Universitatea Transilvania din Brașov

Facultatea de Matematică și Informatică

Specializarea Informatică

Proiect de Licență

Tehnici de Machine Learning în procesarea și recunoașterea imaginilor

Autor:
Draghia Alin-Mădălin

Profesor coordonator:
Lect. Dr. Sasu Lucian Mircea

Brașov

Iulie 2014

Cuprins

Capitolul 1

Introducere

Prin intermediul acestei lucrări doresc sa 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 automata.

Totodată aceasta lucrare vine însoțita 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 algoritmi de recunoaștere consacrați.

1.1 Motivație

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 sa depună prea mult efort, chiar dacă în aceste imagini obiectele prezintă variații de perspectiva, de dimensiune, sunt translatate, rotite sau chiar obstrucționate. Cu toate ca, de-a lungul timpului au fost studiați și dezvoltați multi algoritmi, sistemele de recunoaștere automata a obiectelor sunt încă departe de performanta unei ființe umane, chiar de cea a unui copil de numai doi ani, deci încă exista 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 aplicați pe niște probleme de dimensiune mai mare, cererea de aplicații a crescut.

Câteva dintre cele mai de succes aplicații sunt:

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

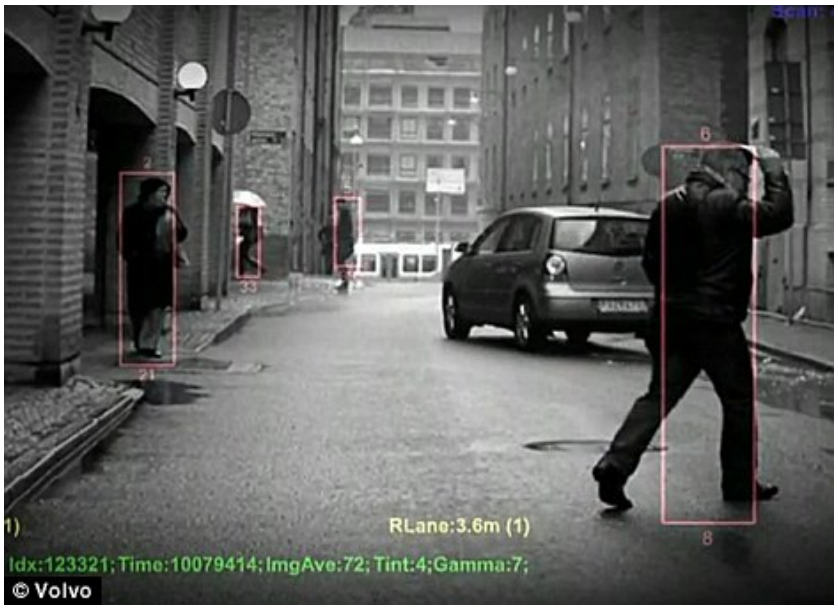

Figura 1.1: Volvo: sistemul de detectie a pietonilor [5]

- Focalizarea automata a camerelor foto pe fețe

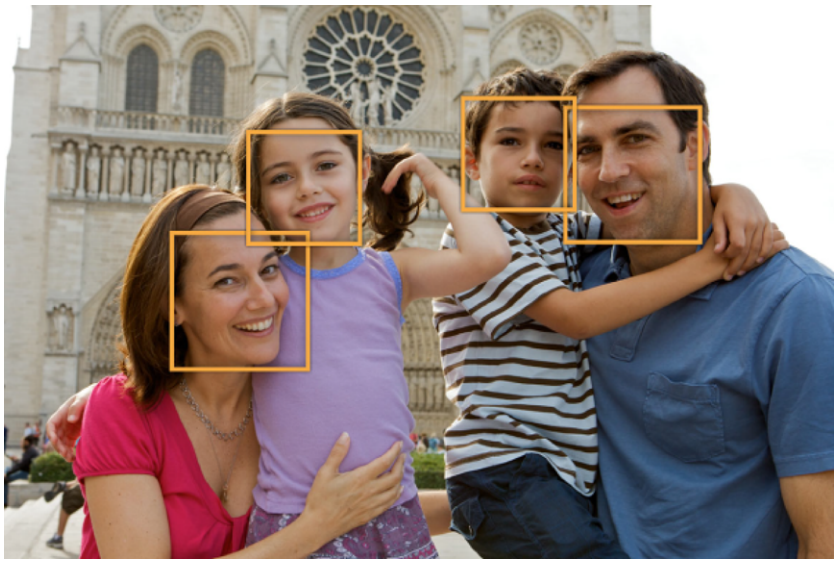

Figura 1.2: Camera foto: focus automat[18]

- Analizarea traficului rutier

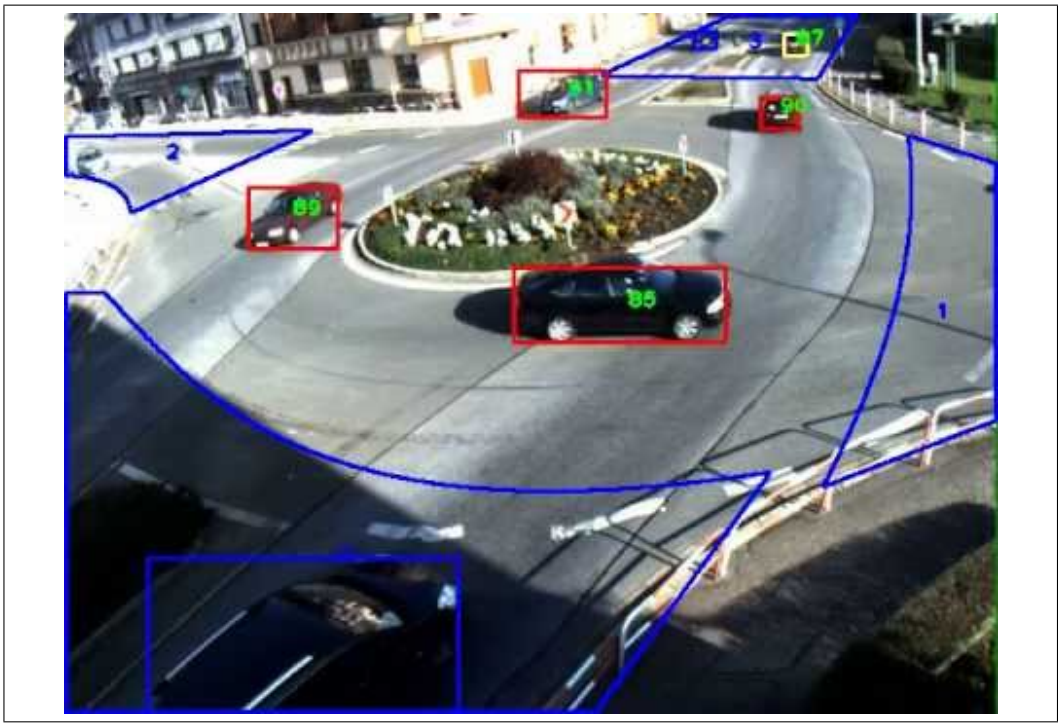

Figura 1.3: Analizarea traficului rutier[1]

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 scrisa 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 distanta intre ele, acestea find uneori foarte greu de găsit. Parcurgerea unui astfel de document, presupune cunoștințe extensive de matematica, statistica, învățare automata, procesarea imaginilor și chiar cunoștințe din domeniul biologic sau medical. Toate acestea fac ca nivelul de la care se intra acest domeniu sa fie unul foarte înalt, ceea ce poate fi descurajant pentru un începător.

Exista câteva biblioteci software bune, open-source, cu care se pot dezvolta aplicații: opencv[2], dlib[3], libccv[4]. Avantajele acestor biblioteci sunt:

[1] `http://youtu.be/PXSiUojhNFg`
[2] `http://opencv.org/`
[3] `http://dlib.net/`
[4] `http://libccv.org/`

- Exista algoritmi de recunoaștere a obiectelor gata implementați.
- Se poate trece direct la dezvoltarea de aplicatii

Totuși sunt și dezavantaje: /

- Componentele care stau la baza acestor implementări nu sunt expuse, reutilizarea lor find imposibila.
- Codul sursa este optimizat cu instituțiuni de asamblare sau cod pentru procesorul grafic, fiind dificil de înțeles.

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

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

Avantajele acestei platforme ar fi:

- Fiecare componenta a unui algoritm de recunoaștere este implementat într-o clasa separata
- Algoritmi pot fi implementați atât în C++ cat și în Python
- Reutilizare sporita
- Pot fi ușor adaptați algoritmi din alte biblioteci pentru utilizare în cadrul platformei

1.2 Enunțul problemei

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

Aceasta biblioteca va fi scrisa într-un mod hibrid, cu componente implementate atât în C++ cat și în Python.

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

Algoritmul va învăța sa recunoască obiecte folosindu-se de un set de imagini cu exemple pozitive adnotate și exemple negative, imagini care nu conțin obiectul pe care dorim sa-l învățam. Algoritmul poate fii 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 aplica pe o imagine data.

1.3 Structura Lucrării

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 se prezintă în detaliu structura unui algoritm de recunoaștere și o tehnica eficienta de antrenare a unui astfel de algoritm.

În capitolul 3 se prezintă în detaliu implementarea bibliotecii de dezvoltare a algoritmilor de recunoaștere a obiectelor.

În capitolul 4 se vor discuta tehnologiile folosite.

În capitolul 5 se prezintă concluzii despre lucrare, precum și posibilități de dezvoltare.

Capitolul 2

Recunoașterea obiectelor

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

Unele dintre cele mai relevante lucrări din domeniu sunt:

- "Robust Real-time Object Detection" [16]
- "Histograms of Oriented Gradients for Human Detection" [3]
- "Object Detection with Discriminatively Trained Part Based Models" [4]

Dacă studiem mai atent algoritmii descriși în aceste lucrări se observa ca toate au o structura comuna și urmăresc o succesiune de operațiuni similare. Aceste operațiuni sunt următoarele: parcurgerea imaginii spațial la diferite scalari, extragerea de trăsături, clasificare și post-procesarea rezultatelor.

