



Universitatea TRANSILVANIA din Braşov
Facultatea de Matematică şi Informatică
Specializarea Informatică

Lucrare de licență

Autor: **Draghia Alin-Madalin**
Coordonator științific: **Lect. univ. dr. Lucian Sasu**

Braşov
Iulie 2014

Universitatea Transilvania din Brașov
Facultatea de Matematică și Informatică
Specializarea Informatică

Proiect de Licență

Tehnici de Machine Learning în procesarea și recunoașterea îmaginelor

Autor:

Draghia Alin-Mădălin

Profesor coordonator:

Lect. Univ. Dr. Lucian Sasu

Brașov

Iulie 2014

Cuprins

1	Introducere	1
1.1	Motivație	1
1.2	Enunțul problemei	4
1.3	Structura Lucrării	5
2	Recunoașterea obiectelor	6
2.1	Parcurgerea imaginii în scara și spațiu	7
2.2	Extragerea de trăsături	11
2.3	Clasificare	17
2.4	Post-procesarea rezultatelor	18
2.5	Algoritmul de recunoaștere	19
3	Tehnologii folosite	21
3.1	Limbajul C++	21
3.2	Limbajul Python	22
3.3	Biblioteca Boost	22
3.4	Biblioteca scikit-learn	22
3.5	Biblioteca numpy	22

3.6	Biblioteca matplotlib	23
3.7	Biblioteca PySide	23
4	Implementarea	24
4.1	Diagrama de clase	26
4.1.1	Core	26
4.1.2	Dataset	27
4.1.3	Image Pyramid	28
4.1.4	Image Scanning	29
4.1.5	Feature Extraction	30
4.1.6	Classification	31
4.1.7	Non Maxima Suppression	32
4.1.8	Detection	32
4.1.9	Python	34
4.2	Algoritm de recunoaștere a persoanelor	35
4.2.1	Cod antrenament	36
4.2.2	Rezultate	40
4.2.3	Aplicație de recunoaștere a obiectelor	42
5	Concluzii	46
	Bibliografie	47

Capitolul 1

Introducere

Prin intermediul acestei lucrări doresc să prezint, din punct de vedere teoretic, pașii necesari în dezvoltarea unui sistem de recunoaștere a obiectelor în imagini, folosind tehnici de procesare a imaginilor și învățare automată.

Totodată, aceasta lucrare vine însăși de implementarea unei biblioteci software pentru dezvoltarea de algoritmi și aplicații de recunoaștere a obiectelor. În plus, pe baza acestei biblioteci, am implementat unul dintre algoritmii de recunoaștere consacrați.

1.1 Motivatie

Recunoașterea obiectelor este una dintre principalele aplicații ale viziunii artificiale și procesarea de imagini.

Oamenii pot recunoaște o mulțime de obiecte într-o imagine fără să depună prea mult efort, chiar dacă în aceste imagini obiectele prezintă variații de perspectivă, de dimensiune, sunt translatate, rotite sau chiar obstrucționate. Cu toate că de-a lungul timpului au fost studiați și dezvoltăți multi algoritmi, sistemele de recunoaștere automată a obiectelor sunt încă departe de performanța unei ființe umane, chiar de cea a unui copil de numai doi ani. Așadar, încă există loc pentru cercetarea și dezvoltarea algoritmilor în acest domeniu. În ciuda performantei relativ scăzute a acestor algoritmi, odată cu dezvoltarea sistemelor hardware, fapt ce a permis aplicarea unor algoritmi mult mai complicați sau au putut fi aplicati pe niște probleme de dimensiune mai mare, cererea de aplicații a crescut.

1.1. MOTIVATIE

Câteva dintre cele mai de succes aplicații sunt:

- Sistemul de frânare automata la detecția pietonilor instalat pe mașinile Volvo.[1]



Figura 1.1: Volvo: sistemul de detectie a pietonilor [11]

- Focalizarea automata a camerelor foto pe fețe

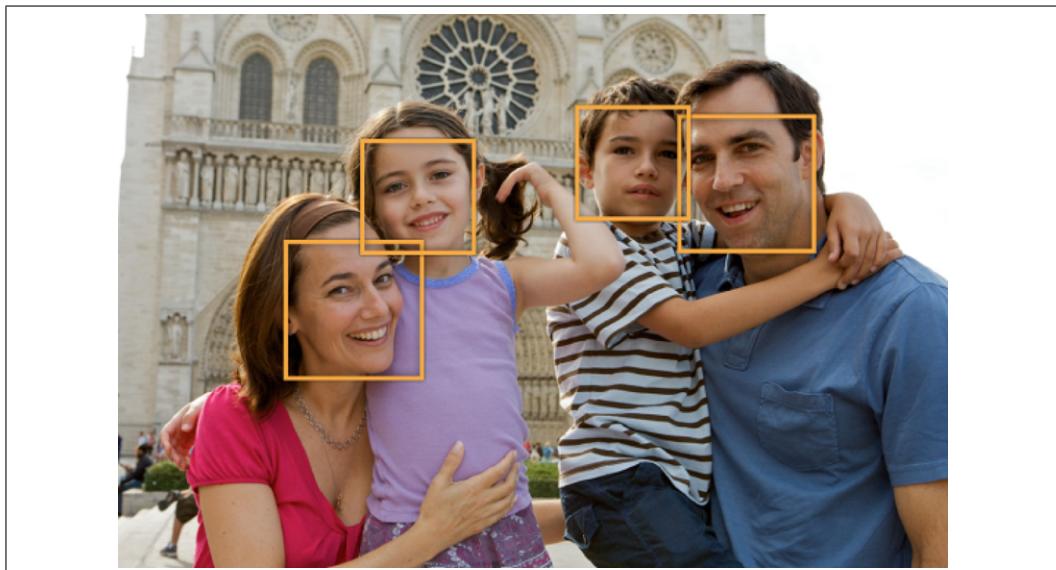


Figura 1.2: Camera foto: focus automat[24]

1.1. MOTIVATIE

- Analizarea traficului rutier

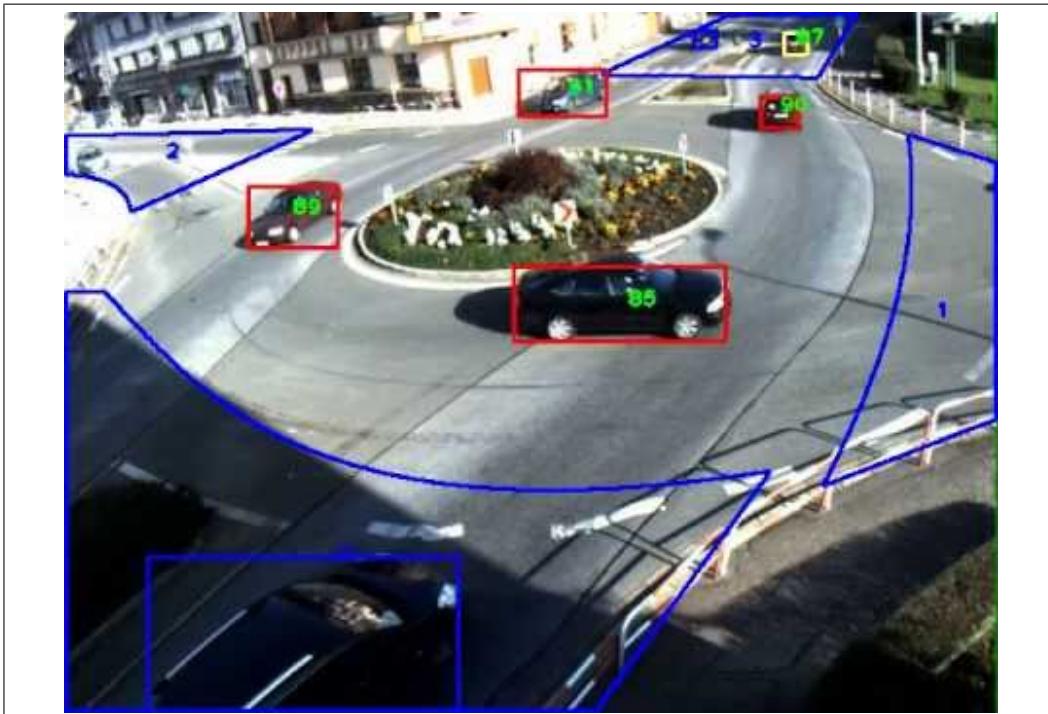


Figura 1.3: Analizarea traficului rutier¹

Înțelegerea și dezvoltarea unui sistem de recunoaștere automata a obiectelor poate fi foarte dificila, mai ales pentru cei care sunt la început de drum în studierea acestui domeniu. Documentația de specialitate, de cele mai multe ori, este scrisă privind problema de la un nivel foarte înalt și nu sunt tratate detaliile algoritmilor. În același timp, în foarte multe lucrări se fac referiri la lucrări anterioare, unele chiar cu zeci de ani distanță între ele, acestea fiind uneori foarte greu de găsit. Parcursul unui astfel de document presupune cunoștințe extensive de matematică, statistică, învățare automată, procesarea imaginilor și chiar cunoștințe din domeniul biologic sau medical. Toate acestea fac ca nivelul de la care se intra acest domeniu să fie unul foarte înalt, ceea ce poate fi descurajant pentru un începător.

Există câteva biblioteci software bune, open-source, cu care se pot dezvolta aplicații: opencv², dlib³, libccv⁴. Avantajele acestor biblioteci sunt:

¹<http://youtu.be/PXSiuojhNFg>

²<http://opencv.org/>

³<http://dlib.net/>

⁴<http://libccv.org/>

1.2. ENUNTUL PROBLEMEI

- Există algoritmi de recunoaștere a obiectelor gata implementați.
- Se poate trece direct la dezvoltarea de aplicații

Totuși sunt și dezavantaje:

- Componentele care stau la baza acestor implementări nu sunt expuse, reutilizarea lor fiind imposibilă.
- Codul sursă este optimizat cu instrucțiuni în cod de asamblare sau pentru procesorul grafic, fiind dificil de înțeles.

Aceste dezavantaje fac ca aceste biblioteci să nu fie foarte utile celor care doresc să dezvolte sau să implementeze algoritmi.

Doresc ca, la finalul acestei lucrări, să obțin o platformă de dezvoltare a algoritmilor pentru recunoașterea obiectelor, pe care să o pot folosi în activitatea mea din domeniu și care să servească ca un punct de plecare pentru cei care doresc să se inițieze în domeniu.

Avantajele acestei platforme ar fi:

- Fiecare componentă a unui algoritm de recunoaștere este implementată într-o clasa separată
- Algoritmii pot fi implementați atât în C++ cât și în Python
- Reutilizare sporită
- Pot fi ușor adaptăți algoritmi din alte biblioteci pentru utilizare în cadrul platformei

1.2 Enunțul problemei

Se scrie o bibliotecă software cu ajutorul căreia să se dezvolte algoritmi și aplicații de recunoaștere a obiectelor.

Aceasta bibliotecă va fi scrisă într-un mod hibrid, cu componente implementate atât în C++ cât și în Python.

Toate componentele bibliotecii vor suporta serializare, pentru a putea fi salvate pe disc, baze de date sau trimise prin rețea în cazul unor programe distribuite.

1.3. STRUCTURA LUCRĂRII

Algoritmul va învăța să recunoască obiecte folosindu-se de un set de imagini cu exemple pozitive adnotate și exemple negative, imagini care nu conțin obiectul pe care dorim să-l învățăm. Algoritmul poate fi personalizat prin alegerea de implementări diferite ale componentelor de către utilizator.

Ca exemplificare, se scrie o aplicație care antrenează un algoritm de recunoaștere și salvează modelul învățat pe disc și o alta aplicație care încarcă modelul și îl aplică pe o imagine data.

1.3 Structura Lucrării

TODO: fix me!

Capitolele care urmează vor trata algoritmul de recunoaștere a obiectelor din punct de vedere teoretic și se va prezenta implementarea unei platforme de dezvoltare a acestora.

În capitolul 2 sunt prezentate în detaliu structura unui algoritm de recunoaștere și o tehnică eficientă de antrenare a unui astfel de algoritm.

În capitolul 3 este prezentată în detaliu implementarea bibliotecii de dezvoltare a algoritmilor de recunoaștere a obiectelor.

În capitolul 4 sunt prezentate tehnologiile folosite.

În capitolul 5 sunt prezentate concluzii despre lucrare, precum și posibilități de dezvoltare.

Capitolul 2

Recunoașterea obiectelor

Problema recunoașterii de obiecte se poate exprima în felul următor: Având o bază de date cu unul sau mai multe modele de obiecte, să se determine dacă există obiectul în imagine și dacă există, să se localizeze.

Unele dintre cele mai relevante lucrări din domeniu sunt:

- "Robust Real-time Object Detection" [22]
- "Histograms of Oriented Gradients for Human Detection" [9]
- "Object Detection with Discriminatively Trained Part Based Models" [10]

Dacă studiem mai atent algoritmii descriși în aceste lucrări se observă că toate au o structură comună și urmăresc o succesiune de operațiuni similare. Aceste operațiuni sunt următoarele: parcurgerea imaginii **spatial** la diferite scări, extragerea de trăsături, clasificare și post-procesarea rezultatelor.

În continuare se va discuta mai detaliat despre fiecare componentă, iar la sfârșit despre algoritmul de recunoaștere.

2.1. PARCURGEREA IMAGINII ÎN SCARA ȘI SPATIU

2.1 Parcurgerea imaginii în scara și spațiu

Obiectele care trebuie recunoscute pot prezenta deviații de la modelul din baza de date. Aceste deviații pot fi de natură geometrică: translație, rotație, scalare și perspectivă.

O soluție pentru aceasta problema ar fi să se construiască un model care să prezinte toate instanțierile obiectului. O dificultate a acestei abordări ar fi că nu se pot ști dinainte toate transformările obiectului. Chiar dacă s-ar ști, se poate deduce că un astfel de model ar putea fi mult prea mare ca să poată fi aplicat practic.

O alta abordare ar fi să se folosească o reprezentare a imaginii, invarianta la aceste transformări. Din literatura se știe că o imagine reprezentată în spațiul Fourier este invarianta la translație și o imagine reprezentată în spațiul Log-Polar este invarianta la scalare și rotație.[21] Există chiar și o combinație între aceste două reprezentări numita Fourier-Mellin care este invarianta la toate cele trei transformări. Totuși s-a observat că utilizarea acestei reprezentări are aplicații limitate, ea fiind folosită mai mult la alinierea imaginilor.[21]

O alta soluție, poate un pic mai naivă, dar în același timp foarte puternică, este folosirea unei combinații între o piramidă de imagini și un algoritm de tip fereastră glisantă¹, acestea fiind aplicate pe imagine, nu pe modelul din baza de date.

