**Universitatea Politehnica Timişoara**

**Facultatea de Mecanică**

**Departamentul de Mecatronică**

# Proiect de diplomă

## *Sistem pentru detectare facială*

Coordonator:

Șl. Dr. Ing. Cristian MOLDOVAN

Student:

Filip Albin Rus

Timişoara, 2017

Cuprins

Plan tematic pentru lucrarea de licenţă

*Lucrarea de licenţă acordată studentului***:** Filip-Albin Rus

1. Tema proiectului: Sistem de detectare facială

2. Termenul de predare al proiectului: Iunie 2017

3. Elemente iniţiale pentru proiect:
   Imagini digitale

4. Conţinutul (enumerarea problemelor care vor fi rezolvate):
   Algoritmi de recunoastere și procesare a imaginilor în OpenCV+Python pentru detectarea personajelor dintr-o imagine

5. Consultaţii pentru proiect (cu indicarea părţilor din proiect care necesită consultarea)

6. Data eliberării temei: Octombrie 2015

*Tema a fost primită pentru rezolvare.*

Data……  Coordonatorul proiectului

Filip-Albin Rus  Șl. Dr. Ing. Cristian Moldovan

*(semnătura)*  *(semnătura)*

Rezumat

Scopul acestei lucrări este de a extrage informații dintr-un sistem foto sau de monitorizare video în scopul identificării în timp real a numărului de persoane din scena observată. Tehnicile folosite în dezvoltarea lucrării pot fi folosite și pentru extragerea/identificarea altor informații.

Aceste sisteme pot fi folosite în scopul creșterii și îmbunătățirii securității, domeniu în permanentă dezvoltare și de interes pentru întreaga societate.

Având în vedere nivelul ridicat de dezvoltare al tehnologiei în acest domeniu, tehnologia permite identificarea de persoane, amprente sau chiar autentificare biometrică.

Astfel, în această lucrare s-a abordat procesul de detectare a numărului de persoane într-un cadru și s-a propus o soluție de sistem automat ce folosește senzorul foto a unui camere si procesul de prelucare si procesare de imagine.

# 1.Introducere

## 1.1. Noțiuni generale

Mecatronica este o cooperare a diferitelor domenii dominante, mecanică, electronică și software, în fază de concepție. Trei faze sunt specifice: faza de concepție, faza de proiectare cu evidențierea disciplinelor specifice, faza de implementare când domină disciplinele tradiționale. [1]

Un sistem mecatronic conține în general 4 sub-sisteme. Acestea sunt:

- De calcul
- Informațional
- Mecanic
- Electric

Mecatronica s-a născut ca tehnologie şi a devenit foarte curând filosofie care s-a răspândit în întreaga lume. În ultimii ani mecatronica este definită simplu: ştiinţa maşinilor inteligente. Apariţia mecatronicii este rezultatul firesc al evoluţiei în dezvoltarea tehnologică. În opinia japonezilor, mecatronica este tehnologia mecanică cerută de societatea informaţională. [8]

## 1.2 Scopul lucrării

Scopul acestei lucrări este de a extrage informații dintr-un sistem foto sau de monitorizare video în scopul identificării în timp real a numărului de persoane din scena observată. Tehnicile folosite în dezvoltarea lucrării pot fi folosite și pentru extragerea/identificarea altor informații.

## 1.3 Motivația alegerii temei

Am ales această temă din dorința de a facilita dezvoltarea de diferite programe prin care se pot prelua anumite informații de la senzori și apoi să se acționeze diferite dispoztive din mediul ambiant, dezvoltarea făcandu-se pe un calculator de mici dimensiuni, fiabil, care este

introdus în tot mai multe școli din străinătate și organizații care promovează acestă idee de a face dezvoltarea software cât mai accesibilă tuturor.

## 2.Generalități

Tehnicile de recunoaștere de imagini digitale folosesc rezultatele și metodele matematice din recunoașterea formelor, inteligența artificială, psiho-fiziologice, știința calculatoarelor, electronica și multe alte discipline științifice. Pentru a simplifica sarcina înțelegerii viziunii computerizate, putem distinge în lanțul algoritmic două nivele: procesarea de nivel scăzut a imaginii și înțelegerea de nivel înalt a imaginii.

Procesarea de nivel înalt se bazează pe cunoaștere , utilizând algoritmi ce duc la un sfârșit scopul propus. Metodele de inteligență artificială sunt folosite în cele mai multe cazuri. Viziunea computerizată de nivel înalt încearcă sa imite cunoașterea umană și capacitatea de a lua decizii potrivit informației conținute in imagine.

Viziunea computerizată și deci procesul de recunoaștere, sunt strâns legate de cunoașterea a-priori a conținutului imaginii. O imagine poate fi descrisă printr-un model formal, dar acest model nu rămâne neschimbat. Deși modelul inițial se poate compune din cunoaștere a-priori generală, procesarea de nivel înalt extrage în mod continuu noi informații din imagini, reînnoiește și clarifică cunoașterea.

Majoritatea metodelor de procesare a imaginii au fost propuse în anii `70. Cercetarea recentă încearcă să găsească algoritmi mai eficienți și să realizeze aplicarea acestora pe mașini paralele care să ușureze cantitatea mare de operații necesare prelucrării.

O problemă complicată si nerezolvată până acum este specificarea automată a etapelor de pre-procesare necesare rezolvării unei anumite sarcini.Operatorul uman este cel care de obicei alege succesiunea de operații, iar atingerea scopului propus depind mult de intuiția și exprerința anterioară a acestuia. În anii ’80 multe proiecte s-au concentrat pe această problemă folosind sistemele expert și permițând astfel folosirea experienței din domeniu.

În ultimii ani, interesul înspre detecția și analiza mișcării a crescut odată cu dezvoltarea unor noi metodologii de analiză precum și a creșterii capacităților de procesare a calculatoarelor. La intrarea unui sistem de analiză a mișcării se regăsește de obicei o secvență de imagini, ceea ce duce la creșterea considerabilă a volumului de date prelucrate. Analiza

mișcării este de cele mai multe ori implementată în sistemele de analiză în timp real, cum ar fi orientarea unui robot autonom. O altă problemă tratată de domeniul analizei mișcării este aceea de a obține informații legate de obiectele prezente în imagini, incluzând atât obiectele aflate în mișcare cât și cele statice.[2]

## 2.1 OpenCV

OpenCV sau Open Computer Vision este o bibliotecă open-source disponibilă pe www.opencv.org. Bibilioteca este scrisa in limbajul de programare C și C++ și poate fi rulată sub următoarele sisteme de operare: Linux, Windows si Mac OS X. Biblioteca este în permanentă dezvoltare pentru inferfețe precum Python, Ruby, Matlab sau alte limbaje de programare.

OpenCV a fost proiectat pentru performanță computațională redusă și cu un accent deosebit pe aplicațiile în timp real. OpenCV este scris în C optimizat și poate profita de procesoarele multicore.

Unul dintre obiectivele OpenCV este acela de a oferi o platformă software de vedere artificială (machine vision) care ajută oamenii să creeze rapid aplicații de viziune destul de sofisticate. Biblioteca OpenCV conține peste 500 de funcții care acoperă multe zone ale vederiii artificiale, inclusiv inspecția vizuală produselor în fabrici, imagistică medicală, securitate, interfața cu utilizatorul, calibrarea camerei, vedere stereo și robotică. [6]

Deoarece computer vision și machine learning nu sunt automatizate, OpenCV conține, de asemenea, o bibliotecă de machine learning (MLL), cu scop general. Această sub-bibliotecă se axează pe recunoașterea și gruparea statistică a modelelor.

Drept definiție ideea de computer vision este procesul de tranformare a datelor dintr-o cameră foto sau o cameră video într-o decizie sau o nouă reprezentare. Aceste transformări sunt efectuate pentru a obținute un anumit rezultat dorit. Datele de intrare pot include anumite informații precum că aparatul foto este instalat într-un autovehicul sau că distanța până la obiectul încadrat în obiectiv este de 1m. O altă reprezentare a conceputului de computer vision este transformarea unei imagini într-o imagine din tonuri de gri sau eliminarea mișcării camerei dintr-o secvență de imagini. [6]

Într-un sistem de machine vision, un procesor primește o matrice de numere de la camera foto, dar în general nu există recunoaștere încorporată a modelului, nici un control automat al focalizării și diafragmei,ca urmare sistemele de viziune sunt încă destul de naive.

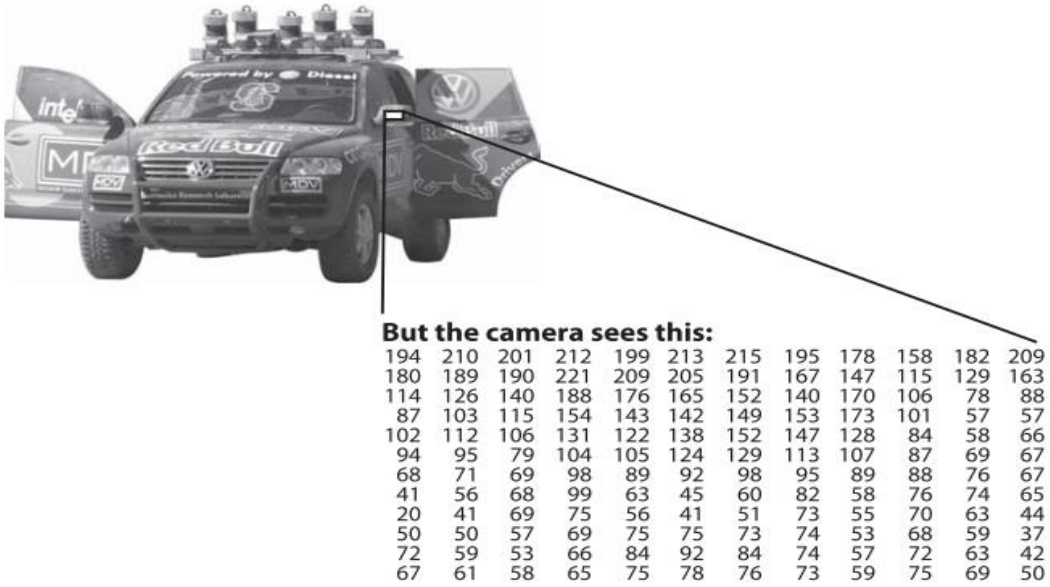


Fig 2.1.1 Codificarea unei imagini [6]

Orice valoare dată în această grilă are o componentă de zgomot destul de mare și astfel, prin ea însăși oferă puține informații, dar acesta este maximul de informații ce se poate obține de la un sistem machine vision. Sarcina devine apoi să se transforme această rețea de cifre în percepție. Figura 2.1.1 prezintă modul în care este codificată imaginea și intuiește dificultățile procesării de imagine.

De fapt, problema așa cum a fost expusă până acum se complică și mai mult și devine aproape imposibil de rezolvat, având în vedere o imagine 2D a unei lumi 3D, și faptul că nu există o modalitate unică de a reconstrui semnalul 3D. [6]

Aceeași imagine 2D ar putea reprezenta oricare dintre combinațiile infinite de scene 3D, chiar dacă datele ar fi perfecte. Cu toate acestea, după cum am menționat deja, datele sunt corupte de zgomot și distorsiuni. Asemenea influență provine din variațiile de vreme, iluminare, reflexii, mișcări, imperfecțiunile în obiectiv și reglarea mecanică, timpul de integrare a senzorului (blur de mișcare), zgomotul electric din senzor.