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

2.1 Parcurgerea imaginii în scara și spațiu

Obiectele care trebuiesc recunoscute pot prezenta deviații de la modelul din baza de date. Aceste deviații pot fi de natura geometrica: translație, rotație, scalare și perspectiva.

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

O alta abordare ar fi sa se folosească o reprezentare a imaginii, invarianta la aceste transformări. Din literatura se știe ca o imagine reprezentata în spațiul Fourier este invarianta la translație și o imagine reprezentata în spațiul Log-Polar este invarianta la scalare și rotație.[15] Exista chiar și o combinație intre aceste doua reprezentări numita Fourier-Mellin care este invarianta la toate cele trei transformări. Totuși s-a observat ca utilizarea acestei reprezentări are aplicații limitate, ea find folosita mai mult la alinierea imaginilor.[15]

O alta soluție, poate un pic mai naiva, dar în același timp foarte puternica, este folosirea unei combinații intre o piramida de imagini și un algoritm de tip fereastră glisantă[1], acestea find aplicate pe imagine, nu pe modelul din baza de date.

Folosirea piramidei de imagini și fereastra glisanta ne permit ca în restul algoritmului de recunoaștere sa tratam problema ca 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 sursa, 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 dimensiune imaginii sursa și $L \in \mathbb{N}^+$ este nivelul piramidei pentru care dorim sa aflam dimensiune.

[1]Eng. sliding window

Se poate vizualiza piramida de imagini în figurile următoare:

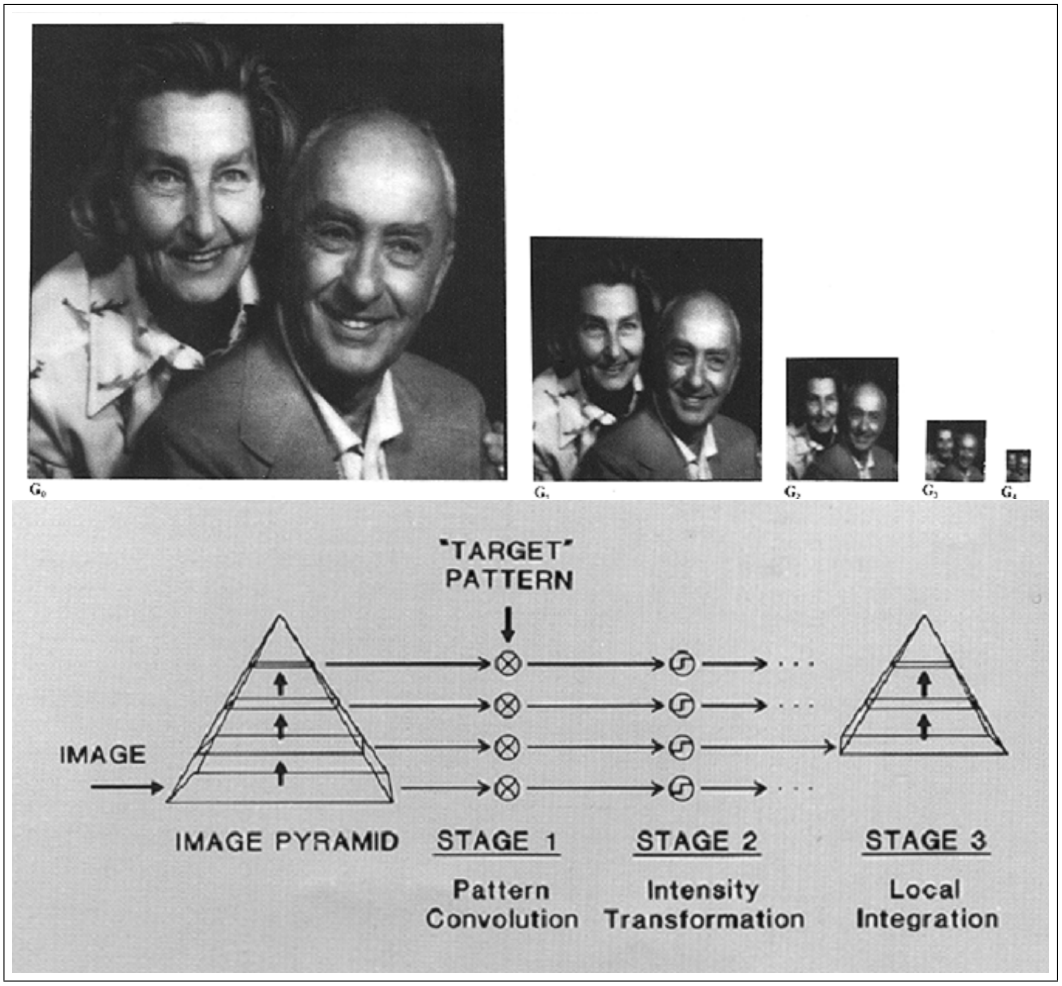

Figura 2.1: Piramida de imagini[2]

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)
```

[2]Pyramid methods in image processing.[2]

```
  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 ca, totuși, acest model nu poate reprezenta toate scările posibile, find un model discret. Aceasta problema poate fi ameliorata prin alegerea unui α potrivi și permițând modelului din baza de date sa reprezinte și el mici variații de scara.

O alta observație ar fi: cu cat α este mai mic, cu atât șansele sa nimerim scara corecta cresc, dar în același timp creste 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 explicita a unei liste de imagini în memorie.

Algoritmul fereastra glisanta se folosește pentru a obține invarianta la translație a modelului. Aici fereastra se refera la o secțiune rectangulara 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:

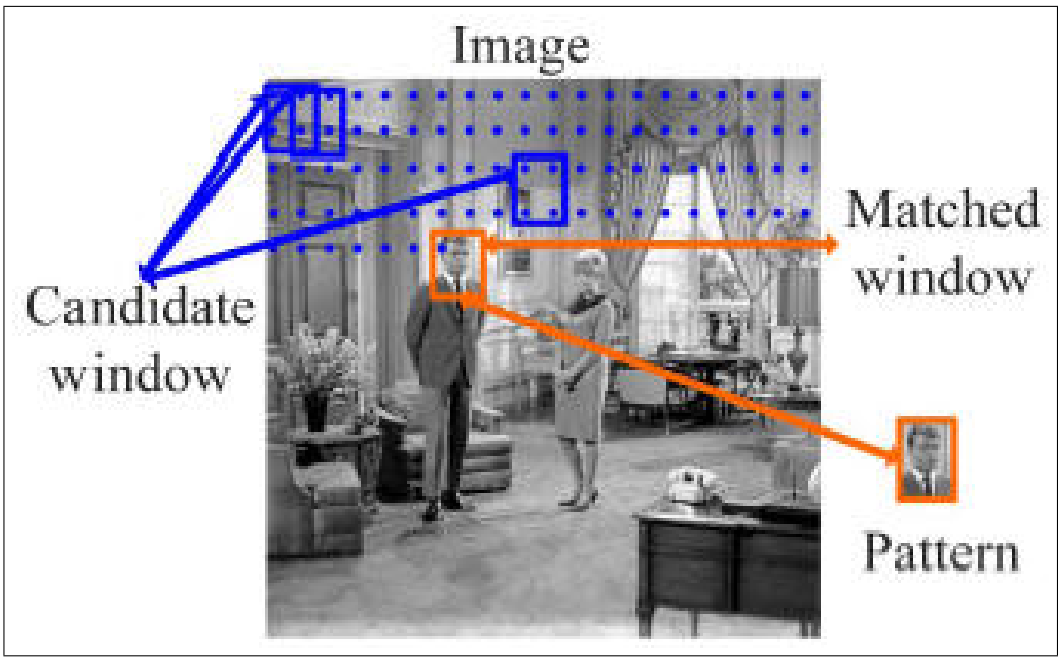

Figura 2.2: Fereastre Glisanta [3]

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

Complexitatea algoritmului piramida combinat cu fereastra glisanta este

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

[3]Imagine din: Segmented Gray-Code Kernels for Fast Pattern Matching[10]

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.

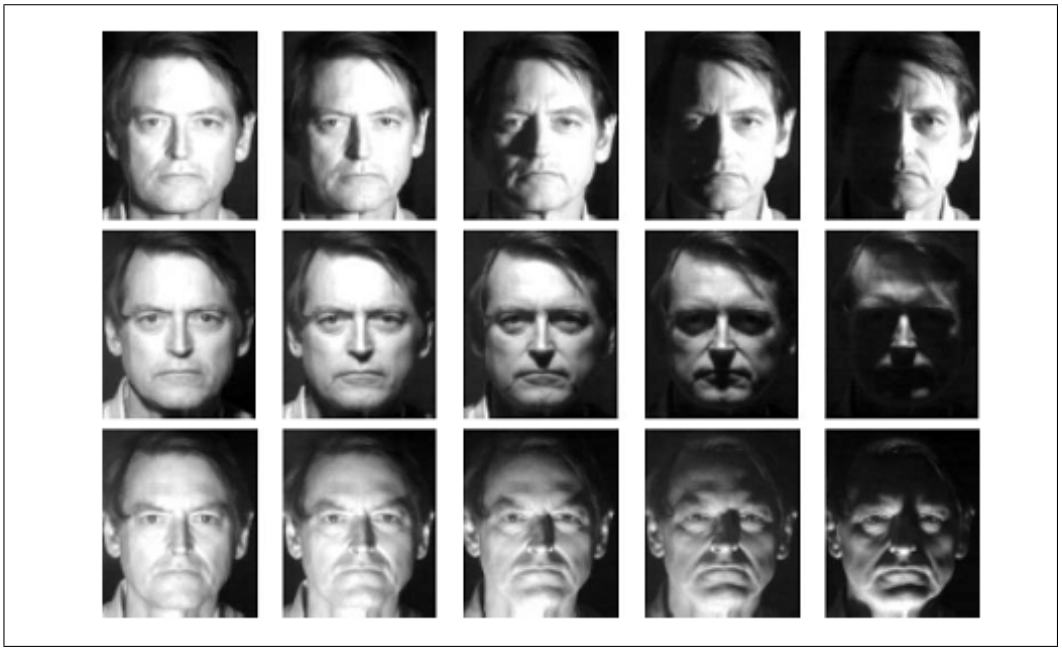

Figura 2.3: Efectul iluminarii[4]