Folosirea piramidei de imagini și fereastra glisantă ne permite să în rezultul algoritmului de recunoaștere să tratăm problema că și cum nu ar exista translații sau scalari, astfel simplificând mult algoritmii aplicați.

O piramida de imagini este o reprezentare multi-scara. Piramida de imagini se formează pornind de la o imagine sursă, prin scalari succesive. Aceste scalari se fac cu un factor $\alpha > 1$, $\alpha \in \mathbb{R}$ și se opresc atunci când se ajunge la o dimensiune minima. Dimensiunea imaginii la un anumit nivel din piramida se calculează astfel:

$$f(D, L) = D \cdot \frac{1}{\alpha^L}$$

Unde $D \in \mathbb{N}^2$ este dimensiunea imaginii sursă și $L \in \mathbb{N}^+$ este nivelul piramidei pentru care dorim să aflăm dimensiunea.

¹Eng. sliding window

2.1. PARCURGEREA IMAGINII ÎN SCARA ȘI SPATIU

Se poate vizualiza piramida de imagini în figurile următoare:

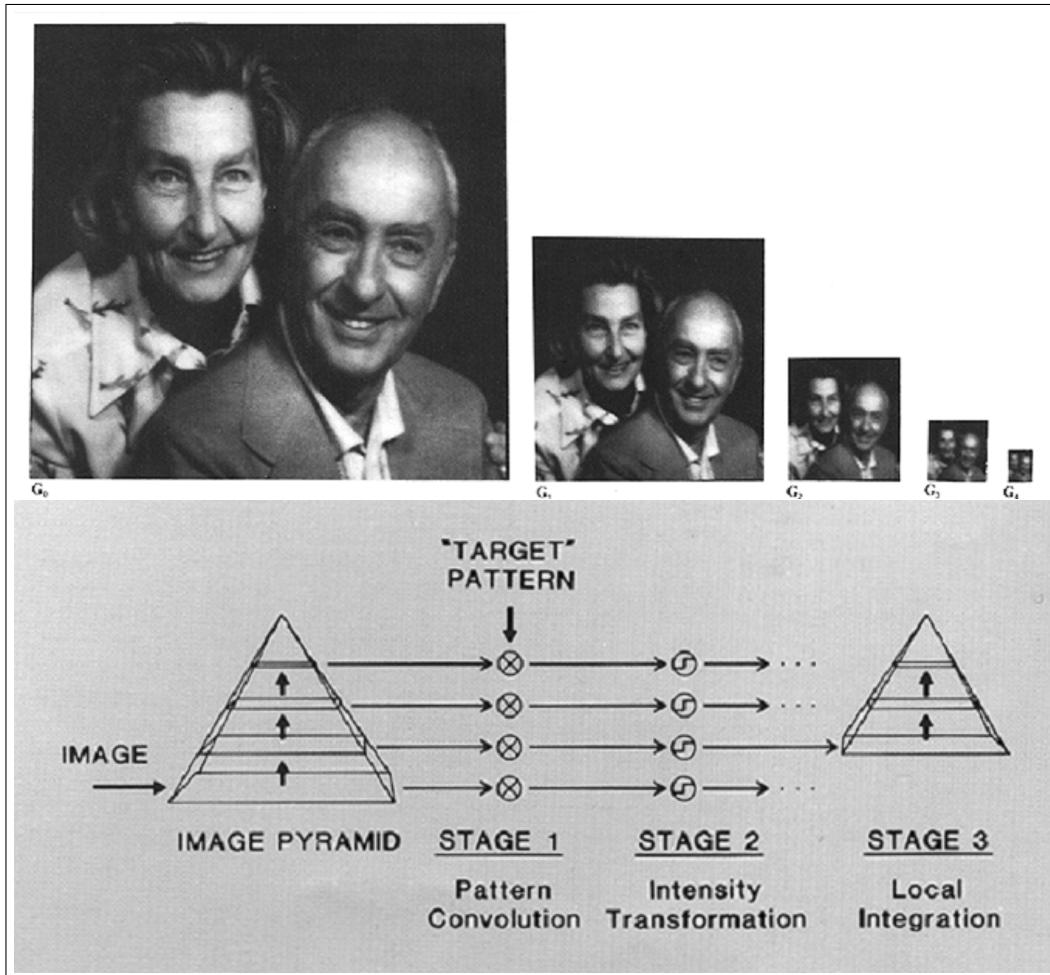


Figura 2.1: Piramida de imagini²

Prezentarea formarii piramidei de imagini în pseudo-cod:

```
sursa = citeste_imagine()
alpha = 6/5
dim_min = (100,100)
piramida = [sursa, ]
L=1
cicleaza
    D = sursa.D * 1/(alpha^L)
```

²Pyramid methods in image processing.[8]

2.1. PARCURGEREA IMAGINII ÎN SCARA ȘI SPATIU

```
daca D < dim_min
    atunci paraseste ciclul
sfarsit daca
nivel = scaleaza(sursa, D)
piramida = insereaza(piramida, nivel)
L = L + 1
sfarsit cicleaza
```

Se poate observa că, totuși, acest model nu poate reprezenta toate scările posibile, fiind un model discret. Aceasta problema poate fi ameliorată prin alegerea unui α potrivit și permitând modelului din baza de date să prezinte și el mici variații de scara.

O alta observație ar fi: cu cat α este mai mic, cu atât spațiile să nimerim scara corecta cresc, dar în același timp crește și consumul de memorie și durata de execuție a algoritmului. Consumul de memorie poate fi evitat dacă algoritmul se executa într-un mod recursiv, eliminând astfel menținerea explicită a unei liste de imagini în memorie.

Algoritmul fereastra glisanta se folosește pentru a obține invarianta la translație a modelului. Aici fereastra se referă la o secțiune rectangulară a imaginii. Fereastra va avea aceeași dimensiune ca și modelul din baza de date. Fereastra glisanta are ca parametri $\Delta_x, \Delta_y \geq 1$, însemnând pasul pe axa x , respectiv pasul pe axa y .

Pseudo-cod fereastra glisanta:

```
dx = 8
dy = 8
I = citeste_imagine()
M = citeste_model()
pentru x de la 0 la dimx(I) - dimx(I)
    pentru y de la 0 la dimy(I) - dimy(M)
        fereastra = sectiune(I, x, y, dimx(M), dimy(M))
        proceseaza(fereastra)
    sfarsit pentru
    sfarsit pentru
```

Se poate vizualiza algoritmul fereastra glisanta în figura următoare:

2.1. PARCURGEREA IMAGINII ÎN SCARA ȘI SPATIU

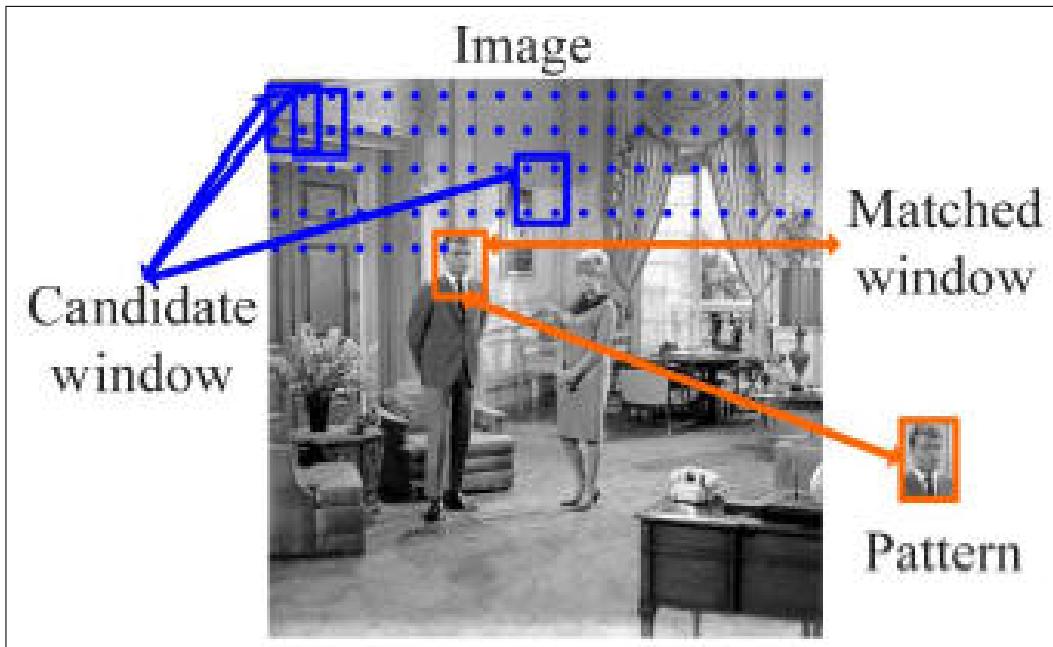


Figura 2.2: Fereastră Glisantă³

Se observă ca aici, ca și în cazul piramidei de imagini, cu cat x și y sunt mai mici, cu atât crește și numărul de ferestre evaluate, ceea ce duce la un timp de execuție mai ridicat.

Complexitatea algoritmului piramida combinată cu fereastra glisantă este

$$O((\dim_x - \Delta_x) \cdot (\dim_y - \Delta_y) \cdot n_{piramida})$$

³Imagine din: Segmented Gray-Code Kernels for Fast Pattern Matching[16]

2.2. EXTRAGEREA DE TRĂSĂTURI

2.2 Extragerea de trăsături

Extragerea de trăsături, în cazul nostru, reprezinta operațiunea de calculare a unei reprezentări a imaginii potrivita pentru recunoaștere.

O imagine este reprezentata ca o matrice de intensități. Aceasta reprezentare este foarte sensibila la condițiile de iluminare, conține informații irelevante și redundante. Se poate observa efectul iluminării în figura 2.3.

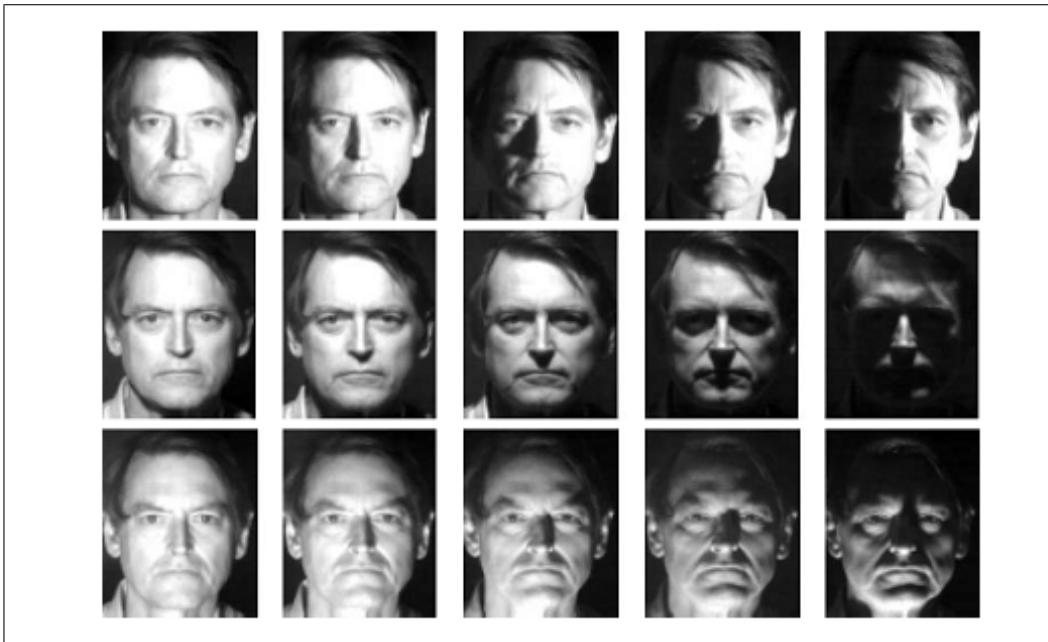


Figura 2.3: Efectul iluminării⁴

Există modalități de a remedia efectul iluminării, cum ar fi egalizarea histogramei(fig. 2.4). O alta modalitate ar fi să se folosească o reprezentare pe baza de gradienți care sunt invariante la iluminare.

⁴Sursa imagine: Computer vision: algorithms and applications[20]

2.2. EXTRAGEREA DE TRĂSĂTURI

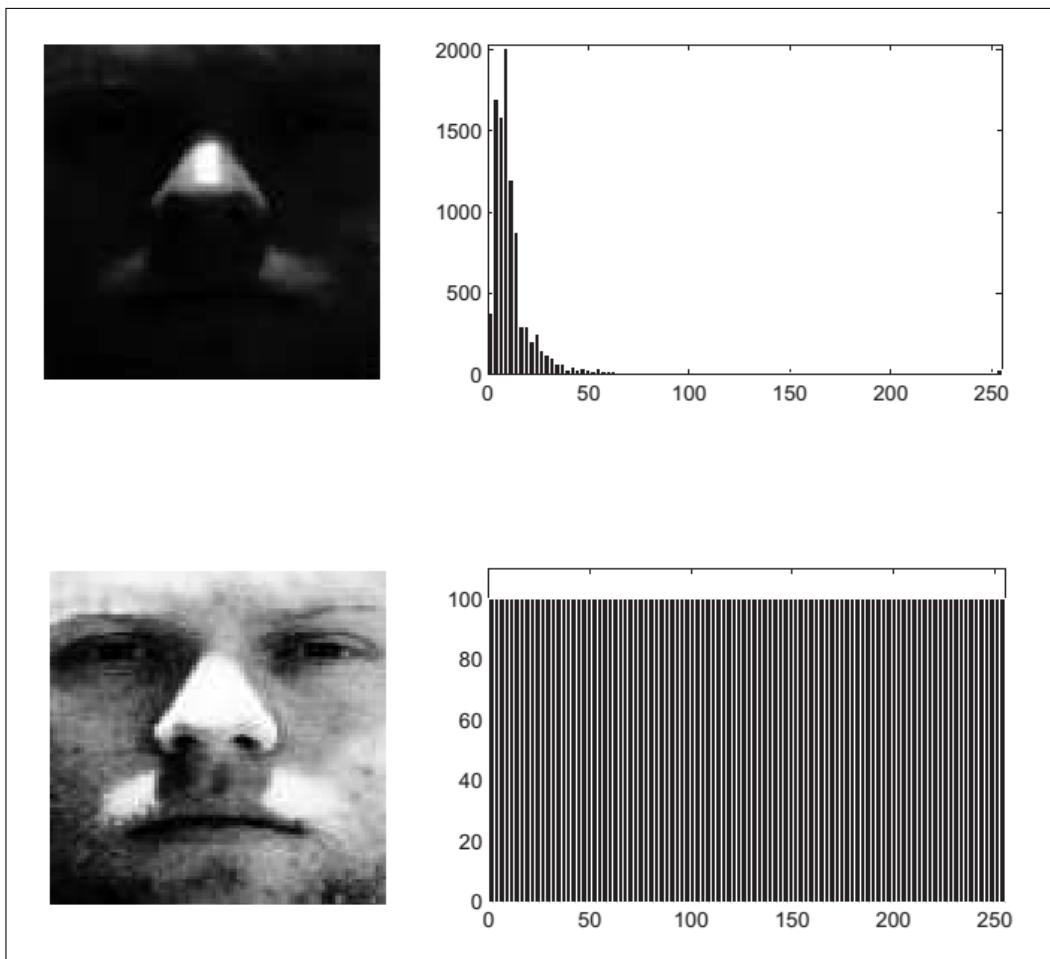


Figura 2.4: Egalizarea Histogramei⁵

Efectul informațiilor irelevante și redundante poate fi ameliorat folosind tehnici de reducere a dimensionalitatii, cum ar fi analiza componentelor principale⁶(fig. 2.5 sau transformata cosinus discreta(fig. 2.6).