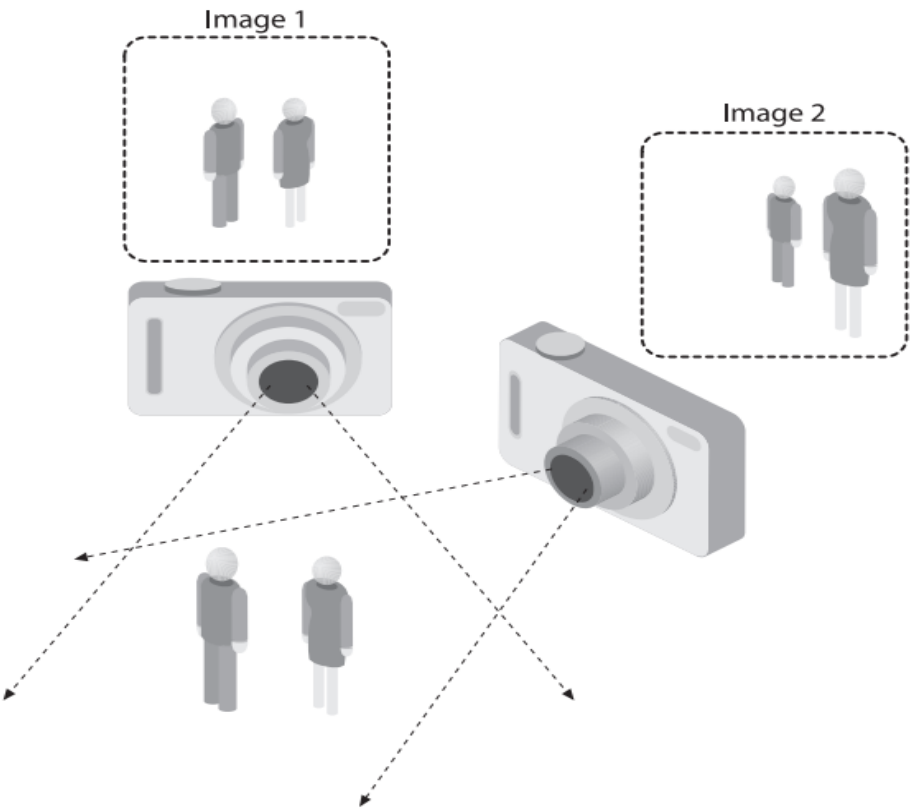


Fig 2.1.2 Perspectiva unei imagini din diferite unghiuri[6]

Următoarea problemă cu care se înfruntă computer vision este zgomotul din imagine. În mod obișnuit, abordăm zgomotul prin utilizarea metodelor statistice. De exemplu, este imposibil să se detecteze o margine dintr-o imagine doar prin compararea unui punct cu vecinii săi imediați. Dar dacă privim statisticile pe o regiune locală, detectarea muchiilor devine mult mai ușoară. O margine reală ar trebui să apară ca un șir de astfel de comparații de pixeli vecini peste o regiune locală, fiecare în concordanță cu vecinii săi. De asemenea, este posibilă compensarea zgomotului prin analiză statistică în timp a imaginii.

OpenCV vizează furnizarea instrumentelor de bază necesare pentru a rezolva problemele de vedere a machine vision. În unele cazuri, funcționalitățile la nivel înalt în bibliotecă vor fi suficiente pentru a rezolva problemele mai complexe ale computer vision. Componentele de bază din bibliotecă sunt suficiente pentru a permite crearea unei soluții complete pentru orice problemă de computer vision.

OpenCV a evoluat dintr-o inițiativă Intel Research pentru a avansa aplicațiile cu solicitare intenstivă a procesoarelor.

OpenCV este în mare parte structurat in cinci mari componente, patru dintre ele fiind reprezentate în Fig 2.1.3

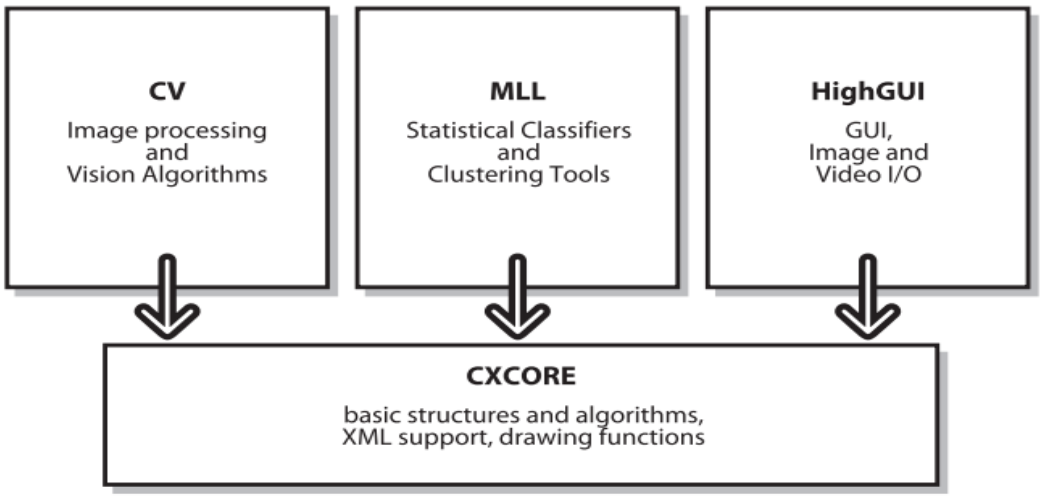


Fig 2.1.3 Componentele OpenCV[6]

Componentele OpenCV conțin procesarea de bază a imaginii și algoritmi de vedere artificială la nivel superior. Biblioteca ML (machine learning) include clasificatoare statistice și instrumente de grupare a datelor. HighGUI conține rutine I/O și funcții pentru stocarea și încărcarea imaginilor, iar CXCore conține structurile de date și conținutul de bază.

Figura 2.1.3 nu include componenta CvAux, care conține recunoașterea chipului încorporat HMM (Hidden Markov model), cât și algoritmi experimentali(segmentare fundal/prim-plan).

## 2.2 Machine Learning

Obiectivul de învățare a mașinilor (ML) * este transformarea datelor în informații. Dacă învățăm dintr-o colecție de date, vrem ca o mașină să poată răspunde la întrebări despre date: Ce alte date sunt cele mai asemănătoare cu aceste date? Există o mașină în imagine? La ce anunț va răspunde utilizatorul? Există o componentă de cost, astfel încât această întrebare ar putea deveni: "Din produsele din care facem cea mai mare sumă de bani, care dintre ele va cumpăra cel mai probabil utilizatorul dacă le arătăm un anunț pentru acesta?". Învățarea prin mașină transformă datele în informații prin extragerea de reguli sau modele din acele date.

Învățarea în mașină funcționează pe date cum ar fi valorile temperaturii, prețurile acțiunilor, intensitățile de culoare și așa mai departe. S-ar putea, de exemplu, să luăm o bază de

date cu 10.000 de imagini de față, să executăm un detector de margini pe fețe și apoi să colectăm caracteristici cum ar fi direcția marginii, rezistența marginilor și distanța din centrul feței pentru fiecare față. Dacă vrem doar să vedem cum se încadrează fețele în grupuri diferite (largi, înguste, etc.), atunci un algoritm de grupare ar fi alegerea potrivită. Dacă vrem să învățăm să prezicem vârsta unei persoane de la (să spunem) modelul marginilor detectate pe fața lui, atunci ar fi potrivit un algoritm clasificator. Pentru a ne atinge obiectivele, algoritmii de învățare automată analizează caracteristicile colectate și ajustează greutățile, pragurile și alți parametri, pentru a maximiza performanța în funcție de aceste obiective. Procesul de ajustare a parametrilor pentru a atinge un obiectiv este ceea ce înțelegem prin termenul de învățare. [6]

Este întotdeauna important să știți cât de bine funcționează metodele de învățare a mașinilor și aceasta poate fi o sarcină subtilă. În mod tradițional, se rupe setul original de date într-un set de antrenament mare (probabil 9000 de fețe, în exemplul nostru) și un set mai mic de testare (restul de 1000 de fețe). Apoi, putem rula clasificatorul nostru peste setul de antrenament pentru a învăța modelul nostru de predicție a vârstei, dat fiind vectorii caracteristicilor de date. Când am terminat, putem testa clasificatorul de predicție a vârstei pe imaginile rămase din setul de testare.

Setul de testare nu este folosit în antrenament și nu lăsăm clasificatorului să "vadă" etichetele de vârstă stabilite pentru test. Rulam clasificatorul peste fiecare dintre cele 1.000 de fețe din setul de date de testare și înregistrăm cât de bine împlinesc vârstele pe care le prezice de la vectorul de caracteristici care corespund vârstelor reale. Dacă clasificatorul nu reușește, am putea încerca să adăugăm noi caracteristici la datele noastre sau să luăm în considerare un tip diferit de clasificator. Vom vedea în acest capitol că există multe tipuri de clasificatori și mulți algoritmi pentru instruirea acestora.

În cazul în care clasificatorul se descurcă bine, acum avem un model potențial valoros pe care îl putem folosi pe datele din lumea reală. Poate că acest sistem va fi folosit pentru a seta un joc video bazat pe vârstă. În timp ce persoana se pregătește să joace, fața lui va fi procesată (direcția muchiei, rezistența margini, setată de la centrul feței). Aceste date vor fi transmise clasificatorului. Vârsta pe care o revine va stabili comportamentul jocului în consecință. După ce a fost desfășurat, clasificatorul vede fețe pe care niciodată nu le-a văzut și ia decizii în funcție de ceea ce a învățat pe setul de antrenament. [6]

Algoritmii de învățare a mașinilor inclusi în OpenCV sunt prezentați în Tabelul 2.2.1. Toți algoritmii se află în biblioteca ML, cu excepția lui Mahalanobis și K-means, care sunt în CVCORE și detectarea feței, care se află în CV.

| Algoritm | Descriere |
|---|---|
| Mahalanobis | O măsură de distanță care explică "întinderea" spațiului de date prin împărțirea covarianței datelor. Dacă covarianța este matricea identității (varianța identică), atunci această măsură este identică cu măsura de distanță Euclidiană. [Mahalanobis36] |
| K-means | Un algoritm de grupare nesupravegheat care reprezintă o distribuție a datelor utilizând centrele K, unde K este aleasă de utilizator. Diferența dintre acest algoritm și maximizarea așteptărilor constă în faptul că aici centrele nu sunt Gaussian.Inventat de către Steinhaus[Steinhaus56] și folosit de către [Lloyd57] |
| Normal/Naïve Bayes classifier | Un clasificator generativ în care se presupune că caracteristicile sunt distribuite și independente din punct de vedere statistic unul de altul, o presupunere puternică care, în general, nu este adevărată. Din acest motiv, se numește adesea un clasificator "naiv Bayes". Cu toate acestea, această metodă funcționează adesea surprinzător de bine. [Maron61; Minsky61] |
| Decision trees | Un clasificator discriminator. Arborele găsește o caracteristică de date și un prag la nodul curent care împarte cele mai bune date în clase separate. Datele sunt împărțite și |

| | |
|---|---|
| | repetăm recursiv procedura de pe ramurile din stânga și din dreapta ale copacului. Este de multe ori primul lucru pe care ar trebui să încercați, deoarece este rapid și are o funcționalitate ridicată. [Breinman84] |
| Boosting | Un grup discriminator de clasificatori. Decizia de clasificare generală se face din deciziile de clasificare ponderate combinate ale grupului de clasificatori. În formare, învățăm un grup de clasificatori unul câte unul. Fiecare clasificator din grup este un clasificator "slab" (doar puțin peste performanța șansei). Aceste clasificatoare slabe sunt în mod obișnuit compuse din arbori de decizie cu o singură variabilă, numiți "stumps". În cursul formării, bătaia de decizie își învață deciziile de clasificare din date și, de asemenea, învață o pondere pentru "votul" său din precizia sa asupra datelor. Între instruirea fiecărui clasificator unul câte unul, punctele de date sunt re-ponderate astfel încât să se acorde o atenție mai mare punctelor de date în care s-au făcut erori. Acest proces continuă până la eroarea totală asupra setului de date, care rezultă din votul ponderat combinat al arborilor de decizie, scade sub un prag stabilit. Acest algoritm este deseori eficient atunci când există o cantitate mare de date de antrenament. [Freud97] |
| Random trees | În timpul învățării, fiecărui nod din fiecare arbore este permis să aleagă variabilele de divizare numai dintr-un subset aleator al |