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

[4]Sursa imagine: Computer vision: algorithms and applications[14]

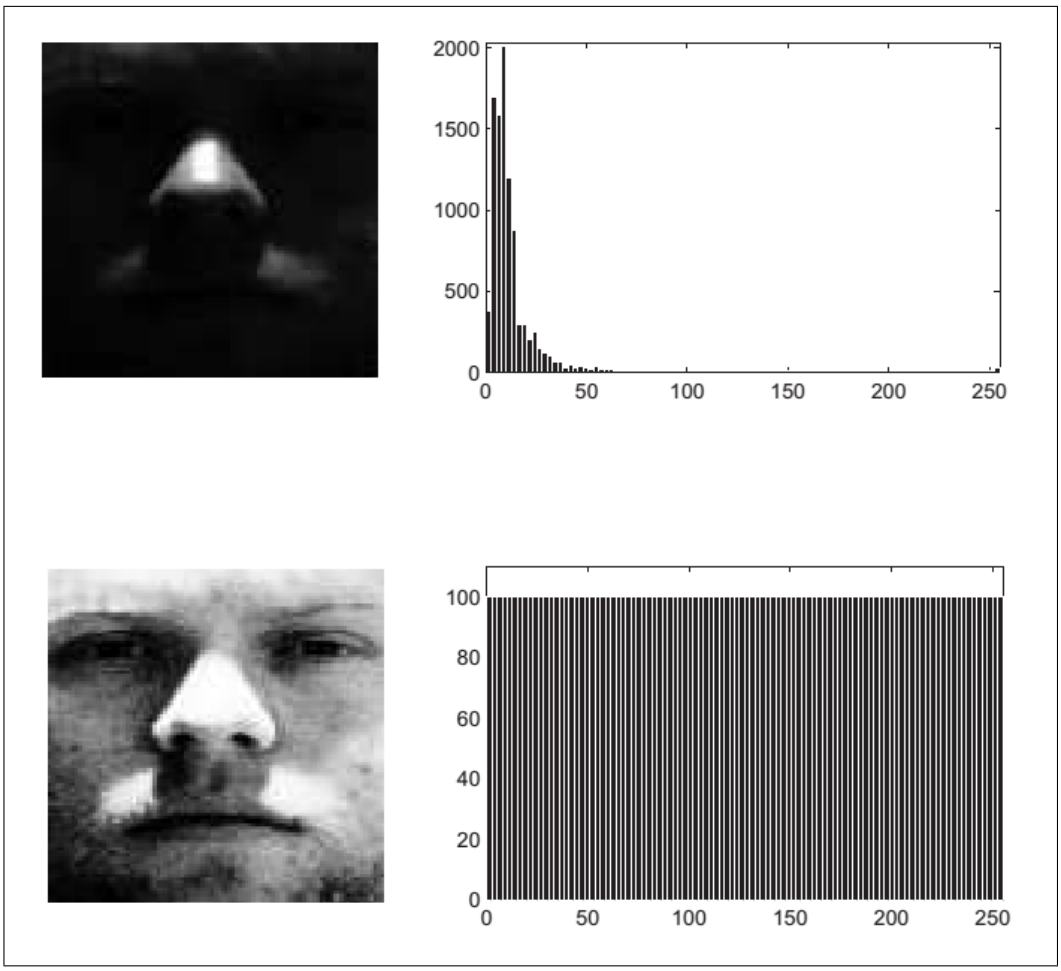

Figura 2.4: Egalizarea Histogramei[5]

Efectul informațiilor irelevante și redundante poate fi ameliorat folosind tehnici de reducere a dimensionalitatii, cum ar fi analiza componentelor principale[6](fig. 2.5 sau transformata cosinus discreta(fig. 2.6).

[5]Sursa imagine: Histogram remapping as a preprocessing step for robust face recognition[12]

[6]Eng. PCA, principal component analisys

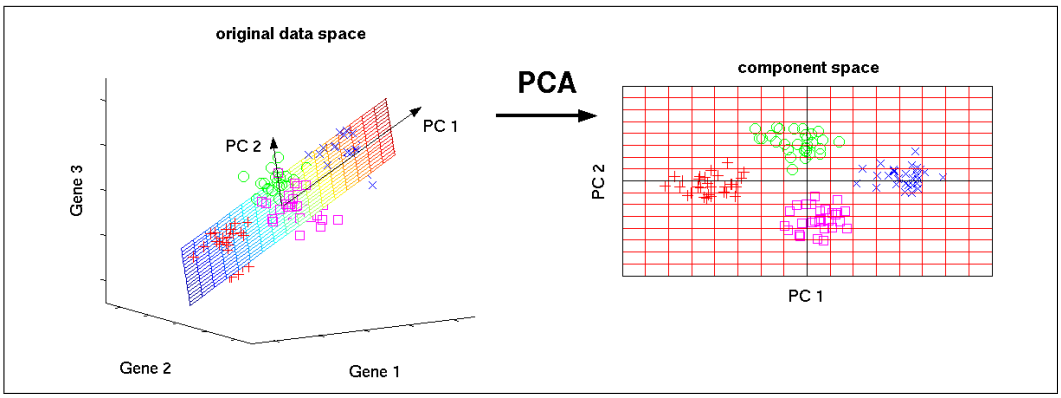

Figura 2.5: Analiza componentelor principale[7]

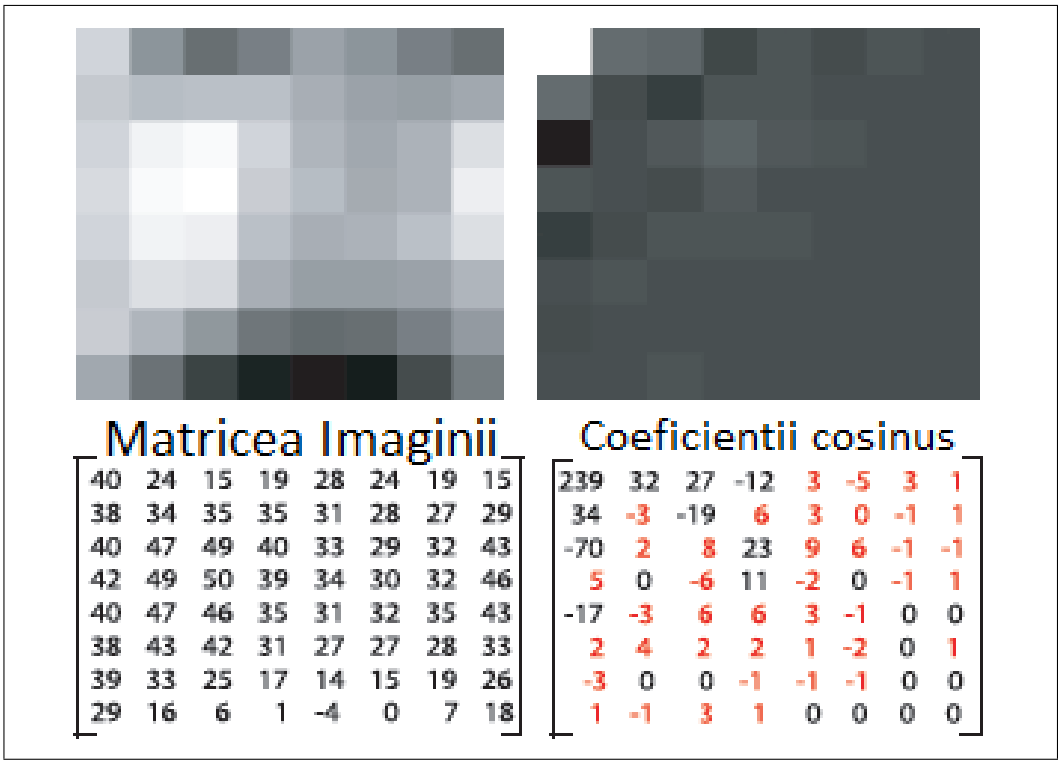

Figura 2.6: Transformata cosinus[8]

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

[7]Sursa imagine: `http://phdthesis-bioinformatics-maxplanckinstitute-molecularplantphys.matthias-scholz.de`

[8]Sursa imagine: `http://cnx.org/content/m13173/1.6/`

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

Aici intervine ceea ce se numește ingineria trăsăturilor[9] care, folosind cunoștințe din fizica, biologie sau chiar neurologie , construiește reprezentări mult mai favorabile recunoașterii. Câteva dinte cele mai cunoscute trăsături sunt HAAR[16], SIFT[8] și HOG[3].

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

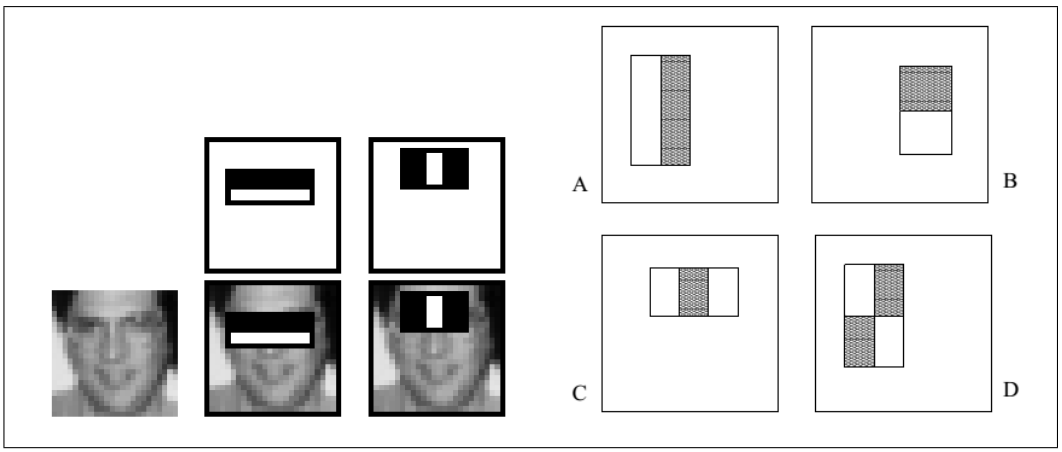

Figura 2.7: Trasaturi HAAR[10]

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 reprezinta trăsătura HOG.