⁵Sursa imagine: Histogram remapping as a preprocessing step for robust face recognition[18]

⁶Eng. PCA, principal component analysis

2.2. EXTRAGEREA DE TRĂSĂTURI

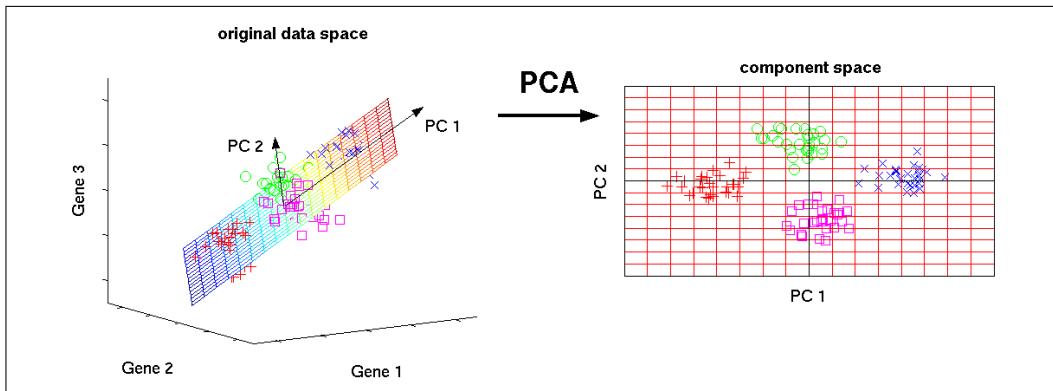


Figura 2.5: Analiza componentelor principale⁷

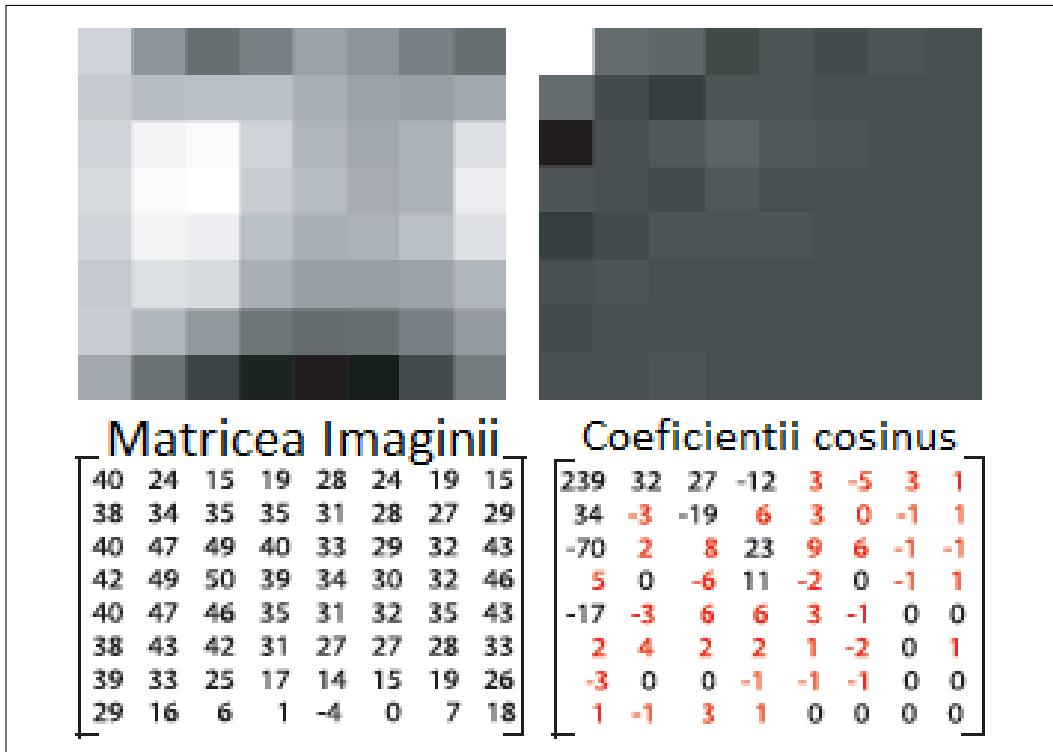


Figura 2.6: Transformata cosinus⁸

Totuși, nici una dintre reprezentările menționate mai sus nu tratează problema discriminării, adică: dacă două imagini conțin același obiect atunci

⁷Sursa imagine: <http://phdthesis-bioinformatics-maxplanckinstitute-molecularplantphys.matthias-scholz.de>

⁸Sursa imagine: <http://cnx.org/content/m13173/1.6/>

2.2. EXTRAGEREA DE TRĂSĂTURI

și reprezentările lor trebuie să fie apropiate, iar dacă sunt imagini ale unor obiecte diferite atunci reprezentările lor să fie distanțate.

Aici intervine ceea ce se numește ingineria trăsăturilor⁹ care, folosind cunoștințe din fizica, biologie sau chiar neurologie, construiește reprezentări mult mai favorabile recunoașterii. Câteva dintre cele mai cunoscute trăsături sunt HAAR[22], SIFT[14] și HOG[9].

Valoare unei trăsături HAAR este diferența dintre suma pixelilor din dreptunghiul negru și suma pixelilor din dreptunghiul alb, normalizată la aria celor două.

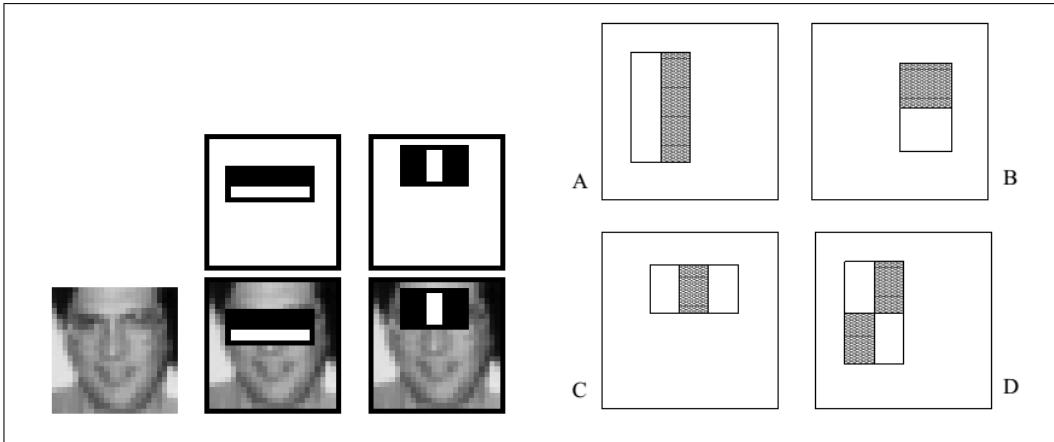


Figura 2.7: Trasaturi HAAR¹⁰

HOG, sau histograma orientărilor de gradienți, se calculează divizând imaginea în zone mai mici, numite celule, apoi se calculează histograma de orientări a gradienților din aceste zone. Concatenarea acestor histograme reprezintă trăsătura HOG.

⁹Eng. Feature Engineering

¹⁰Sursa imagine: Robust Real-time Object Detection[22]

2.2. EXTRAGEREA DE TRĂSĂTURI

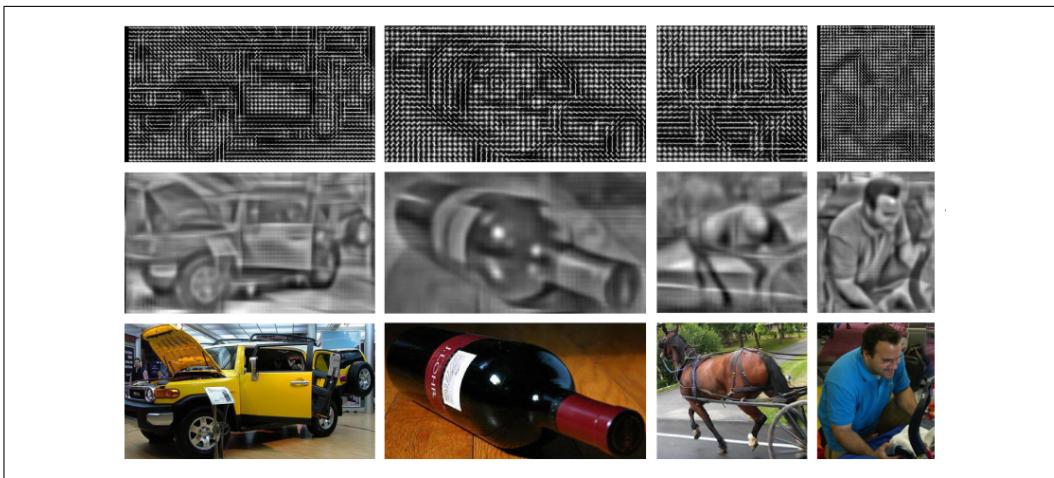


Figura 2.8: Trasaturi HOG¹¹

Descriptorul SIFT este similar cu HOG, acesta fiind în plus și invariant la rotație.

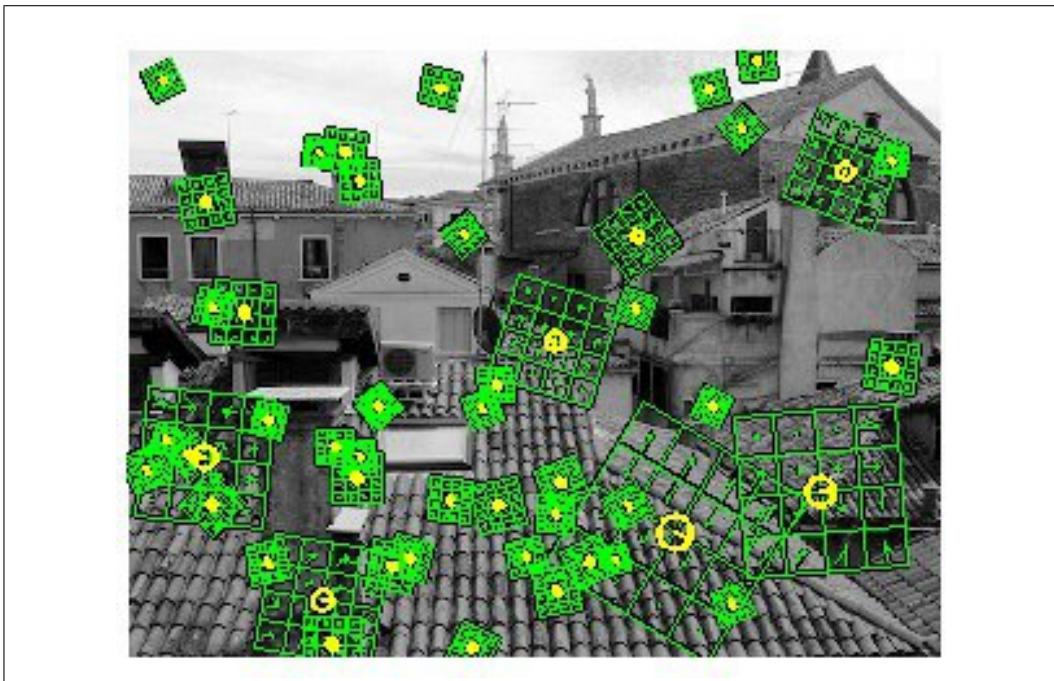


Figura 2.9: Descriptorul SIFT¹²

¹¹Sursa imagine: HOGgles: Visualizing Object Detection Features[23]

¹²Sursa imagine: <http://www.vlfeat.org/overview/sift.html>

2.2. EXTRAGEREA DE TRĂSĂTURI

Recent a apărut o nouă abordare în ceea ce privește extragerea de trăsături. Aceasta folosește reprezentarea cruda a imaginii, adică matricea de intensități a pixelilor și se bazează pe algoritmul de clasificare să extragă trăsături mai puternice, un exemplu fiind rețeaua neuronală convolutională.[13]

2.3 Clasificare

Din perspectiva recunoașterii obiectelor, clasificarea se va realiza cu ajutorul unei funcții care evaluează un vector de trăsături și decide dacă este sau nu obiectul pe care încercam să îl recunoaștem. Acest tip de clasificare se numește clasificare binara, fiindcă răspunsul nu poate lua decât două valori.

Aceasta funcție de decizie poate fi, în cazurile cele mai simple, o funcție de prag peste o distanță euclidiana sau chiar o rețea neuronală cu sute de neuroni.

În cazul nostru aceasta funcție este rezultatul unui algoritm de învățare automată.¹³ Învățarea automată, o ramură a inteligenței artificiale, este preocupată cu studiul și construcția sistemelor care pot învăță din date. Algoritmii de învățare automată sunt împărțiți în multe categorii, însă noi ne vom axa doar pe cei de învățare supervizată. Se numesc algoritmi de învățare supervizată acei algoritmi care folosesc la antrenament seturi de perechi de date (x, y) unde x reprezintă trăsăturile sau atributele unui exemplar, iar y reprezintă răspunsul dorit. După ce a avut loc învățarea, algoritmul va fi capabil să producă un răspuns și în cazul unor exemplare pe care nu le-a mai întâlnit, de aceea în literatura de specialitate clasificatorii se mai numesc și predictori.

