

Universitatea Transilvania din Braşov  
Facultatea de Matematică şi Informatică  
Specializarea Informatică

Proiect de disertaţie

# **Tehnici de Machine Learning în procesarea şi recunoaşterea imaginilor**

*Autor:*

Draghia Alin-Madalin

*Profesor coordonator:*

Lect. dr. Sasu Lucian Mircea

Braşov

Iulie 2014

# Cuprins

<b>1</b>	<b>Introducere</b>	<b>1</b>
1.1	Motivație . . . . .	1
1.2	Enunțul problemei . . . . .	4
1.3	Tehnologii folosite . . . . .	5
1.4	Structura Lucrării . . . . .	10
<b>2</b>	<b>Recunoașterea obiectelor</b>	<b>11</b>
2.1	Parcurgerea imaginii în scara și spațiu . . . . .	12
2.2	Extragerea de trăsături . . . . .	16
2.3	Clasificare . . . . .	21
2.4	Post-procesarea rezultatelor . . . . .	22
2.5	Algoritmul de recunoaștere . . . . .	23
<b>3</b>	<b>Implementare Librăriei</b>	<b>25</b>
3.1	Diagrama de clase . . . . .	25
3.2	Interoperabilitatea cu Python . . . . .	31
3.3	Serializarea . . . . .	32
<b>4</b>	<b>Concluzii</b>	<b>33</b>



# Capitolul 1

## Introducere

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

### 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 dezvoltati 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 inca 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.

Cateva dintre cele mai de succes aplicatii sunt:

- Sistemul de frânare automata la detecția pietonilor de pe mașinile Volvo.