[9]Eng. Feature Engineering

[10]Sursa imagine: Robust Real-time Object Detection[16]

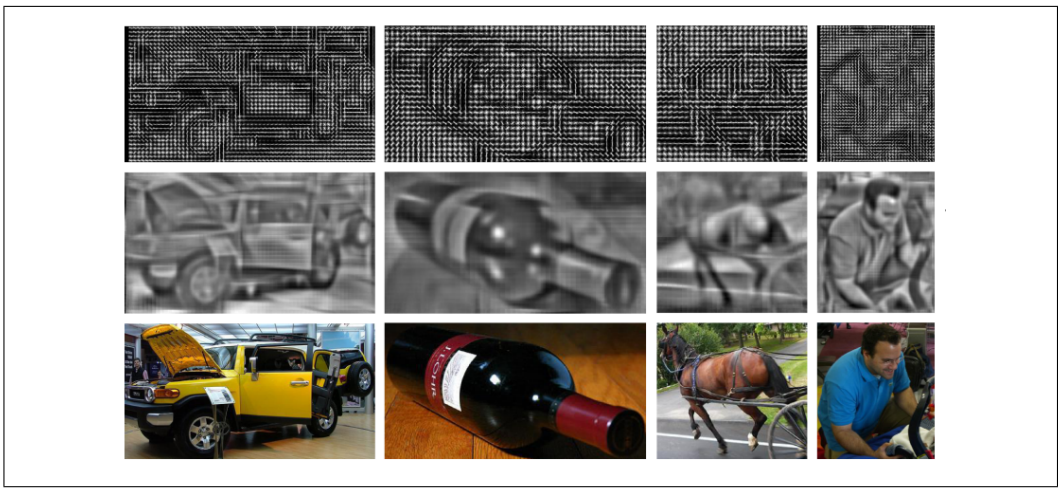

Figura 2.8: Trasaturi HOG[11]

Descriptorul SIFT este similar cu HAAR, acesta find în plus și invariant la rotație.

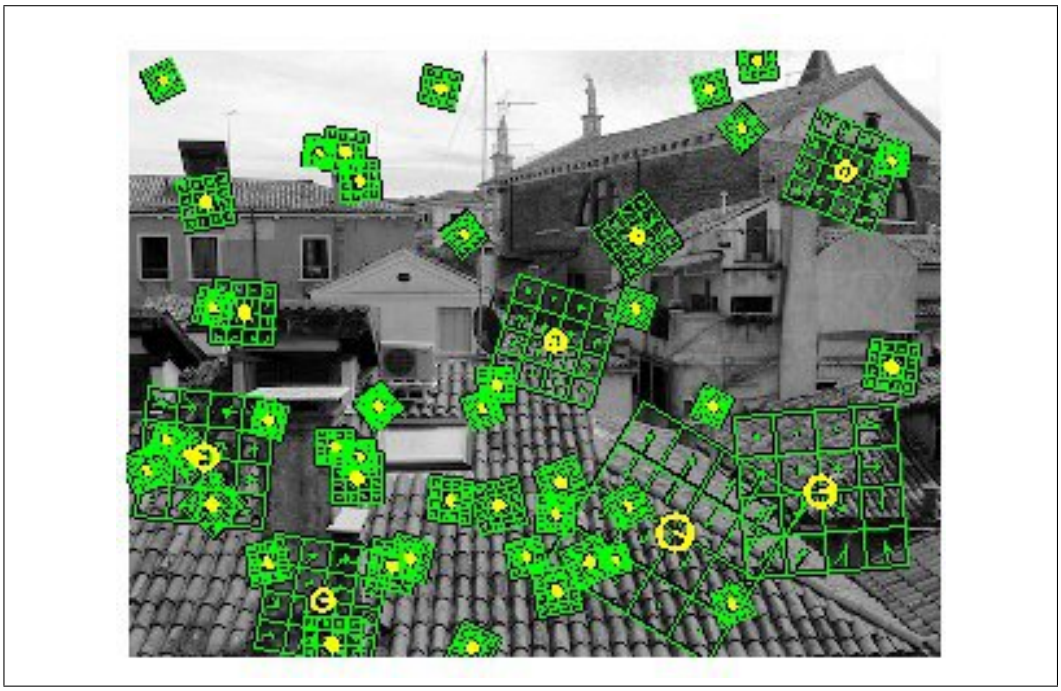

Figura 2.9: Descriptorul SIFT[12]

[11]Sursa imagine: HOGgles: Visualizing Object Detection Features[17]
[12]Sursa imagine: `http://www.vlfeat.org/overview/sift.html`

Recent a apărut o noua 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 sa extragă trăsături mai puternice, un exemplu fiind rețeaua neuronala convolutionala.[7]

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 sa îl recunoaștem. Acest tip de clasificare se numește clasificare binara, findcă răspunsul nu poate lua decât doua valori.

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

În cazul nostru aceasta funcție este rezultatul unui algoritm de învățare automata.[13] Învățarea automata, o ramura a inteligentei artificiale, este preocupata cu studiul și construcția sistemelor care pot învață din date. Algoritmii de învățare automata sunt împărțiți în multe categorii, însa noi ne vom axa doar pe cei de învățare supervizata. Se numesc algoritmi de învățare supervizata acei algoritmi care folosesc la antrenament seturi de perechi de date (x, y) unde x reprezinta trăsăturile sau atributele unui exemplar, iar y reprezinta răspunsul dorit. După ce a avut loc învățarea, algoritmul va fi capabil sa 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 un plan care sa separe cele doua clase intre ele. (fig: 2.10)

Cel mai des întâlniți algoritmi de învățare în viziunea artificiala sunt: automatul cu vectori de suport[13][14] si rețeaua neuronala.

[13] Machine Learning
[14] Eng. Support Vector Machines

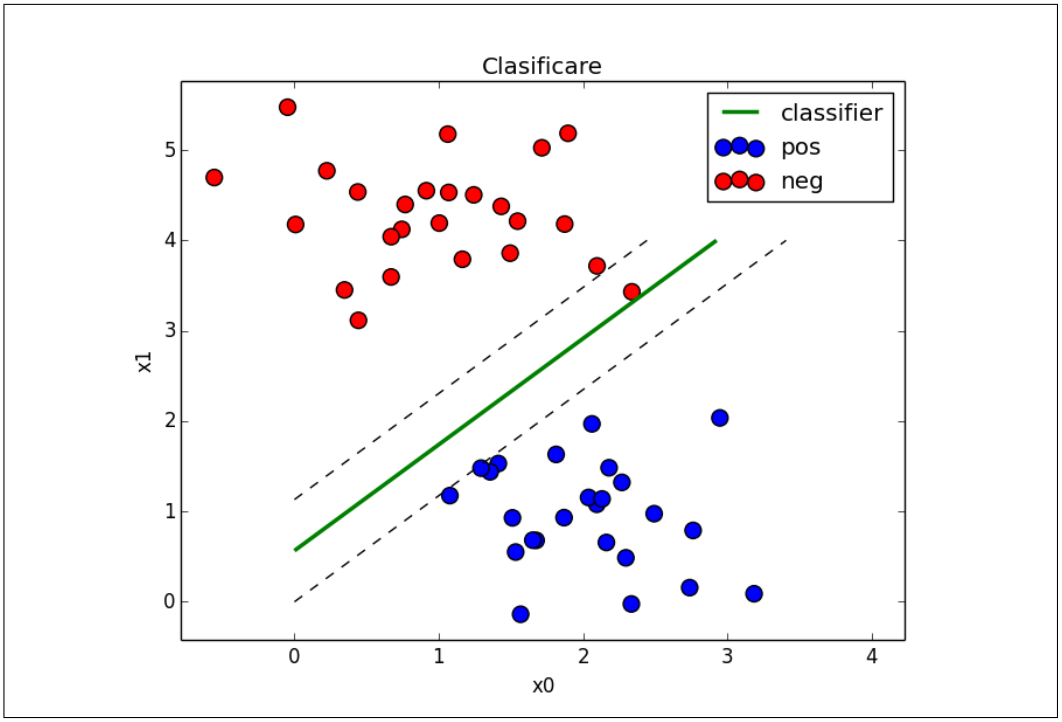

Figura 2.10: Clasificare

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 ca modelul învățat recunoaște și obiecte cu mici translații și scalari. Totuși, se dorește ca fiecare obiect prezent în imagine sa fie detectat doar o singura data. Acest lucru se realizează cu ajutorul unui algoritm de grupare a detecțiilor suprapuse.

2.5 Algoritmul de recunoaștere

Folosindu-ne de componentele 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 rasuns 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 doua seturi de imagini. Un set va conține imagini decupate cu obiectul pe care dorim sa 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 comuna prin redimensionare. Din setul de imagini negative se vor extrage exemplare folosind scanarea în scara și spațiu de la algoritmul de recunoaștere. Pentru ca setul de negative este de obicei foarte mare, nu este practic ca la antrenare sa se folosească toate exemplarele posibile. Exemplarele negative se vor extrage printr-un proces iterativ. În prima faza se extrage un număr ales de exemple negative și se antrenează clasificatorul. Apoi, folosind clasificatorul antrenat la pasul anterior, se scanează 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

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 rasuns este 'afirmativ' atunci
          X = adauga(X, xi)
          y = adauga(y, 'negativ')
        sfarsit daca
      sfarsit
    sfarsit
  sfarsit

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

Capitolul 3

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.

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`.