Scopul învățării automate, dacă privim problema din punct de vedere geometric, este acela de a găsi o linie care să separe cele două clase între ele. (fig: 2.10)

Cel mai des întâlniți algoritmi de învățare în viziunea artificială sunt: automatul cu vectori de suport^[19]¹⁴ și rețeaua neuronală.

¹³Eng. Machine Learning

¹⁴Eng. Support Vector Machines

2.4. POST-PROCESAREA REZULTATELOR

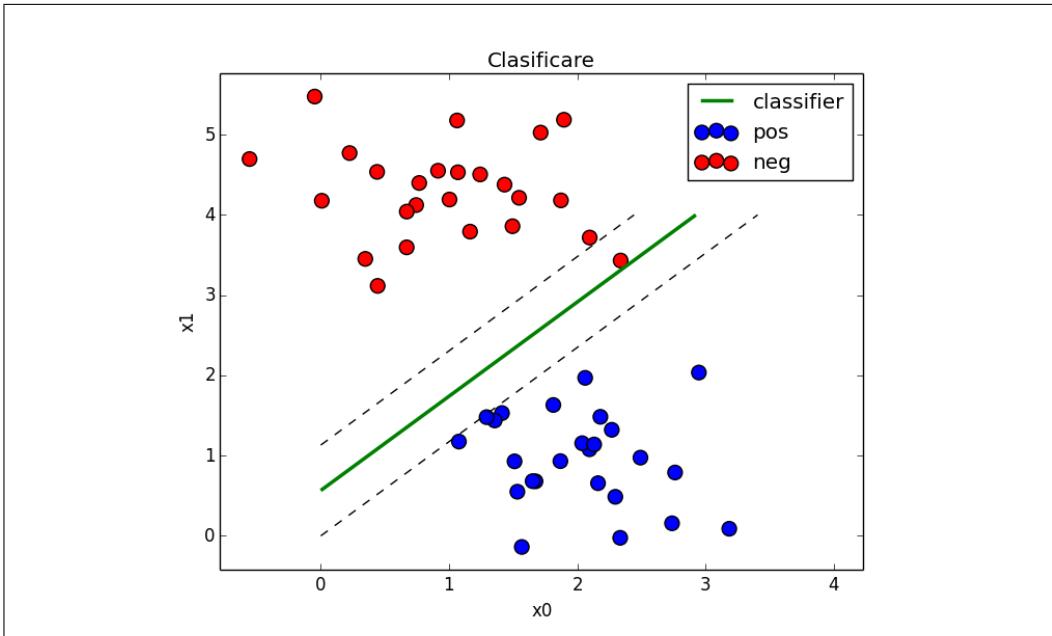


Figura 2.10: Clasificare

2.4 Post-procesarea rezultatelor

O situație foarte des întâlnita în cazul algoritmilor de recunoaștere a imaginilor este ca același obiect este detectat de mai multe ori. Aceste detecții sunt suprapuse și se datorează faptului că modelul învățat recunoaște și obiecte cu mici translații și scalari. Totuși, se dorește ca fiecare obiect prezent în imagine să fie detectat doar o singura data. Acest lucru se realizează cu ajutorul unui algoritm de grupare a detecțiilor suprapuse.



Figura 2.11: Efectul post-procesarii

2.5 Algoritmul de recunoaștere

Folosindu-ne de componentelete descrise în acest capitol putem discuta despre algoritmii de recunoaștere și antrenarea lor.

Algoritmul de recunoaștere a obiectelor poate fi descris cu ajutorul pseudocodului urmator:

```
detectii = lista_goale()

I = citeste_imaginea()
P = construieste_piramida(I)

pentru fiecare nivel din P
    pentru fiecare fereastra din extrage_ferestele(P)
        trasaturi = extrage_trasaturi(fereastra)
        raspuns = clasificare(trasaturi)
        daca raspuns este afirmativ atunci
            detectii = adauga(detectii, locatie(fereastra))
        sfarsit daca
        sfarsit
    sfarsit

detectii = grupare_suprapuse(detectii)
```

Pentru antrenarea unui algoritm de recunoaștere a obiectelor avem nevoie de o baza de date cu două seturi de imagini. Un set va conține imagini decupate cu obiectul pe care dorim să îl recunoaștem, iar al doilea va fi constituit din imagini care nu conțin obiectul. Aceste seturi se numesc setul de exemplare pozitive, respectiv negative. Setul de pozitive este adus la o mărime comună prin redimensionare. Din setul de imagini negative se vor extrage exemplare folosind scanarea în scara și spațiul de la algoritmul de recunoaștere. Pentru ca setul de negative este de obicei foarte mare, nu este practic ca la antrenare să se folosească toate exemplarele posibile. Exemplarele negative se vor extrage printr-un proces iterativ. În prima fază se extrage un număr ales de exemple negative și se antrenează clasificatorul. Apoi, folosind clasificatorul antrenat la pasul anterior, se scanăzează imaginile negative. Fiecare exemplar negativ care a fost clasificat pozitiv se adaugă la lista de antrenare și se antrenează clasificatorul din nou. Pasul acesta se repeta de un număr de ori specificat de utilizator, sau pana când nu se mai

2.5. ALGORITMUL DE RECUNOAȘTERE

pot extrage exemplare negative din setul de date. Acest procedeu se numește bootstrapping.

```
P = citeste_setul_de_exemplare_positive()
N = citeste_setul_de_exemplare_negative()

X = lista()
y = lista()

X = adauga(X, P)
y = adauga(y, selecteaza_aleator(N))

Cls = antreneaza_clasificator(X,y)

iter = citeste_nr_iteratii()

pentru i = 1 pana la iter
    pentru I din N
        P = construieste_piramida(I)
        pentru fiecare nivel din P
            pentru fiecare fereastra din extrage_ferestele(P)
                xi = extrage_trasaturi(xi)
                raspuns = clasificare(Cls, xi)
                daca raspuns este 'afirmativ' atunci
                    X = adauga(X, xi)
                    y = adauga(y, 'negativ')
                    sfarsit daca
                sfarsit
                sfarsit
                sfarsit

    Cls = antreneaza_clasificator(X,y)
sfarsit
```

Capitolul 3

Tehnologii folosite

În acest capitol voi enumera tehnologiile folosite și voi oferi cate o scurta descriere pentru fiecare.

3.1 Limbajul C++

C++ este un limbaj de programare general, multi-paradigma, compilat cu verificare statică a tipurilor. Suporta programarea: procedurala, orientată pe obiecte și generică. Limbajul oferă acces la facilități de manipulare a memoriei pana la cel mai scăzut nivel.

Câteva dintre trăsăturile limbajului care îl fac potrivit pentru aceasta lucrare sunt:

- Performanta. Multi dintre algoritmi ce vor fi descris au de procesat un volum foarte mare de date. Este important ca execuția să fie cat mai rapidă.
- Portabilitatea. Ne oferă posibilitatea de a porta codul către alte platforme.
- Biblioteci. Sunt scrise multe biblioteci pentru acest limbaj. În plus faptul că este compatibil cu C ne pune practic la dispoziție un întreg arsenal de biblioteci.
- Modern. În ultimi ani limbajul a trecut printr-o întreaga serie de schimbări care l-au transformat într-un limbaj mult mai ușor de folosit.

3.2 Limbajul Python

Limbajul Python este un limbaj interpretat, dinamic, foarte ușor de folosit. Ca și în cazul C++, Python este un limbaj multi-paradigma și suportă programarea orientată pe obiecte, procedurală și generică. Este portabil, și vine însăși de o librărie standard vastă, cu facilități de lucru cu fișiere XML până la programare distribuită. Vine integrat cu un manager de pachete: bibliotecile pot fi instalate mult mai ușor. Are toate beneficiile unui limbaj modern: management automat de memorie, serializare automată a obiectelor și reflexia tipurilor.

Python este un limbaj foarte popular în domeniul învățării automate, de aceea sunt disponibile foarte multe biblioteci care susțin acest domeniu. Este și un limbaj care se scalează de la mici aplicații demonstrative până la aplicații distribuite cu sute de noduri.

3.3 Biblioteca Boost

Este una dintre cele mai mari și de calitate biblioteci C++. Din biblioteca Boost am folosit Boost.Serialization și Boost.Python, cu ajutorul cărora am implementat serializarea obiectelor și interoperabilitatea între C++ și Python pentru sistemul dezvoltat în aceasta lucrare.

3.4 Biblioteca scikit-learn

Este cea mai cunoscută bibliotecă Python de învățare automată. Sunt implementații zeci de algoritmi de învățare, selecție de trăsături, validare și reducere a dimensionalității.

3.5 Biblioteca numpy

Biblioteca numpy a transformat Python-ul într-un adevarat rival al MATLAB. Aceasta bibliotecă face lucru cu vectori și matrici în Python foarte simplu și eficient.

3.6 Biblioteca matplotlib

Matplotlib este o biblioteca Python de vizualizare a datelor. Se pot afișa imagini, trase grafice sau vizualiza grafuri.

3.7 Biblioteca PySide

Biblioteca PySide este interfața de programare în Python a bibliotecii Qt, una dintre cele mai populare biblioteci de dezvoltare a interfețelor de utilizator portabile. Deși este cunoscută pentru dezvoltarea de interfețe grafice, Qt este de fapt un cadru de lucru complet de dezvoltare de aplicații cu facilități de lucru cu baze de date, fire de execuție, comunicare în rețea sau grafica 3d.

Capitolul 4

Implementarea

În acest capitol va fi prezentata implementarea bibliotecii software pentru dezvoltarea de algoritmi și aplicații de recunoaștere a obiectelor.

Biblioteca a fost implementata într-un mod hibrid. Interfețele au fost definite în limbajul C++, iar implementările au fost făcute atât în C++ cat și în Python. Biblioteca a fost scrisă și testată cu Visual Studio 2013.

Structura de directoare a bibliotecii este:

```
+---include
|   \---object-recognition-toolkit
|       +---classification
|       +---core
|       +---dataset
|       +---detection
|       +---feature-extraction
|       +---image-pyramid
|       +---image-scanning
|       +---non_maxima_suppression
|       \---python
\---src
    \---object-recognition-toolkit
        +---classification
        +---core
        +---dataset
        |   \---imglab
```

```
+---detection
+---feature-extraction
+---image-pyramid
+---image-scanning
+---non_maxima_suppression
\---python
```

Fiecare pachet este situat in propriul director. Declaratiile se gasesc in directorul "include", iar implementarile in "src".

4.1. DIAGRAMA DE CLASE

4.1 Diagrama de clase

4.1.1 Core

Pachetul "core" conține interfețe de baza ale bibliotecii.

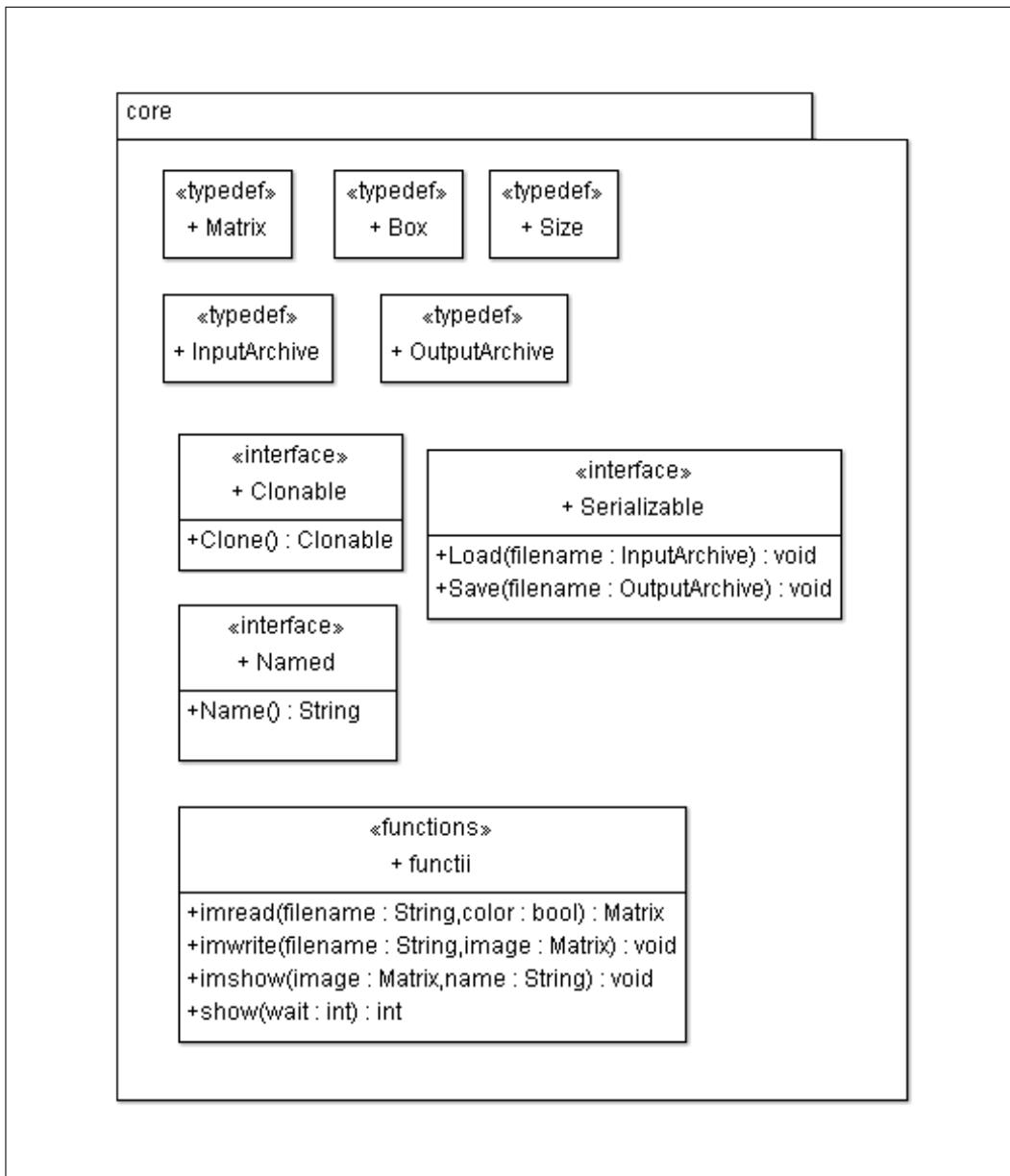


Figura 4.1: core Class Diagram

4.1. DIAGRAMA DE CLASE

4.1.2 Dataset

Pachetul "dataset" conține clase care modelează baza de date pentru antrenament și implementează funcționalități de importare unor formate uzuale.

