trăsături. Deoarece necesită destul de mult timp aplicarea tuturor trăsăturilor a fost introdus conceptul clasificatorilor cascadati. Astfel, trăsăturile sunt grupate în mai multe stagii de clasificare și se aplică fiecare clasificator, unul câte unul, pe imagine. Clasificatorii au din ce în ce mai multe trăsături de testat. Dacă la un anumit stagiu clasificatorul curent decide că imaginea nu este o față după testarea trăsăturilor, atunci procedeul nu continuă până la ultimul clasificator și algoritmul decide că imaginea nu conține o față, iar dacă sunt testate toate trăsăturile și se regăsesc atunci algoritmul decide că în imagine se află o față. [5]

3.1.2 Clasificatori cascadați bazați pe caracteristici LBP

La fel ca în cazul precedent, din imaginile folosite pentru antrenarea algoritmului de detecție bazat pe caracteristici LBP trebuie extras un set de trăsături. Pentru extragerea trăsăturilor, inițial se împarte imaginea în mai multe regiuni cu suprafața egală și cu aceeași formă (regiune

patratică/dreptunghiulară). Trăsăturile vor fi extrase pentru fiecare regiune în parte, apoi se va compune vectorul de trăsături pentru întreaga imagine.

Se parcurge fiecare regiune în parte, pixel cu pixel. Pentru fiecare pixel se compară valoarea lui cu valorile celor 8 pixeli vecini (vecinătate de 3x3 cu centrul în pixelul curent), iar dacă valoarea curentă este mai mare decât valoarea vecinului respectiv, acestuia este codat cu 0, altfel, vecinul respectiv este codat cu 1. Pentru fiecare pixel obtinet o serie de 8 valori de 0 sau 1, care pot fi citite în sensul acelor de ceasornic sau sens contrar. Numărul format este convertit în baza 10, astfel obținându-se un număr cu valoarea între 0 și 255, inclusiv acestea. Pentru fiecare pixel din regiune se realizează operațiile menționate anterior, pentru fiecare pixel în parte, apoi valorile obținute în baza 10 sunt folosite pentru a realiza histograma regiunii. La final obținem câte o histogramă, ce poate fi privită ca un vector de trăsături LBP, pentru fiecare regiune. Aceste histograme sunt concatenate și obținem un vector de trăsături corespunzător pentru întreaga imagine.

Pentru rezultate cât mai bune trebuie folosite cât mai multe imagini pozitive, ce conțin fețe, dar și de imagini ce nu conțin fețe. Astfel, poate fi antrenat un model ce recunoaște o față într-o imagine în funcție de distribuția trăsăturilor LBP.

3.2 Recunoașterea facială

La momentul actual, recunoașterea faciala este una dintre cele mai folosite metode în domeniul securității, fie că este vorba de verificarea facială pentru accesul într-o anumită companie, fie că este vorba de accesul în propria locuința sau chiar pentru a debloca telefonul mobil.

Pentru recunoașterea facială au fost studiați 3 algoritmi: Eigenfaces, descriptori Fisher (Fisherfaces) și histograme de descriptori binari (LBPH).

Trebuie menționat faptul că algoritmii prezentați folosesc ca date de intrare vectori. Fiecare element al vectorilor este corespondentul unui pixel, astfel că imaginea trebuie convertită în cadrul de luminanță mai întâi, apoi transformată din matrice în vector.

Pentru convertirea în spațiu de luminanță este folosită medierea ponderată a valorilor pentru R, G și B, pentru fiecare pixel în parte. Noua valoare a pixelului este calculată folosind următoarea relație:

$$valoare\_gray = 0.299 * R + 0.587 * G + 0.114 * B$$

3.2.1 Eigenfaces

Metoda Eigenfaces reprezintă implementarea în domeniul recunoașterii faciale a metodei matematice PCA. Cea din urmă este o procedură statistică ce folosește o transformare ortogonală pentru a converti un set de variabile posibil corelate într-un set de variabile necorelate. Acest lucru

este necesar pentru a reduce semnificativ numărul de calcule ce trebuie efetuate. Valorile generate de variabilele necorelate obținute poartă numele de componente principale.

Transformarea este definită astfel încât primei componente principale să îi corespundă cea mai mare variantă posibilă, adică să fie responsabilă de cât mai multă variabilitate posibilă pentru date, și fiecare din componentele principale succesoare să aibă cea mai mare variantă posibilă sub constrângerea ca vectorii lor proprii corespunzători trebuie să fie ortogonali față de vectorii proprii ai componentelor anterioare. Vectorii proprii astfel rezultați formează o bază ortogonală necorelată. Acești vectori pot fi priviți ca fiind o grupare de caracteristici generale ale variațiilor imaginilor din bază de date, deoarece fiecare imagine va avea o reprezentare exactă, după găsirea componentelor principale, printr-o combinație liniară a vectorilor proprii. Vectorii proprii ce vor fi păstrați poartă numele de “

3.2.1.1 Pașii algoritmului Eigenfaces

Presupunem că avem imaginile ce le vom folosi pentru antrenare și că le-am transformat pe fiecare într-un vector coloană, notat cu x_i. Notăm cu n numărul acestora.

1. Se calculează media vectorilor:

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i \qquad 3.2$$

2. Se calculează matricea de covariație, notată cu S:

$$S = \frac{1}{n}\sum_{i=1}^{n} (x_i - \mu)(x_i - \mu)^T \qquad 3.3$$

3. Calculăm valorile proprii, notate cu λ_i:

$$\det(S - \lambda I_n) = 0 \qquad 3.4$$

Trebuie menționat faptul că o matrice de m x n va avea un număr de valori proprii nenule egal cu min (m, n).

4. Calculăm vectorii proprii, notați cu Θ_i, ai matricii de covariație S, folosind relația următoare și ținând cont de relația de ortogonalitate a vectorilor proprii (relatia 3.6):

$$S\Theta_i = \lambda_i \Theta_i, \; i = 1, 2, ..., n \qquad 3.5$$

$$\Theta_i^T \Theta_j = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases} \qquad 3.6$$

5. Se ordonează descrescător valorile proprii obținute si se păstrează cele mai mari k valori proprii și vectorii lor proprii corespunzători

6. Se compune matricea ortogonală corespunzatoare, W:

$$W = (\Theta_1, \Theta_2, ..., \Theta_k)^T \qquad 3.7$$

7. Se proiectează vectorul ce se dorește a fi testat, notat cu x, în subspațiul PCA determinat:

$$y = W(x - \mu) \qquad 3.8$$

Metoda Eigenfaces realizează recunoașterea facială prin proiectarea tuturor imaginilor (ca vectori) în subspațiul PCA, apoi proiectează imaginea de test în subspațiu și găsește cel mai apropiat vecin al vectorului de test, folosind distanța Euclidiană. Astfel, rezultatul clasificării imaginii de test este dat de eticheta celui mai apropiat vecin găsit.

Numărul optim al componentelor principale ce trebuie păstrate pentru o bună recunoaștere facială variază în funcție de imaginile folosite pentru antrenare, neexistând o valoare exactă. [6]

3.2.1.2 Reconstrucția imaginii

Din vectorii obținuți prin proiecția imaginilor în subspațiul PCA se poate reconstrui imaginea în spațiul inițial, fidelitatea imaginii reconstruite variind cu numărul de componente principale folosite pentru reconstrucție. Presupunând că se dorește reconstrucția folind m componente principale, cu m<k, atunci relația folosită pentru reconstrucție este următoarea, obținută din 3.8, cu mențiunea că matricea ortogonală W' conține primii m vectori proprii din W:

$$x' = W'^T y + \mu \qquad 3.9$$

În figura următoare se poate observa calitatea unei imagini reconstruite folosind de la 10 până la 300 de componente principale:

Figura 3.2 Reconstrucția imaginii

Metoda Eigenfaces este sensibilă la variații de scalare, iluminare, rotație sau translație și necesită o reînvățare completă a datelor de antrenare pentru a adăuga noi subiecți în baza de date.[7]

3.2.2 Fisherfaces

Analiza Componentelor Principale, nucleul metodei Eigenfaces, găsește o combinație liniară a trăsăturilor care maximizează varianța totală a datelor. În timp ce aceasta este o metodă puternică pentru reprezentarea datelor, nu ține cont de clasele imaginilor așa că o mare parte din informația distincte poate fi pierdută când se deduc componetele principale și se renunță la celelalte componente. Într-o situație în care varianța datelor este determinată de o sursă externă, precum iluminarea, componetele identificate de PCA nu conțin neapărat informație distinctivă, ceea ce determină imposibilitatea de a obține o diferențiere clară între imaginile proiectate în subspațiul PCA. Astfel a apărut nevoie găsirii unei metode care să reducă semnificativ influența condițiilor de iluminare. [6]

Metoda Fisherfaces este compusă din două metode care sunt folosite în cascadă, PCA și LDA (Linear Discriminant Analysis). LDA execută o reducere a dimensionalitatii specifică fiecărei clase în parte. Pentru a găsi combinația de trăsături ce realizează cel mai bine separarea claselor, LDA urmărește maximizarea raportului dintre împrăștierea inter-clase și împrăștierea în interiorul acestora., în schimbul maximizării împrăștierii generale. Acest lucru se reduce la ideea de a grupa clase ce se aseamănă foarte mult, în timp ce clase foarte diferite trebuie să fie cât mai distanțate, toate acestea întâmplându-se în spațiul cu mai puține dimensiuni. [6]