3.1 Diagrama de clase

3.1.1 Core

Pachetul "core" conține interfețe de baza ale bibliotecii.

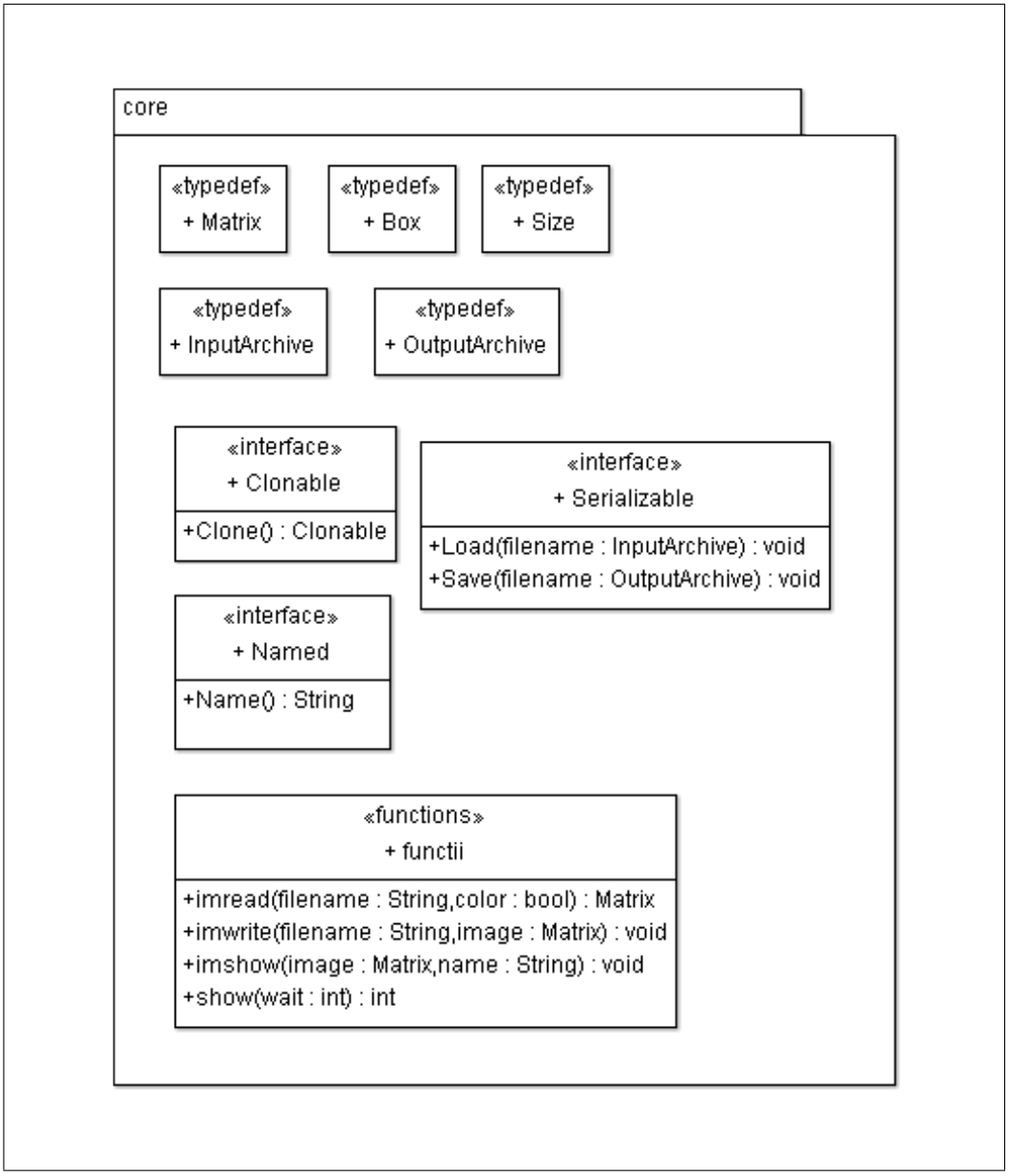

Figura 3.1: core Class Diagram

3.1.2 Dataset

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

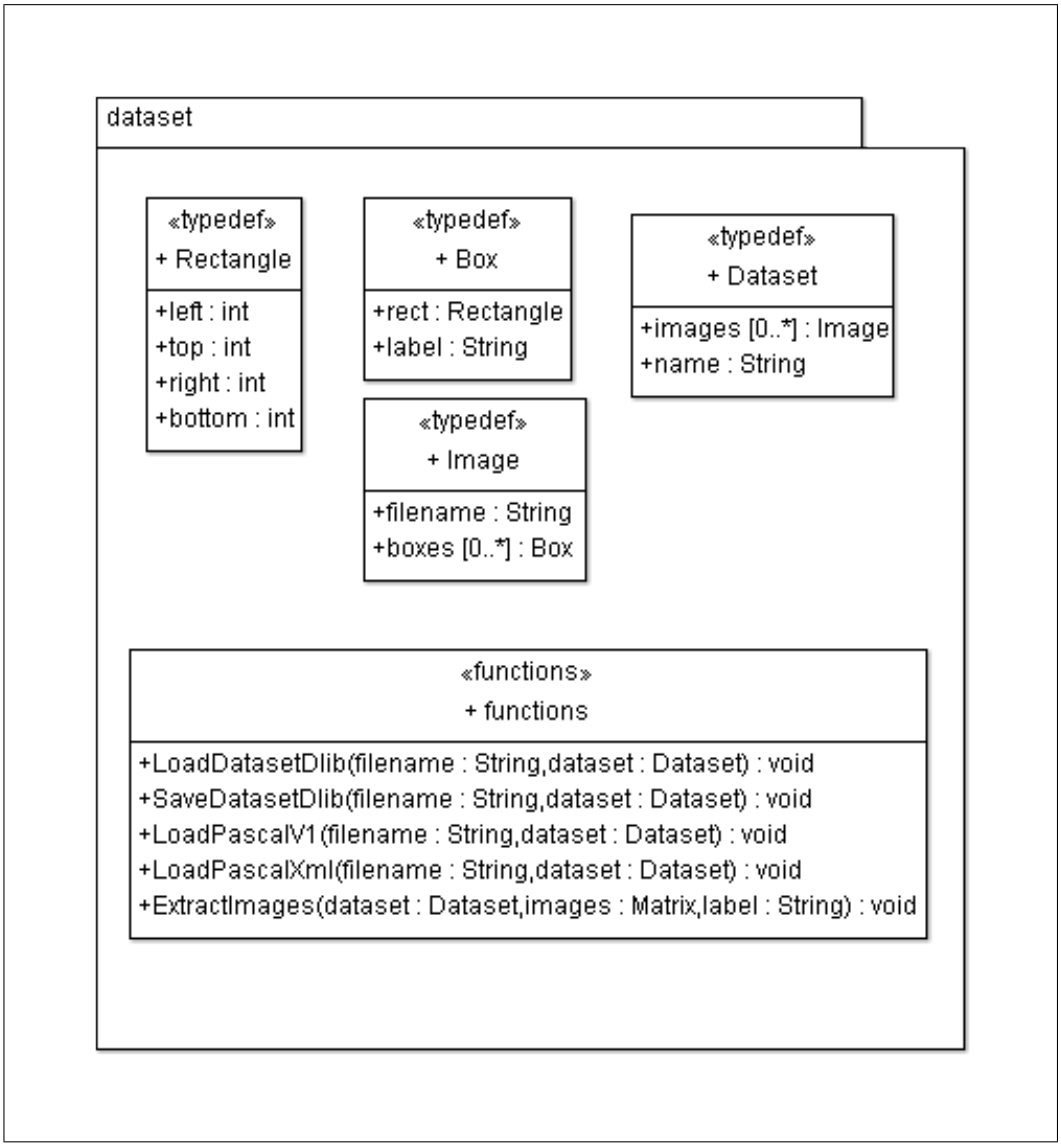

Figura 3.2: Diagrama de clase: dataset

3.1.3 Image Pyramid

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

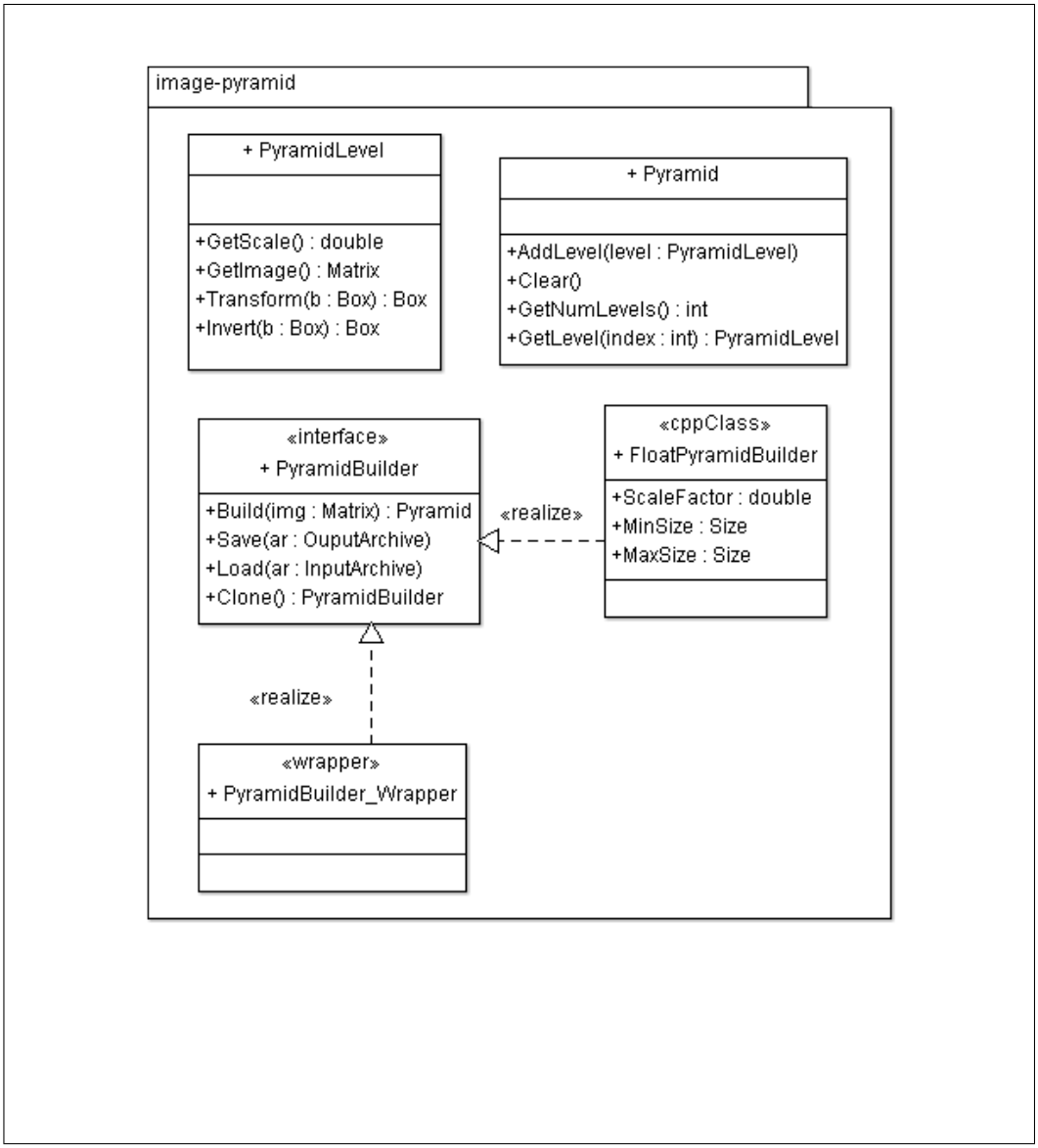

Figura 3.3: Diagrama de clase: image-pyramid

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

Metodele Transform si Invert din clasa PyramidLevel transforma coordonate din spațiul imaginii sursa în cel al nivelului, respectiv invers.