## 1.1. MOTIVAȚIE

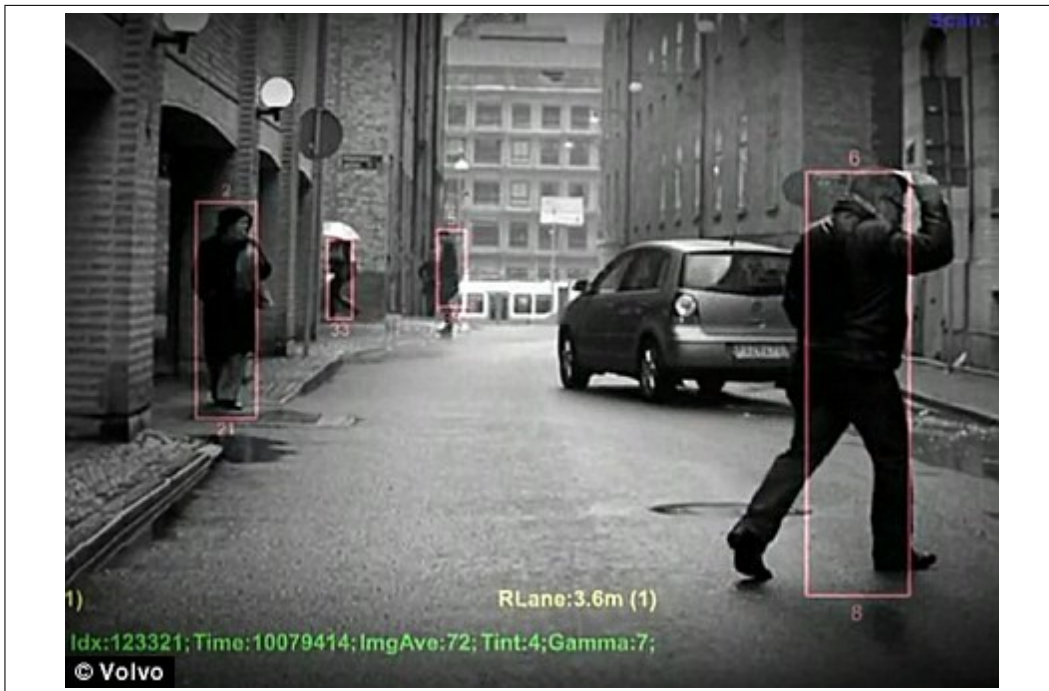


Figura 1.1: Volvo: sistemul de detectie a pietonilor

- Focalizarea automata a camerelor foto pe fețe

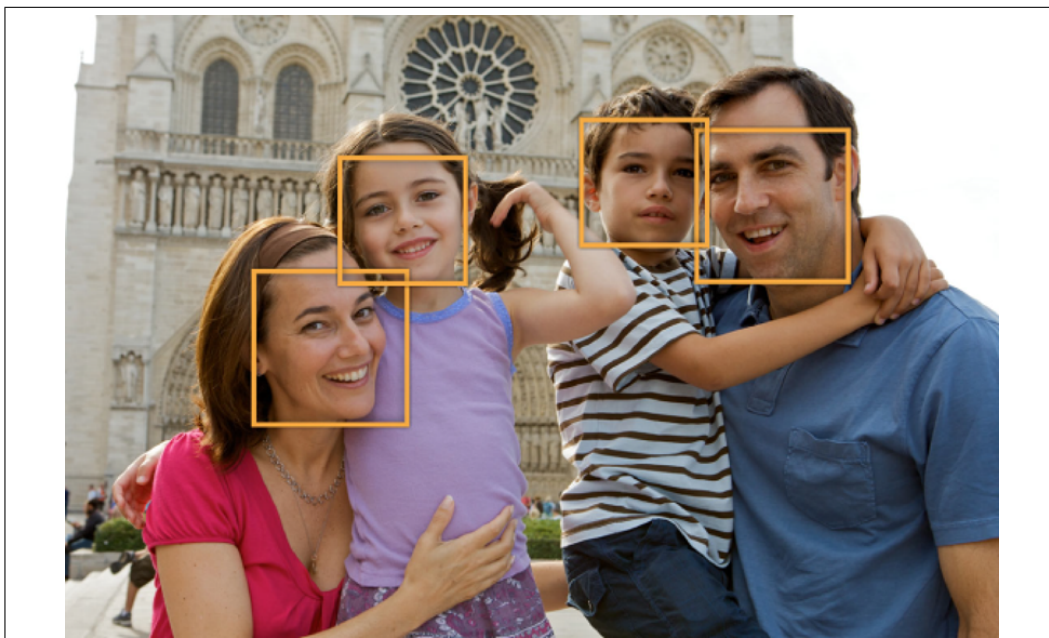


Figura 1.2: Camera foto: focus automat

## 1.1. MOTIVAȚIE

---

- Analizarea traficului rutier

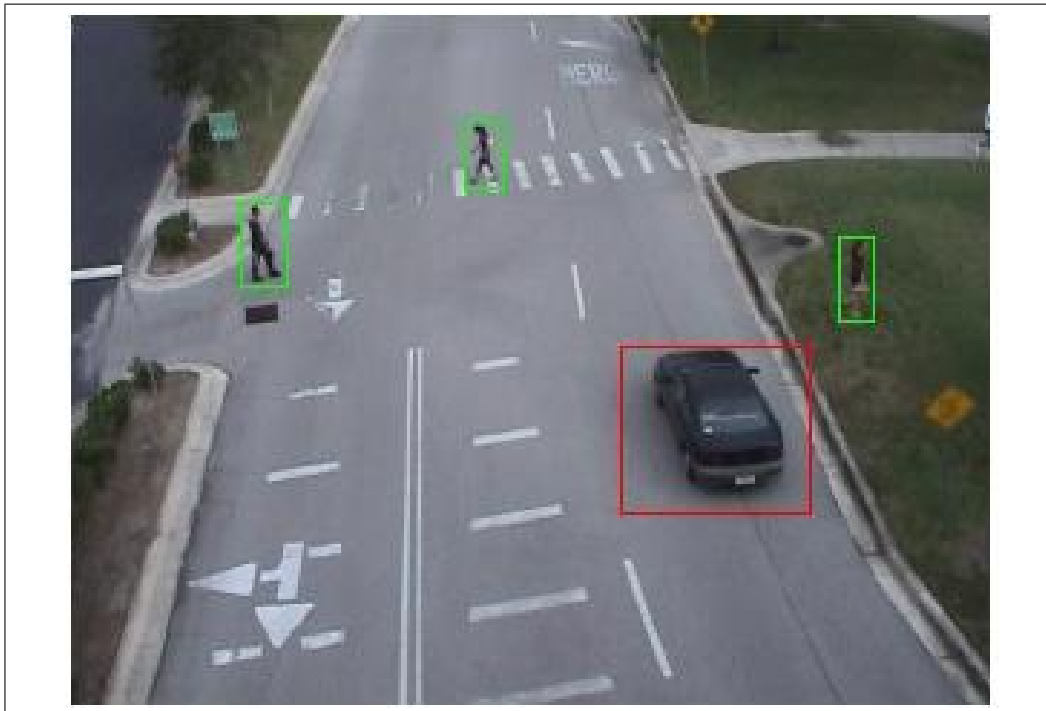


Figura 1.3: Analizarea traficului rutier

Înțelegerea și dezvoltarea unui sistem de recunoaștere automată a obiectelor poate fi foarte dificilă, mai ales pentru cei care sunt la început de drum în studiarea 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. Parcurgerea 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 intră în acest domeniu să fie unul foarte înalt, ceea ce poate fi descurajant pentru un începător.

Există câteva **librării** software bune, open-source, cu care se pot dezvolta **aplicații**. Acestea conțin algoritmi de recunoaștere a obiectelor gata implementați. Totuși componentele care stau la baza acestor implementări nu sunt expuse, reutilizarea lor fiind imposibilă. Nici codul sursă nu este ușor de **urmări**, acești algoritmi sunt optimizați cu instrucțiuni de asamblare, cod pe mai multe fire de execuție și chiar cod care se execută pe procesorul grafic. Din aceasta

## 1.2. ENUNȚUL PROBLEMEI

---

cauza, consider ca nu sunt 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.

## 1.2 Enunțul problemei

Se scrie o **librărie** 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 învățăm.**

**Se scrie** o aplicatie 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 Tehnologii folosite

### Limbajul C++

[Am rezervat o pagina]



#### **Limbajul Python**

[Am rezervat o pagina]

### 1.3. TEHNOLOGII FOLOSITE

---

#### **Librăria** Boost

[Am rezervat o pagina]

### 1.3. TEHNOLOGII FOLOSITE

---

#### **Librăria scikit-learn**

[Am rezervat o pagina]

### 1.3. TEHNOLOGII FOLOSITE

---

#### **Librăria Qt**

[Am rezervat o pagina]

### 1.4 Structura Lucrării

Capitolele care urmează vor trata algoritmul de recunoaștere a obiectelor din punct de vedere teoretic și se va prezenta implementarea unui astfel de algoritm.