3.2.2.1 Pașii algoritmului LDA

1. Se calculează media vectorilor folosind relația 3.2.
2. Se calculează matricea de covariație, notată cu S, folosind 3.3.
3. Se calculează valorile proprii ale lui S, notate cu λ_i, folsind 3.4.
4. Se determină vectorii proprii, notați cu Θ_i, ai matricii de covariație S, folosind relația 3.5 și ținând cont de relația de ortogonalitate a vectorilor proprii, 3.6
5. Se ordonează descrescător valorile proprii obținute și se păstrează cele mai mari k valori proprii și vectorii lor proprii corespunzători.
6. Se compune matricea ortogonală corespunzătoare pentru PCA, W_{PCA}, folosind 3.7.
7. Se proiectează fiecare vector inițial în subspațiul PCA determinat și se notează cu y_i proiecția vectorului x_i.
8. Având cele c clase cunoscute, fiecare conținând un număr N_i de proiecții obținute anterior, $i \in \{1,2,...,c\}$, și N numărul total de vectori proiectați, se calculează media totală a vectorilor:

$$\mu_t = \frac{1}{N}\sum_{i=1}^{N} x_i \qquad 3.10$$

9. Se calculează vectorul medie al fiecarei clase:

$$\mu_i = \frac{1}{N_i}\sum_{j=1}^{N_i} x_j \qquad 3.11$$

10. Se calculează matricea de împrăștiere inter-clase, S_B:

$$S_B = \sum_{i=1}^{c} N_i(\mu_i - \mu)(\mu_i - \mu)^T \qquad 3.12$$

11. Se calculează matricea de împrăștiere în interiorul claselor, S_W:

$$S_W = \sum_{i=1}^{c}\sum_{j=1}^{N_i}(x_j - \mu_i)(x_j - \mu_i)^T \qquad 3.13$$

12. Se determină valorile proprii, notate cu λ', ale matricii $S_B^{-1}S_W$, folosind relația:

$$\det(S_B^{-1}S_W - I_k) = 0 \qquad 3.14$$

13. Se calculează vectorii proprii, notați cu $\Theta_i^{'}$, ai matricii $S_B^{-1}S_W$, folosind relația următoare și ținând cont de relația de ortogonalitate a vectorilor proprii:

$$S_B^{-1}S_W\Theta_i' = \lambda_i'\Theta_i', \ i = 1,2,...,k \qquad 3.15$$

14. Se ordonează descrescător valorile proprii calculate anterior și se păstrează cele mai mari k' valori proprii, împreună cu vectorii lor proprii corespunzători.
15. Se compune matricea ortogonală corespunzătoare transformării LDA, W_{LDA}:

$$W_{LDA} = (\Theta_1^{'}, \Theta_2^{'}, ..., \Theta_{k'}^{'})^T \qquad 3.16$$

16. Se calculează matricea transformării Fisherfaces, W, folosind relația:

$$W = W_{LDA}W_{PCA} \qquad 3.17$$

17. Se proiectează vectorii obținuți după reducerea dimensiunilor folosind PCA în subpațiul determinat de LDA, astfel:

$$z_i = W_{LDA}y_i \qquad 3.18$$

Aplicarea PCA înainte de aplicarea LDA este necesară deoarece în cazul în care vectorii de intrare ar fi trimiși direct către LDA, matricea de împrăștiere în interiorul claselor ar fi avut determinantul nul. Acest fapt este generat de rang-ul mic ce l-ar fi avut matricea SW, și anume maxim N-c, pentru că numărul imaginilor folosite, N, ar fi cu mult mai mic decât numărul de pixeli din fiecare imagine. PCA rezolvă această problemă prin reducerea semnificativă a numărului de componente, proiectând setul de imagini într-un spațiu cu mai puține dimensiuni și pentru a permite aplicarea LDA. La rândul sau, LDA reduce dimensiunea vectorilor la c-1, unde c este numărul de clase inițiale. [6]

Pentru realizarea clasificării unei imagini de test, se proiectează aceasta în spațiul c-1 dimensional, folosind matricea transformării Fisherfaces calculată la 3.17:

$$z = W(x - \mu) \qquad 3.19$$

Următorul pas este reprezentat de găsirea celui mai apropiat vecin prin calcularea distanței Euclidiene. Astfel, eticheta imaginii testate va fi aceeași cu eticheta imaginii corespunzătoare distanței Euclidiene minime.

Aplicarea PCA înainte de aplicarea LDA este necesară deoarece în cazul în care vectorii de intrare ar fi trimiși direct către LDA, matricea de împrăștiere în interiorul claselor ar fi avut determinantul nul. Acest fapt este generat de rang-ul mic ce l-ar fi avut matricea SW, și anume maxim N-c, pentru că numărul imaginilor folosite, N, ar fi cu mult mai mic decât numărul de pixeli din fiecare imagine. PCA rezolvă această problemă prin reducerea semnificativă a numărului de componente, proiectând setul de imagini într-un spațiu cu mai puține dimensiuni și pentru a permite aplicarea LDA. La rândul său, LDA reduce dimensiunea vectorilor la c-1, unde c este numărul de clase inițiale. [6]

Analiza Discriminantă nu este influențată de condițiile de iluminare la fel de mult ca și metoda Eigenfaces. Aceasta găsește trăsăturile feciale pentru a diferenția persoanele. Performanța oferită de algoritmul Fisherfaces depinde foarte mult de datele de intrare, precum Eigenfaces, și este o metodă sensibilă la variațiile de fundal, scalare, rotație sau translație. [7]

3.2.3 LBPH

În cazul metodelor Eigenfaces și Fisherfaces datele se regăsesc sub forma unui vector într-un spațiu cu multe dimensiuni. Ambele metode identifică un spațiu cu mai puține dimensiui în care încearcă să prezerve informația utilă din imaginile inițiale. Totuși, metoda Eigenfaces poate întâmpina probleme, precum varianța datelor generată de un factor extern. Pentru a prezeva cu succes informație utilă se folosește metoda LDA, după cum a fost descris în cazul metodei Fisherfaces. [6]

Cu toate acestea, există și situații în care imaginile folosite pentru antrenare nu au o iluminare suficient de bună sau situații în care nu dispunem de multe imagini pentru fiecare subiect. În aceste situații, metodele descrise anterior ar avea rezultate aproape complet eronate. În imaginea următoare se poate observa un grafic ce arată numărul de imagini pentru antrenare de care ar avea nevoie metodel anterioare pentru a obține o acuratețe cât mai mare, folosind baza de data AT&T Facedatabase. Se observă faptul că pentru a atinge o acuratețe bună sunt necesare aproximativ 8 imagini. [6]

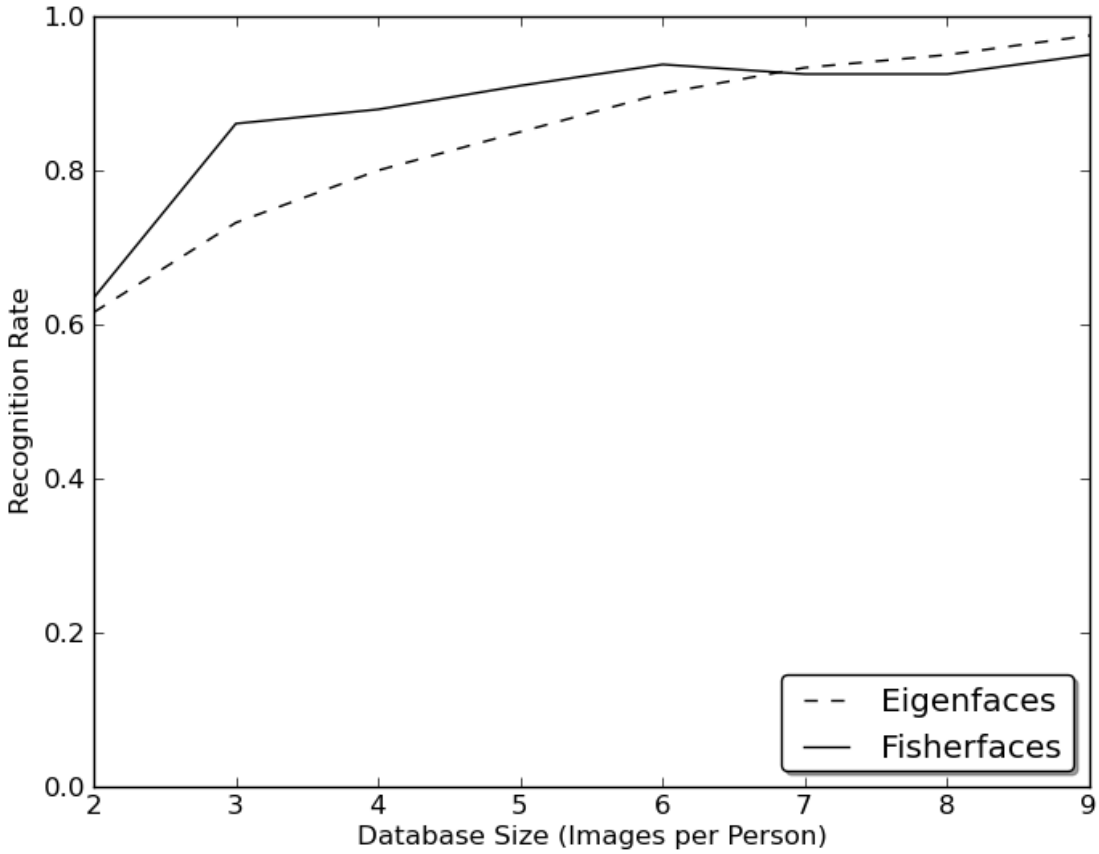

Figura 3.3 Dependența Eigenfaces și Fisherfaces de numărul de imagini

Metoda LBPH este axată pe extragerea de trăsături locale din imagini. Ideea metodei nu este de a privi întreaga imagine ca pe un vector cu multe dimensiuni, ci să descrie doar trăsăturile locale ale unui obiect. Aceste trăsături extrase astfel vor avea, implicit, o dimensionalitate redusă. Descrierea locală trebuie să fie robustă împotriva factorilor precum scalarea, translația sau rotația imaginilor. [6]

Ideea de bază a LBPH este să rezume structura locală dintr-o imagine la compararea fiecărui pixel cu vecinătatea lui. Se alege un pixel drept centrul vecinătății și se compară cu pixelii din jurul său. Dacă valoarea pixelului din centru este mai mare sau egală decât cea a vecinului ales, atunci vecinul respectiv va fi codat cu valoarea 1, altfel cu 0. Astfel obținem în jurul pixelului ales drept centru, într-o vecinătate de 3x3, un număr binar format din valorile 0 și 1 cu care am codat vecinii acestuia. Cu 8 vecini obținem 256 combinații posibile, ce poartă numele de Local Binary Patterns, sau coduri LBP. Un exemplu de cod LBP se regăsește în figura următoare:

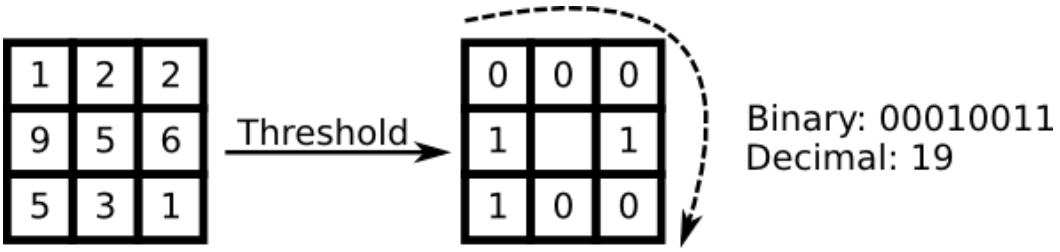

Figura 3.4 Cod LBP

3.2.3.1 Pașii algoritmului LBPH

1. Se împarte imaginea în regiuni cu suprafață egală și cu aceeași formă (regiune patratică/dreptunghiulară).
2. Fiecare regiune se parcurge pixel cu pixel și se calculează codul LBP corespunzător, în baza zece, folosind relația următoare, în care x_c și y_c sunt coordonatele pixelului central cu valoarea vc, iar vp reprezintă valoarea pixelilor vecini, pe rând:

$$LBP(x_c, y_c) = \sum_{p=0}^{p-1} 2^p s(i_p - i_c) \qquad 3.20$$

$$s(x) = \begin{cases} 1 & \text{daca } x \geq 0 \\ 0 & \text{altfel} \end{cases} \qquad 3.21$$

3. Se realizează histograma pentru fiecare regiune.
4. Se concatenează histogramele obținute anterior și se obține astfel un vector de trăsături corespunzator imaginii.