3.1.4 Image Scanning

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

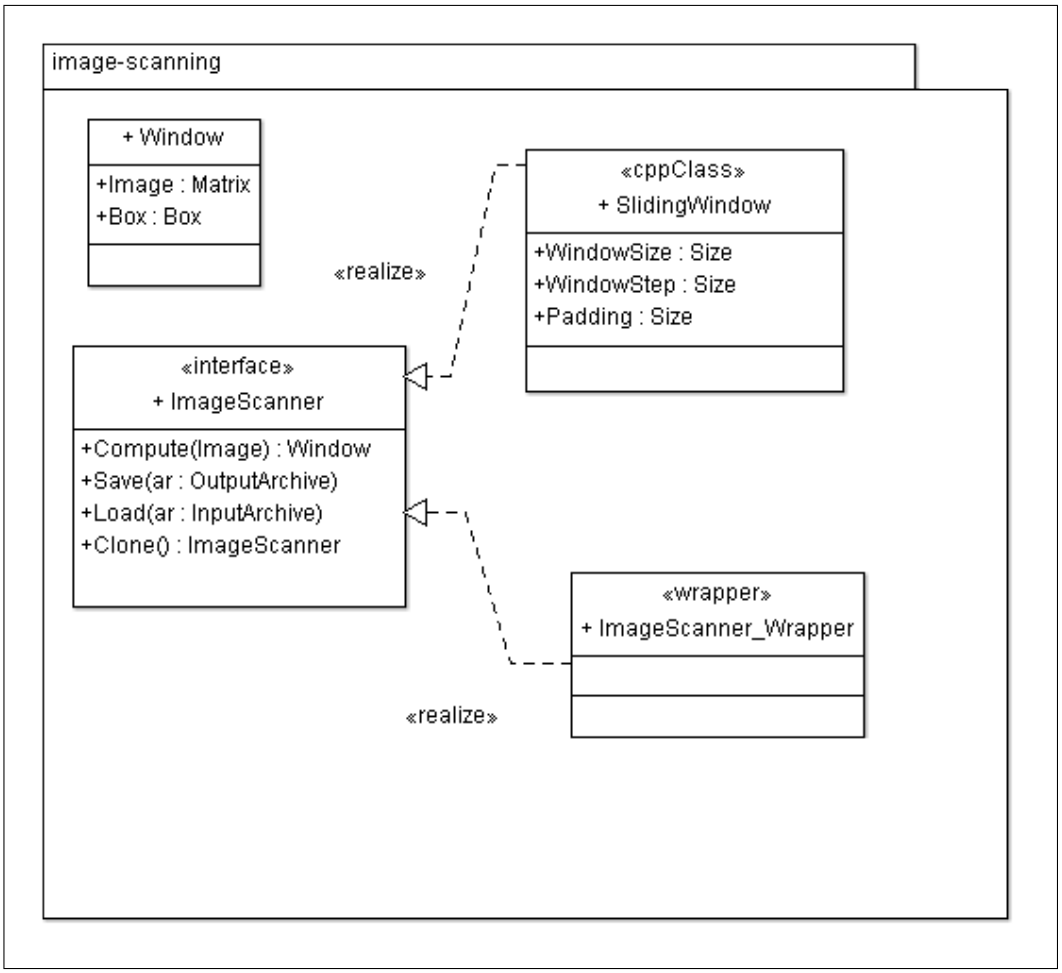

Figura 3.4: Diagrama de clase: image-scanning

3.1.5 Feature Extraction

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

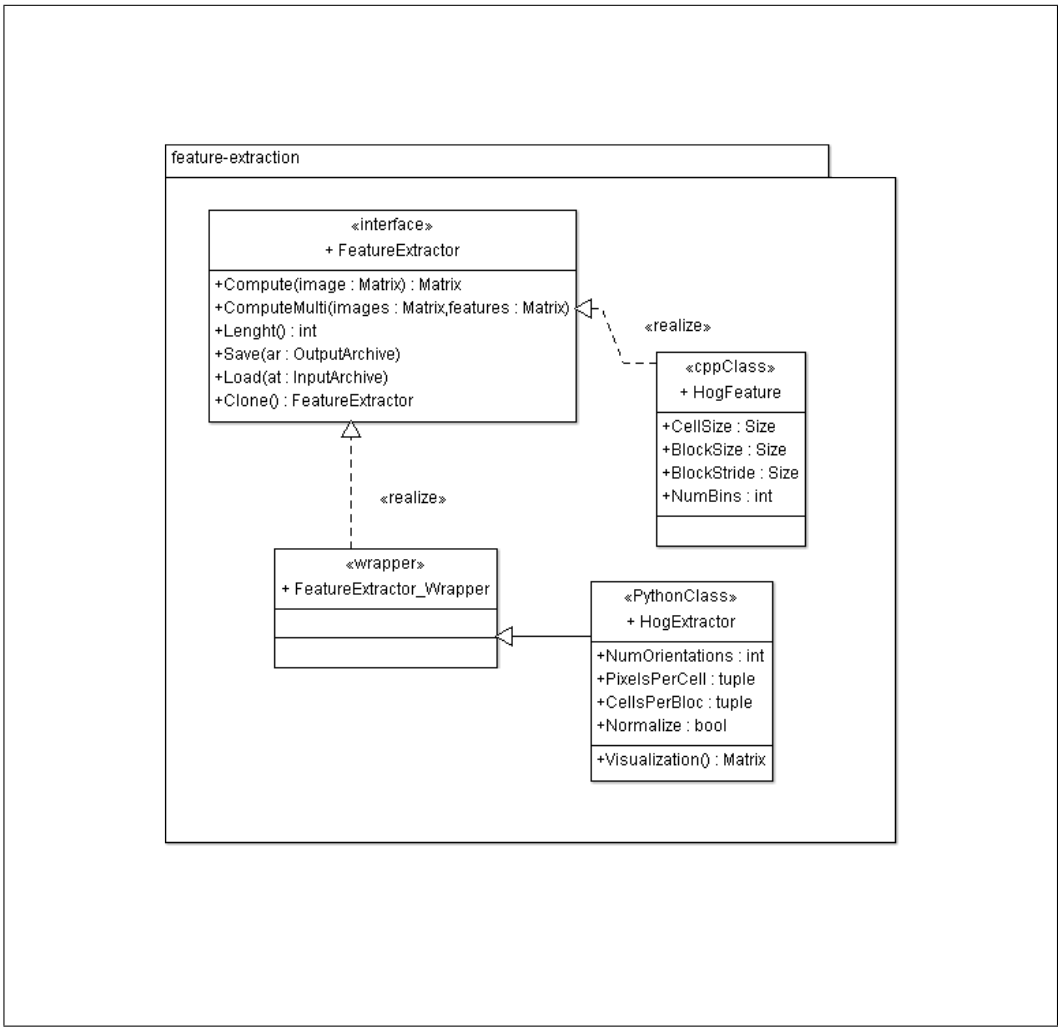

Figura 3.5: Diagrama de clase: feature-extraction

3.1.6 Classification

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