| | |
|---|---|
| | caracteristicilor de date. Acest lucru ajută la asigurarea faptului că fiecare copac devine factor decizional independent din punct de vedere statistic. În modul de funcționare, fiecare copac primește un vot nepăsător. Acest algoritm este adesea foarte eficient și poate de asemenea să efectueze regresia prin medierea numerelor de ieșire din fiecare arbore. [Breiman01] |
| Face detector /Haar classifier | O aplicație de detectare a obiectelor bazată pe o utilizare inteligentă a amplificării. Distribuția OpenCV vine cu un detector de față frontal instruit care funcționează remarcabil de bine. Puteți programa algoritmul pe alte obiecte cu software-ul furnizat. Funcționează bine pentru obiecte rigide. [Viola04] |
| Expectation maximization (EM) | Un algoritm generator nesupravegheat care este utilizat pentru gruparea. Aceasta poate fi o modalitate eficace de a reprezenta o distribuție mai complexă, cu doar câțiva parametri (mijloace și variante). Deseori folosit în segmentare. [Dempster77] |
| K-nearest neighbors | Cea mai simplă clasificare discriminatorie posibilă. Datele de instruire sunt pur și simplu stocate cu etichete. Ulterior, un punct de date de testare este clasificat în funcție de votul majoritar al celor mai apropiate puncte de date K (într-un sens euclidian de apropiere). Acesta este probabil cel mai simplu lucru pe care îl puteți face. Este adesea eficient, dar este lent și necesită multă memorie. [Fix51]. |

| | |
|---|---|
| Neural networks /Multilayer perceptron (MLP) | Un algoritm discriminativ care (aproape întotdeauna) are "unități ascunse" între nodurile de ieșire și de intrare pentru a reprezenta mai bine semnalul de intrare. Poate fi lent pentru tren, dar este foarte rapid pentru a alerga. Totuși, performantul de top pentru lucruri precum recunoașterea literelor [Werbos74; Rumelhart88] |
| Support vector machine (SVM) | Un clasificator discriminativ care poate face regresie. Este definită o funcție de distanță între oricare două puncte de date dintr-un spațiu de dimensiuni mai mari. Tind să fie printre cele mai bune, cu date limitate, pierzându-se la algoritmul de Boosting sau cel de Random trees doar atunci când sunt disponibile seturi de date mari [Vapnik95]. |

Tabel 2.2.1 Clasificare algoritmi Machine Learning [6]

## 2.3 Detectarea feței sau clasificatorul Haar

Caracteristicile de tip Haar sunt caracteristicile imaginii digitale folosite în recunoașterea obiectelor.

Viola și Jones au adaptat ideea de a folosi funcțiile Haar și au dezvoltat așa-numitele caracteristici Haar. O caracteristică Haar consideră regiunile dreptunghiulare adiacente într-o locație specifică într-o fereastră de detecție, însumează intensitățile pixelilor în fiecare regiune și calculează diferența dintre aceste sume. Această diferență este apoi utilizată pentru a clasifica subsecțiunile unei imagini. De exemplu, să spunem că avem o bază de date cu chipuri umane. Este o observație obișnuită că între toate fețele regiunea ochilor este mai întunecată decât cea a obrajilor. Prin urmare, o caracteristică obișnuită de haos pentru detectarea feței este un set de două dreptunghiuri adiacente care se află deasupra ochiului și a regiunii obrazului. Poziția acestor dreptunghiuri este definită în raport cu o fereastră de detecție care acționează ca o casetă de legare la obiectul țintă (fața în acest caz).[11]

În faza de detecție a cadrului de detectare a obiectelor Viola-Jones, o fereastră a dimensiunii țintă este deplasată peste imaginea de intrare, iar pentru fiecare subsecțiune a imaginii se calculează caracteristica Haar. Această diferență este apoi comparată cu un prag învățat care separă non-obiectele de obiecte. Deoarece o astfel de caracteristică Haar este doar un cursant slab sau un clasificator (calitatea sa de detecție este puțin mai bună decât ghicirea aleatoare), un număr mare de caracteristici asemănătoare lui Haar sunt necesare pentru a descrie un obiect cu o precizie suficientă. În cadrul de detectare a obiectelor Viola-Jones, caracteristicile Haar sunt astfel organizate în ceva numit cascadă de clasificatori pentru a forma un cursant puternic sau un clasificator.

Avantajul cheie al unei caracteristici asemănătoare lui Haar față de cele mai multe alte caracteristici este viteza de calcul. Datorită utilizării imaginilor integrale, o caracteristică asemănătoare cu Haar de orice dimensiune poate fi calculată în timp constant.[11]

Una dintre contribuțiile lui Viola și Jones a fost să utilizeze tabelele cu sumare, pe care le-au numit imagini integrale. Imaginile integrale pot fi definite ca tabele de căutare bidimensionale sub forma unei matrice cu aceeași dimensiune a imaginii originale. Fiecare element al imaginii integrate conține suma tuturor pixelilor localizați în regiunea stângă sus a imaginii originale (în funcție de poziția elementului). Aceasta permite calcularea sumei

suprafețelor rectangulare din imagine, în orice poziție sau scară, folosind doar patru căutări.Găsirea sumei zonei dreptunghiulare umbrite.

$sum = I(C) + I(A) - I(B) + I(D)$, unde punctele A, B, C, D aparțin imaginii integrale I [11]

Punctele sunt prezentate în figură.

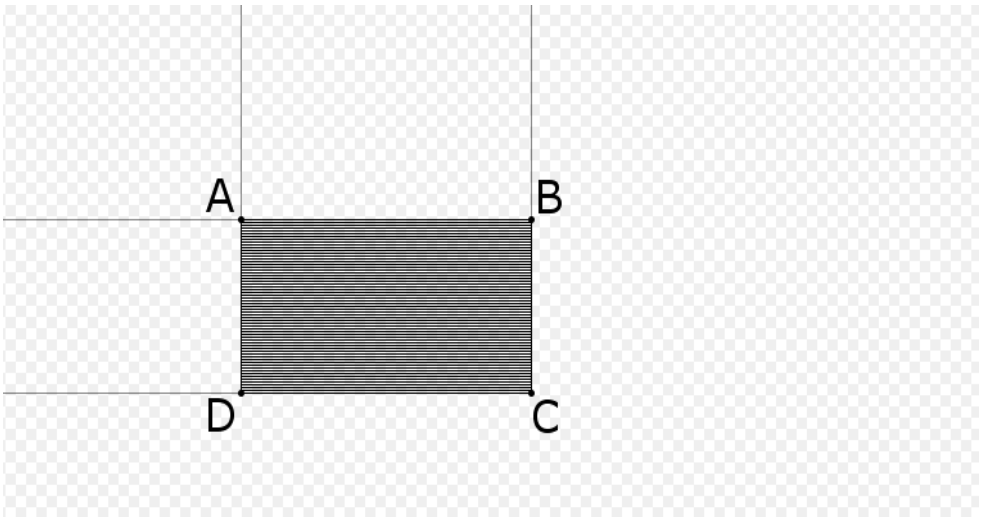


Fig 2.3.1 Suma zonei umbrite [11]

Fiecare funcție Haar poate necesita mai mult de patru căutări, în funcție de modul în care a fost definit. Caracteristicile lui Viola și Jones cu 2 dreptunghiuri au nevoie de șase căutări, funcțiile cu 3 dreptunghiuri necesită opt căutări, iar funcțiile cu 4 dreptunghiuri necesită nouă căutări.[11]

## 2.4 Teorie clasificatorului Viola-Jones

Clasificatorul Viola-Jones utilizează AdaBoost la fiecare nod din cascadă pentru a afla o rată ridicată de detecție la costul clasificatorului cu mai multe tensiuni la fiecare nod al cascadei. Acest algoritm include câteva caracteristici inovatoare.

1) Utilizează caracteristicile de intrare asemănătoare lui Haar: un prag aplicat sumelor și diferențelor diferențiale ale regiunilor imaginii dreptunghiulare.

2) Tehnica sa integrală de imagine permite calcularea rapidă a valorii regiunilor dreptunghiulare sau a unor astfel de regiuni rotind la 45 de grade. Această structură de date este utilizată pentru a accelera calculul caracteristicilor de intrare ca Haar.
3) Utilizează amplificarea statistică pentru a crea noduri de clasificare binare (față-non-față) care se caracterizează printr-o detecție ridicată și o respingere slabă.
4) Organizează nodurile slab clasifere ale unei cascade de respingere. Cu alte cuvinte: primul grup de clasificatori este selectat care detectează cel mai bine regiunile imaginii care conțin un obiect, permițând în același timp multe detectări greșite; Următorul grup clasificator este al doilea cel mai bun la detectarea cu respingere slabă; si asa mai departe. În modul de testare, un obiect este detectat numai dacă îl face prin întreaga cascadă. [6]

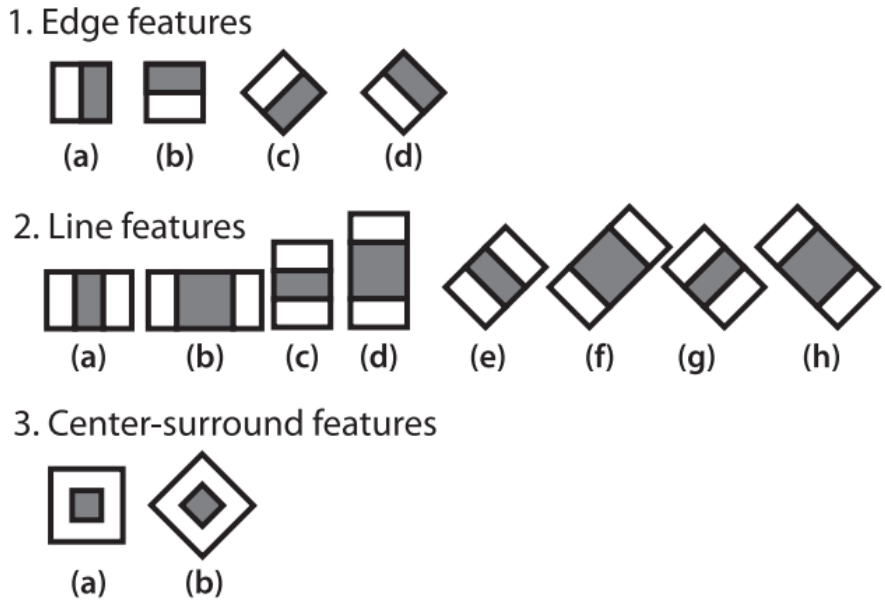


Fig 2.4.1 Clasificatorea Haar din biblioteca OpenCV

Viola și Jones au organizat fiecare grup clasificator în noduri dintr-o cascadă de respingere. Fiecare nod $F_j$ conține o întreagă cascadă de grupare a pașilor de decizie instruiți cu privire la caracteristicile Haar de la caracteristicile unei fețe sau non-fețe. În mod obișnuit, nodurile sunt ordonate de la cele mai puțin complexe la cele mai complexe, astfel încât calculele să fie reduse la minimum. De obicei, creșterea în fiecare nod este reglată pentru a avea o rată foarte mare de detectare.

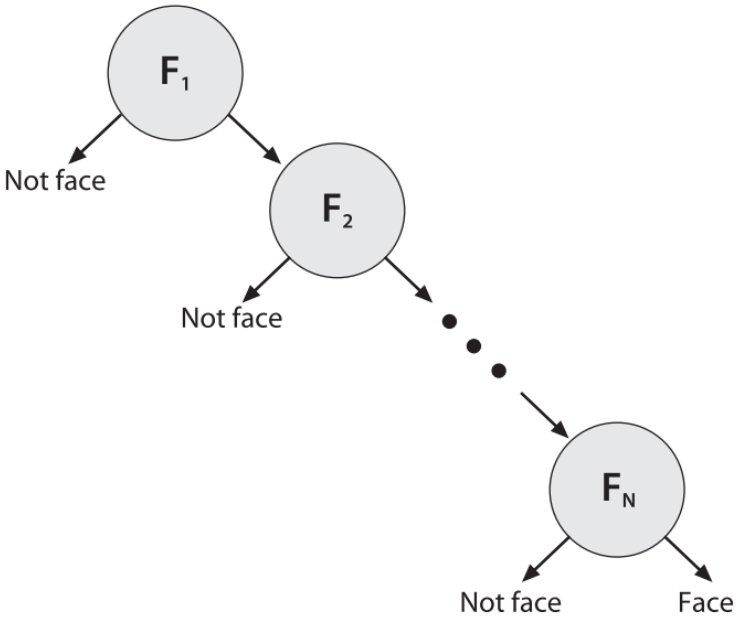


Fig 2.4.2 Cascada de respingere clasificată de către Viola-Jones [6]

## 2.5 Python

Python este un limbaj de programare dinamic multi-paradigmă, creat în 1989 de programatorul olandez Guido van Rossum. Van Rossum este și în ziua de astăzi un lider al comunității de dezvoltatori de software care lucrează la perfecționarea limbajul Python și implementarea de bază a acestuia, CPython, scrisă în C. Python este un limbaj multifuncțional folosit de exemplu de către companii ca Google sau Yahoo! pentru programarea aplicațiilor *web*, însă există și o serie de aplicații științifice sau de divertisment programate parțial sau în întregime în Python. Popularitatea în creștere, dar și puterea limbajului de programare Python au dus la adoptarea sa ca limbaj principal de dezvoltare de către programatori specializați și chiar și la predarea limbajului în unele medii universitare. Din aceleași motive, multe sisteme bazate pe Unix, inclusiv Linux, BSD și Mac OS X includ din start interpretatorul CPython.