În figura următoare este exemplificată obținerea vectorului de trăsături locale dintr-o imagine:

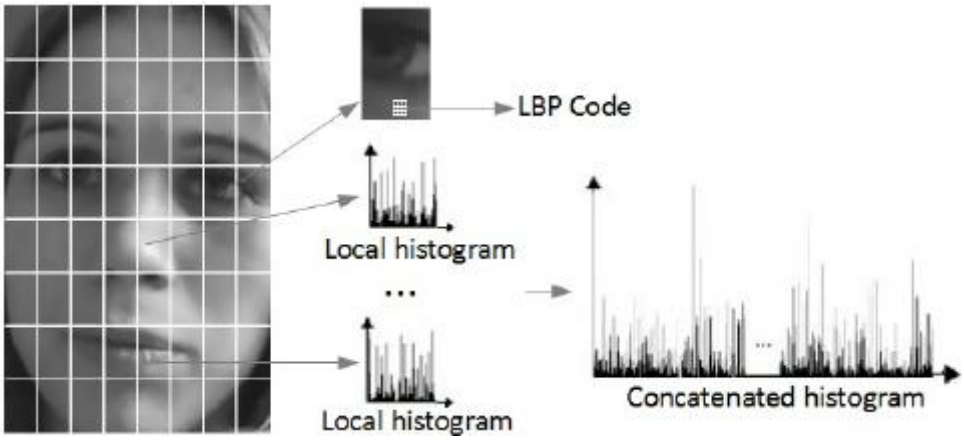

Figura 3.5 Vector trasaturi

Pentru a testa o imagine se urmează pașii menționați anterior și apoi se ia decizia folosind distanța Euclidiană, în raport cu cel mai apropiat vecin, care, în acest caz, va fi imaginea a cărei histogramă rezultantă este cea mai asemănătoare cu histograma obținută pentru imaginea de test.

Aplicarea operatorului LBP asupra unei imagini în spațiu de luminanță se referă, de fapt, la generarea unei noi imagini care conține pe o anumită poziție, plasată la coordonatele x_c și y_c, codul LBP calculat pentru pixelul situat la poziția (x_c, y_c) în imaginea originală.

Prin definiție, operatorul LBP este robust impotriv transformarilor liniare în plan de luminanță, în figura următoare fiind exemplificat acest lucru. Imaginea de bază din figură a fost modificată artificial, prin transformări liniare, și se observă că imaginile obținute în urma LBP se aseamănă foarte mult, fiind invariante aproape complet la transformările liniare. [6]

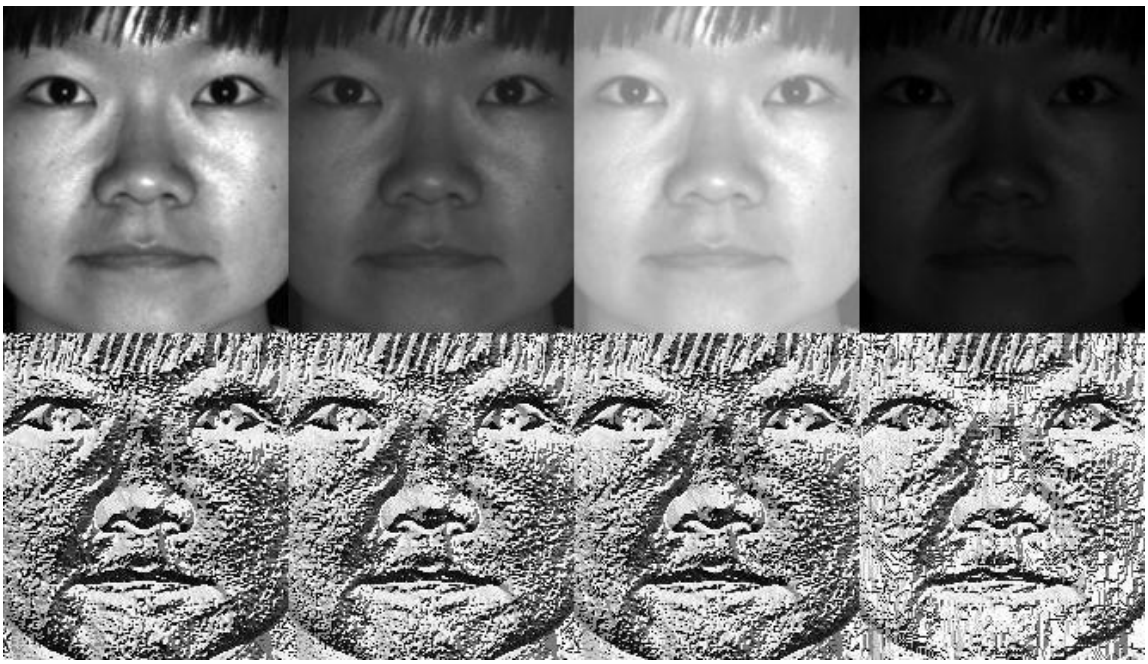

Figura 3.6 Invarianța LBP

3.3 OpenCV

OpenCV este o bibliotecă de tip open-source ce pune la dispoziția oricărui utilizator o gamă largă de funcții pentru prelucrarea imaginilor și de algoritmi folosiți în inteligența artificială. Biblioteca poate fi folosită împreună cu limbajele C/C++, Java, Python, Matlab și poate rulă pe sisteme de operare precum Windows, distribuții de Linux, Mac OS X. Fiind de tip open-source, utilizatorii au acces la codul sursă și îl pot folosi și modifica după bunul plac în scopuri necomerciale. [8]

În această bibliotecă se găsesc peste 2500 de algoritmi optimizați, printre care sunt și algoritmi clasici, dar și algoritmi state-of-the-art în domeniul inteligenței articiale. Algoritmii pot fi folosiți pentru:

- a detecta și recunoaște fețe
- a identifica obiecte
- a determina acțiunile oamenilor din videoclipuri
- a urmări obiecte care se mișcă
- a extrage modele 3D ale obiectelor
- a realiza o imagine cu rezoluție mare din mai multe imagini cu rezoluție mică ale unei împrejurimi
- a urmări mișcările ochiului
- a recunoaște împrejurimi
- a implementa conceptul de realitate augmentată

În cadrul proiectului se folosesc prelucrările simple de imagine puse la dispoziție de bibliotecă, precum redimensionare și conversia dintr-o imagine cu 3 planuri de culori către o imagine cu niveluri de luminanță, dar și algoritmi pentru detecția facială, și anume clasificatorii cascadati bazați pe caracteristici Haar și LBP, dar și algoritmii de recunoaștere Eigenfaces, Fisherfaces și LBPH.

3.4 Qt

Qt este un set de instrumente open-source folosit în general pentru dezvolatrea de interfețe grafice cu utilizatorul precum și aplicații cross-platform pentru rularea pe diverse combinații de platforme software și hardware, Windows, Linux, macOS, Android și chiar în sisteme embedded, cu cât mai puține modificări sau chiar deloc în codul sursă.

Qt suportă diverse compilatoare, precum GCC și cel al suitei Visual Studio, are acces la baze de date SQL, poate analiza fișiere în format XML, JSON. De asemenea, Qt oferă poate gestiona și firele de execuție.

În cadrul proiectului a fost folosit Qt pentru a crea o interfață simplistă și intuitivă pentru programul ce rulează pe server și căruia îi este atribuită funcția de a adăuga noi persoane, prin efectuarea de capturi ale persoanei și reantrenarea sistemului pentru a obține noul model folosit de algoritm la rularea pe youBot. [9]

3.5 libcurl

Decizia de a rula programul pentru a recalcula modelul pe un calculator ce rulează drept server pune problema transferării modelului pe youBot. Aducerea manuală, fie prin transfer prin rețea, fie prin intermediul unui mediu de stocare extern, precum stick-ul USB, nu este comodă pentru utilizator.

Libcurl este o bibliotecă de transfer prin URL orientată către client ce suportă mai multe protocoale pentru transfer, cele mai importante fiind HTTP, HTTPS, FTP, TFTP, SCP. Biblioteca este portabilă, rulând pe platforme precum Android, distribuții de Linux, OpenBSD, Symbian, Windows. [10]

Capitolul 4. Optimizări

În realizarea practică a proiectului cel mai mare impas l-au reprezentat resursele limitate ale lui youBot. Rularea secvențială a programului realizat inițial, excluzând componența ce se ocupă cu deplasarea robotului, a pus probleme chiar și unui calculator modern, cu resurse mult superioare calculatorului intern al robotului. Achiziția imaginilor de la camera video a calculatorului s-a făcut la o rezoluție de 1280x720 pixeli, iar părțile de imagine cu conținut detectat ca fiind o față umană fiind reduse la dimensiunea de 192x168, deoarece aceasta era dimensiunea inițială a imaginilor folosite pentru obținerea modelului utilizat de algoritmul de recunoaștere facială. În această situație, numărul maxim de cadre procesate pe secundă era de aproximativ 16. Acest număr poate părea suficient de satisfăcător, dar luând în considerare faptul că execuția programului nu s-a făcut pe calculatorul intern al robotului putem deduce că pe youBot programul ar fi rulat cu mult mai lent. În această situație, numărul de cadre pe secunde atins în timpul execuției secvențiale pe youBot a fost de 4-5, în condițiile menționate anterior. Acest lucru a alimentat nevoia de a găsi și aplica diverse optimizări asupra codului pentru a face posibilă rularea în timp real a programului pe youBot.