Capitolul 1 prezintă enunțul problemei și motivația acesteia

Capitolul 2 se prezintă problema generală a unui algoritm de recunoaștere. Vor fi prezentate în detaliu componentele din structura algoritmului. Se va prezenta o tehnică eficientă de antrenare a acestor algoritmi.

Capitolul 3 se prezintă în detaliu implementarea unei **biblioteci** de dezvoltare a algoritmilor de recunoaștere a obiectelor.

Capitolul 4 prezintă un manual de utilizare pentru aplicația de recunoaștere a **obiectelor**.

Capitolul 5 prezintă concluzii despre lucrare, precum și posibilități de dezvoltare.

## Capitolul 2

# Recunoașterea obiectelor

Problema recunoașterii de obiecte se poate exprima în felul următor: Având un 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" [VJ01]
- "Histograms of Oriented Gradients for Human Detection" [DT05]
- "Object Detection with Discriminatively Trained Part Based Models" [FGMR]

Dacă studiem mai atent algoritmi 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 în scara și spațiu, 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 trebuie recunoscute pot prezenta deviații de la modelul din baza de date, aceste deviații pot fi de natura geometrică: translație, rotație, scalare și perspectivă.

O soluție pentru această problemă ar fi să se construiască un model care să prezinte toate instanțierile obiectului. O dificultate cu această abordare 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 altă abordare ar fi să se folosească o reprezentare a imaginii invariantă la aceste transformări. Din literatură se știe că o imagine reprezentată în spațiul Fourier este invariantă la translație și o imagine reprezentată în spațiul Log-Polar este invariantă la scalare și **rotație**. Există chiar și o combinație între aceste două reprezentări numită Fourier-Mellin care este invariantă la toate cele trei transformări. **Totuși s-a observat** că utilizarea acestei reprezentări are aplicații limitate, ea fiind folosită mai mult la alinierea imaginilor.

O altă soluție, poate un pic mai naivă, dar în același timp foarte puternică este folosirea unei combinații de piramidă de imagini și un algoritm de tip fereastră glisantă<sup>1</sup>, acestea fiind aplicate pe imagine, nu pe modelul din baza de date.

Folosirea piramidei de imagini și fereastra glisantă ne permite ca în rezultatul algoritmului de recunoaștere să tratăm problema ca și cum nu ar exista translații sau scalări, astfel simplificând mult algoritmi aplicați.

O piramidă de imagini este o reprezentare multi-scală. Piramida de imagini se formează, pornind de la o imagine sursă, prin scalări succesive. Aceste scalări se fac cu un factor și se opresc atunci când se ajunge la o dimensiune minimă. Dacă factorul de scalare este  $\alpha > 1$  atunci avem funcția care calculează dimensiunea unui nivel este

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

, unde  $D$  este dimensiunea imaginii sursă și  $L$  este nivelul piramidei pentru care dorim să aflăm dimensiunea.

Se poate vizualiza piramida de imagini în figurile următoare:

---

<sup>1</sup>eng. sliding window

## 2.1. PARCURGerea IMAGINII ÎN SCARA ȘI SPAȚIU

---

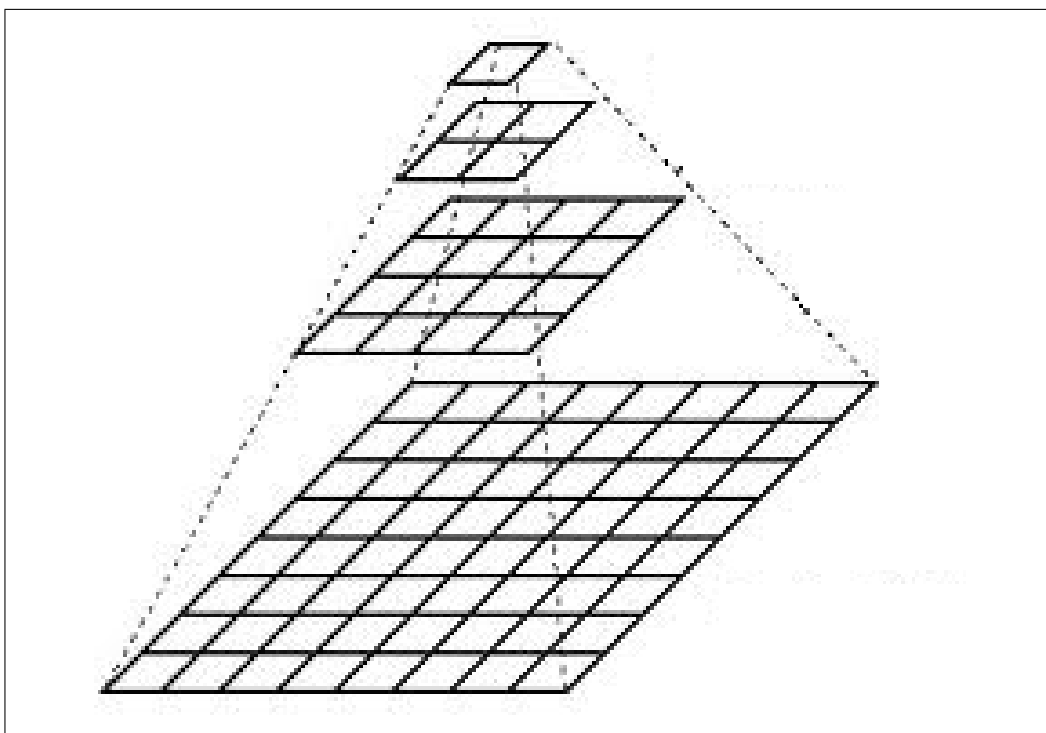


Figura 2.1: Piramida de imagini



Figura 2.2: Exemplu piramida de imagini



## 2.1. PARCURGEREA IMAGINII ÎN SCARA ȘI SPAȚIU

---

Prezentarea formarii piramidei de imagini în pseudo-cod:

```
sursa = citește_imagine()
alpha = 6/5
dim_min = (100,100)
piramida = [sursa, ]
L=1
cicleaza
    D = sursa.D * 1/(alpha^L)
    dacă D < dim_min
        atunci paraseste ciclul
    sfarsit dacă
    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, fiind un model discret, aceasta problema poate fi ameliorata prin alegerea unui  $\alpha$  potrivit și permițând modelului din baza de date să reprezinte și el mici variații de scară.

O altă observație ar fi, cu cât  $\alpha$  este mai mic, cu atât șansele să nimerim scara corectă 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 execută într-un mod recursiv, astfel eliminând menținerea explicită a unei liste de imagini în memorie.

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

Pseudo-cod fereastră glisantă:

```
dx = 8
dy = 8
I = citește_imagine()
M = citește_model()
pentru x de la 0 la dimx(I) - dimx(M)
    pentru y de la 0 la dimy(I) - dimy(M)
```

## 2.1. PARCURGerea IMAGINII ÎN SCARA ȘI SPAȚIU