Python pune accentul pe curățenia și simplitatea codului, iar sintaxa sa le permite dezvoltatorilor să exprime unele idei programatice într-o manieră mai clară și mai concisă decât în alte limbaje de programare ca C. În ceea ce privește paradigma de programare, Python poate servi ca limbaj pentru software de tipul *object-oriented*, dar permite și programarea imperativă, funcțională sau procedurală. Sistemul de tipizare este dinamic iar administrarea memoriei

decurge automat prin intermediul unui serviciu „gunoier” (*garbage collector*). Alt avantaj al limbajului este existența unei ample biblioteci standard de metode.

Implementarea de referință a Python este scrisă în *C* și poartă deci numele de *CPython*. Această implementare este software liber și este administrată de fundația *Python Software Foundation*. [5]

## 2.6 Raspberry Pi

Raspberry Pi este un SBC(Single-Board-Computer) de dimensiunile unui card de credit, produs in UK de către Raspberry Pi Foundation cu scopul de a promova învățarea noțiunilor de bază din domeniul informaticii în școli. [3]

Chiar dacă are dimensiuni reduse (85mm x 56mm), Raspberry Pi este un calculator complet permițând funcționalități obișnuite precum rularea unui sistem de operare (Linux sau Windows) și rularea de aplicații utilizator (jocuri, editoare de text, medii de programare, redarea de muzică și filme, aplicații de teleconferință, aplicații Internet). Diferențele între o placă Raspberry Pi și un calculator personal (PC) sau laptop constau atât în dimensiunea redusă a plăcii cât și în puterea mai mică de calcul a acesteia – nu are aceleași performanțe de calcul precum un PC desktop care are un cost și o dimensiune de câteva ori mai mari. Putem compara placa Raspberry Pi cu o tabletă sau cu un sistem de tip NetBook dar fără a dispune de ecran și tastatură. În plus, placa Raspberry Pi oferă posibilitatea de a conecta diverse componente electronice specifice sistemelor embedded: senzori, butoane, ecrane LCD sau pe 7 segmente, drivere de motoare, relee etc. Posibilitatea de a personaliza sistemele de programe (sistemul de operare, aplicațiile) și posibilitatea de interconectare cu alte componente electronice fac din placa Raspberry Pi un sistem de calcul ce poate sta la baza unor proiecte personale extrem de interesante și de puternice – un calculator ce poate fi integrat în sisteme electronice și mecanice proiectate și realizate de utilizator.

În ciuda dimensiunii reduse placa Raspberry Pi 3 dispune de periferice integrate numeroase acoperind complet funcționalitatea unui sistem de calcul (audio, video, porturi USB, conectivitate de rețea):

- Procesor SoC pe 64 de biți din familia ARMv8-A, Broadcom BCM2837, ce lucrează la o frecvență de 1.2GHz și dispune de 4 nuclee de tip ARM Cortex-A53;

- 1GB de memorie RAM (folosită și ca memorie video, partajată cu procesorul grafic);
- Procesor grafic Broadcoam VideoCore IV 3D integrat pe aceeași pastilă de siliciu ca și procesorul principal;
- Ieșire digitală video / audio HDMI;
- Ieșire analogică video (composite video) / audio mixtă prin intermediul unei mufe jack 3.5mm;
- Mufă de rețea RJ45 Ethernet 10/100 Mbit/s;
- Conectivitate WiFi 802.11n;
- Conectivitate Bluetooth 4.1 / BLE;
- 4 porturi USB 2.0;
- 40 de pini de intrare / ieșire (GPIO);
- Slot card de memorie microSD
- Conectori dedicați pentru cameră video (CSI) și afișaj (DSI);

Pentru a pune în funcțiune placa Raspberry Pi 3 avem nevoie de următoarele componente suplimentare:

- Cablu HDMI și un monitor / televizor cu intrare HDMI. În cazul în care nu dispunem de un monitor / televizor cu intrare HDMI putem utiliza un adaptor HDMI-DVI sau un adaptor HDMI-VGA.
- Alimentator de rețea cu ieșire de 5V, minim 2.5A și mufă microUSB. Este recomandată utilizarea alimentatorului oficial sau a unui alimentator de calitate care asigură o tensiune corectă și un curent suficient pentru alimentarea plăcii Raspberry Pi 3.
- Tastatură și mouse USB. Sunt necesare pentru instalarea și configurarea inițială a sistemului.
- Card de memorie microSD, capacitate minimă 8GB, clasă de viteză 10.
- Dacă sistemul va fi utilizat într-o rețea locală pe cablu este necesar și un cablu de rețea UTP – patch-cord. Dacă se utilizează placa într-o rețea locală WiFi nu este necesar.

- Opțional, dar recomandat, este utilizarea și a unei carcase pentru placa Raspberry Pi. [4]

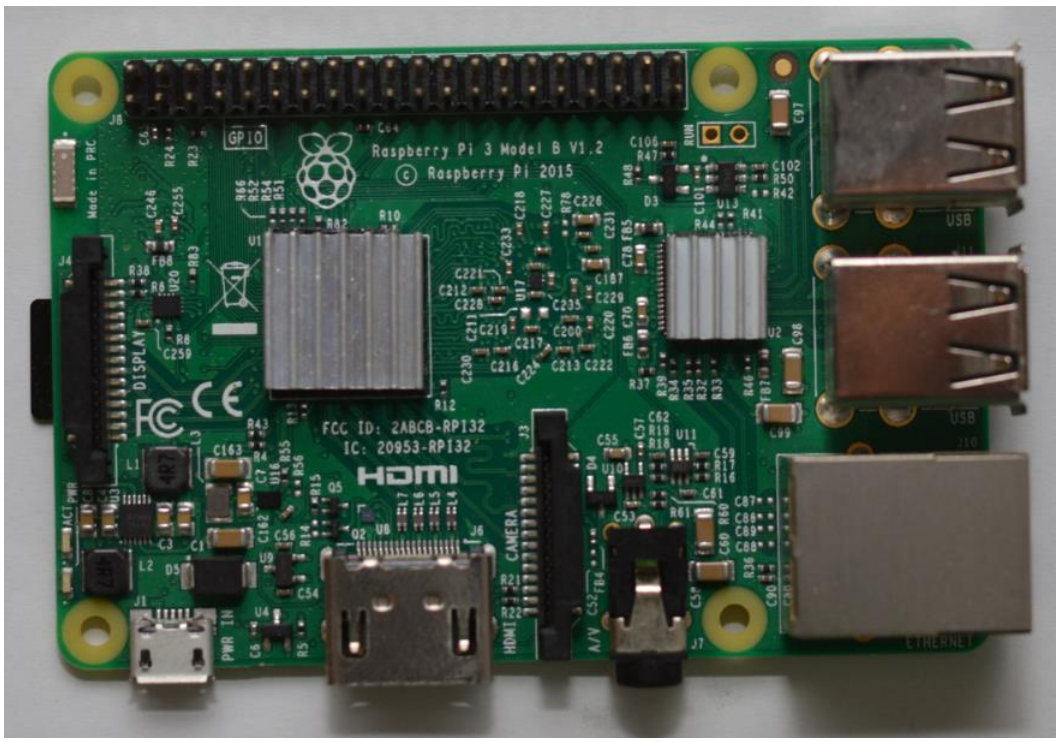


Fig 2.6.1 Placa Raspberry Pi 3 Model B

## 2.7 PiCamera

Modulul de cameră RaspberryPi este un add-on personalizat pentru Raspberry Pi. Se atașează la Raspberry Pi în portul alăturat ieșirii digitale HDMI. Locul este indicat prin afișarea cuvântului Camera.Această interfață utilizează interfața dedicată CSI, care a fost concepută special pentru interfața cu camerele. Magistrala CSI este capabilă de rate de date extrem de ridicate și poartă exclusiv date pixel.

Placa în sine este mică, de aproximativ 25mm x 20mm x 9mm. De asemenea, cântărește puțin peste 3g, făcând-o perfectă pentru aplicații mobile sau alte aplicații în care mărimea și greutatea sunt importante. Se conectează la Raspberry Pi printr-un cablu de panglică scurt. Aparatul foto este conectat la procesorul BCM2835 de pe magistrala Pi prin magistrala CSI, o legătură de lățime de bandă mai mare, care transportă date pixel din camera foto înapoi la procesor. Această magistrala transport informația de-a lungul cablului de panglică care atașează placa camerei de luat vederi la dispozitivul Pi.

Senzorul însuși are o rezoluție nativă de 5 megapixeli și are un obiectiv fix pe placă. În ceea ce privește imaginile statice, aparatul foto este capabil de imagini statice de 2592 x 1944 pixeli și susține, de asemenea, filme video de 1080p30, 720p60 și 640x480p60.

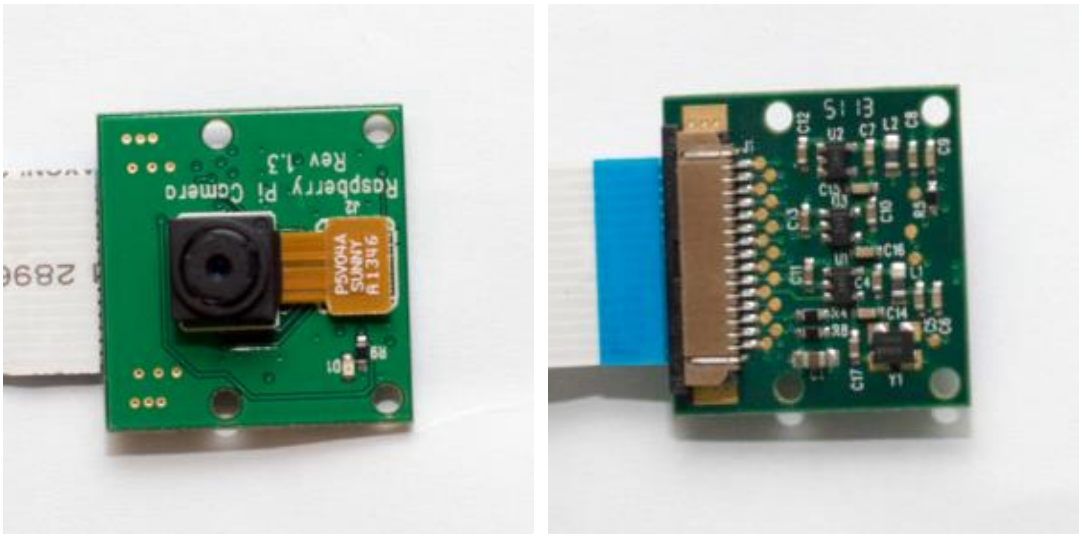


Fig 2.7.1 Camera Raspberry Pi V1

# 3. Inițializarea sistemului

Pentru a putea începe lucrul efectiv, în placa Raspberry Pi trebuie să fie introdus cardul de memorie microSD. Înainte de a face acest lucru însă trebuie să se pregătească sistemul de operare dacă acesta nu vine pre-instalat. Raspberry Pi poate rula diverse distribuții de sisteme de operare, precum Linux și o versiune minimală a sistemului de operare Microsoft Windows 10, respectiv Windows 10 IoT Core. În următorii pași se poate observa cum se instalează distribuția Linux Raspbian, sistemul de operare oficial oferit de către Raspberry Pi a cărui grad de folosință este foarte ușor și e recomandat utilizatorilor noi în acest domeniu.