4.1 Reducerea rezoluției imaginilor captate

Camera video folosită, Orbbec Astra Pro, permite ahizitia imaginilor la o rezoluție maximă de 1280x720 pixeli. Prin efectuarea unui simplu test de captare de imagini și apoi afișarea lor, pe youBot, la rezoluția maximă permisă, am observat că se obțin aproximativ 10 cadre pe secundă. Acest lucru înseamnă că achiziția și afișarea unei imagini la acea rezoluție se realiza în 100ms. Măsurând timpul de execuție al ambelor instrucțiuni am ajuns la concluzia că achiziția imaginii ocupă aproape tot acel timp de 100ms.

Următoarea cea mai mare rezoluție admisă de camera video este de 640x480 pixeli. Efectuarea aceluiași test folosind această rezoluție a dus la obținerea unor rezultate favorabile. În acest caz, am obținut 30 de cadre pe secundă, acesta fiind și numărul maxim admis de specificațiile camerei video.

4.2 Reducerea dimensiunii imaginilor din baza de date

Reducerea rezoluției camerei video la 640x480 pixeli are un efect direct asupra dimensiunii imaginilor folosite pentru obținerea modelului utilizat pentru recunoașterea facială. Utilizarea rezoluției native pentru imaginile din baza de date, și anume de 192x168 pixeli, generează 2 cazuri posibile:

1) Când persoana este situată suficient de aproape de camera video, iar fața detectată a acesteia depășește rezoluția de 192x168 pixeli, întâlnim cazul favorabil. Imaginea

obținută după redimensionarea feței la 192x168 pixeli va fi clară deoarece vom converti o imagine de la o rezoluție mai mare către o rezoluție mai mică.

2) În celălalt caz, când persoana este departe de camera video, rezoluția feței detectate ar fi sub cea necesară, fapt ce ar determina că imaginea obținută după redimensionarea către 192x168 să fie neclară și să conțină informații redundante, ce ar favoriza recunoașterea incorectă. Acest lucru se întâmplă deoarece, în acest caz, s-ar produce redimensionarea de la imagine cu rezoluție mică la o imagine cu rezoluție mare.

Pentru a rezolva această problemă și pentru a permite totodată recunoașterea de la o distanță cât mai mare am efectuat un test pentru a afla dimensiunile, în pixeli, a unei fețe detectate. La rezoluția folosită pentru achiziția imaginii, de 640x480, am observat că unei distanțe de aproximativ 2.5m față de cameră îi corespund dimensiunile feței de, aproximativ, 50 pixeli lățime și 50 pixeli înălțime. Astfel am decis că este necesară redimensionarea imaginilor din baza de date. Prin reducerea fiecărei dimensiuni a imaginilor de 4 ori obținem imagini cu dimensiunile de 48, respectiv 42, pixeli, de la o rezoluție de 192x168. Aceste dimensiuni sunt suficient de aproape de cele determinate experimental, și anume 50 pixeli lățime și 50 înălțime Astfel păstrăm suficiente componente din imagine pentru recunoaștere și creștem semnificativ rata de succes pentru recunoașteri de la distanțe de 2.5 până la 3m.

4.3 Detecția facială

4.3.1 Redimensionarea imagini utilizate pentru detecție

Inițial, după stabilirea rezoluției de 640x480 pentru camera video și micșorarea imaginilor din baza de date, am continuat cu testarea programului și am analizat durata de timp a instrucțiunii ce realizează detecția facială. S-a observat faptul că analiza imaginii pentru a detecta fețele din aceasta necesită un timp îndelungat, comparativ cu celelalte instrucțiuni, precum achiziția imaginii și convertirea în spațiu de luminanță. Căutarea fețelor într-o imagine cu rezoluția de 640x480 pixeli necesită între 100 și 120 ms, pe youBot, ceea ce în cazul în care am rula doar detecția, fără alte prelucrări, ar limita rularea programului la cel mult 10 cadre pe secundă.

Soluția acestei probleme a venit prin redimensionarea imaginii pentru detecție la 320x240 pixeli, de 4 ori mai mică decât cea inițială. Astfel, timpul necesar căutării fețelor în imaginea redimensionată s-a redus în intervalul de 50-60 ms. Pozițiile fețelor detectate, ca și coordonate, în imaginea redimensionată pot fi translatate pentru a se potrivi în imaginea inițială prin simpla înmulțire cu 2. Coordonatele unui dreptunghi, returnate de instrucțiunea ce realizează detecția, ce reprezintă o față detectată sunt pozițiile pe orizontală, respectiv verticală, ale punctului stânga-sus al dreptunghiului, urmat de lățimea și lungimea dreptunghiului, toate în pixeli. În figura următoare am exemplificat translatarea poziției feței detectate către imaginea inițială folosind pozițiile identificate în imaginea micșorată.

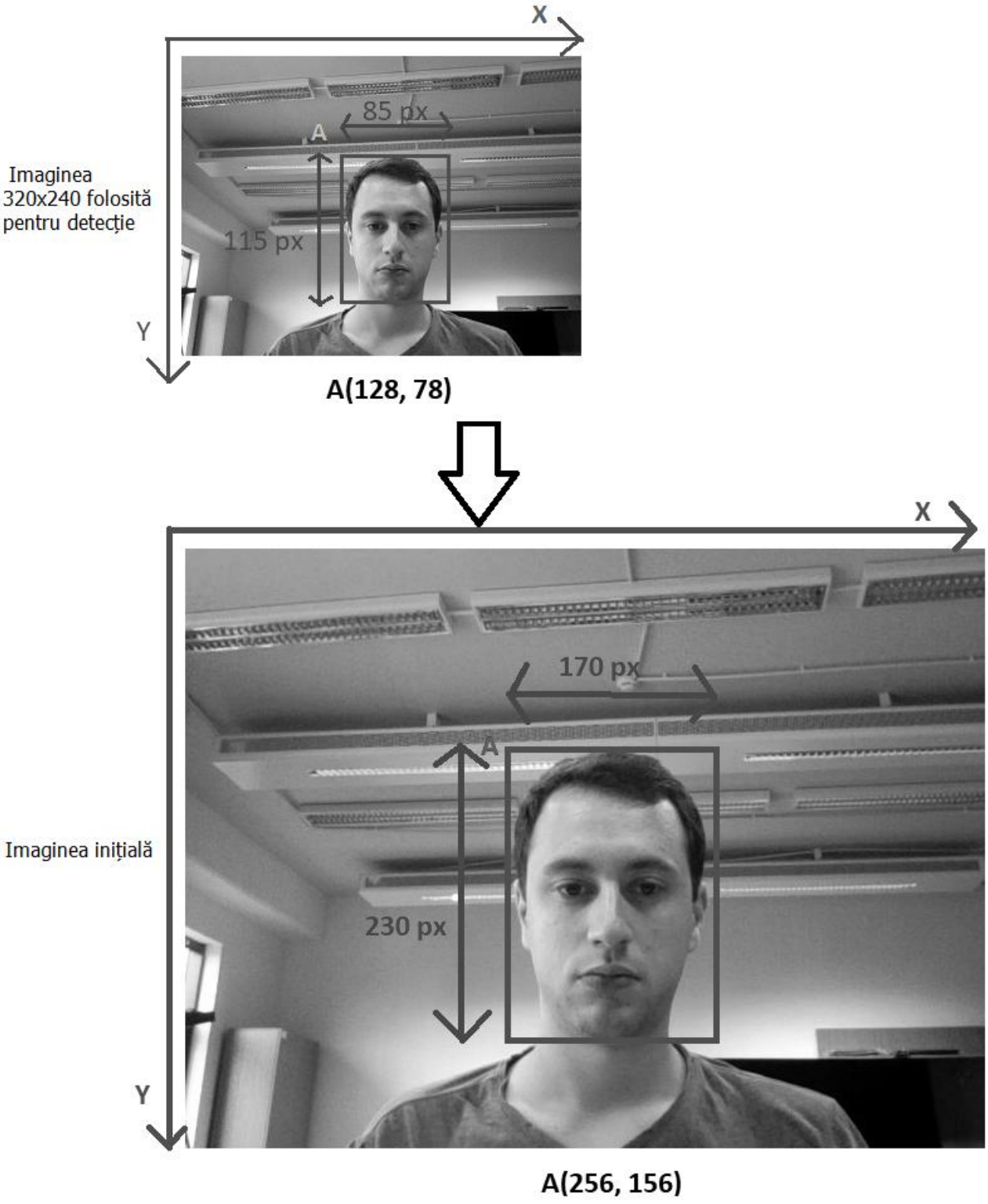

Figura 4.1 Translatarea coordonatelor feței

4.3.2 Clasificatori utilizați

Detecția fețelor în imaginea redimensionată se realizează cu ajutorul clasificatorilor cascadati prezentați în subcapitolul 3.1. OpenCV pune la dispoziția utilizatorului modele preantrenate, în format xml, pentru a putea folosi fie caracteristicile Haar, fie cele LBP pentru detecție.

Ambele modele pot detecta fețe într-o imagine până la dimensiunea minimă de 24x24 pixeli. Acest lucru înseamnă că orice față care ar fi reprezentată pe o suprafață mai mică decât cea menționată nu va fi detectată de clasificatori. Acele dimensiuni sunt raportate la imaginea micșorată, de 320x240. Translatandu-le în dimensiuni pentru imaginea inițială, obținem că limita inferioară de detecție fețe cu dimensiunea de 48x48 pixeli, ceea ce satisface și condiția de a avea fețe cu dimensiunea mai mare sau egală cu 48x42 pixeli, aceasta fiind rezoluția imaginilor din baza de date.

Diferența între cele 2 modele, respectiv tip de clasificatori, este că modelul bazat pe caracteristici Haar realizează calculele folosind numere reale, în timp ce modelul bazat pe caracteristici LBP execută calculele folosind numere întregi. Acest lucru afectează timpul necesar pentru efectuarea calculelor, deci modelul Haar solicită mai mult sistemul, însă oferă o precizie cu 2-3 procente mai bună decât cea a modelului LBP. Avantajul modelului LBP este rapiditatea. În consecință, când fața subiectului autorizat este mai aproape de cameră, programul folosește modelul LBP pentru detecție, iar când aceasta depășește pragul impus (lățime feței, în pixeli, se micșorează, deci fața se depărtează în cadru) sau când trec 40 de cadre în care programul nu a identificat fața utilizatorului autorizat, indiferent dacă ultima data l-a identificat aproape sau departe, programul comută pe folosirea modelului Haar pentru detecție.