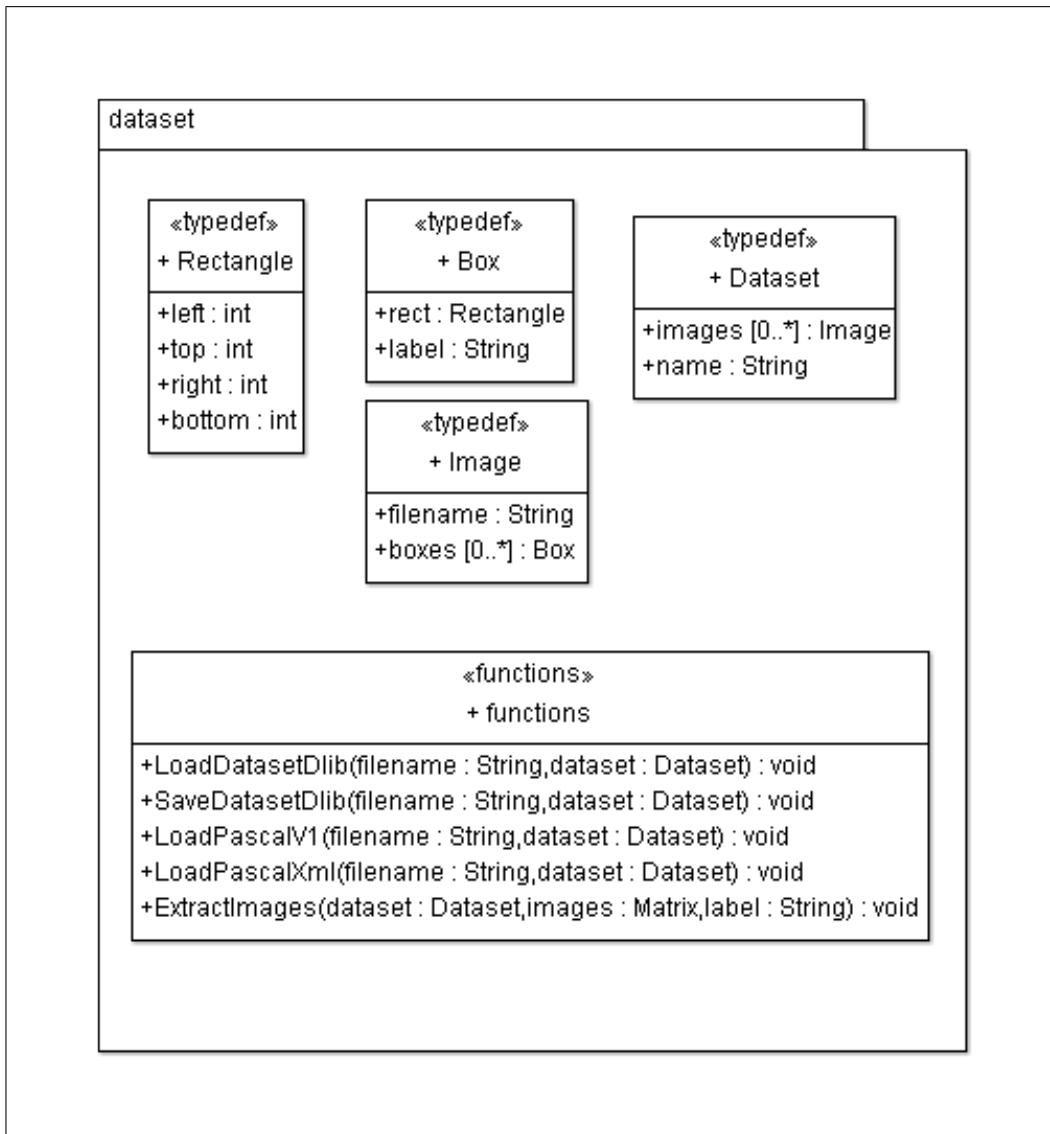


Figura 4.2: Diagrama de clase: dataset

4.1. DIAGRAMA DE CLASE

4.1.3 Image Pyramid

Pachetul "image-pyramid" conține interfețe și implementări care servesc la construcția piramidei de imagini.

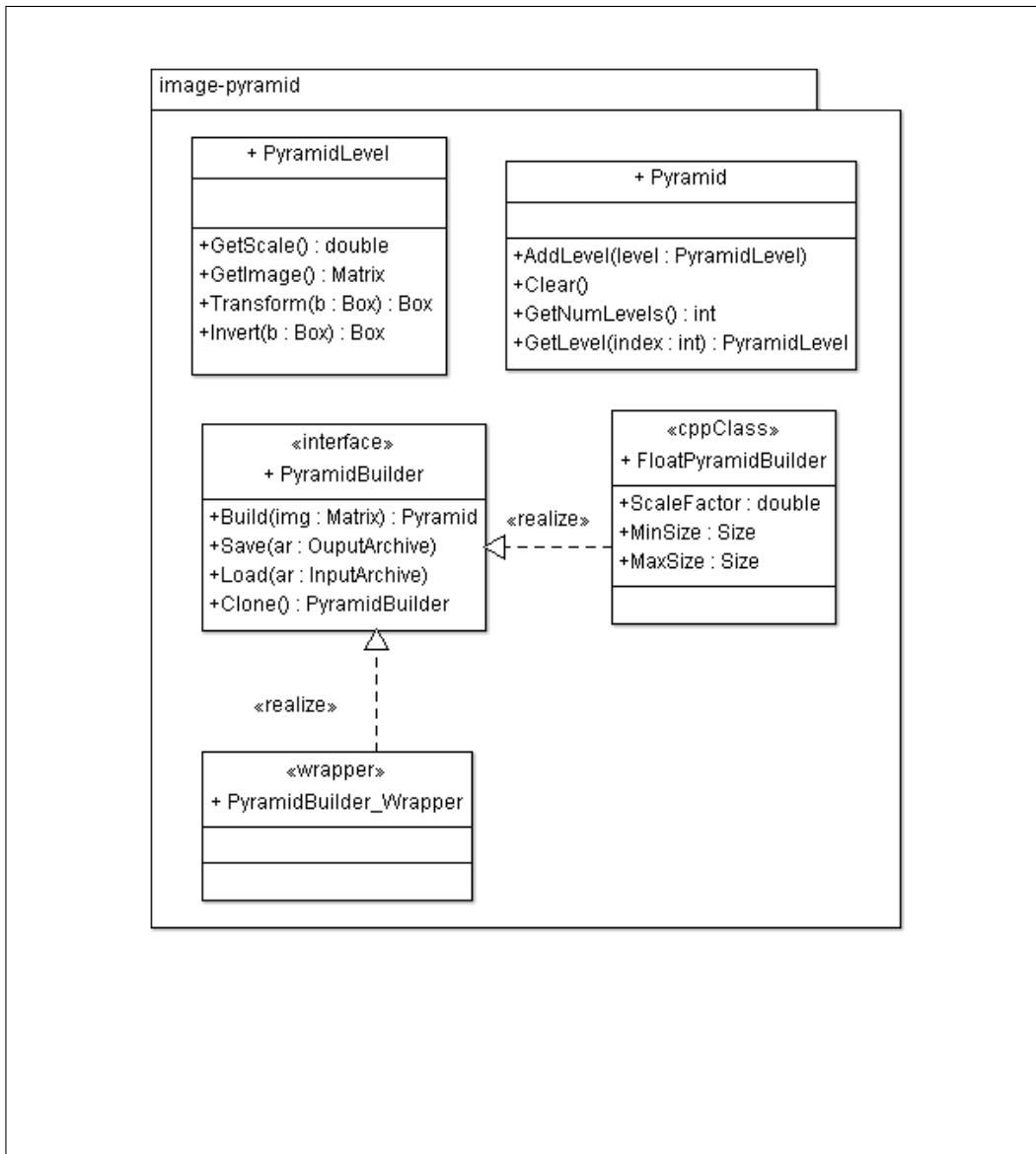


Figura 4.3: Diagrama de clase: image-pyramid

Clasa `FloatPyramidBuilder` construiește o piramidă de imagini folosind `ScaleFactor` ca factor de scalare și `MinSize`, `MaxSize` ca criterii de terminare.

4.1. DIAGRAMA DE CLASE

Metodele Transform și Invert din clasa PyramidLevel transformă coordinate din spațiul imaginii sursă în cel al nivelului, respectiv invers.

4.1.4 Image Scanning

Pachetul "image-scanning" conține interfețe și implementări care servesc la scanarea imaginilor

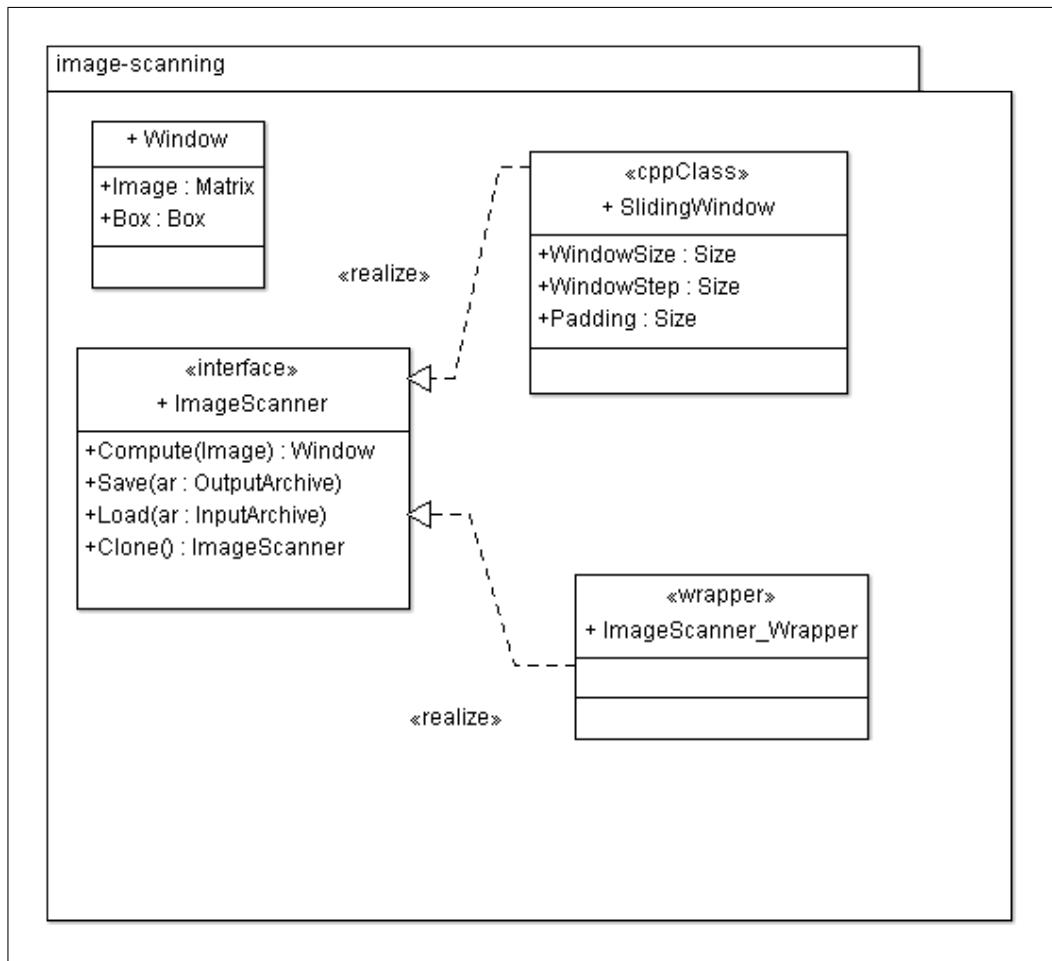


Figura 4.4: Diagrama de clase: image-scanning

4.1. DIAGRAMA DE CLASE

4.1.5 Feature Extraction

Pachetul "feature-extraction" conține interfețe și implementări care servesc la extragerea de trăsături din imagini.

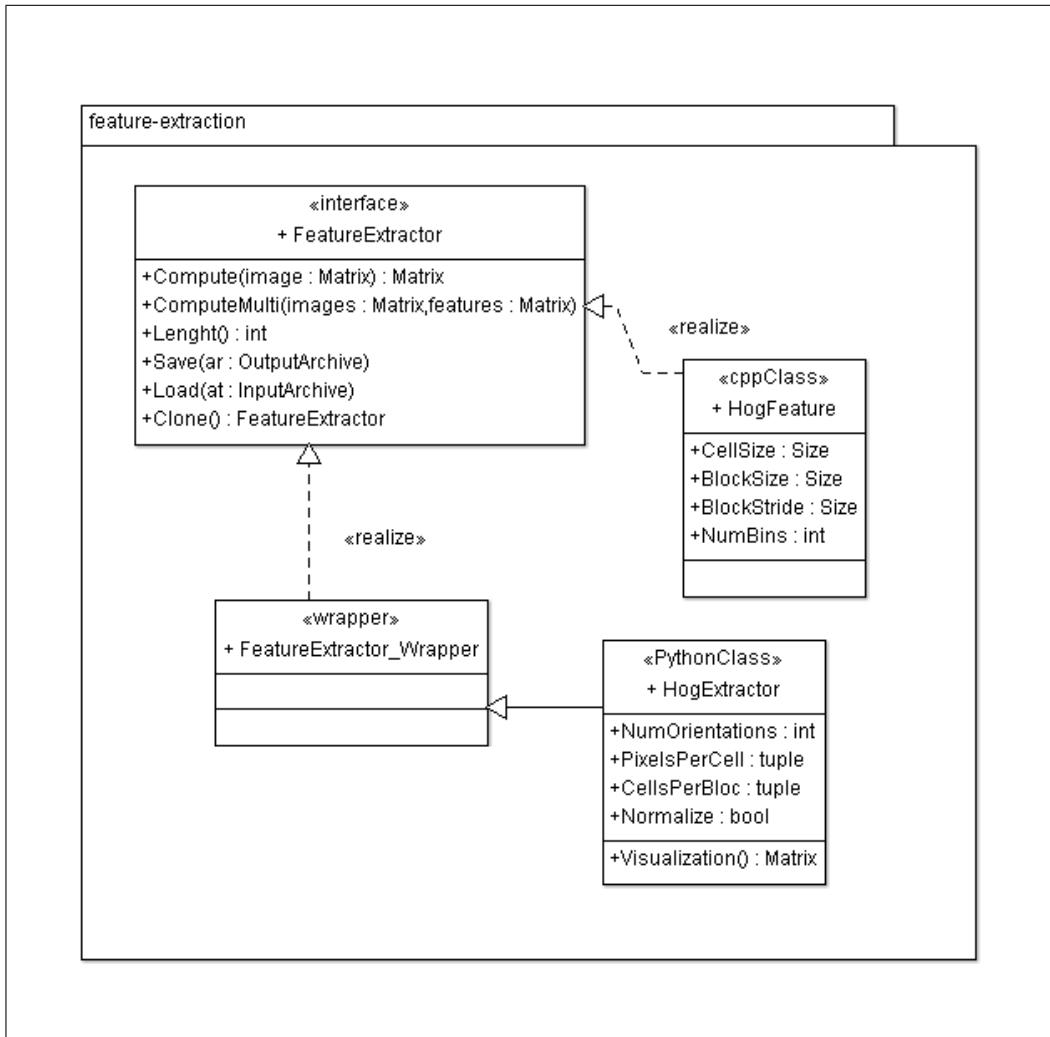


Figura 4.5: Diagrama de clase: feature-extraction

4.1. DIAGRAMA DE CLASE

4.1.6 Classification

Pachetul "classification" conține interfețe și implementări care servesc la clasificare și antrenarea clasificatorilor.