```
fereastră = secțiune(I, x, y, dimx(M), dimy(M))  
proceseaza(fereastră)  
sfarsit pentru  
sfarsit pentru
```

Se poate vizualiza algoritmul fereastră glisanta în figura următoare:



Figura 2.3: Fereastră glisanta

Se observa ca și aici, ca și în cazul piramidei de imagini, cu cât  $x$  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 combinat cu fereastră glisanta este

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

### 2.2 Extragerea de trăsături

Extragerea de trăsături se ocupa cu, în cazul nostru, calcularea unei reprezentări a imaginii potrivite pentru recunoaștere.

O imagine este reprezentată ca o matrice de intensități. Aceasta reprezentare este foarte sensibilă la condițiile de iluminare, conține informații irelevante și redundante. Se poate observa efectul iluminării în figura 2.4.

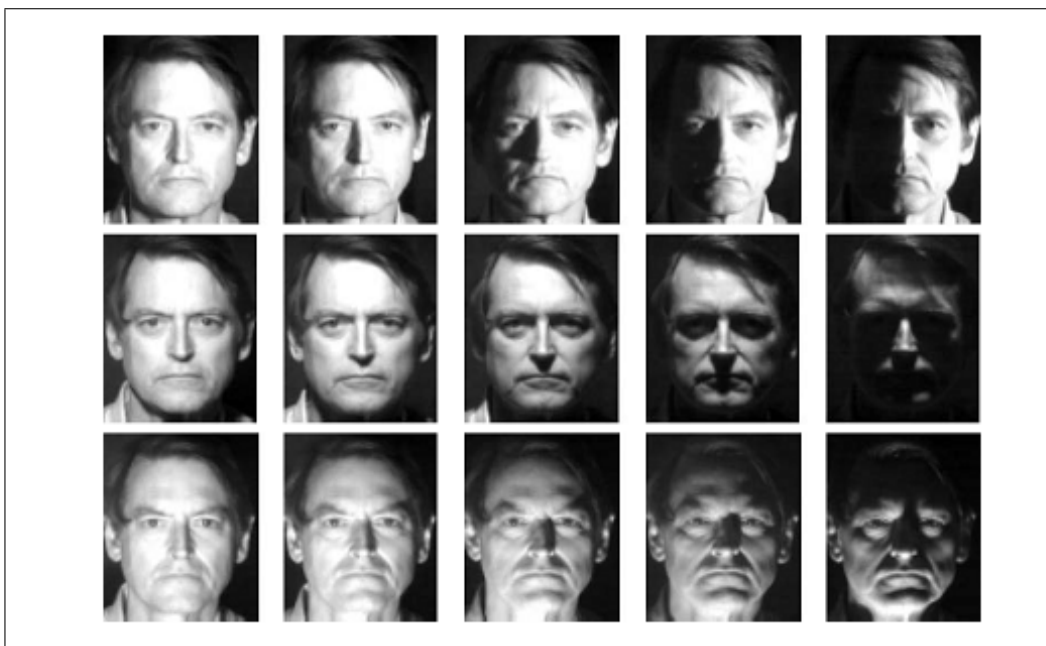


Figura 2.4: Efectul iluminării

Există modalități de a remedia efectul iluminării, cum ar fi egalizarea histogramei(fig. 2.5). O altă modalitate ar fi să se folosească o reprezentare pe baza de gradienti care sunt invariante la iluminare.

## 2.2. EXTRAGEREA DE TRĂSĂTURI

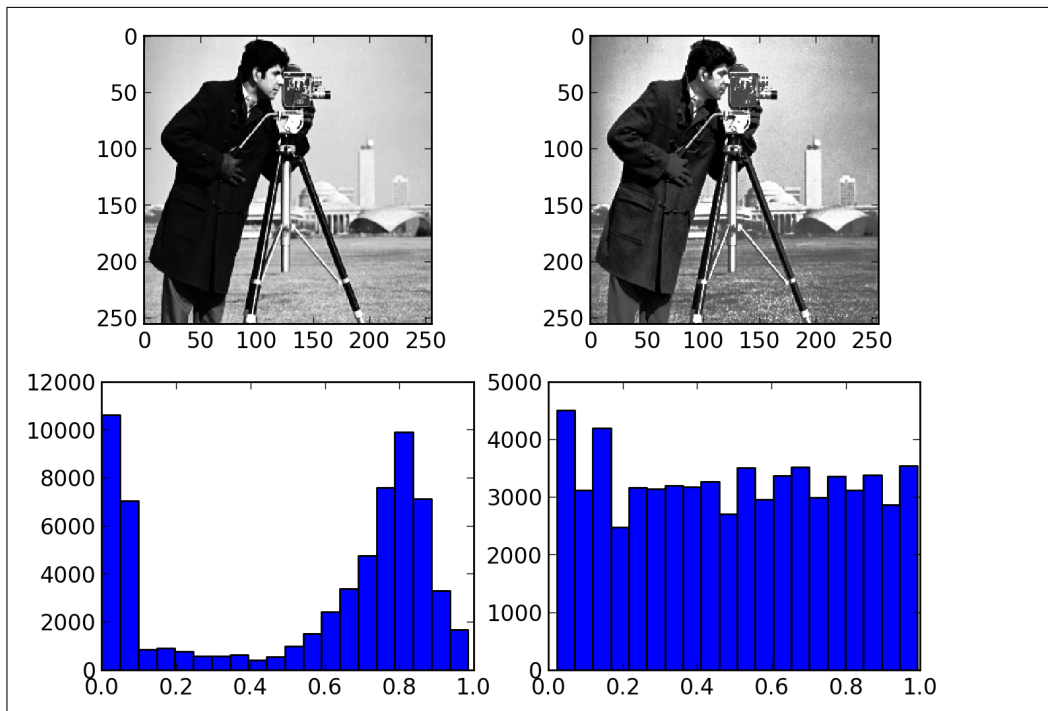


Figura 2.5: Egalizarea Histogramei

Efectul informațiilor irelevante și redundante poate fi ameliorat folosind tehnici de reducerea de dimensionalitate, cum ar fi analiza componentelor principale<sup>2</sup>(fig. 2.6 sau transformata cosinus discreta(fig. 2.7).