4.3.3 Utilizarea poziției anterioare a feței autorizate

Optimizarea anterioară oferă deja rezultate suficient de bune pentru efectuarea recunoașterii în timp real. Pentru a micșora și mai mult timpul necesar detecției am folosit coordonatele feței autentificate pentru a obține un chenar cu lățimea și înălțimea de 130 pixeli, ce este poziționat în afară chenarului feței. Coordonatele (punctul din stânga-sus) acestuia sunt calculate în funcție de centrul feței autorizate, prin scăderea din fiecare coordonată (x și y) a valorii 65, dar lățimea și înălțimea acestuia va fi mereu de 130 pixeli. Cele două chenare au formă de pătrat (cel exterior, ce reprezintă poziția feței autentificate anterioare, și cel interior ce reprezintă fața detectată în imaginea curentă) și sunt concentrice. Astfel, folosind chenarul exterior, în imaginea următoare se execută detecția doar în porțiunea marcată de el. Fețele detectate vor avea drept origine punctul stânga-sus al chenarului calculat. Pentru a obține poziția fețelor detectate raportată la originea întregii imagini folosite pentru detecție se adună la coordonata x a fiecărei fețe găsite coordonata x a chenarului. În același mod se procedează și pentru coordonata y. Acest lucru este necesar pentru a putea decupa din imagine fețele ce vor fi trimise către procesul de recunoaștere.

Funcția utilizată pentru detecție are setată ca dimensiuni minime pentru detecție 24x24 pixeli, iar pentru dimensiuni maxime are setate dimensiunile de 120x120 pixeli. Acest fapt garantează că dimensiunea chenarului care marchează poziția anterioară a feței autorizate este suficient de mare pentru a cuprinde chiar și o față situată foarte aproape de cameră.

Poziția anterioară nu este păstrată la nesfârșit. Dacă a fost detectată persoana autorizată în cadrul actual, în următoarele 10 cadre programul va căuta în acea zonă de imagine fața autorizată. În cazul în care în niciunul din cele 10 cadre nu a fost găsită persoana respectivă, atunci poziția anterioară este resetată, iar programul va caută în întreagă imagine. În situația în care într-unul din următoarele 10 cadre este detectată persoana autorizată, atunci se calculează noua poziție ce va fi folosită drept poziție anterioară și contorul celor 10 cadre se resetează la 0.

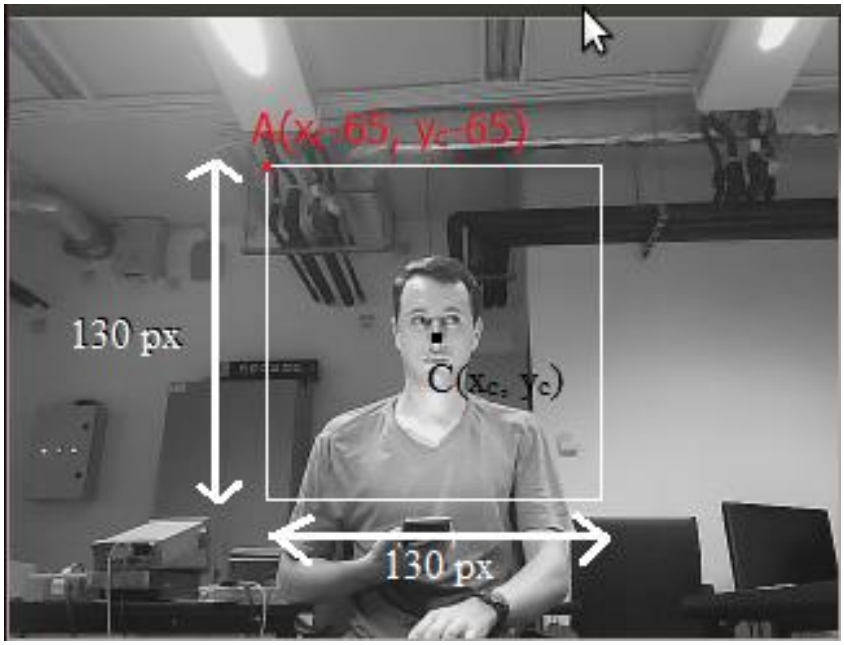

Figura 4.2 Poziția anterioară în imaginea folosită pentru detecție

4.4 Paralelizarea codului

Folosirea tehnicilor de paralelizare a codului a permis o îmbunătățire semnificativă a rulării programului. Prin intermediul firelor de execuție POSIX (pthread) și a tehnicii benzii rulante, execuția codului a fost împărțită în 4 fire de execuție:

1. Firul principal al programului ce are rolul de a menține bucla principală a programului, de a lansa în execuție celelalte 3 fire de execuție și de a rula recunoașterea fețelor găsite de firul asignat pentru detecție.
2. Un fir de execuție se ocupă exclusiv cu achiziția imaginilor de la camera video cu rezoluția 640x480.
3. Acest thread se ocupă cu transformarea imaginii captate în spațiu de luminanță și redimensionarea ei la rezoluția de 320x240.
4. Ultimul fir de execuție este responsabil cu detectarea fețelor dintr-o imagine.

Astfel, la o iterație a buclei principale:

- Se pregătesc variabilele pentru fiecare thread în parte
- Se pune în funcțiune thread-ul pentru achiziția unei noi imagini
- Se procesează imaginea captată în iterația anterioară

- Se execută recunoașterea pentru fiecare față identificată în imaginea obținută cu 3 iterații în urmă
- Se execută detecția facială pe imaginea obținută cu 2 iterații în urmă

Comunicația între cele 4 fire de execuție se face prin intermediul unor variabile globale.

În cazul în care este detectată și recunoscută persoana autorizată, firul principal de execuție pune în mișcare platforma robotului în funcție de cât de departe sau aproape este situată persoana și modifică unghiul sub care se află camera prin modificarea articulației 4 a brațului.

Capitolul 5. Experimente realizate

5.1 Baza de date

5.1.1 Baza de date inițială

Pe parcursul dezvoltării proiectului a fost necesară utilizarea unei baze de date cu imagini faciale care să conțină diverse persoane. Această bază de date a fost folosită în scopul pregătirii modelelor utilizate de algoritmi pentru a-i putea analiza și pentru a putea rula programul, deoarece aceștia au nevoie de imagini faciale de la cel puțin 2 persoane pentru a face recunoașterea.

Baza de date folosită este Extended Yale Face Database B. [11] Aceasta conține un număr de 38 de persoane, fiecare dintre ele având atașate 64 de imagini. Imaginile disponibile pentru fiecare subiect reprezintă diverse condiții de iluminare și sunt stocate în format pgm, fiind reprezentate în spațiul de luminanță. Rezoluția acestora este de 192x168 pixeli. Toate imaginile din baza de date au fost prelucrate manual, fiind aliniate, decupate și redimensionate către rezoluția menționată anterior de către autorii lucrării „Acquiring Linear Subspaces for Face Recognition under Variable Lighting”, Kuang-Chih Lee, Jeffrey Ho, și David Kriegman. [12]

Figura următoare conține câteva exemple din imaginile disponibile pentru subiectul B02 din baza de date:

Figura 5.1 Exemple de imagini din Yale Face Database B

5.1.2 Baza de date ulterioară

Pentru a putea efectua diverse teste și pentru a putea observa comportamentul programului în diferite stagii a fost nevoie de extinderea bazei de date. Astfel, am integrat inițial în baza de date imagini preluate folosind telefonul mobil, apoi am decupat și convertit manual fiecare imagine către spațiul de luminanță și către o rezoluție identică cu cea a imaginilor din baza de date. Ulterior, după obținerea cu succes a noului model, care să conțină și informații extrase din imaginile adăugate de mine, am integrat detecția facială pentru a folosi camera video a calculatorului pentru achiziția de noi imagini. În timpul rulării programul executa convertirea imaginii în spațiul de luminanță, apoi aplică procedeul de detecție facială, iar fața detectată era decupată, redimensionată către rezoluția necesară și apoi stocată în format pgm.

În figură se pot observa câteva din imaginile achiziționate inițial. Obiectivul a fost capturarea acestora în diferite condiții de iluminare.

Figura 5.2 Exemple de imagini proprii adăugate

În forma finală a proiectului, persoanele noi sunt adăugate în baza de date prin intermediul aplicației ce rulează pe server.

5.2 Interfața grafică

5.2.1 Interfața grafică a programului pe server

Folosind setul de dezvoltare Qt am putut implementa o interfață grafică pentru a face utilizarea programului mai facilă. Acest program este menit a fi folosit doar de către un administrator pentru a nu permite oricui să adauge persoane în sistem.

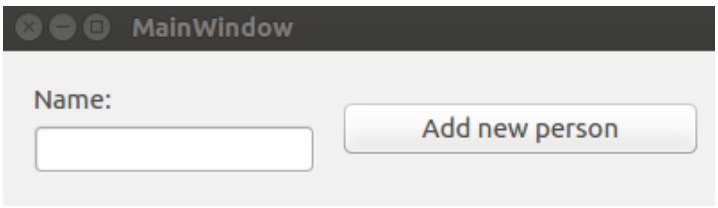

Figura 5.3 Interfața programului rulat pe server

Interfața conține o casetă pentru introducerea textului, ce va fi folosită pentru stocarea numelui subiectului ce va fi adăugat, și un buton, ce are rolul de a porni achiziția cadrelor. Nu toate cadrele sunt păstrate, ci la interval de o secundă se salvează doar unul, iar numărul total de cadre păstrate este 20. În cazul în care nu se introduce un nume, programul nu va rula și va afișa un mesaj care să evidențieze lipsa numelui introdus. În momentul achiziției, cadrele păstrate sunt convertite în spațiu de luminanță. După achiziția cadrelor, programul le parcurge, pe rând, execută detecția, decupează fețele detectate și le salvează în baza de date. Următorul pas este citirea imaginilor existente în baza de date, urmat de recalcularea modelului utilizat de recunoașterea facială folosind și fețele detectate în timpul achiziției curente, apoi salvează noul model. Rezoluția utilizată pentru achiziție este de 1280x720, iar numărul reprezentat în cadru semnifică numărul imaginii achiziționate.