Dacă avem un card de memorie microSD oficial de la producătorul Raspberry Pi, instalarea sistemului de operare este mai facilă deoarece cardul de memorie conține utilitarul NOOBS(New Out of the Box Software) ce pune la dispoziția utilizatorului o interfață grafică de unde se pot alege diferitele sisteme de operare suportate de către Raspberry Pi ce pot fi instalate. Pentru a porni instalarea cardul microSD trebuie să fie introdus în slot iar apoi se poate alimenta placa Raspberry Pi.

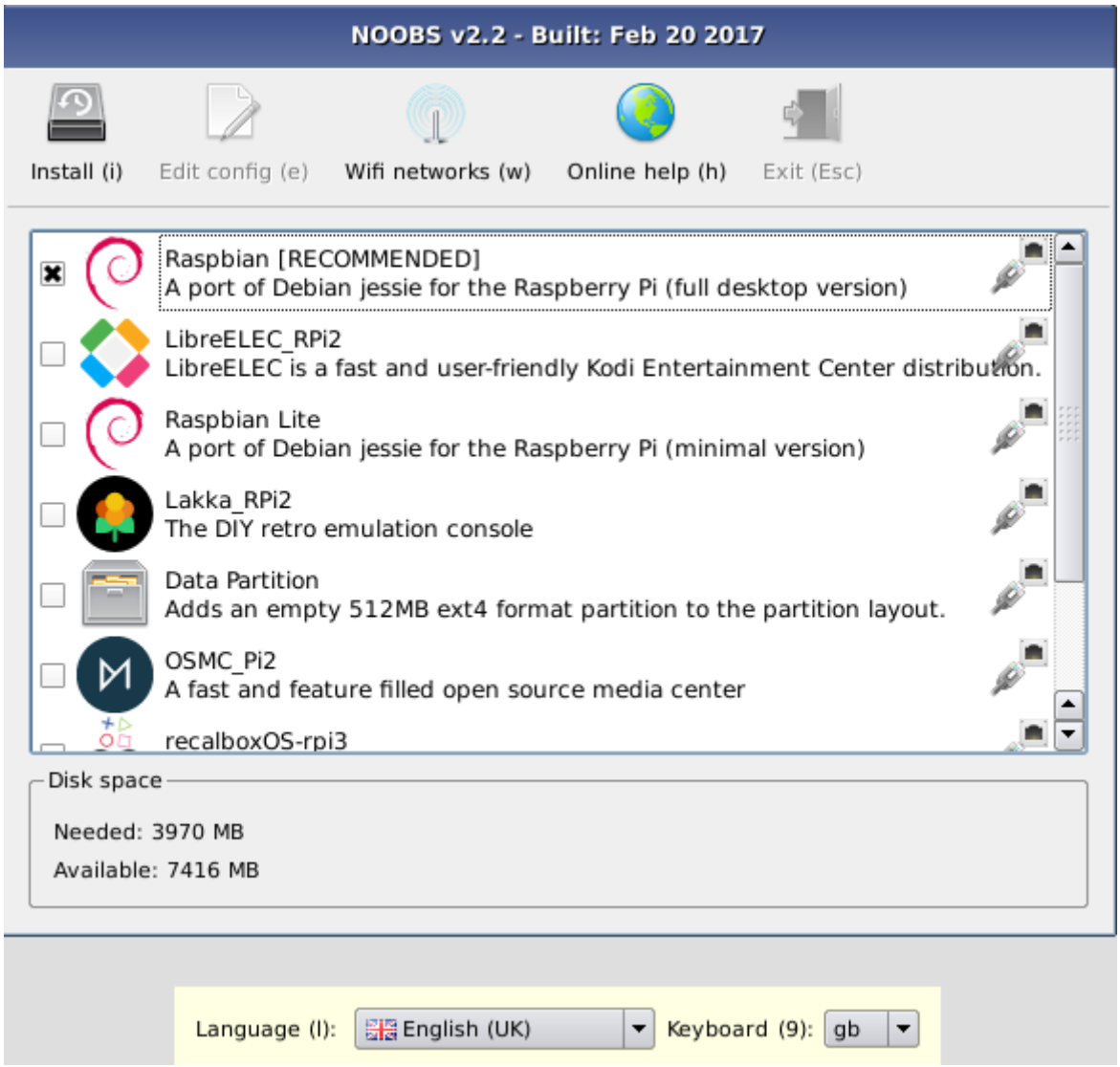


Fig 3.1 Sisteme de operare suportate

Pentru a instala alt sistem de operare decât Linux Raspbian trebuie să vă asigurați că placa este conectată la rețeaua de internet, fie prin cablu sau prin Wifi.

După ce se termină instalarea sistemului de operare, sistemul va reporni.

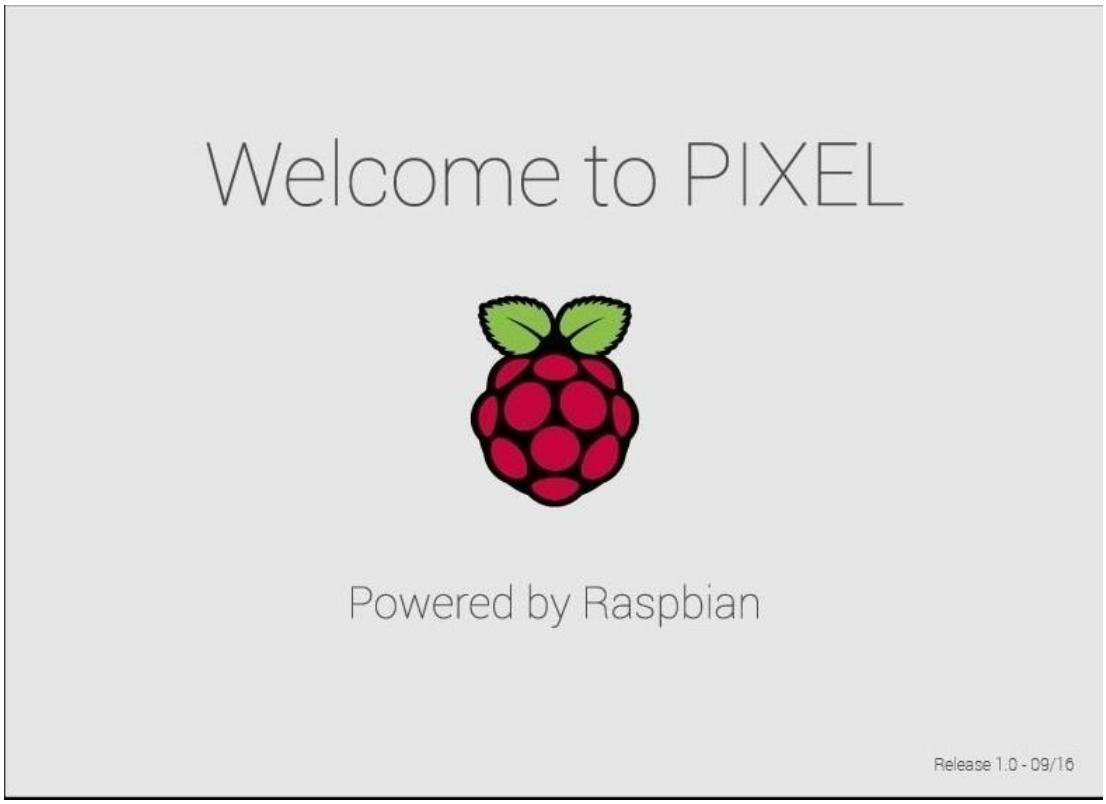


Fig 3.2 Ecran de pornire

Dacă aveți cardul oficial oferit de fundația Raspberry Pi, iar voi dețineți un card normal de clasă înaltă (10) trebuie să urmați următorul scenariu. Se descarcă imaginea oficial Linux Raspbian de pe adresa oficială a producătorului, https://www.raspberrypi.org/downloads/raspbian/ iar apoi se descărcă pe mașina personală programul Etcher, de pe următoarea adresă https://etcher.io/.

Acest program nu necesită instalare ci doar o simplă rulare a programului. În inferfața grafică a programului se alege locația unde s-a descărcat sistemul de operare Rasbpian, dispozitivul pe care dorim să se monteze imaginea, în cazul nostru cardul microSD iar apoi apăsăți comanda “Flash”.După ce procesul s-a finalizat cu succes puteți scoate cardul microSD din mașina personală și il puteți introduce în placa Raspberry Pi unde urmăriți pașii mai sus

menționați.

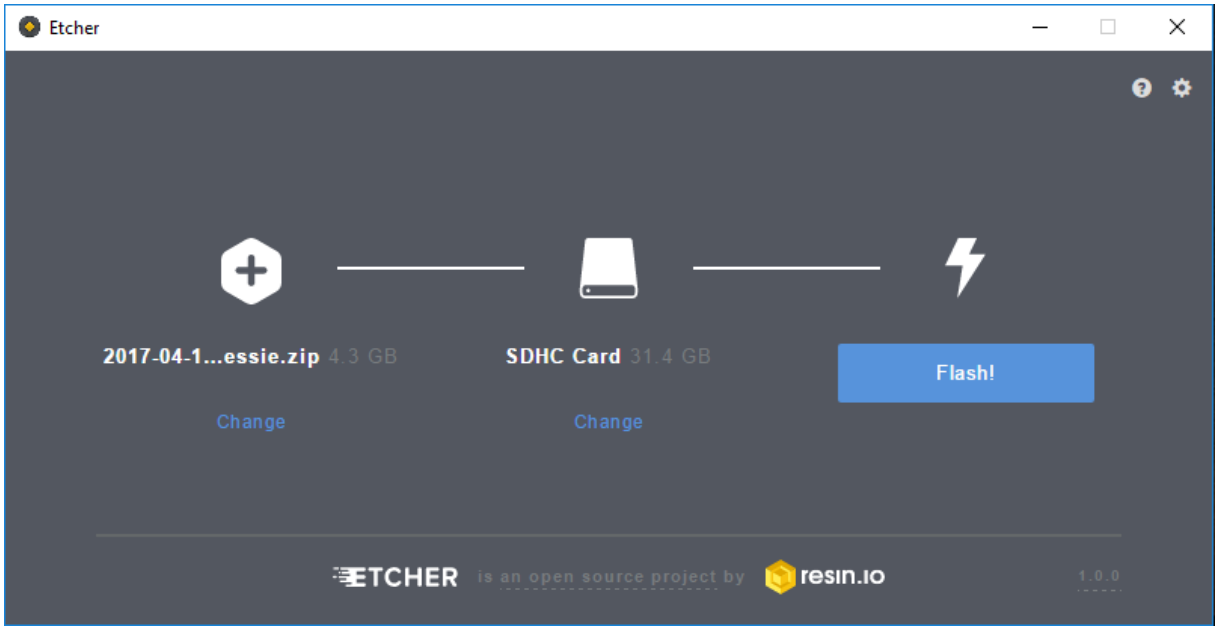


Fig 3.3 Interfață Etcher

Observație, pentru a monta imaginea sistemul de operare pe cardul microSD este necesar un adaptor de card, microSD-SD.