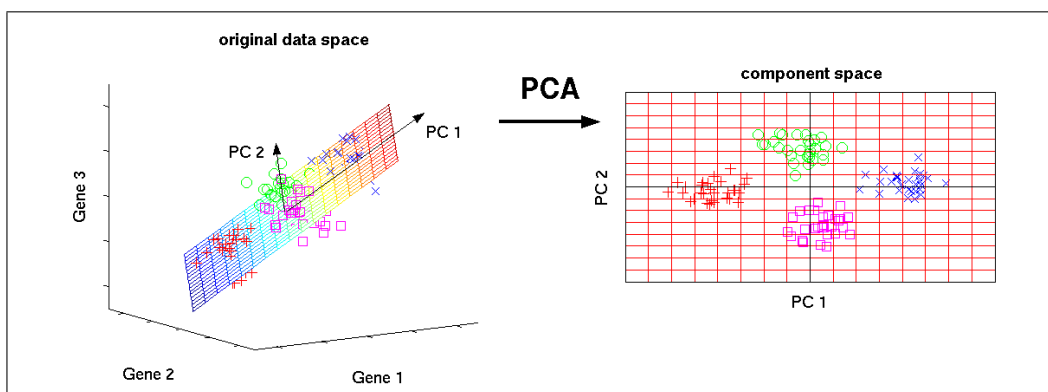


Figura 2.6: Analiza componentelor principale

<sup>2</sup>eng. PCA, principal component analysis

## 2.2. EXTRAGEREA DE TRĂSĂTURI

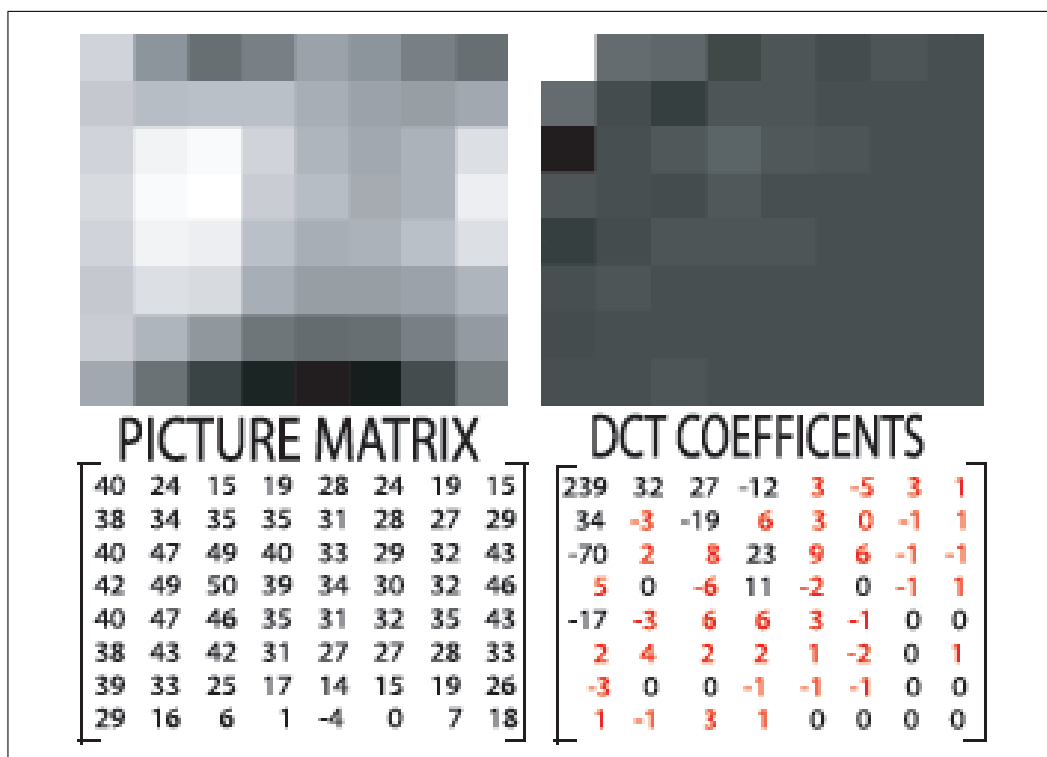


Figura 2.7: Transformata cosinus

Totuși nici una dintre reprezentările menționate mai sus nu tratează problema discriminării, adică dacă două imagini conțin același obiect atunci 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 trăsăturilor<sup>3</sup> care, folosind cunoștințe din fizică, biologie sau chiar neurologie construiește reprezentări mult mai favorabile recunoașterii. Câteva dintre cele mai cunoscute trăsături sunt Haar[VJ01], Sift[Low99] și Hog[DT05].

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

---

<sup>3</sup>eng. Feature Engineering

## 2.2. EXTRAGEREA DE TRĂSĂTURI

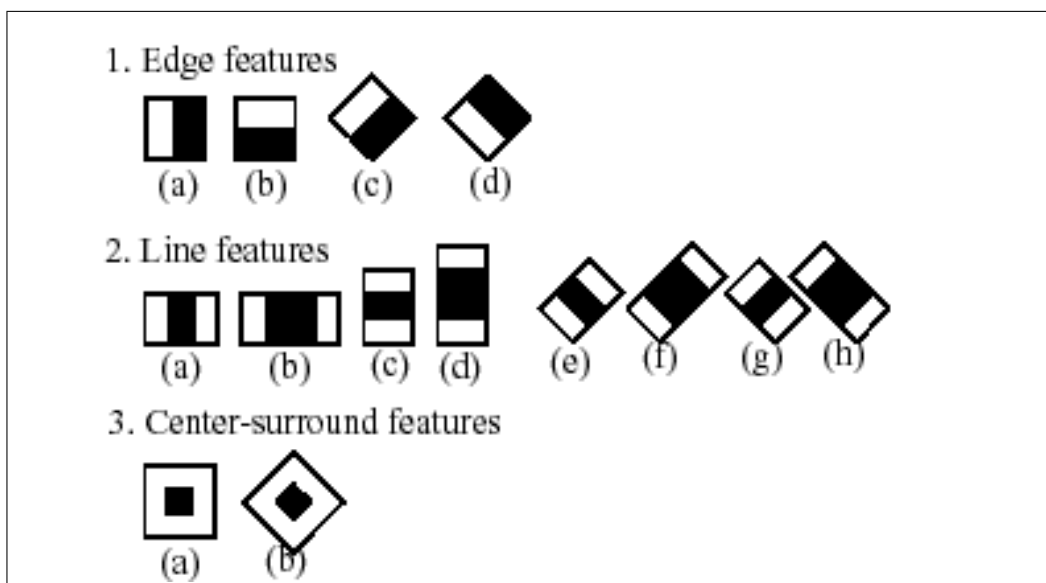


Figura 2.8: Trăsături Haar

Hog, sau histograma orientărilor de gradienti, se calculează divizând imaginea în zone mai mici, numite celule, apoi se calculează histograma de orientări a gradientilor din aceste zone. Concatenarea acestor histograme reprezentând trăsătura hog.

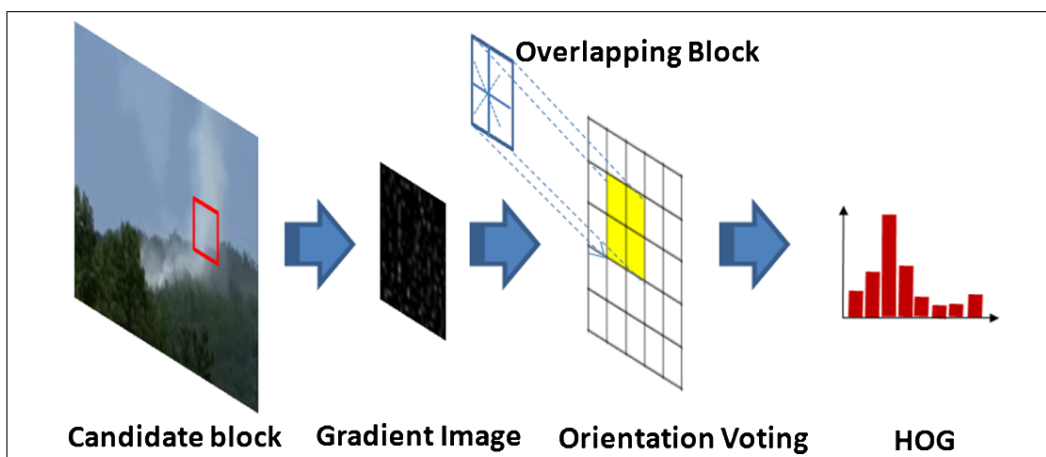


Figura 2.9: Trăsături hog

Descriptorul Sift este similar cu Hog, acesta fiind în plus și invariant la rotație.

## 2.2. EXTRAGEREA DE TRĂSĂTURI

---

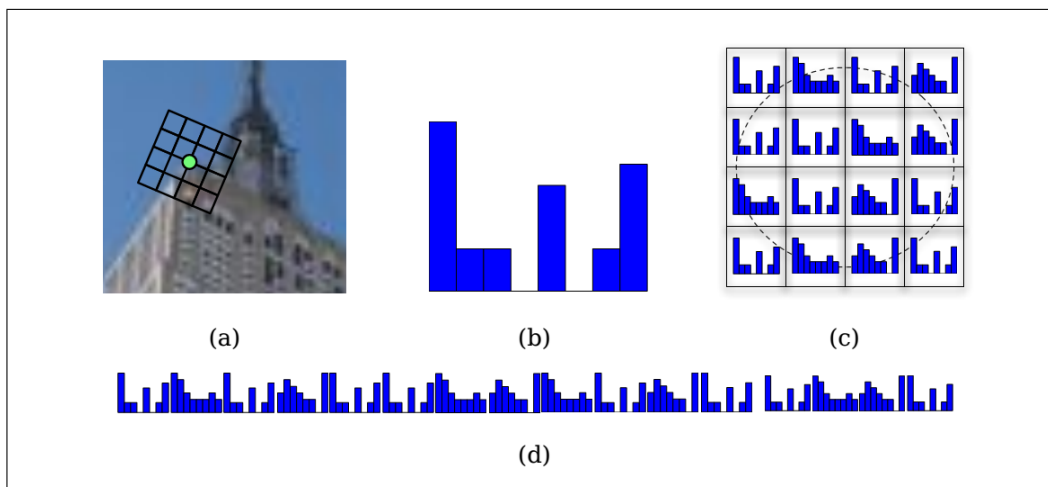