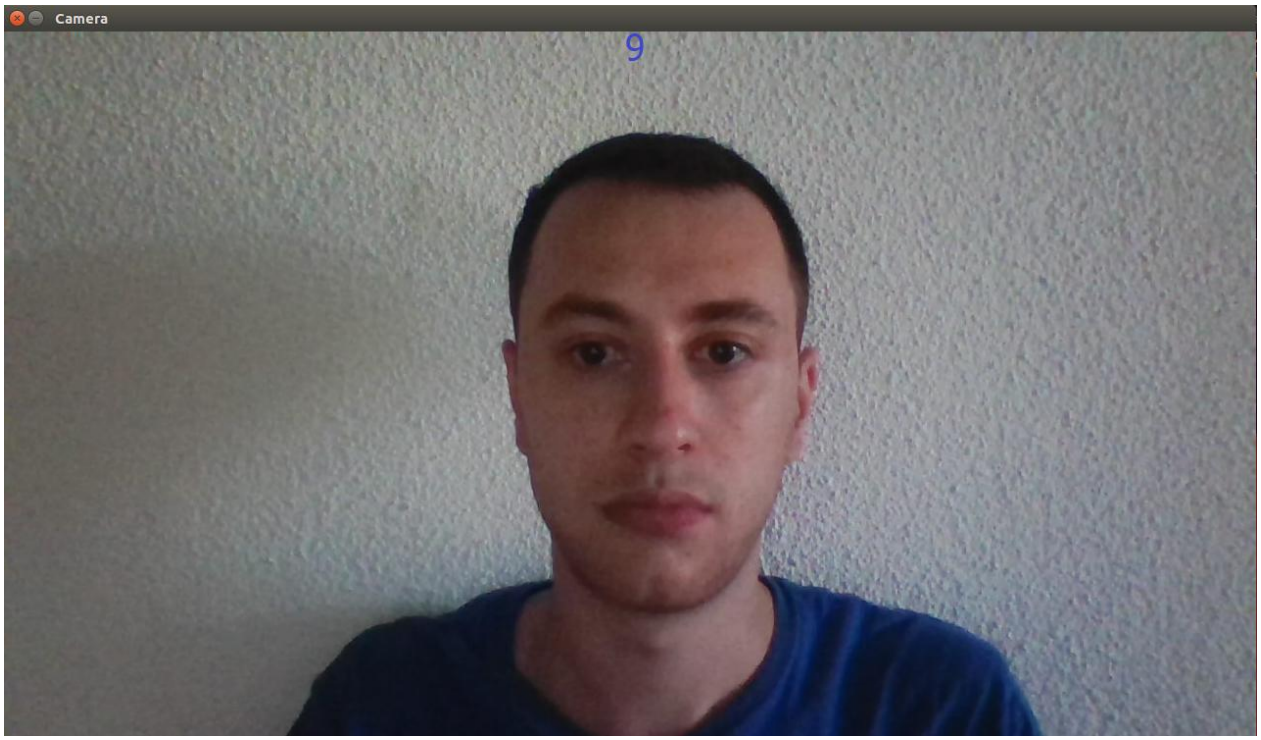

Figura 5.4 Achiziția imaginilor pentru subiect nou

5.2.2 Interfața grafică a programului pe youBot

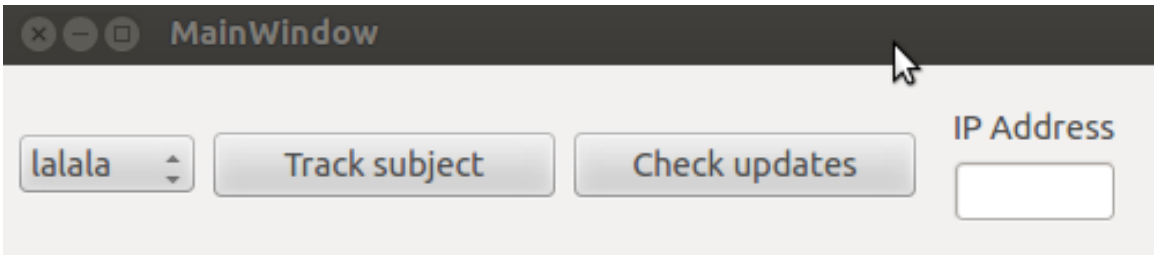

Figura 5.5 Interfața pe youBot

Interfața este compusă din 4 elemente: casetă text pentru introducerea adresei IP a calculatorului folosit pentru adăugarea persoanelor noi în sistem, buton pentru verificare dacă există pe server un model calculat mai recent decât cel actual pe youBot, acest buton având și rolul de a încărca lista subiecților disponibili pentru autentificare și urmărire, o listă verticală ce permite selectarea unui subiect și un buton pentru a lansa execuția programului de urmărire. În cazul în care adresa IP introdusă nu este validă, se va afișa un mesaj la apăsarea butonului ce verifică dacă există actualizări ale modelului. Dacă există o actualizare disponibilă, programul va transfera modelul nou prin protocolul HTTP, folosindu-se de biblioteca libcurl.

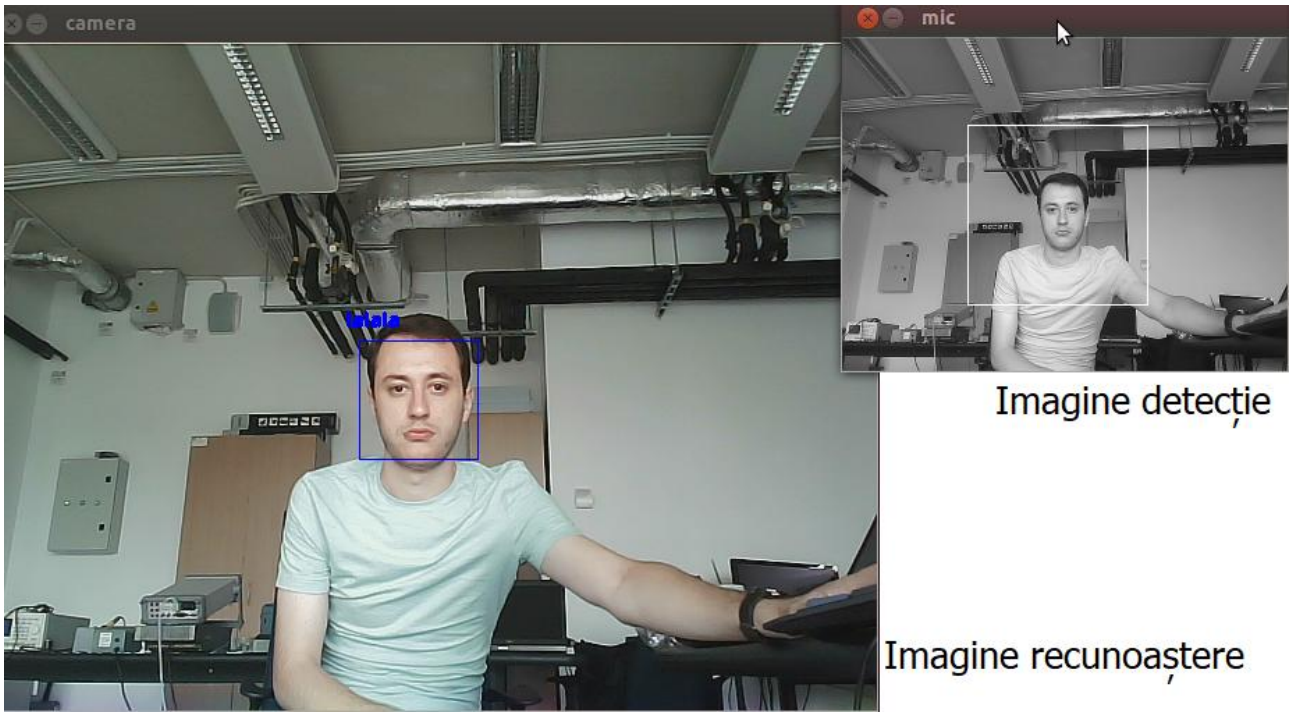

Figura 5.6 Urmărirea unui subiect

În imaginea anterioară este exemplificată rularea progrmaului de recunoaștere pe youBot, captura de ecran fiind făcută pe acesta. În imaginea mică, alb-negru, afișată în partea dreaptă, observăm apariția chenarului ce marchează poziția precendenta a feței autentificate, în timp ce în imaginea din stânga este recunoscută fața și se aplică un chenar albastru pentru a marca recunoașterea acesteia.

5.2.3 Mișcarea robotului

În urma autentificării, robotul va răspunde, efectuând diverse mișcări, în funcție de poziția în imagine a feței autentificate, raportată la imaginea folosită pentru detecție, adică la imaginea cu rezoluția de 320x240. Mișcările efectuate pot fi:

- Rotire în plan orizontal
- Ridicare sau coborâre a camerei
- Deplasare pe direcția înainte sau înapoi

Figura 5.7 Limitele analizate pentru mișcarea robotului

În figura anterioară sunt marcate cu roșu limitele orizontale, folosite pentru a determina dacă este necesară rotirea în plan orizontal, cu verde sunt reprezentate limitele folosite pentru a determina dacă este necesară ridicarea sau coborârea camerei, iar punctul albastru reprezintă centrul imaginii de 320x240 pixeli folosită în cadrul detecției, acesta fiind situat la coordonatele (160, 120).

Pentru rotire, programul analizează poziționarea pe orizontală a feței determinate, mai exact, centrul acesteia, din care folosește coordonata pentru planul orizontal. Cunoscând următoarele coordonate ale feței: punctul stânga sus al chenarului ce o marchează, cu coordonatele (x_f, y_f) și lățimea și înălțimea acestuia, centrul feței se poate determina ca aflându-se în poziția (x_f + lățime/2, y_f + înălțime/2). Dreapta verticală roșie din partea dreaptă a imaginii este perpendiculară pe abscisă în punctul 180, iar dreaptă roșie din partea stângă este perpendiculară pe abscisă în punctul 140. Pentru a stabili dacă trebuie să efectueze rotirea, programul compară coordonata x a centrului feței, xf + lățime/2, cu coordonata x a celor două drepte roșii, pe rând. Astfel că dacă centrul feței are valoarea mai mare decât 180, robotul va efectua o mișcare de rotire spre dreapta. Dacă centrul feței are valoarea mai mică decât 140, robotul va efectua o mișcare de rotire spre stânga. Mișcările se efectuează până când în următoarele cadre centrul orizontal al feței ajunge la centrul orizontal al imaginii, adică la poziția de 160.

Comandarea ridicării sau coborârii camerei se face prin compararea coordonatei y a centrul feței, situată la poziția y_f + înălțime/2, cu coordonata y a dreptelor marcate cu verde. Dreapta de deasupra centrului are coordonata y egală cu 90, iar cealaltă are coordonata egală cu 150. Astfel, dacă centrul determinat al feței are coordonata y mai mică decât 90, robotul va ridica camera video și va aduce centrul feței în centrul imaginii în cadrele următoare. Dacă centrul feței are coordonata y mai mare decât 150, robotul va coborî camera.