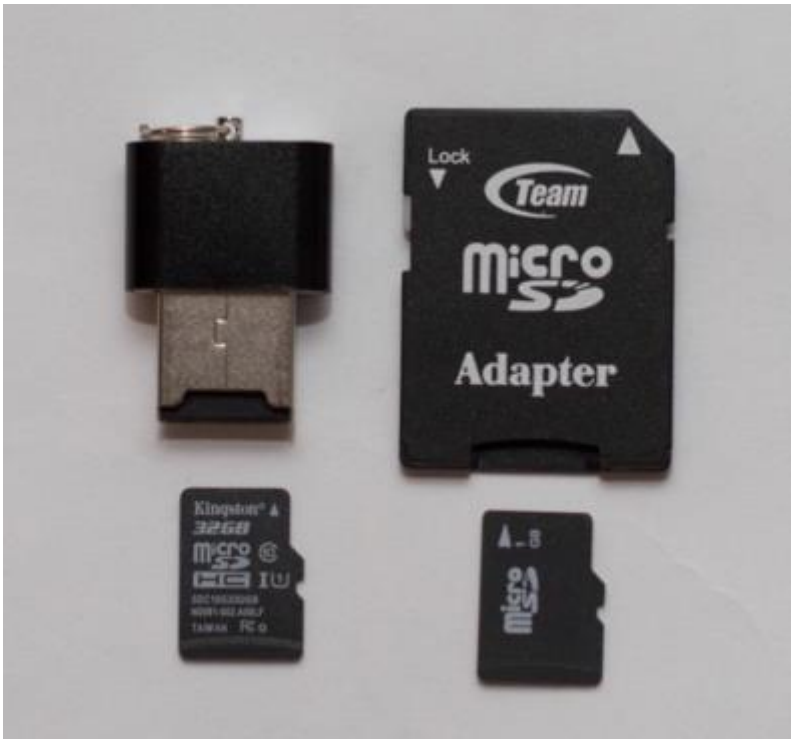


Fig 3.4 2 tipuri de cititor de card microSD

După ce s-a efectuat procesul de instalare a sistemului de operare se pot face anumite configurări. Aceste modificări pot fi făcute fie din interfața grafică sau din comenzi scrise în terminal.


Fig 3.5 Meniu de configurare

Pentru a deschide terminalul, putem fie să accesam pictrograma sistemului de operare->Accessories>Terminal sau apasând pe tastatură combinația Ctrl+Alt+T iar pentru a accesa

exact această fereastră se poate scrie in terminal, următoarea comandă sudo raspi-config.

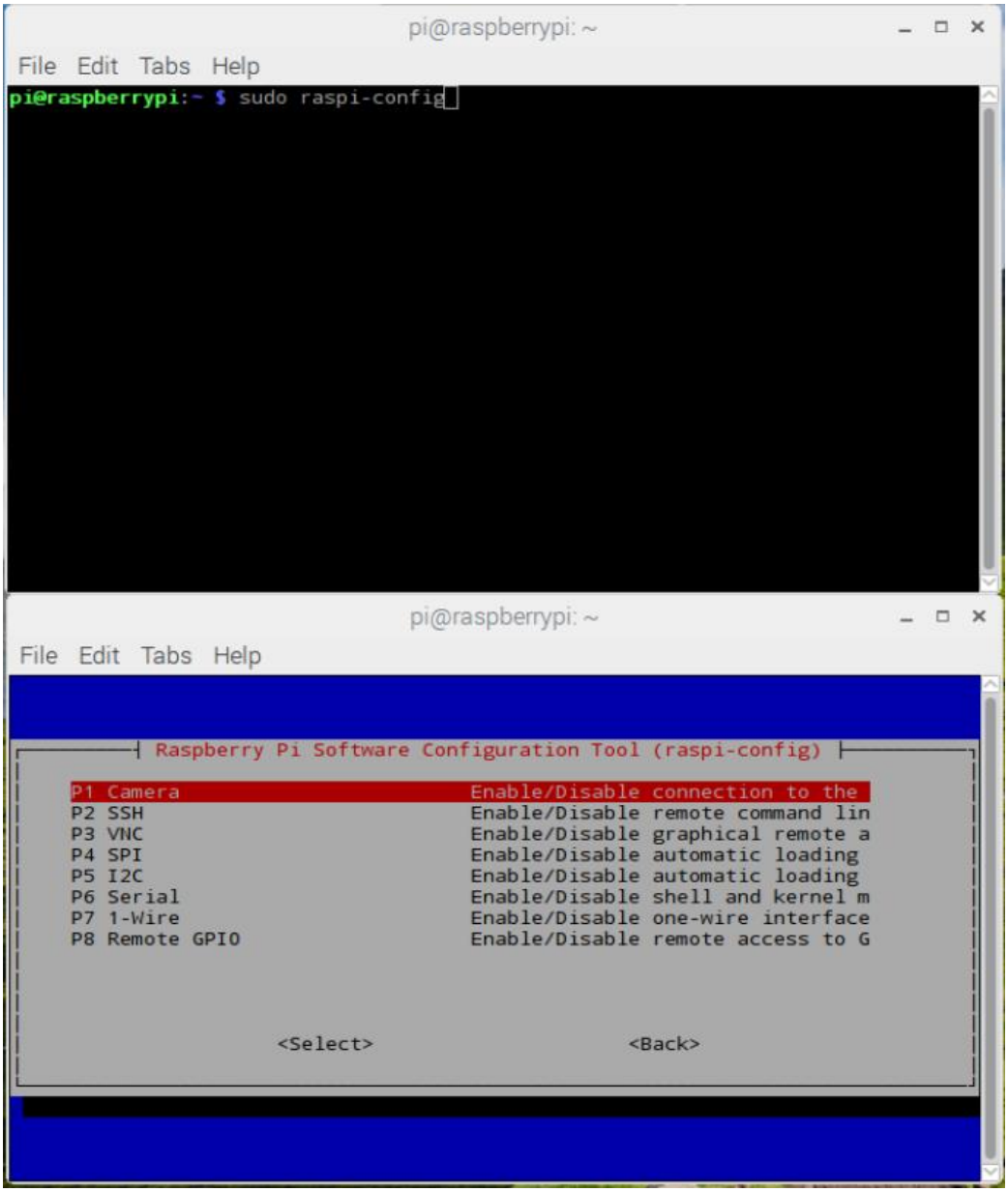


Fig 3.6 Meniu de configurare in terminal

Comanda sudo permite execuția unor comenzi privilegiate, adică doar cu drepturi de adminstrare a sistemului.

După instalarea sistemului de operare este recomandat procesul de schimbare a parolei predefinte de către sistemul de operare pentru a nu avea anumite neplăceri. Predefinit, utilizatorul are numele de pi iar parola este raspberry.

Dacă nu sunt folosite metode de a accesa placa Raspberry Pi de la distanța, serviciile SSH și VNC sunt recomandate a fi oprite.

De altfel, pentru a vă asigura că sistemul de operare are ultimele pachete instalate, este recomandată rularea a următoarelor comenzi în terminal.

Sudo apt-get update

Sudo apt-get upgrade

În cazul în care există o versiune mai nouă a sistemului de operare și se dorește o înnoire a acestuia se rulează următoarele comenzi

Sudo apt-get update

Sudo apt-get dist-upgrade

Pentru a putea continua această lucrare este necesar să se conecteze camera la placa Raspberry Pi iar apoi să se activeze modulul de cameră. Acesta se face din fereastra RaspberryPi Configuration.Pentru a conecta camera la placă trebuie să se introducă panglica în portul dedicat de comunicare CSI. Se ridică conectorul de pe placă iar pe panglica se poate observa o parte colorată cu albastru. Partea colorată trebuie să fie îndreptată spre portul Ethernet iar după ce se introduce panglica, conectorul se împinge la loc pentru a împiedica ieșirea panglicii din port. Această operație se face cu placa deconectată de la rețeaua de electricitate.


Fig 3.7 Conectarea cablului la interfața CSI

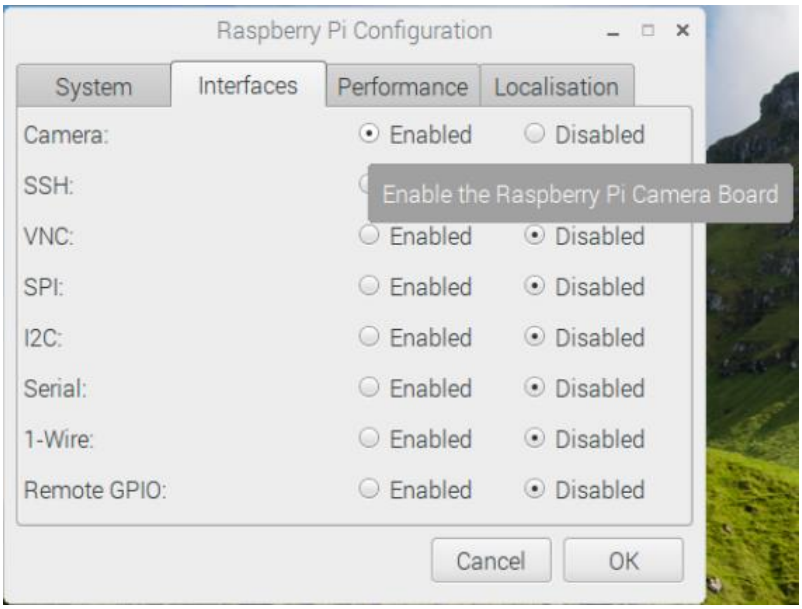


Fig 3.8 Fereastră activare modul cameră

Iar ca să se asigure că pachetele folosite de către modulul de camera este recomandat rularea comenziilor sudo apt-get update și sudo apt-get upgrade.

Pentru a verifica pașii efectuați până în acest moment și totul funcționează corespunzător, se recomandă rularea următoarei comenzi în terminal: python -c “import picamera”. Dacă nu se primește nici un mesaj de eroare, înseamna că pașii au fost executați corect.

# 4.Sistem de detectare facială

După ce s-au efectuat comenzile de inițializare iar sistemul este montat si funcțional, se rulează urmatoarele comenzi:

Sudo apt-get update

Sudo apt-get upgrade

Sudo apt-get install python-opencv libopencv-dev -y

Ultima comandă va instala pachetele necesare pentru a rula scripturi în limbajul Python din biblioteca OpenCV.

În cazul în care se întâmplă ca pachetele să nu pot fi instalate din cauza că serverul de pe care se descarcă acestea, fie este offline, fie placa Raspberry Pi nu se poate conecta la el, se poate executa următoarea commandă:

Cat /etc/apt/sources.list

Această comandă arată lista de servere pe care Raspberry Pi le accesează când descarcă pachete sau înnoiește versiunea anumitor programe.

Pentru a edita această listă rulăm următoarea comandă:

Sudo nano /etc/apt/sources.list

Și stergem simbolul # din fața liniei a3-a iar apoi se salvează.

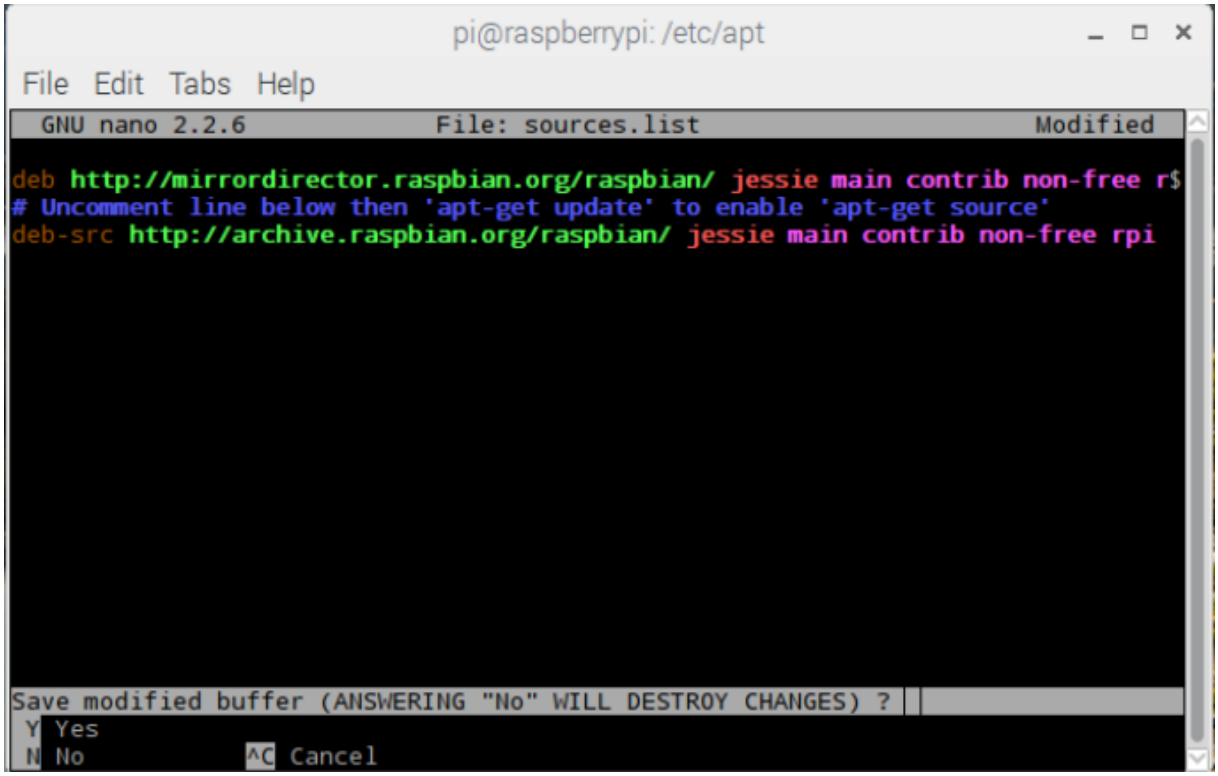


Fig 4.1 Modificare listă de surse disponbile

Se deschide o instanță a programului Python 2.7.9 din următoarea cale de acces Start->Programming->Python 2 (IDLE).