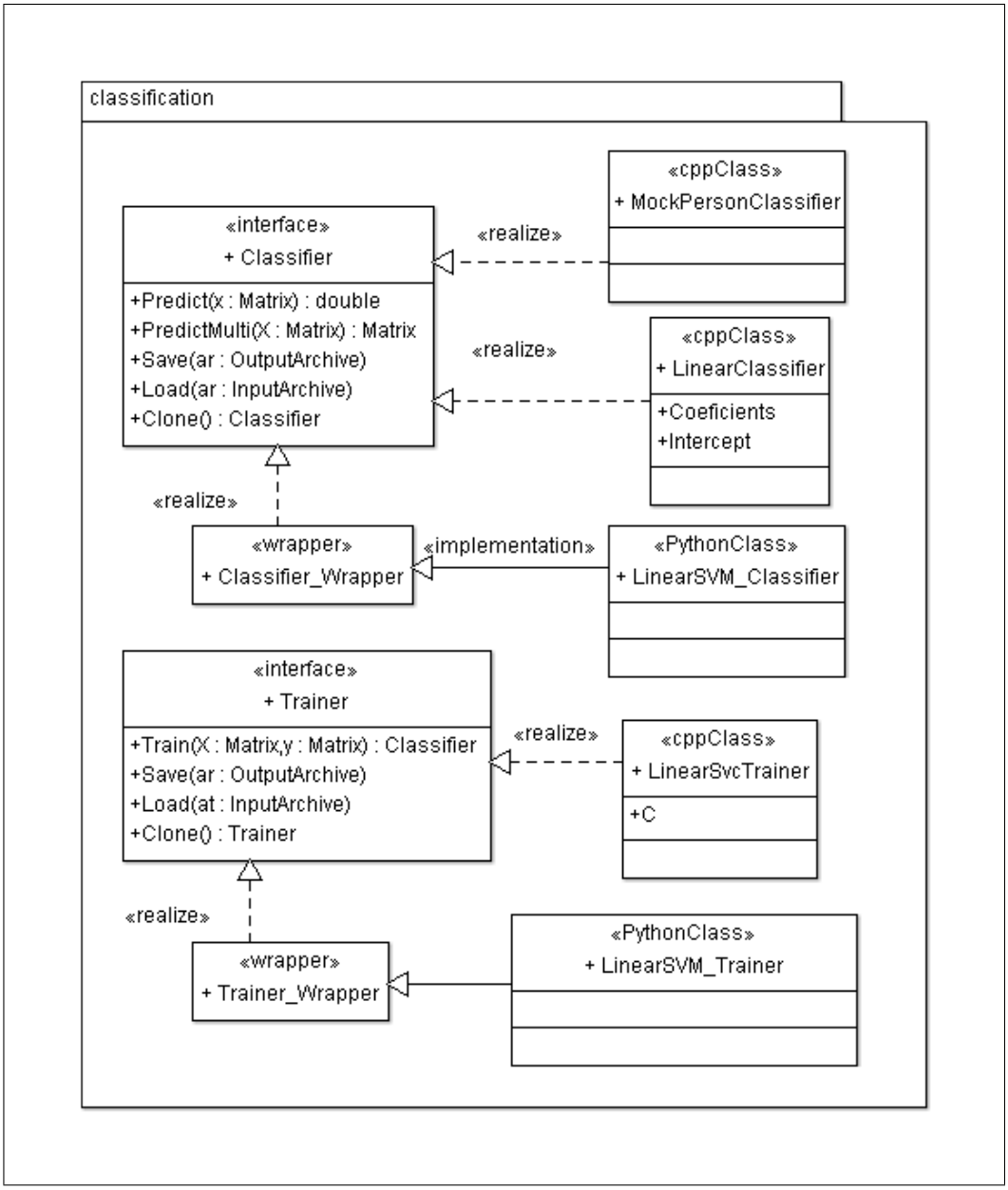

Figura 3.6: Diagrama de clase: classification

3.1.7 Non Maxima Suppression

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

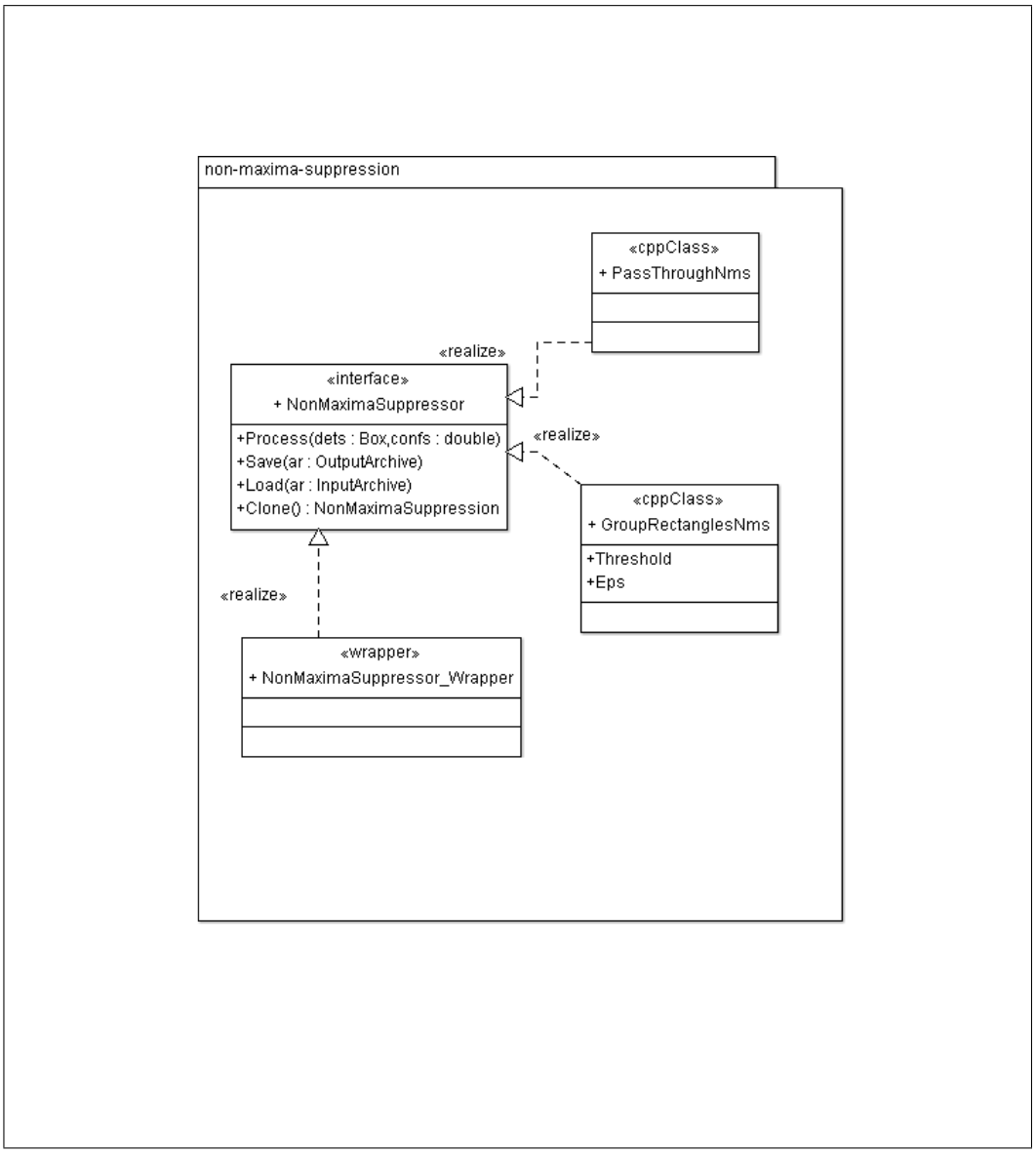

Figura 3.7: Diagrama de clase: non-maxima-suppression

3.1.8 Detection

Pachetul "detection" conține interfețe și implementări care servesc la recunoasterea obiectelor in imagini si la antrenarea algoritmilor.