Mișcarea pe direcția înainte sau înapoi se realizează prin comparare lățimii dreptunghiului ce încadrează fața cu 2 valori determinate experimental. Astfel, dacă lățimea chenarului este mai mică decât valoarea de 75, atunci robotul se va mișca înainte până când lățimea feței va fi mai mare

decât 80. Dacă lățimea chenarului este mai mare decât valoarea 95, atunci robotul se va mișca înapoi până când lățimea acestuia va deveni mai mică decât valoarea 90.

Codul sursă utilizat pentru efectuarea celor 3 mișcări se regăsește atașat în Anexa 1.

5.2.4 Serverul Web și transferul modelului

Metoda implementată pentru a transfera modelul pe youBot se bazează pe rularea unui server web pe calculatorul folosit pentru adăugarea persoanelor noi în sistem. Astfel, a fost instalat pe server programul Apache2.

Configurarea serverului este foarte simplă, acesta necesitând doar specificarea unui director ce va fi folosit drept rădăcina serverului. Acest director poate conține la rândul lui fișiere și alte directoare cu conținut divers. Folosindu-mă de serverul apache pentru a transfera 3 fișiere, ce se pot observa în imagine, nu a fost nevoie de crearea unor directoare suplimentare, fișierele fiind plasat în directorul rădăcină folosit de server.

Serverul web poate fi accesat și prin browser, folosindu-se protocolul HTTP. Browser-ul știe cum să interpreteze un director dintr-un server web ce conține doar fișiere sau alte directoare, afișându-l precum în figură:

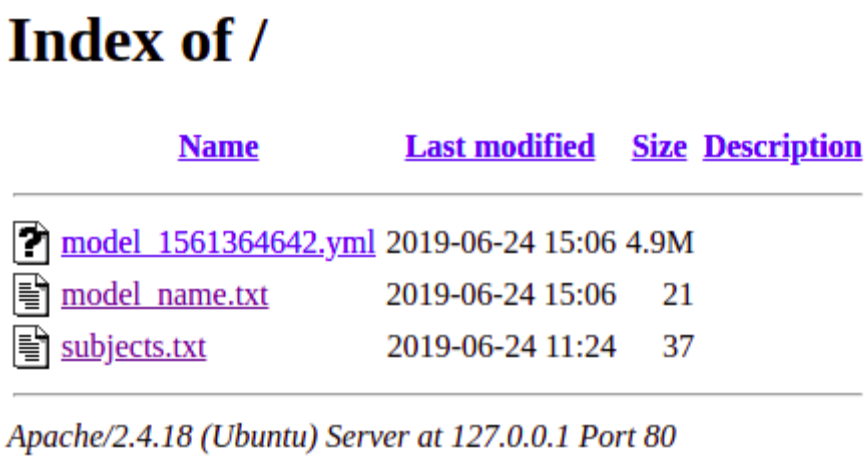

Figura 5.8 Conținutul serverului web

Fișierul „model_name.txt” conține numele modelului, fișierul „subjects.txt” conține linii ce descriu persoanele folosite în calculul modelului, persoane ce se regăsesc și în baza de date, iar fișierul „model_1561364642.yml” reprezintă modelul în sine. Deoarece numele modelului variază în funcție de timpul local al sistemului, în secunde, calculat de la data de 1 Ianuarie 1970, a trebuit găsită o metodă pentru a ști pe youBot care este numele acestuia pentru a-l transfera. Astfel, a fost creat și utilizat fișierul „model_name.txt”, fișier ce are mereu același nume, însă conținutul acestuia se modifică. Când se inițiază procedura de actualizare pe youBot, acesta aduce mai întâi fișierul „model_name.txt”, verifică conținutul acestuia și dacă marcajul său de timp este mai recent față de marcajul de timp regăsit în continul vechiului fișierul „model_name.txt” (care se regăsea până atunci pe youBot), atunci este transferat modelul de pe server împreună cu noul fișier „subjects.txt”. Fișierul „subjects.txt” conține linii sub formă „index;Nume_Subiect;Număr_Poze” și este folosit pe server pentru a ști ce subiecți se regăsesc în baza de date și câte poze are fiecare, iar pe youBot este folosit pentru a ști ce nume corespunde subiectului cu un anumit index pentru a-l afișa în imagine

când fața lui este recunoscută. Indexul reprezintă, de fapt, clasa fiecărui subiect folosită în calculul modelului.

Pentru a putea accesa serverul web prin browser sau pentru a transfera fișierele folosind procedura de actualizare a modelului este necesar ca youBot și calculatorul folosit drept server să fie conectate la aceeași rețea.

5.3 Algoritmul ales

Pentru a alege algoritmul optim utilizat de recunoașterea facială în timpul rulării pe youBot au fost impuse următoarele condiții:

- Acuratețe cât mai ridicată
- Timp necesar pentru clasificarea unei imagini cât mai redus

Astfel, pentru a putea măsura parametrii enumerați anterior, au fost realizate 6 teste pentru fiecare algoritm. Un test este reprezentat de alegerea a câte 2 imagini pentru testare pentru fiecare subiect în parte și calcularea modelului cu restul de imagini. Imaginile alese pentru testare sunt aceleași pentru fiecare din cei 3 algoritmi. Toate testele se efectuează în aceeași manieră, ținând cont de faptul că trebuie alese alte 2 imagini, reprezentând altă condiție de iluminare, pentru testare, pentru fiecare subiect.

Testele au fost realizate pe baza de date Yale Face Database B, fără persoane adăugate de mine, folosind imaginile redimensionate de la rezoluția 192x168 la 48x42. Având 38 de subiecți și folosind 2 imagini pentru testarea algoritmului de recunoaștere pentru subiect, obținem numărul de 76 de imagini testate. Procentajul marcat în tabelele ce conțin rezultate este raportat la cele 76 de imagini și se calculează folosind raportul dintre numărul cazurilor favorabile și numărul cazurilor posibile.

Trebuie menționat și faptul că testele au fost rulate pe un calculator cu resurse superioare față de resursele calculatorului intern al lui youBot.

5.3.1 Testarea algoritmului Eigenfaces

Condiții de iluminare	Acuratețe (%)	Timp mediu de recunoaștere per imagine (ms)
Iluminare bună	88.15	6.4
Iluminare bună spre medie	100	8.2
Iluminare medie	100	7.4
Iluminare medie spre slabă	96.05	6.9
Iluminare slabă	100	7.3
Iluminare foarte slabă	61.84	7.5

Tabel 5.1 Rezultatele testării algoritmului Eigenfaces

5.3.2 Testarea algoritmului Fisherfaces

Condiții de iluminare	Acuratețe (%)	Timp mediu de recunoaştere per imagine (ms)
Iluminare bună	100	0.6
Iluminare bună spre medie	100	0.6
Iluminare medie	100	0.7
Iluminare medie spre slabă	100	0.6
Iluminare slabă	100	0.6
Iluminare foarte slabă	100	0.6

Tabel 5.2 Rezultatele testării algoritmului Fisherfaces

5.3.3 Testarea algoritmului LBPH

Condiții de iluminare	Acuratețe (%)	Timp mediu de recunoaştere per imagine (ms)
Iluminare bună	100	89.6
Iluminare bună spre medie	100	89.7
Iluminare medie	100	88.2
Iluminare medie spre slabă	100	88.3
Iluminare slabă	100	88.5
Iluminare foarte slabă	100	88.9

Tabel 5.3 Rezultatele testării algoritmului LBPH

Ținând cont de rezultatele obținute, observăm că ambii algoritmi, Fisherfaces și LBPH, au obținut procentaj maxim în ceea ce privește acuratețea, însă diferența majoră dintre aceștia este data de timpul de execuție necesar. Astfel, observăm că folosirea algoritmului LBPH pentru recunoaștere ar limita rularea programului pe calculatorul folosit la 11-12 cadre pe secundă. Luând în calcul și timpul necesar execuției celorlalte procedee, precum detecția facială, achiziția și conversia imaginii, am obține un număr aproape înjumătățit al cadrelor pe secundă. Durata necesară se datorează faptului că pentru fiecare imagine ce trebuie testată, cu rezoluția de 48x42, trebuie împărțită imaginea în regiuni identice, apoi pentru fiecare regiune trebuie comparat fiecare pixel cu vecinii săi și trebuie obținut codul LBP al acestuia. Următorul pas este reprezentat de realizarea histogramei regiunii, iar în final, de concatenarea histogramelor pentru a obține vectorul de trăsături corespunzător imaginii testate. Clasa determinată de algoritmul LBPH a imaginii testate este dată de clasa celui mai asemănător vector de trăsături determinat pentru fiecare imagine folosită în calculul modelului. Vectorul de trăsături ar conține, în acest caz, peste 1900 de componente. Calculul acestora și apoi determinarea celui mai asemănător vector de trăsături necesită timpul determinat în timpul testării.

Pe de altă parte, Fisherfaces are nevoie doar de matricea transformării cu același nume obținută în timpul calculului modelului pentru a găsi proiecția vectorului imaginii în subspațiul determinat. Astfel, se realizează reducerea numărului de componente de la 2016 (48x42) la 37

(numărul de clase – 1). În acest caz se determină mult mai rapid cel mai asemănător vector de trăsături Fisher, iar imaginea este clasificată în aproximativ 0.7 ms.

Utilizând condițiile enumerate anterior, se deduce că algoritmul optim pentru realizarea recunoașterii faciale pe youBot este Fisherfaces, deoarece am obținut o acuratețe ridicată și un timp de procesare scurt, nefiind necesare compromisuri.

Capitolul 6. Concluzii

6.1 Concluzii generale

Obiectivul principal al acestei lucrări a fost dezvoltarea unui sistem autonom care să folosească ca metodă de autentificare pe cea facială, iar apoi să permită controlarea unei platforme robotice prin mișcarea persoanei autorizate în cadrul captat de camera utilizată. Alcătuirea proiectului este făcută din 2 module:

- modulul de recunoaștere facială ce a integrat concepte de prelucrări de imagini și clasificare
- modulul de deplasare a robotului ce permite mișcarea acestuia în funcție de poziția curentă a persoanei autorizate

În configurația actuală, programul dezvoltat pe youBot își poate găsi utilitatea în cadrul companiilor ce folosesc deja platforme și roboți pentru automatizarea completă sau aproape completă a muncii.