Se creează o noua fereastră de lucru, fie prin opțiunea de File->New sau combinația de tastură Ctrl+N.

Primul pas este de a importa bibliotecile ce vor urma să fie folosite.

```
import io
import picamera
import cv2
import numpy
```

Apoi se va crea o memorie virtuală astfel încât fotografiile să nu fie salvate într-un fișier.

```
stream = io.BytesIO()
```

În acest pas se vor stabili setările folosite de către cameră.

```
with picamera.PiCamera() as camera:
```

Camera.resolution = (320,240)

Camera.capture(stream, format=’jpeg’)

Aici se poate modifica de exemplu rezoluția sau extensia sub care imaginea va fi salvată.

Următorul pas este transformarea imaginii într-o matrice

```
buff = numpy.fromstring(stream.getvalue(), dtype=numpy.uint8)
```

Urmează crearea imaginii în OpenCV

```
image = cv2.imdecode(buff, 1)
```

Încărcarea clasificatoarelor Haar

```
face_cascade=cv2.CascadeClassifier('/usr/share/opencv/haarcascades/haarcascade_frontalface_alt.xml')
eye_cascade = cv2.CascadeClassifier('/usr/share/opencv/haarcascades/haarcascade_eye.xml')
```

Convertire imagine în tonuri de gri

```
gray = cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
```

Detectarea în imagine a fețelor

```
faces = face_cascade.detectMultiScale(gray, 1.1, 5)
```

Desenerea unui contur în jurul feței și a ochiilor

```
for (x,y,w,h) in faces:
   cv2.rectangle(image,(x,y),(x+w,y+h),(255,255,0),2)
   roi_gray = gray[y:y+h,x:x+w]
   roi_color = image[y:y+h,x:x+w]
   eyes = eye_cascade.detectMultiScale(roi_gray)
   for (ex,ey,ew,eh) in eyes:
      cv2.rectangle(roi_color,(ex,ey),(ex+ew,ey+eh),(0,255,0),1)
```

Afișare mesaj

print "Found "+str(len(faces))+" face(s)"

Stocarea imaginii pe cardul de memorie

cv2.imwrite('Imagine.jpg',image)

În imaginile alăturate se pot observa rezultatele rulării acestui script.

Se folosește următoarea imagine drept sursă

Fig 4.2 Cadru sursă

Iar rezultatul este următorul

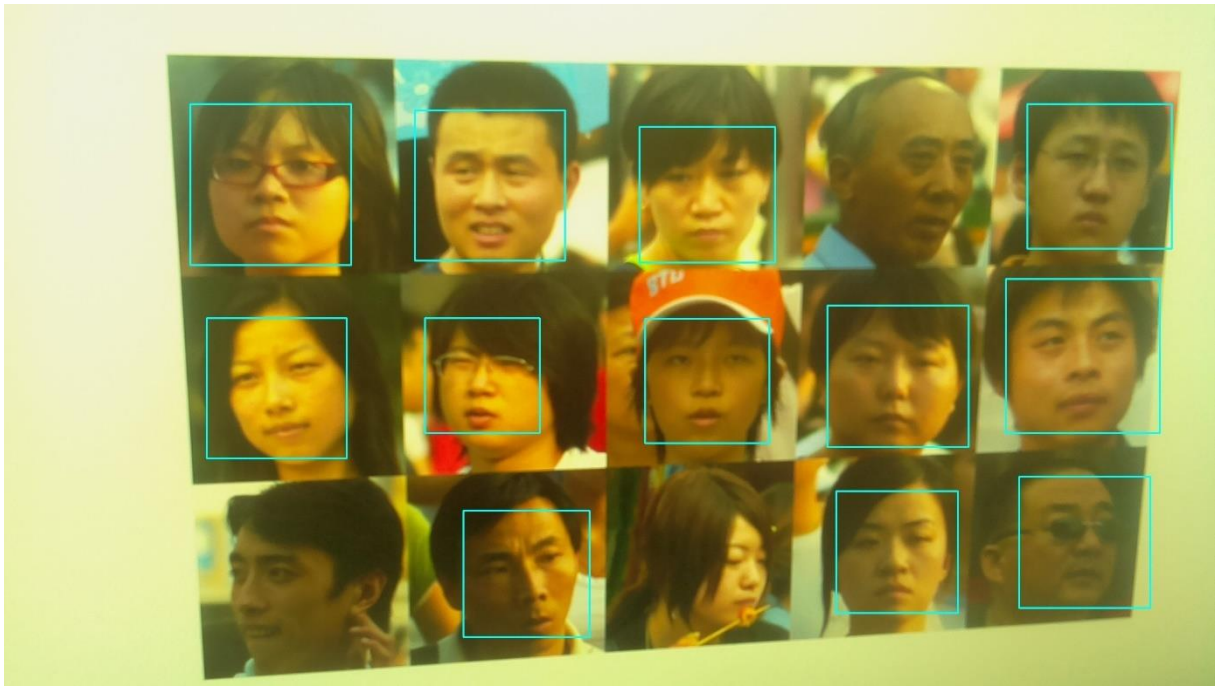

Fig 4.3 Imagine rezultat

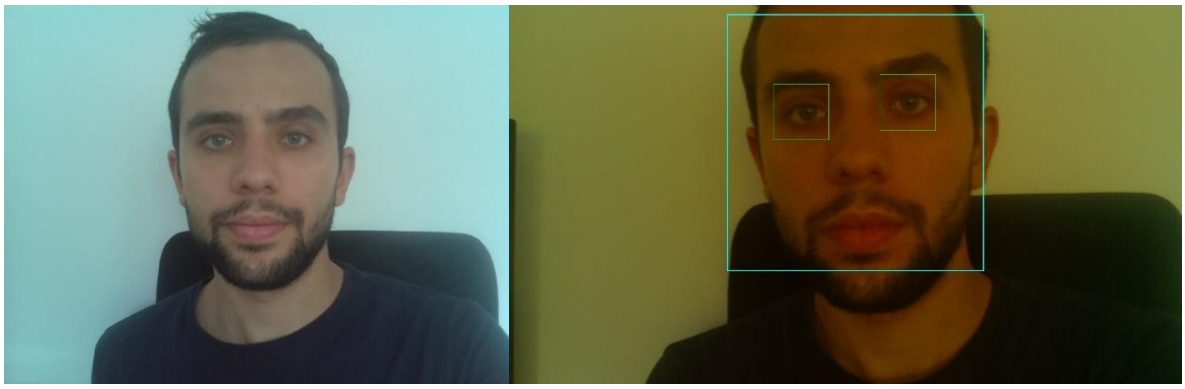

Fig 4.4 Rezultat imagine încadrare față și ochi

# 5.Concluzii

Sistemul funcționează, însă după cum se poate observa și în figura 4.3 însă cu anumite limitări și anume că persoanele fotografiate trebuie să fie poziționate cât mai perpendicular spre cameră și nu trebuie să existe alte obiecte în fața acestora.

Pentru identificarea unui alt sistem trebuie să se pregătească un nou set de date. Pentru instruirea unei cascade amplificate de clasificatori slabi avem nevoie de un set de eșantioane pozitive (care conțin obiecte reale pe care doriți să le detectați) și un set de imagini negative (conținând tot ceea ce nu doriți să detectați). Setul de probe negative trebuie să fie pregătit manual, în timp ce setul de probe pozitive este creat folosind aplicația opencv_createsamples. [9]

Probele negative sunt luate din imagini arbitrare, fără obiecte pe care doriți să le detectați. Aceste imagini negative, din care sunt generate eșantioanele, ar trebui să fie listate într-un fișier de imagine negativ special care conține o singură cale de imagine pe linie (poate fi absolută sau relativă). Rețineți că eșantioanele negative și imaginile de eșantion sunt numite și eșantioane de fundal sau imagini de fundal.

Probele pozitive sunt create de aplicația opencv_createsamples. Ele sunt folosite de procesul de stimulare pentru a defini ceea ce modelul ar trebui să caute de fapt atunci când încearcă să găsească obiectele de interes. Aplicația acceptă două moduri de a genera un set de date pozitiv.

1) Puteți genera o mulțime de eșantioane pozitive dintr-o singură imagine obiect pozitiv.
2) Puteți furniza toate pozițiile și utiliza instrumentul numai pentru a le elimina, a le redimensiona și a le pune în formatul binar necesar în formatul opencv.

În timp ce prima abordare funcționează pentru obiecte fixe, cum ar fi logo-ul, aceasta tinde să eșueze pentru obiecte mai puțin rigide. În acest caz, sugerăm să folosim a doua abordare.

1) Rețineți că aveți nevoie de mai mult de un singur eșantion pozitiv înainte de a-l da aplicației menționate, deoarece se aplică numai transformarea perspectivelor.
2) Dacă doriți un model robust, luați mostre care acoperă o gamă largă de soiuri care pot apărea în clasa obiect. De exemplu, în cazul fețelor ar trebui să luați în considerare

diferite rase și grupe de vârstă, emoții și, probabil, stiluri de barbă. Acest lucru se aplică și în cazul utilizării celei de-a doua abordări.

Prima abordare ia o imagine a unui obiect unic, de exemplu, cu o siglă a companiei și creează un set mare de eșantioane pozitive din imaginea obiectului dat, prin rotirea aleatorie a obiectului, schimbarea intensității imaginii și plasarea imaginii pe fundaluri arbitrare. Cantitatea și intervalul de aleatorie pot fi controlate de argumentele liniei de comandă din aplicația opencv_createsamples.