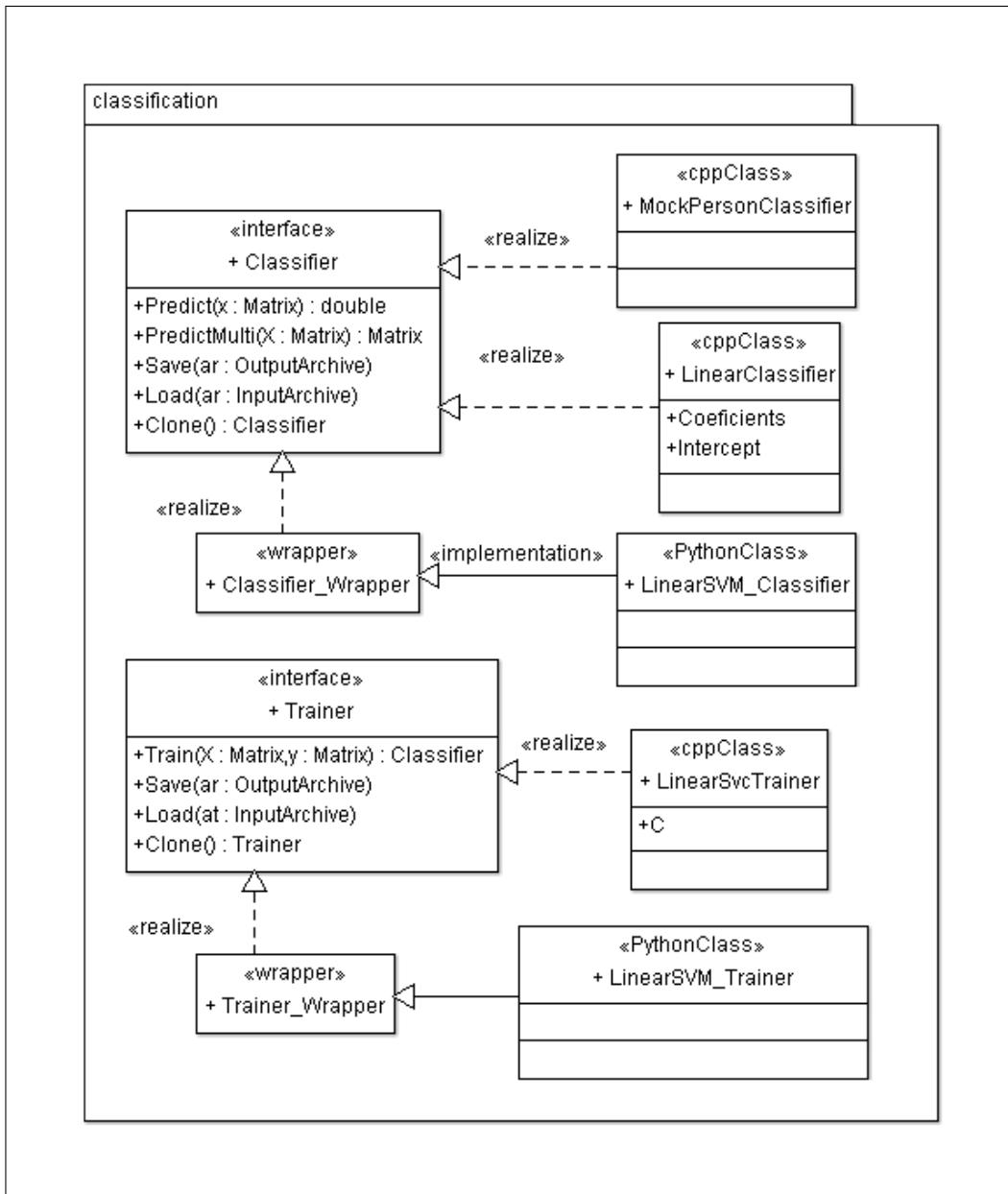


Figura 4.6: Diagrama de clase: classification

4.1. DIAGRAMA DE CLASE

4.1.7 Non Maxima Suppression

Pachetul "non-maxima-suppression" conține interfețe și implementări care servesc la post-procesarea rezultatelor.

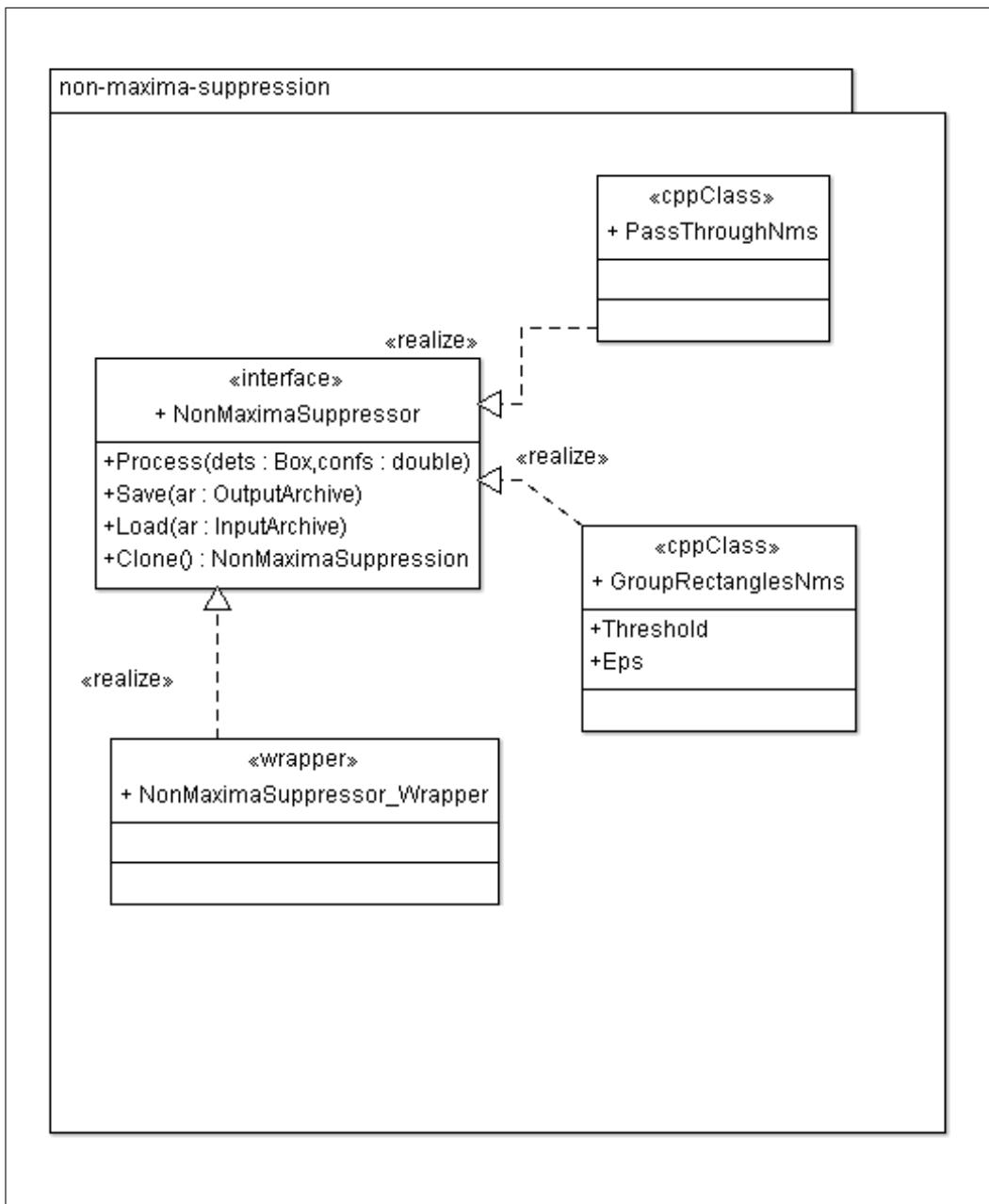


Figura 4.7: Diagrama de clase: non-maxima-suppression

4.1. DIAGRAMA DE CLASE

4.1.8 Detection

Pachetul "detection" conține interfețe și implementări care servesc la recunoașterea obiectelor în imagini și la antrenarea algoritmilor.

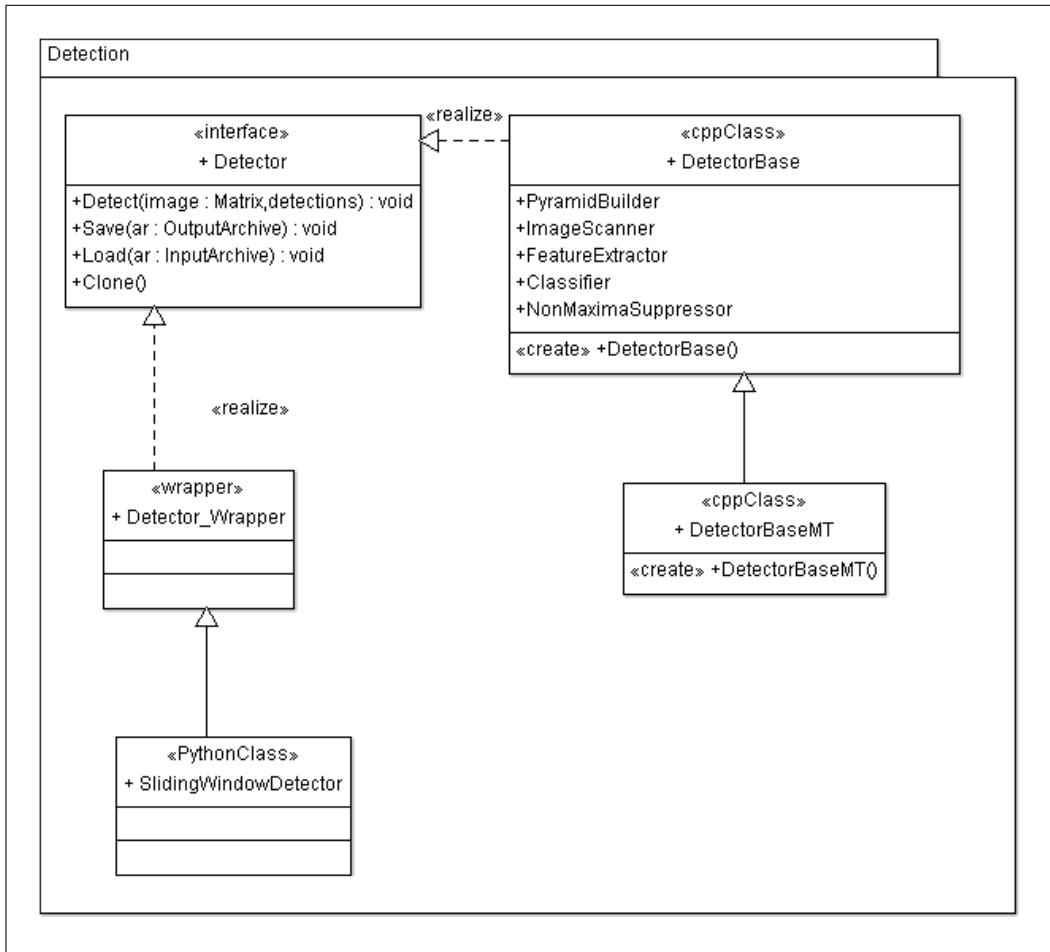


Figura 4.8: Diagrama de clase: detection

4.1. DIAGRAMA DE CLASE

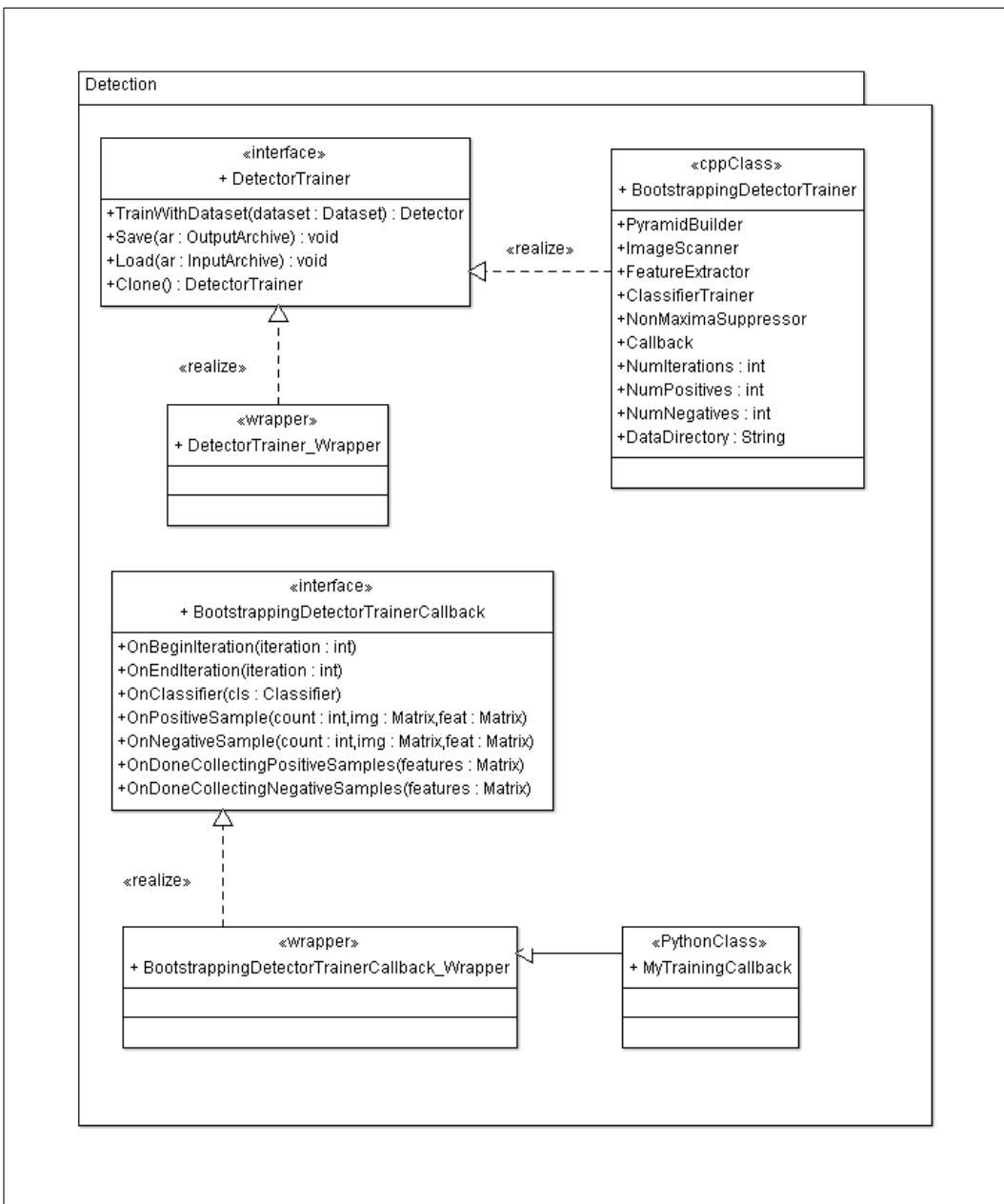


Figura 4.9: Diagrama de clase: detection

4.1. DIAGRAMA DE CLASE

4.1.9 Python

Pachetul "python" conține suportul necesar pentru interoperabilitatea cu limbajul Python.