Figura 2.10: Descriptorul Sift

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 să extragă trăsături mai puternice, un exemplu ar fi rețeaua neurală convolutională.[LBBH98]

### 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 încercăm să îl recunoaștem. Acest tip de clasificare se numește clasificare binară, fiindcă răspunsul nu poate lua decât două valori.

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

În cazul nostru această funcție este rezultatul unui algoritm de învățare automată.<sup>4</sup> Învățarea automată, o ramură a inteligenței artificiale, este preocupată cu construcția și studiul sistemelor care pot învăța 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 clasificatori se mai numesc și predictorii.

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.

Cel mai des întâlniți algoritmi de învățare în viziunea artificială sunt: automatul cu vectori de suport<sup>5</sup>, rețeaua neuronală.

---

<sup>4</sup>Machine Learning

<sup>5</sup>eng. Support Vector Machines



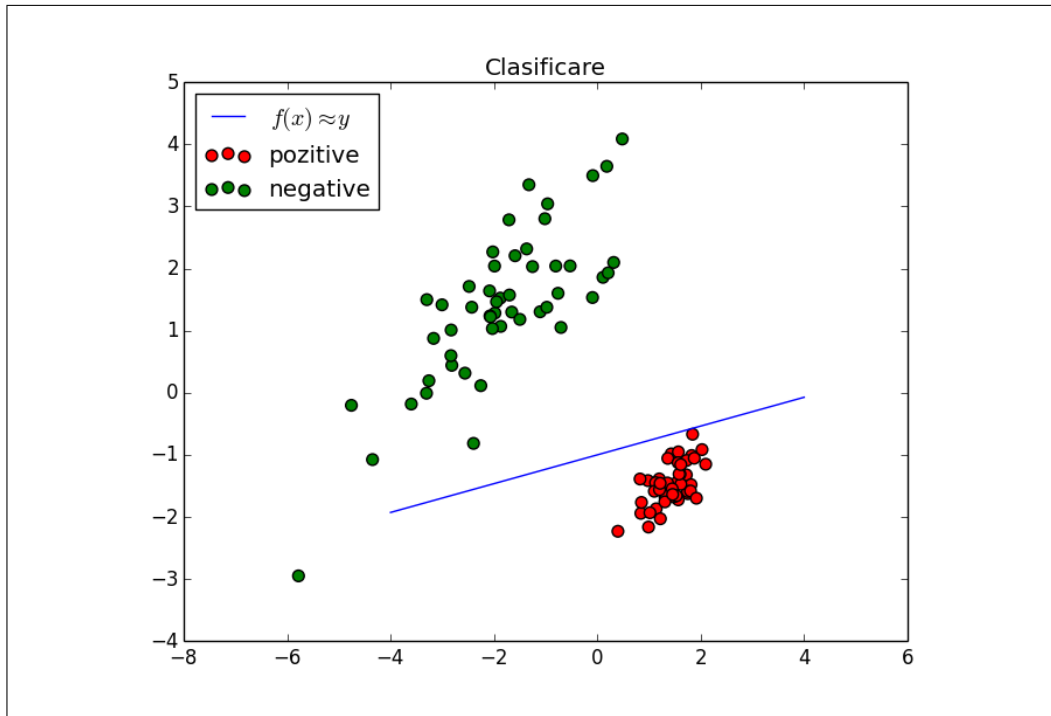


Figura 2.11: Clasificare

## 2.4 Post-procesarea rezultatelor

O situație foarte des întâlnită 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 să fie detectat doar o singură dată. 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 algoritmi de recunoaștere și antrenarea lor.

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