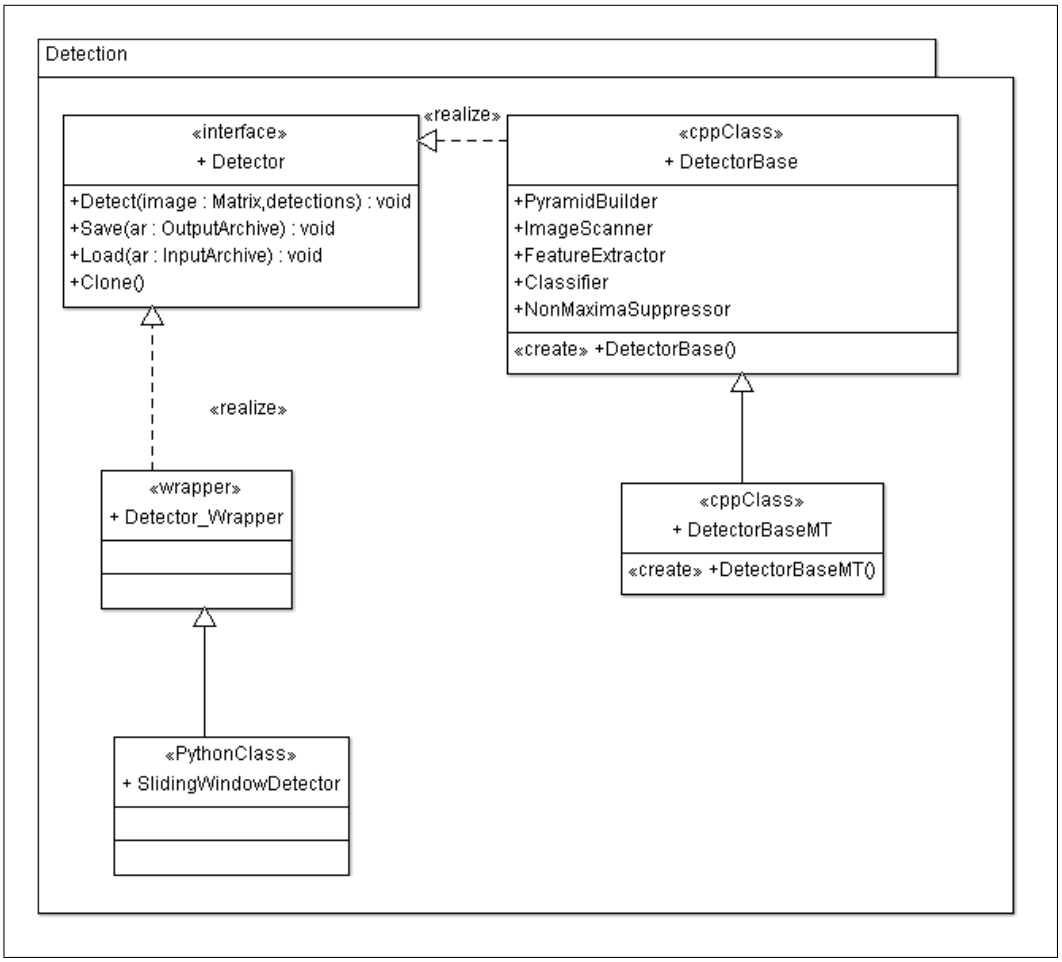

Figura 3.8: Diagrame de clase: detection

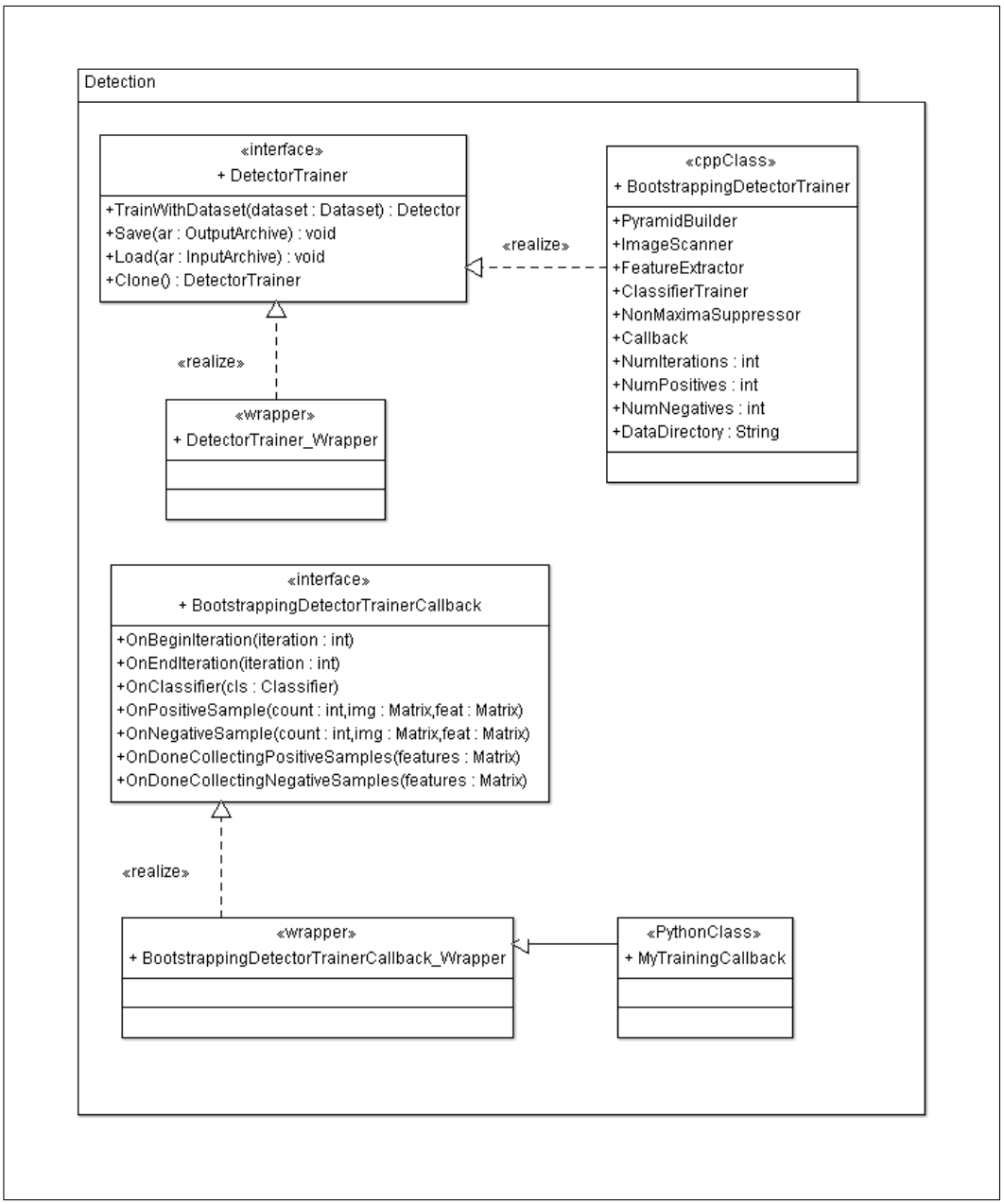

Figura 3.9: Diagrama de clase: detection

3.1.9 Python

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

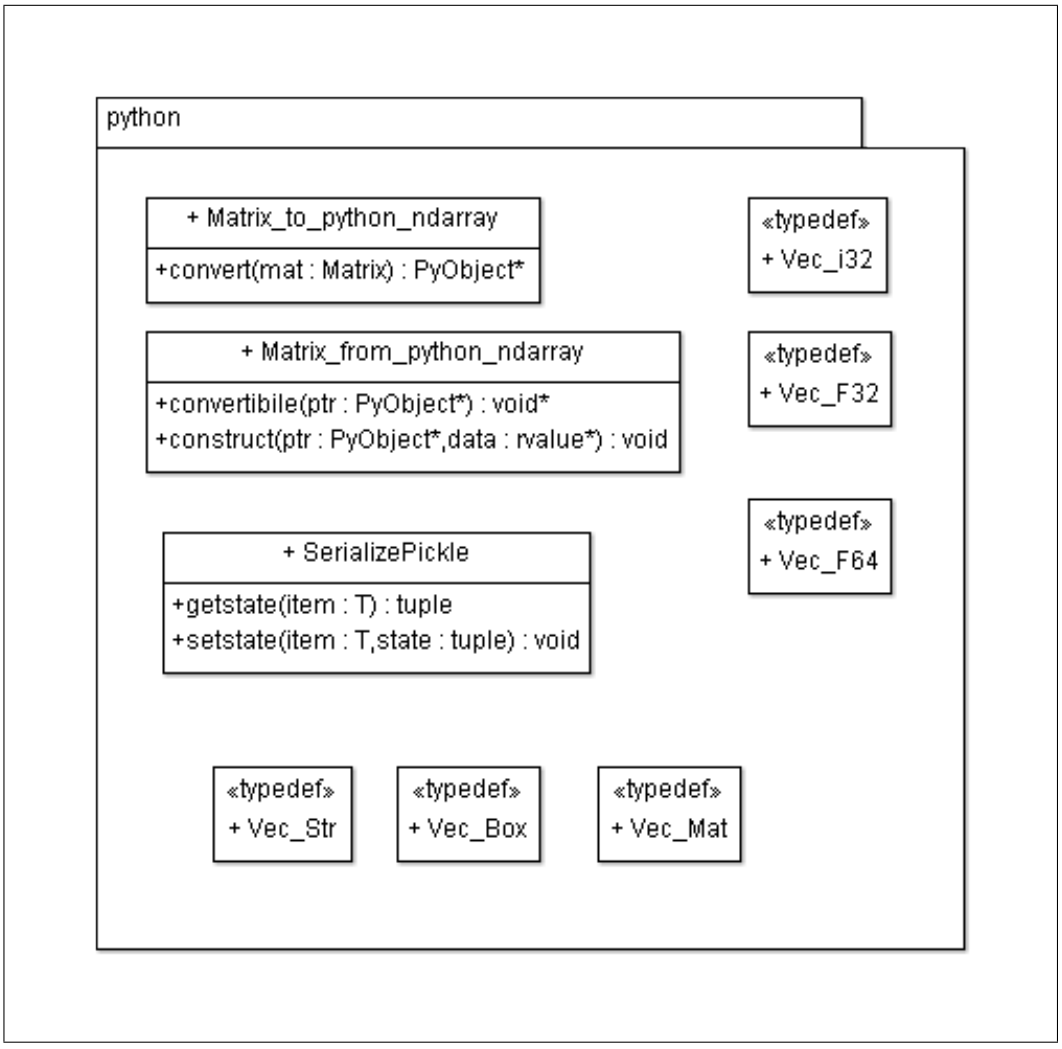

Figura 3.10: Diagrama de clase python

3.2 Interoperabilitatea cu Python

3.3 Serializarea

Capitolul 4

Tehnologii folosite

4.1 Limbajul C++

[Am rezervat o pagina]

4.2 Limbajul Python

[Am rezervat o pagina]

4.3 Biblioteca Boost

[Am rezervat o pagina]

4.4 Biblioteca scikit-learn

[Am rezervat o pagina]

4.5 Biblioteca numpy

[Am rezervat o pagina]

4.6 Biblioteca matplotlib

[Am rezervat o pagina]

4.7 Biblioteca PySide

[Am rezervat o pagina]

Capitolul 5

Concluzii

Bibliografie

[1]

[2] 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.

[3] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *In CVPR*, pages 886–893, 2005.

[4] Pedro F. Felzenszwalb, Ross B. Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part based models.

[5] Smith Graham. Smashing idea: Volvo installs pedestrina detection systems that brakes car automatically, February 2011.

[6] 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.

[7] 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.

[8] David G. Lowe. Object recognition from local scale-invariant features, 1999.

[9] Tom Michael Mitchell. *Machine Learning*. McGraw-Hill Education, 1997.

[10] 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.

[11] Phil Simon. *Too Big to Ignore: The Business Case for Big Data*, volume Volume 72 of Wiley and SAS Business Series. Wiley, 2013.

[12] Vitomir Štruc, Janez Žibert, and Nikola Pavešić. Histogram remapping as a preprocessing step for robust face recognition. *image*, 7:9, 2009.

[13] Johan AK Suykens and Joos Vandewalle. Least squares support vector machine classifiers. *Neural processing letters*, 9(3):293–300, 1999.

[14] Richard Szeliski. *Computer vision: algorithms and applications*. Springer, 2010.

[15] Marco Alexander Treiber. *An Introduction to Object Recognition*. Springer, 2010.

[16] Paul Viola and Michael Jones. Robust real-time object detection. In *International Journal of Computer Vision*, 2001.

[17] C. Vondrick, A. Khosla, T. Malisiewicz, and A. Torralba. HOGgles: Visualizing Object Detection Features. *ICCV*, 2013.

[18] Michael Zhang. Canon face recognition feature gives friends preferential treatment, January 2012.