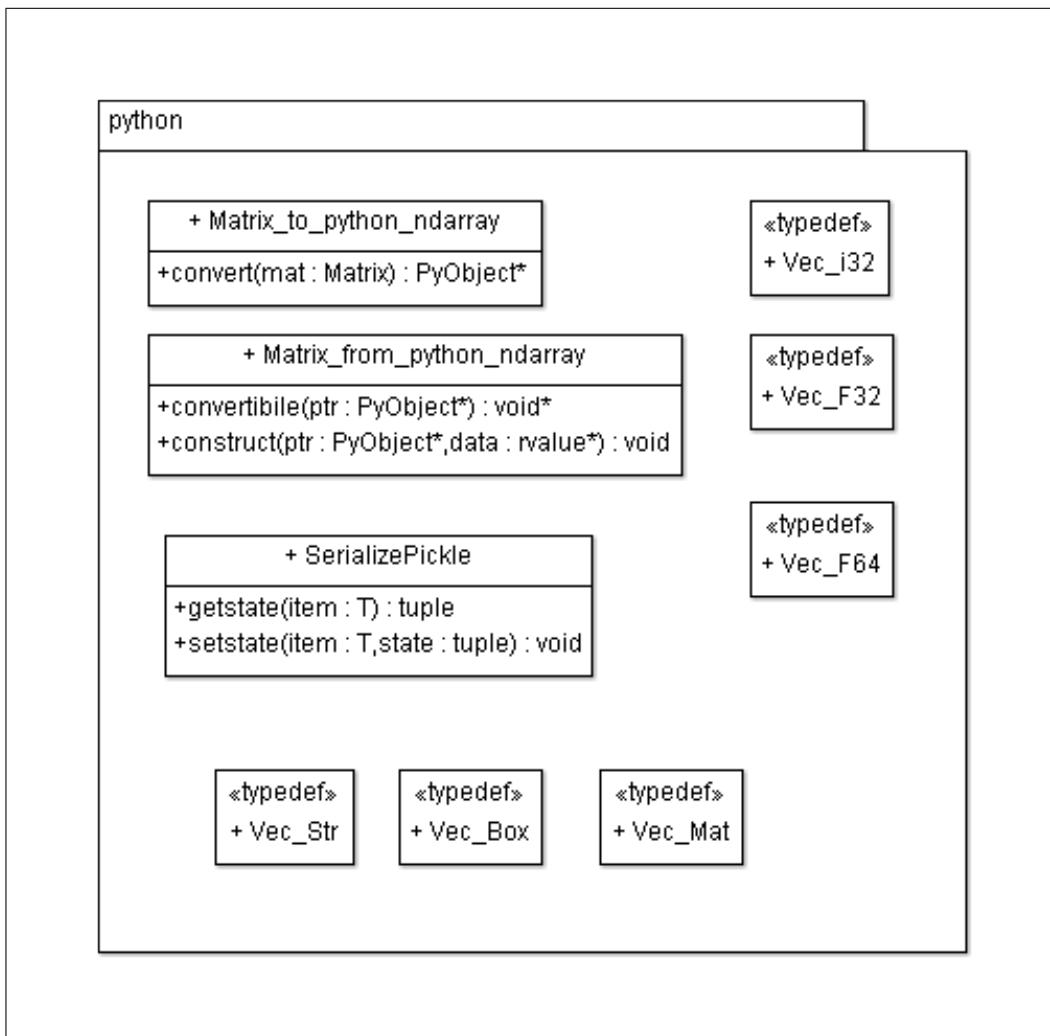


Figura 4.10: Diagrama de clase python

4.2 Algoritm de recunoaștere a persoanelor

Cu ajutorul bibliotecii descrise în lucrare am implementat algoritmul de recunoaștere a persoanelor în imagini descris în lucrarea "Histograms of Oriented Gradients for Human Detection", scrisă de Navneet Dalal și Bill Triggs.[9] Acesta lucrare a introdus și descris extragerea trăsăturilor de imagine: histograma de gradienți orientați.

Pentru implementarea algoritmului s-au folosit următoarele componente:

- Parcurgea imaginii: FloatPyramidBuilder și SlidingWindow
- Extragerea de trăsături: HogFeature
- Clasificare: LinearSVM_Classifier și LinearSVM_Trainer
- Post-procesare: GroupRectanglesNms
- Baza de date: Dataset
- Antrenament: BootstrappingDetectorTrainer

Setul de imagini folosit pentru antrenarea algoritmul este cel oferit de institutul Inria(todo: cite) și poate fi descărcat gratuit de pe pagina web: <http://pascal.inrialpes.fr/data/human/>. Sunt oferite imagini și adnotări în formatul PASCAL VOC(todo: cite). Datele conțin exemplare pozitive și negative.

Înainte de antrenare imaginile exemplarelor pozitive au fost decupate și redimensionate la o mărime de 64 pixeli lățime și 128 pixeli înălțime. Exemplarele negative au fost folosite în întregime. (fig. 4.11)



Figura 4.11: Exemplare pozitive din setul de date

4.2. ALGORITM DE RECUNOAȘTERE A PERSOANELOR

Antrenarea cu "bootstrapping" a fost efectuata folosind 2500 de exemplare pozitive într-un număr de 10 iterări fiecare adăugând 1.000 de exemplare negative la modelul învățat. În total s-au folosit 10.000 de exemplare negative.

Lungimea vectorului de trăsături pentru o fereastră de dimensiune 64x128, folosindu-se celule de 8x8, blocuri de 16x16 și histograma cu 9 valori, este de 3870 de componente.

Factorul de scalare a piramidei de imagini este setat la $6/5 = 1.2$, iar pasul ferestrei glisante este de 8 pixeli în ambele direcții.

În timpul antrenării modelul rezultat după fiecare iterărie intermediara a fost salvat.

4.2.1 Cod antrenament

Codul programului de antrenare în Python este:

```
1 from __future__ import division, print_function
2
3 import sys
4 import os
5 import numpy as np
6 from math import floor
7 import pickle
8 from object_recognition_toolkit import *
9 from my_hog import *
10 from my_svm import *
11
12
13 class MyTrainingCallback(BootstrappingDetectorTrainerCallback):
14     def __init__(self, num_iter, num_pos, num_neg, data_dir):
15         BootstrappingDetectorTrainerCallback.__init__(self)
16         self.num_iterations = num_iter
17         self.num_positives = num_pos
18         self.num_negatives = num_neg
19         self.data_dir = data_dir
20         self.current_iteration = 0;
21     return
22
23     def OnBeginIteration(self, iteration):
24         self.current_iteration = iteration + 1;
25         print()
26         print('Begin iteration {} / {}'.format(
27             self.current_iteration,
28             self.num_iterations))
29         print()
```

4.2. ALGORITM DE RECUNOAȘTERE A PERSOANELOR

```
30     return
31
32 def OnClassifier(self, classifier):
33     print()
34     cls_fn = os.path.join(self.data_dir,
35                           'classifier_{}.pkl'.format(self.current_iteration))
36     with open(cls_fn, mode='w') as f:
37         pickle.dump(classifier, f)
38
39     del classifier
40
41     print('Got classifier->{}{}'.format(cls_fn))
42     print()
43     return
44
45 def OnEndIteration(self, iteration):
46     print()
47     print('End iteration{} / {}'.format(
48         self.current_iteration,
49         self.num_iterations))
50     print()
51     return
52
53 def OnPositiveSample(self, count, image, features_vector):
54     msg = 'collecting positive samples{} / {}'.format(
55         count,
56         self.num_positives)
57     sys.stdout.write('\r' + msg)
58     sys.stdout.flush()
59     return
60
61 def OnDoneCollectingPositiveSamples(self, features):
62     print()
63     msg = 'done collecting positive samples{} / {}'.format(
64         features.shape[0],
65         self.num_positives)
66     print(msg)
67     fn = os.path.join(self.data_dir, 'positive_samples_{}.npy'.format(
68         self.current_iteration))
69     np.save(fn, features);
70     msg = 'positive samples saved to:{}'.format(fn)
71     print(msg)
72     print()
73     return
74
75 def OnNegativeSample(self, count, image, features_vector,
76                      score):
77     msg = 'collecting negative samples{} / {}'.format(
```

4.2. ALGORITM DE RECUNOAȘTERE A PERSOANELOR

```
77         count ,
78         self.num_negatives)
79     sys.stdout.write('\r' + msg)
80     sys.stdout.flush()
81     return
82
83 def OnDoneCollectingNegativeSamples(self, features):
84     print()
85     msg = 'done\u00e2\u00e0 collecting\u00e2\u00e0 negative\u00e2\u00e0 samples\u00e2\u00e0{} / {}'.format(
86         features.shape[0],
87         self.num_negatives)
88     print(msg)
89     fn = os.path.join(self.data_dir, 'negative_samples_{}.npy'.format(
90         self.current_iteration))
91     np.save(fn, features);
92     msg = 'negative\u00e2\u00e0 samples\u00e2\u00e0 saved\u00e2\u00e0 to : {}'.format(fn)
93     print(msg)
94     print()
95     return
96
97 def main():
98
99     pid = os.getpid()
100    print('PID={}'.format(pid))
101
102    raw_input('Press\u00e2\u00e0 any\u00e2\u00e0 key\u00e2\u00e0 begin\u00e2\u00e0 training ... ')
103
104    num_iterations = 10
105    num_positives = 2500
106    num_negatives = 1000
107    data_dir = './training_10'
108
109    if os.path.exists(data_dir) == False:
110        os.makedirs(data_dir)
111
112    scale_factor = 1.2
113    win_size = Size(64,128)
114    win_step = Size(8,8)
115    block_size = Size(16,16)
116    block_stride = win_step
117    cell_size = Size(8,8)
118    n_bins = 9
119
120    pyramid_builder = FloatPyramidBuilder(
121        scale_factor=scale_factor,
122        min_size=win_size,
123        max_size=Size())
124    image_scanner = DenseImageScanner(
```

4.2. ALGORITM DE RECUNOAȘTERE A PERSOANELOR

```
125     win_size=win_size ,  
126     win_step=win_step ,  
127     padding=Size ())  
128     feature_extractor = HogFeatureExtractor (  
129         win_size ,  
130         block_size ,  
131         block_stride ,  
132         cell_size ,  
133         n_bins)  
134     trainer = LinearSVM_Trainer (C=0.1)  
135     nms = PassThroughNms ()  
136     callback = MyTrainingCallback (num_iterations ,  
137         num_positives ,  
138         num_negatives ,  
139         data_dir)  
140  
141     feature_extracotr_path = os.path.join (data_dir , '   
142         feature_extracotr.pkl')  
143     with open (feature_extracotr_path , 'w') as f:  
144         pickle.dump (feature_extractor , f)  
145  
146     detector_trainer = BootstrappingDetectorTrainer ()  
147  
148     detector_trainer . num_iterations=num_iterations  
149     detector_trainer . num_positives=num_positives  
150     detector_trainer . num_negatives=num_negatives  
151     detector_trainer . detector_size=win_size  
152     detector_trainer . data_directory=data_dir  
153     detector_trainer . pyramid_builder=pyramid_builder  
154     detector_trainer . image_scanner=image_scanner  
155     detector_trainer . feature_extractor=feature_extractor  
156     detector_trainer . trainer=trainer  
157     detector_trainer . non_max_supppersor=nms  
158     detector_trainer . callback = callback  
159  
160     positive_dataset = Dataset ()  
161     negative_dataset = Dataset ()  
162  
163     LoadDatasetDlib ("positive_train_dataset_crop.xml" ,  
164         positive_dataset)  
165     LoadDatasetDlib ("negative_train_dataset.xml" ,  
166         negative_dataset)  
167  
168     detector = detector_trainer . TrainWithDataset (  
169         positive_dataset ,  
             negative_dataset)
```

4.2. ALGORITM DE RECUNOAȘTERE A PERSOANELOR

```
170 | if __name__ == '__main__':
171 |     main()
```

4.2.2 Rezultate

În continuare voi prezenta câteva rezultate ale algoritmului implementat.