6.2 Concluzii specifice

Pentru implementarea proiectului a fost necesară analiza algoritmilor de recunoaștere facială pentru stabilirea celui mai eficient. Eficiența a constat în obținerea unei acurateți cât mai mari și a unui timp minim necesar până la luarea unei decizii. După testarea fiecăruia, ajungând la concluzia că pentru sistemul limitat disponibil, cea mai bună alegere este reprezentată de algoritmul Fisherfaces.

Inițial, a fost dezvoltat un algoritm secvențial care să realizeze achiziția imaginilor, apoi procesarea lor, realizarea detecției pentru a obține pozițiile fețelor încadrate în imagine pentru a testa autentificarea, folosind algoritmul de recunoaștere facială, pentru fiecare în parte. Timpul necesar analizei unei imagini fiind destul de mare, a apărut necesitatea integrării unor soluții de optimizare pentru îmbunătățirea programului, astfel că:

- a fost redusă rezoluția cu care sunt captate imaginile la 640x480
- imaginile din baza de date și imaginile pentru persoanele noi adăugate au fost redimensionate de la 192x168 pixeli la 48x42 pixeli
- detecția a fost realizată pe o imagine de 320x240 pixeli, pozițiile fețelor fiind translatate ușor pentru a se potrivi pe imaginea inițială, de 640x480
- consecință a micșorării imaginii pentru detecție a fost faptul că rezoluția minimă a fețelor ce puteau fi detectate de algoritm era de 24x24, acest lucru corespunzând unei depărtări față de robot de aproximativ 2.5-3 m, însă acest lucru a fost suficient pentru testarea în laborator
- au fost folosiți ambii clasificatori pentru detecția facială, încercând utilizarea predominantă a clasificatorului LBP, însă în situațiile cu lumină ambientală nedistribuită uniform pe fața subiectului a fost nevoie de aplicarea clasificatorului Haar pentru a obține detecția
- a fost păstrată poziția în imaginea anterioară a feței persoanei autorizate, incercandu-se mai întâi detecția și recunoașterea în aceeași regiune în noua imagine
- a fost folosită biblioteca Phtread și principiul benzii rulante pentru a putea paraleliza procesele de achiziție, respectiv prelucrare, detecție, recunoaștere și comandarea mișcărilor robotului, folosind 4 fire de execuție

După îmbunătățirea codului rulat pe youBot a apărut problema adăugării noilor persoane în sistem. Fiind o sarcină intens computațională, am găsit soluția de a folosi un calculator cu resurse superioare lui youBot pentru recalcularea modelului folosit de algoritm. Astfel, a fost dezvoltat un program pentru a fi rulat pe acel calculator folosit drept server, pentru a capta imaginile persoanelor ce se doresc a fi adăugate în baza de date și pentru a recalcula modelul.

Următoarea problemă întâmpinată a fost modul de obținere al modelului nou determinat pe youBot. Soluția acesteia este reprezentată de folosirea unui server web pe calculatorul utilizat drept server pentru putea aduce pe youBot, prin protocolul HTTP, modelul nou. În cadrul programului rulat pe youBot a fost folosită biblioteca libcurl pentru transferul fișierelor.

Ultimul pas a fost implementarea unei interfețe grafice pentru cele 2 aplicații, făcând utilizarea lor intuitivă și ușoară, folosind setul de dezvoltare Qt.

Bibliografie

[1] -, „KUKA – Wikipedia” disponibil online la adresa https://en.wikipedia.org/wiki/KUKA, accesat la data: 15.06.2019
[2] -, „Orbbec Company Profile – Orbbec” disponibil online la adresa https://orbbec3d.com/orbbec-company-profile/, accesat la data: 15.06.2019
[3] -, „YouBot Detailed Specifications - youBot wiki” disponibil la adresa http://www.youbot-store.com/wiki/index.php/YouBot_Detailed_Specifications, accesat la data: 15.06.2019
[4] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001, Kauai, HI, USA, 2001, pp. I-I., doi: 10.1109/CVPR.2001.990517
[5] -, „OpenCV: Face Detection using Haar Cascades” disponibil la adresa https://docs.opencv.org/3.4.1/d7/d8b/tutorial_py_face_detection.html, accesat la data: 16.06.2019
[6] -, „Face Recognition with OpenCV - OpenCV 2.4.13.7 documentation”, disponibil la adresa https://docs.opencv.org/2.4/modules/contrib/doc/facerec/facerec_tutorial.html, accesat la data: 17.06.2019
[7] -, „Victor Neagoe-Curs RFIA-PCA Si LDA - [PDF Document]” disponibil la adresa https://vdocuments.mx/victor-neagoe-curs-rfia-pca-si-lda.html, accesat la data: 17.06.2019
[8] -, „OpenCV – Wikipedia” disponibil la adresa https://en.wikipedia.org/wiki/OpenCV, accesat la data: 17.06.2019
[9] -, „Qt (software)” disponibil la adresa https://en.wikipedia.org/wiki/Qt_(software), accesat la data: 17.06.2019
[10] -, „cURL – Wikipedia” disponibil la adresa https://en.wikipedia.org/wiki/CURL, accesat la data: 18.06.2019
[11] -, „vision.ucsd.edu/~iskwak/ExtYaleDatabase/ExtYaleB.html” disponibil la adresa http://vision.ucsd.edu/~iskwak/ExtYaleDatabase/ExtYaleB.html, accesat la data: 18.06.2019
[12] K.C. Lee, J. Ho ,D. Kriegman, „Acquiring Linear Subspaces for Face Recognition under Variable Lighting”, IEEE Trans. Pattern Anal. Mach. Intelligence, No. 5, Vol. 27, 2005, Pages 684-698

.

Anexa 1

```
	// variabila enable_move_direction imi arata daca se desfasoara deja o
miscare inainte/inapoi
	//    pentru a stii sa nu incep o alta miscare inainte/inapoi
	// variabilele forward si backward imi spun ce miscare se realizeaza cand
enable_move_direction este setat ca true
	static void move_direction(bool &enable_move_direction, bool &forward,
bool &backward, int &face_width, quantity<si::velocity> &longitudinalVelocity) {
		if (!enable_move_direction) {
			// daca latimea fetei este mai mica decat 76, atunci robotul
trebuie deplasat inainte
			if (face_width <= 75) {
				longitudinalVelocity = 0.15 * meter_per_second;
				enable_move_direction = true;
				forward = true;
				backward = false;
			}
			// daca latimea fetei este mai mare decat 94, atunci robotul
trebuie deplasat inapoi
			if (face_width >= 95) {
				longitudinalVelocity = -0.15 * meter_per_second;
				enable_move_direction = true;
				forward = false;
				backward = true;
			}
		}
		else {
			// daca este setata variabila forward, atunci a inceput
miscarea inainte
			if (forward)
				// executa miscarea inainte pana cand fata se apropie si
are latimea mai mare decat 80
				if (face_width > 80) {
					enable_move_direction = false;
					longitudinalVelocity = 0 * meter_per_second;
					forward = false;
					backward = false;
				}
				else
					longitudinalVelocity = 0.15 * meter_per_second;
			// daca este setata variabila backward, atunci a inceput
miscarea inapoi
			if (backward)
				// executa miscarea inapoi pana cand fata departeaza si
are latiema mai mica decat 90
				if (face_width < 90) {
					enable_move_direction = false;
					longitudinalVelocity = 0 * meter_per_second;
					forward = false;
					backward = false;
				}
			else
				longitudinalVelocity = -0.15 * meter_per_second;
		}
	}
```

```
	// functie de calcul a valorii pentru variabila angularVelocity, ce
specifica viteza de rotatie
	// semnul acesteia semnifica directia rotatiei:
	// semn pozitiv inseamna ca rotirea se va face in sens trigonometric, iar
semn pozitiv semnifica rotirea in sensul acelor
	// de ceasornic
	static void rotate_body(bool &enable_rotate_body, bool &rotate_right, bool
&rotate_left, int &face_cols, quantity<si::0.15ocity> &angularVelocity) {
		int centru_cols = 160;
		// daca nu ruleaza deja rotatia intr-un sens
		if (!enable_rotate_body) {
			// daca pozitia centrului fetei pe orizontala se afla la mai
mult de valoarea 180
			// incep rotatia in sensul antitrigonometric
			if (centru_cols + 20 <= face_cols) {
				angularVelocity = -0.15 * radian_per_second;
				rotate_right = true;
				rotate_left = false;
				enable_rotate_body = true;
			}
			// daca pozitia centrului fetei pe orizontala se afla la mai
putin de valoarea 140
			// incep rotatia in sens trigonometric
			if (centru_cols - 20 >= face_cols) {
				angularVelocity = 0.15 * radian_per_second;
				rotate_right = false;
				rotate_left = true;
				enable_rotate_body = true;
			}
		}
		// daca ruleaza deja miscarea de rotatie
		else {
			// daca robotul se roteste in sens antitrigonometric
			if (rotate_right)
				// daca fata a depasit centrul orizontal opresc miscarea
				if (centru_cols > face_cols) {
					rotate_right = false;
					rotate_left = false;
					enable_rotate_body = false;
					angularVelocity = 0 * radian_per_second;
				}
			// daca robotul se roteste in sens trigonometric
			if (rotate_left)
				// daca fata a depasit centrul orizontal opresc miscarea
				if (centru_cols < face_cols) {
					rotate_right = false;
					rotate_left = false;
					enable_rotate_body = false;
					angularVelocity = 0 * radian_per_second;
				}
		}
	}


	// functie de calcul a unghiului sub care trebuie situat actuatorul 4, cel
ce controleaza pozitionarea
	// verticala a camerei
	static void move_camera(int face_rows, double& vert_center_radians,
JointAngleSetpoint& desiredJointAngle) {
		// obtin distanta de la centrul fetei pana la centrul imaginii
		double pixel_distance_vert = face_rows - 120;
		double vert_multiplier = 0.0006;
		double next_vert_center_radians;
```

```
            // daca distanta este mai mare decat 30 atunci calculez noul unghi
              if (abs(pixel_distance_vert) > 30) {
                    next_vert_center_radians = vert_center_radians +
pixel_distance_vert * vert_multiplier;
                  // unghiurile extreme admise sunt 2.7 si 3.2, orice unghi nou
ce nu este cuprins
                  // in acest interval nu va fi folosit
                  if (next_vert_center_radians > 2.7 && next_vert_center_radians
< 3.2)
                        vert_center_radians = next_vert_center_radians;
            }
              desiredJointAngle.angle = vert_center_radians * radian;
      }
```