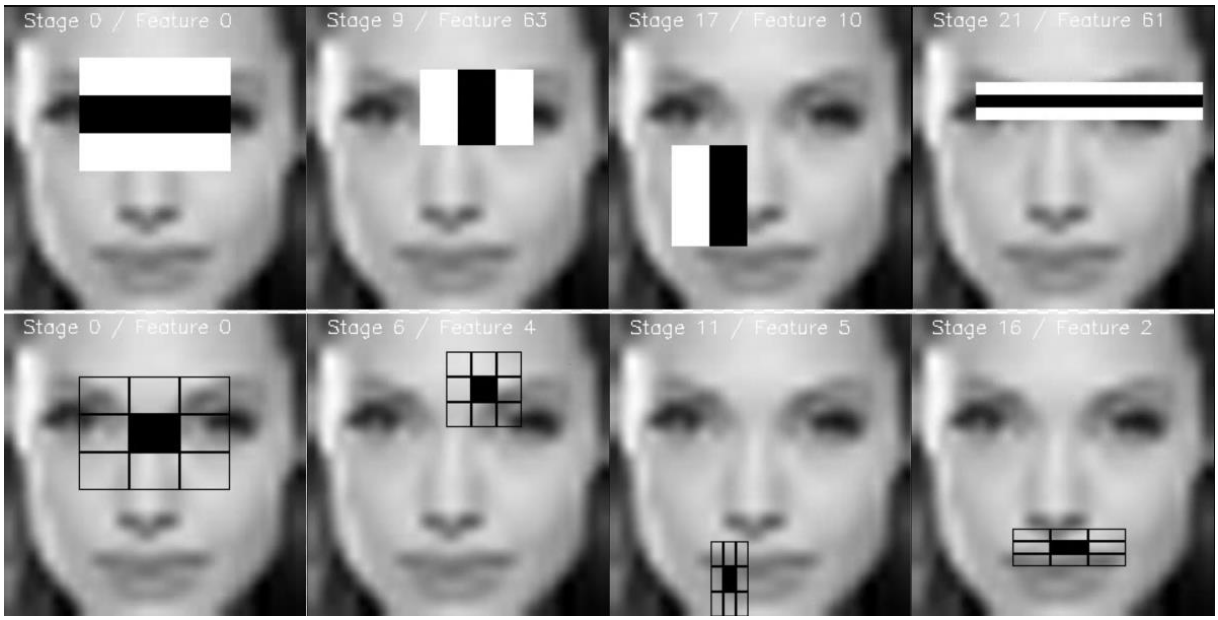

Fig 5.1 Proces de antrenare a clasificatorului [10]

# 6.Anexe

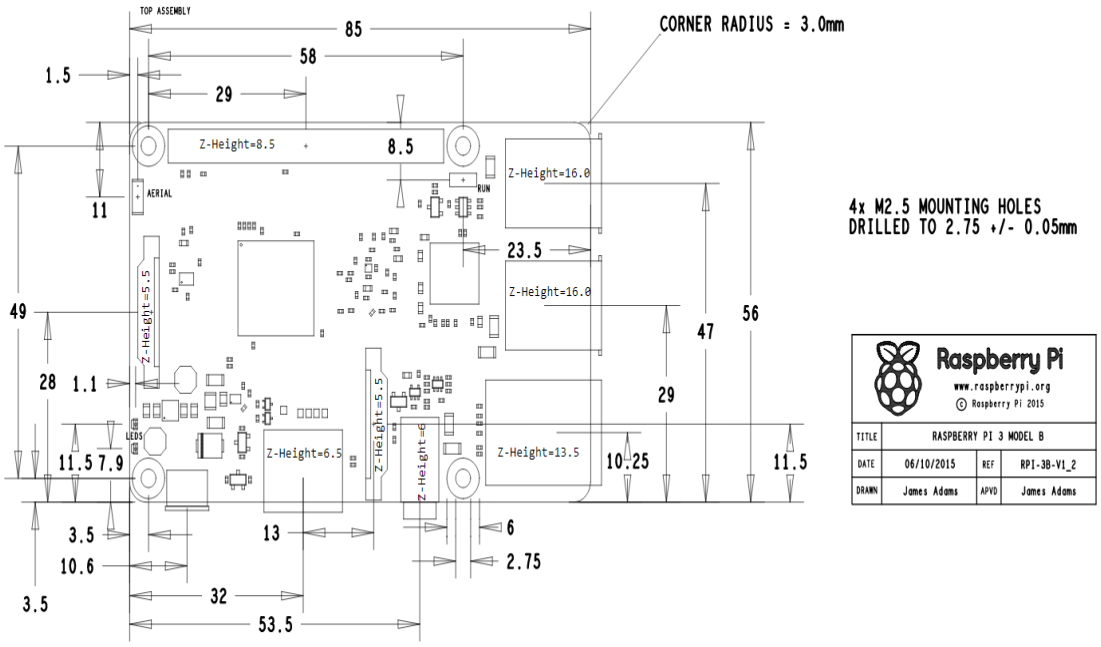


Fig 6.1 Cote de gabarit Raspberry Pi 3 Model B

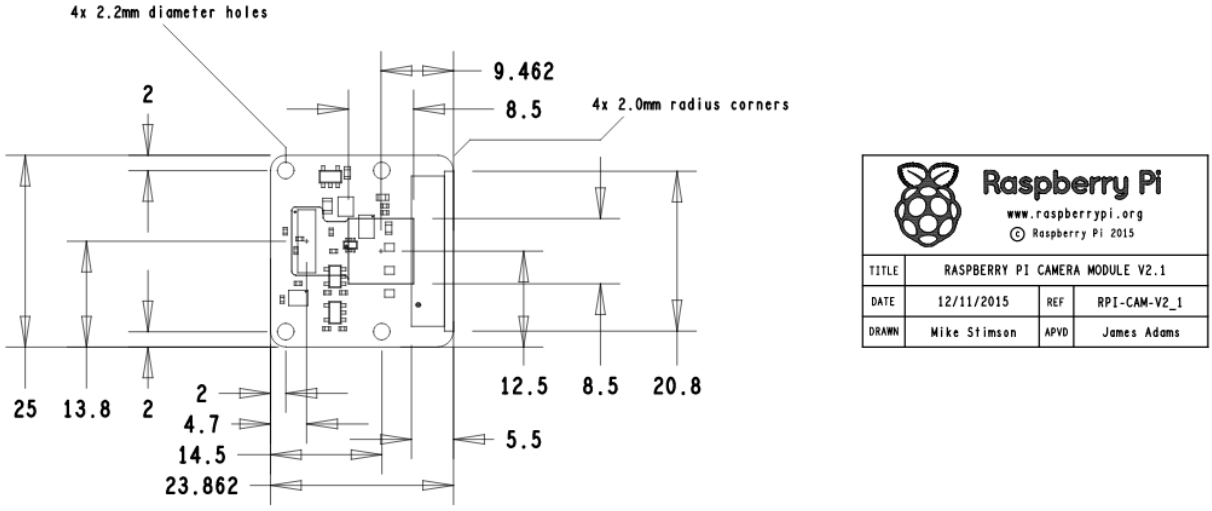


Fig 6.2 Cote de gabarit Raspberry Pi Camera

| | **Camera Module v1** | **Camera Module v2** |
|---|---|---|
| Preț | $25 | $25 |
| Mărime | Around 25 × 24 × 9 mm | |
| Greutate | 3g | 3g |
| Rezoluție | 5 Megapixels | 8 Megapixels |
| Moduri video | 1080p30, 720p60 and 640 × 480p60/90 | 1080p30, 720p60 and 640 × 480p60/90 |
| Integrare Linux | V4L2 driver available | V4L2 driver available |
| Programare C API | OpenMAX IL and others available | OpenMAX IL and others available |
| Senzor | OmniVision OV5647 | Sony IMX219 |
| Rezolutie sensor | 2592 × 1944 pixels | 3280 × 2464 pixels |
| Dimensiune pixel | 1.4 µm × 1.4 µm | 1.12 µm x 1.12 µm |
| Dimensiune optică | 1/4" | 1/4" |
| Echivalent lentilă Full-frame SLR | 35 mm | |
| Sensitivity | 680 mV/lux-sec | |
| Diafragmă | 2.9 | 2.0 |

Tabel 6.1 Specificații Hardware

| | |
|---|---|
| Format foto | JPEG (accelerated), JPEG + RAW, GIF, BMP, PNG, YUV420, RGB888 |
| Format video | raw h.264 (accelerated) |
| Efecte | negative, solarise, posterize, whiteboard, blackboard, sketch, denoise, emboss, oilpaint, hatch, gpen, pastel, watercolour, film, blur, saturation |
| Moduri de expunere | auto, night, nightpreview, backlight, spotlight, sports, snow, beach, verylong, fixedfps, antishake, fireworks |
| Moduri de măsurare | average, spot, backlit, matrix |
| Balans de alb | off, auto, sun, cloud, shade, tungsten, fluorescent, incandescent, flash, horizon |
| Declanșare | Keypress, UNIX signal, timeout |
| Moduri extra | demo, burst/timelapse, circular buffer, video with motion vectors, segmented video, live preview on 3D models |

Tabel 6.2 Caracteristici Software

## 6.1 Cod sursă

```
import io
import picamera
import cv2
import numpy
#Create a memory stream so photos doesn't need to be saved in a file
stream = io.BytesIO()
#Get the picture (low resolution, so it should be quite fast)
#Here you can also specify other parameters (e.g.:rotate the image)
with picamera.PiCamera() as camera:
    camera.resolution = (320,240)
    camera.capture(stream, format='jpeg')
#Convert the picture into a numpy array
buff = numpy.fromstring(stream.getvalue(), dtype=numpy.uint8)
#Now creates an OpenCV image
image = cv2.imdecode(buff, 1)
#Load a cascade file for detecting faces
face_cascade=cv2.CascadeClassifier('/usr/share/opencv/haarcascades/haarcascade_frontalface
_alt.xml')
eye_cascade = cv2.CascadeClassifier('/usr/share/opencv/haarcascades/haarcascade_eye.xml')
#Convert to grayscale
gray = cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
#Look for faces in the image using the loaded cascade file
faces = face_cascade.detectMultiScale(gray, 1.1, 5)
```

```
#Draw a rectangle around every found face
for (x,y,w,h) in faces:
    cv2.rectangle(image,(x,y),(x+w,y+h),(255,255,0),2)
    roi_gray = gray[y:y+h,x:x+w]
    roi_color = image[y:y+h,x:x+w]
    eyes = eye_cascade.detectMultiScale(roi_gray)
    for (ex,ey,ew,eh) in eyes:
        cv2.rectangle(roi_color,(ex,ey),(ex+ew,ey+eh),(0,255,0),1)
print "Found "+str(len(faces))+" face(s)"
#Save the result image
cv2.imwrite('Imagine.jpg',image)
```

## 6.2 Schema logică

[7]

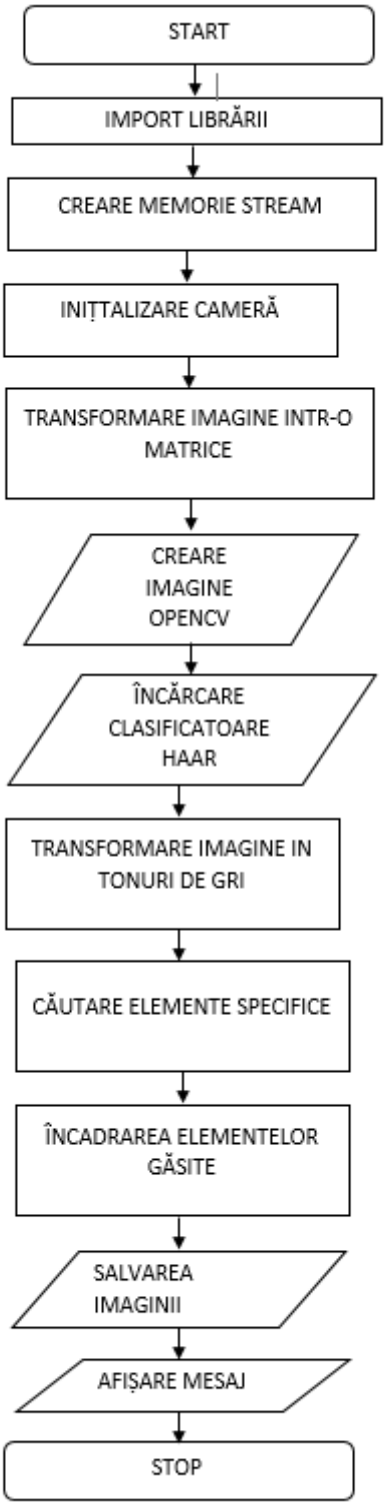


Fig 6.2.1 Schema logică

# 7.Bibliografie


1) Dolga V., Proiectarea sistemelor mecatronice, Timisoara, 2007
2) Remus B., Procesare de imagine și elemente de computer vision, Editura Universității "Lucian Blaga" din Sibiu, 2003
3) https://ro.wikipedia.org/wiki/Raspberry_Pi
4) https://www.robofun.ro/raspberry-pi-si-componente/raspberry-pi-v3
5) https://ro.wikipedia.org/wiki/Python
6) Learning OpenCV - Gary Bradski and Adrian Kaehler
7) http://staff.cs.upt.ro/~cami/upc/scheme/scheme-logice.htm
8) http://mdm.utcluj.ro/old/mecatronica.html
9) http://docs.opencv.org/trunk/dc/d88/tutorial_traincascade.html
10) http://docs.opencv.org/trunk/visualisation_video.png
11) https://en.wikipedia.org/wiki/Haar-like_features