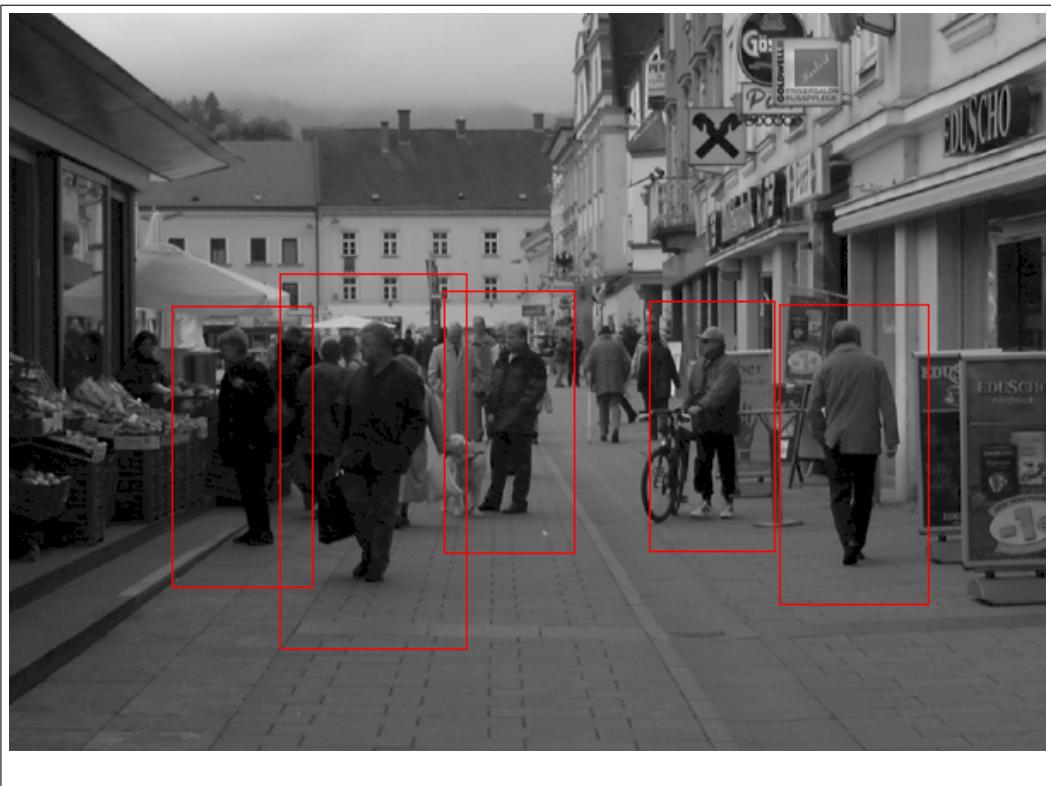


Figura 4.12: Rezultate recunoastere persoane

4.2. ALGORITM DE RECUNOAȘTERE A PERSOANELOR



Figura 4.13: Rezultate recunoastere persoane

4.2. ALGORITM DE RECUNOAȘTERE A PERSOANELOR

4.2.3 Aplicație de recunoaștere a obiectelor

În continuare voi prezenta aplicația de recunoaștere a obiectelor.

Aceasta aplicație are rolul de a demonstra algoritmi de recunoaștere dezvoltăți cu cadrul de lucru descris în aceasta lucrare.

Aplicația suportă următoarele operațiuni:

- Încărcare și salvare algoritm
- Încărcare imagine
- Configurare parametrii algoritm
- Executarea algoritmului

Interfața cu utilizatorul a fost dezvoltată folosind biblioteca PySide și poate fi observată în figura 4.15.

Un algoritm poate fi configurat cu ajutorul dialogului de configurare din figura 4.16.

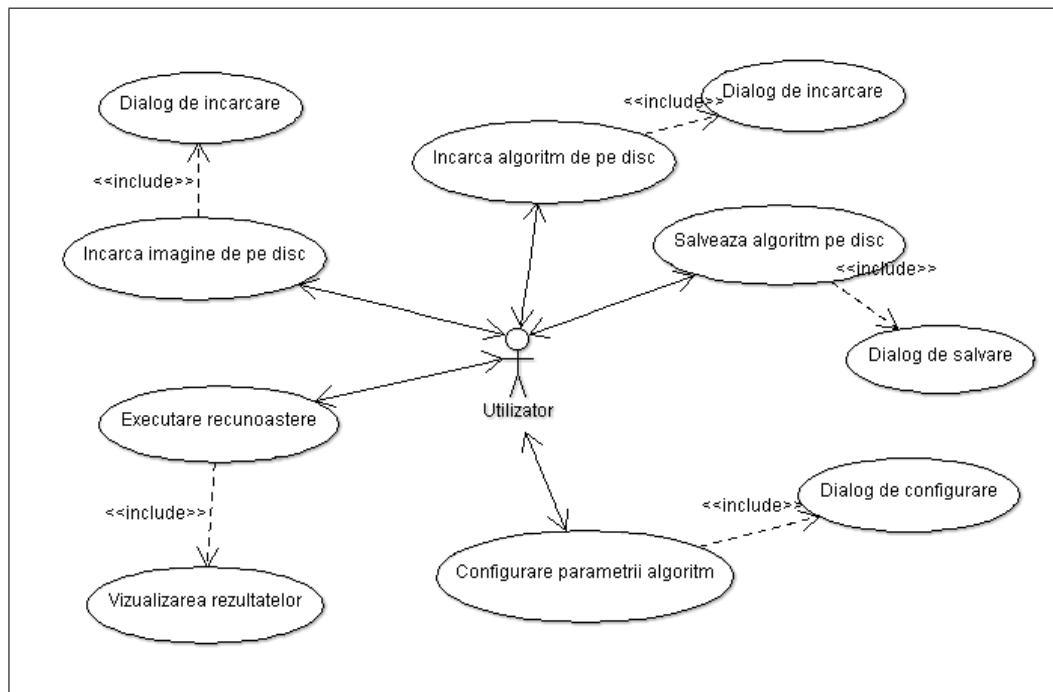


Figura 4.14: Diagrama Use Case a aplicatiei

4.2. ALGORITM DE RECUNOAȘTERE A PERSOANELOR

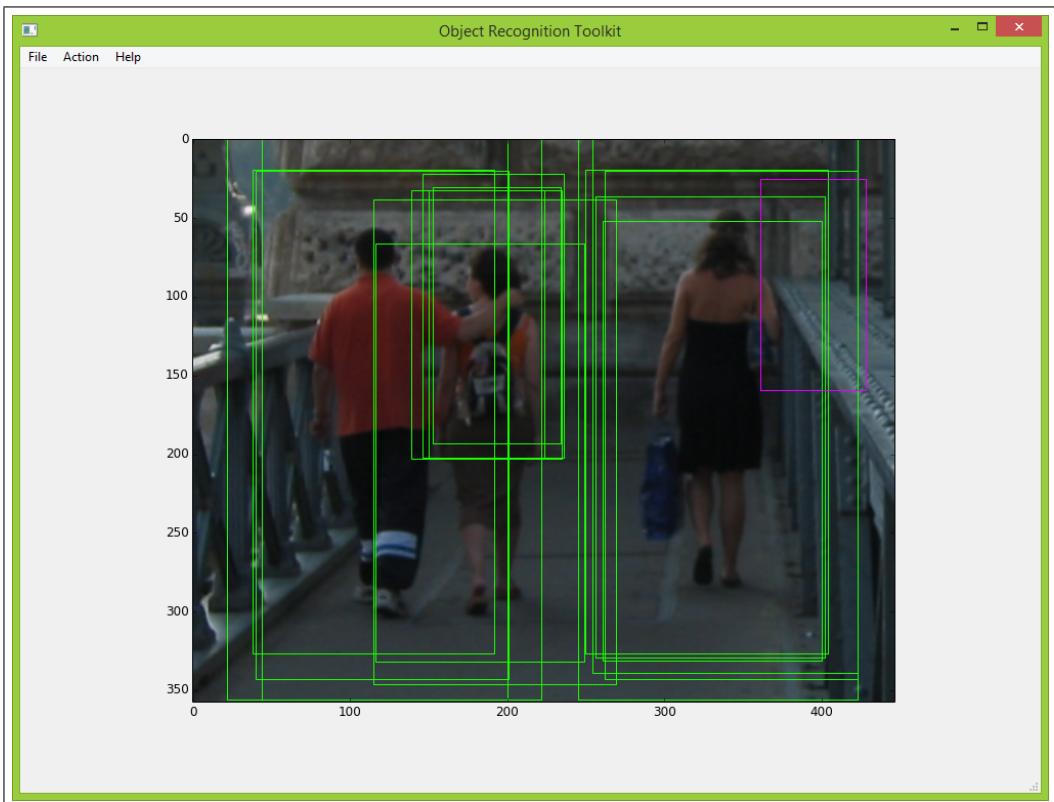


Figura 4.15: Aplicatia de recunoastere

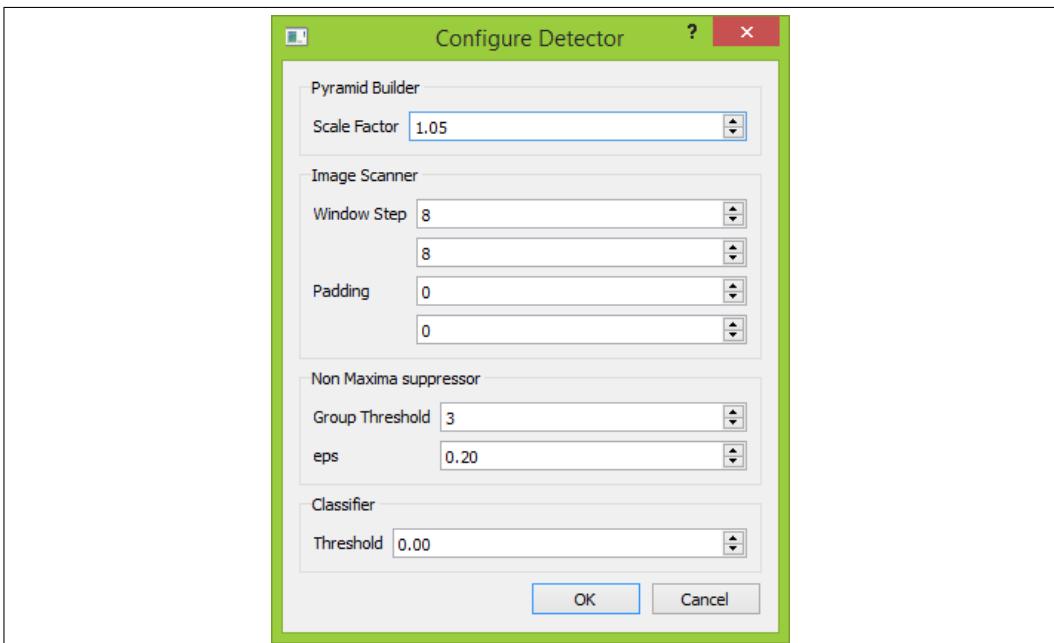


Figura 4.16: Dialog de configurare

Capitolul 5

Concluzii

În aceasta lucrare am prezentat în ce constă dezvoltarea unui sistem de recunoaștere automata a obiectelor în imagini. Au fost prezentate în detaliu componentele din care este alcătuit un astfel de sistem. De asemenea am prezentat un cadru de lucru pentru dezvoltarea de algoritmi și aplicații de recunoaștere a obiectelor.

Algoritmii și aplicațiile implementate folosind acest cadru de lucru pot fi scrise în C++ și Python. Acest lucru permite dezvoltarea rapidă a aplicațiilor, iar atunci când este nevoie de performanță se pot scrie parti din algoritm sau aplicație în C++. Totuși există foarte mulți algoritmi implementați în limbi de programare precum Java și Matlab pe care nu îi putem folosi în mod direct. O direcție de dezvoltare ar ca cadrul de lucru să permită dezvoltarea și în alte limbi de programare. Acest lucru s-ar putea realiza folosind tehnici din programarea distribuită: RMI, CORBA sau COM.

Folosind acest cadru am implementat un algoritm de recunoaștere a persoanelor în imagini, o aplicație de antrenare și una care să-l folosească. Astfel cadrul de lucru și-a atins scopul de a servi în dezvoltarea de algoritmi și aplicații. Totuși algoritmul nu poate recunoaște decât obiecte rigide. O direcție de dezvoltare ar fi: identificarea și implementarea altor algoritmi sau tehnici de recunoaștere a obiectelor.

O alta direcție de dezvoltare a fi implementarea unei aplicații care să permită oricui să antreneze algoritmi fără a avea cunoștințe în domeniu. Aceasta aplicație ar putea avea o interfață web și ar putea funcționa ca un software as a service.

Bibliografie

- [1] <http://www.volvo.com>.
- [2] <https://docs.python.org/2/c-api>.
- [3] <http://www.swig.org>.
- [4] <http://www.cosc.canterbury.ac.nz/greg.ewing/python/Pyrex>.
- [5] <https://docs.python.org/2/library/ctypes.html>.
- [6] <http://riverbankcomputing.co.uk/software/sip/intro>.
- [7] http://www.boost.org/doc/libs/1_55_0/libs/python/doc
- [8] Edward H Adelson, Charles H Anderson, James R Bergen, Peter J Burt, and Joan M Ogden. Pyramid methods in image processing. *RCA engineer*, 29(6):33–41, 1984.
- [9] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *In CVPR*, pages 886–893, 2005.
- [10] Pedro F. Felzenszwalb, Ross B. Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part based models.
- [11] Smith Graham. Smashing idea: Volvo installs pedestrina detection systems that brakes car automatically, February 2011.
- [12] Christoph H Lampert, Matthew B Blaschko, and Thomas Hofmann. Beyond sliding windows: Object localization by efficient subwindow search. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

BIBLIOGRAFIE

- [13] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998.
- [14] David G. Lowe. Object recognition from local scale-invariant features, 1999.
- [15] Tom Michael Mitchell. *Machine Learning*. McGraw-Hill Education, 1997.
- [16] Wanli Ouyang, Federico Tombari, Stefano Mattoccia, Luigi Di Stefano, and Wai-Kuen Cham. Segmented gray-code kernels for fast pattern matching. *IEEE Trans. Image Processing*, 22(4):1512–1525, Apr. 2013.
- [17] Phil Simon. *Too Big to Ignore: The Business Case for Big Data*, volume Volume 72 of Wiley and SAS Business Series. Wiley, 2013.
- [18] Vitomir Štruc, Janez Žibert, and Nikola Pavešić. Histogram remapping as a preprocessing step for robust face recognition. *image*, 7:9, 2009.
- [19] Johan AK Suykens and Joos Vandewalle. Least squares support vector machine classifiers. *Neural processing letters*, 9(3):293–300, 1999.
- [20] Richard Szeliski. *Computer vision: algorithms and applications*. Springer, 2010.
- [21] Marco Alexander Treiber. *An Introduction to Object Recognition*. Springer, 2010.
- [22] Paul Viola and Michael Jones. Robust real-time object detection. In *International Journal of Computer Vision*, 2001.
- [23] C. Vondrick, A. Khosla, T. Malisiewicz, and A. Torralba. HOGgles: Visualizing Object Detection Features. *ICCV*, 2013.
- [24] Michael Zhang. Canon face recognition feature gives friends preferential treatment, January 2012.