```
detectii = lista_goale()

I = citește_imaginea()
P = construiește_piramida(I)

pentru fiecare nivel din P
    pentru fiecare fereastră din extrage_ferestele(P)
        trasaturi = extrage_trasaturi(fereastră)
        raspuns = clasificare(trasaturi)
        dacă raspuns este afirmativ atunci
            detectii = adaugă(detectii, locație(fereastră))
        sfarsit dacă
    sfarsit
sfarsit

detectii = grupare_suprapuse(detectii)
```

Pentru antrenarea unui algoritm de recunoaștere a obiectelor avem nevoie de o bază 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 scară și spațiu de la algoritmul de recunoaștere. Pentru că 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 extrag un număr ales de exemple negative și se antrenează clasificatorul. Pe urma 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 asta se repeta

## 2.5. ALGORITMUL DE RECUNOAȘTERE

---

de un număr de ori specificat de utilizator, ori pana când nu se mai pot extrage exemplare negative din setul de date. Acesta se procedează se numește bootstrapping.

```
P = citește_setul_de_exemplare_positive()
N = citește_setul_de_exemplare_negative()

X = lista()
y = lista()

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

Cls = antrenează_clasificator(X,y)

iter = citește_nr_iterații()

pentru i = 1 până la iter
    pentru I din N
        P = construiește_piramidă(I)
        pentru fiecare nivel din P
            pentru fiecare fereastră din extrage_ferestrele(P)
                xi = extrage_trasaturi(xi)
                răspuns = clasificare(Cls, xi)
                dacă răspuns este 'afirmativ' atunci
                    X = adauga(X, xi)
                    y = adauga(y, 'negativ')
                sfarsit dacă
            sfarsit
        sfarsit
    sfarsit

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

# Capitolul 3

## Implementare **Librăriei**

În acest capitol va fi prezentată implementarea **librăriei** software pentru dezvoltarea de algoritmi și aplicații de recunoaștere a obiectelor.

### 3.1 Diagrama de clase

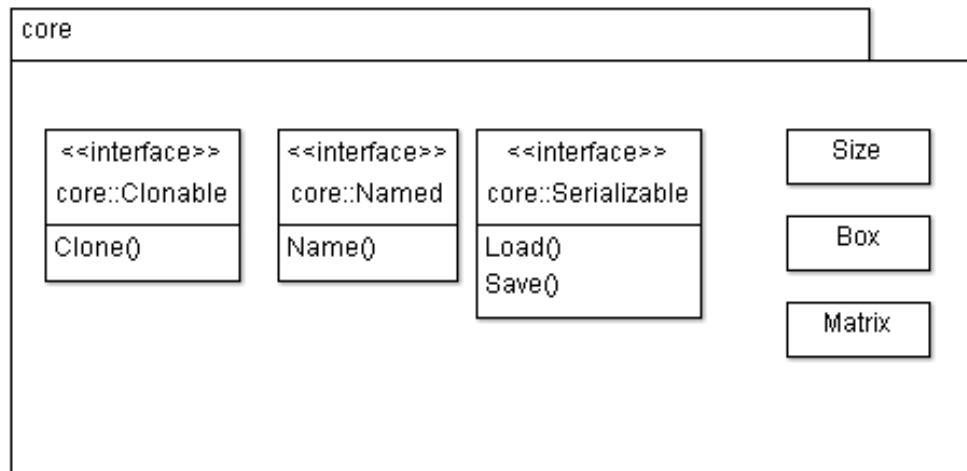
Librăria este împărțită în următoarele pachete:

- core
- dataset
- image-pyramid
- image-scanning
- feature-extraction
- classification
- non-maxima-suppression
- detection
- python

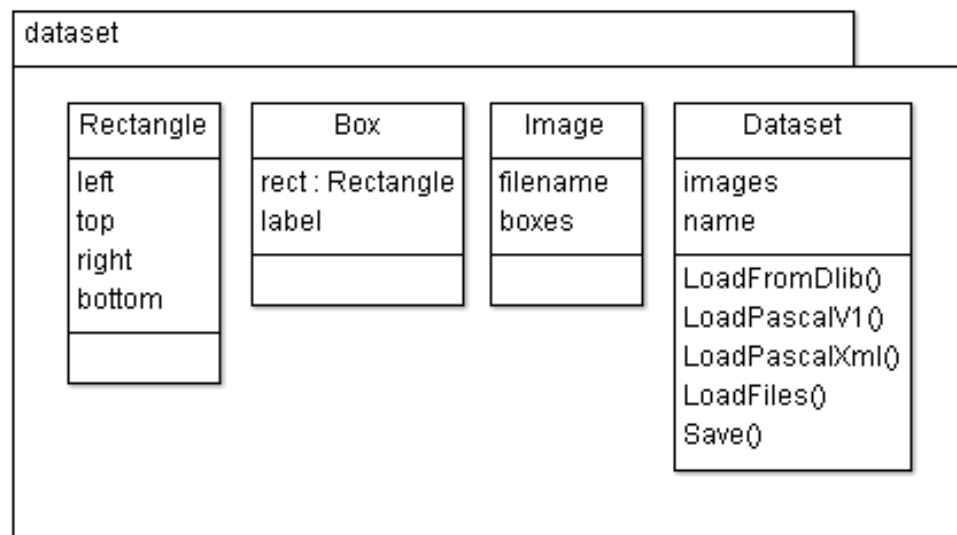
### 3.1. DIAGRAMA DE CLASE

---

Pachetul "core" conține interfețe de baza ale librăriei.

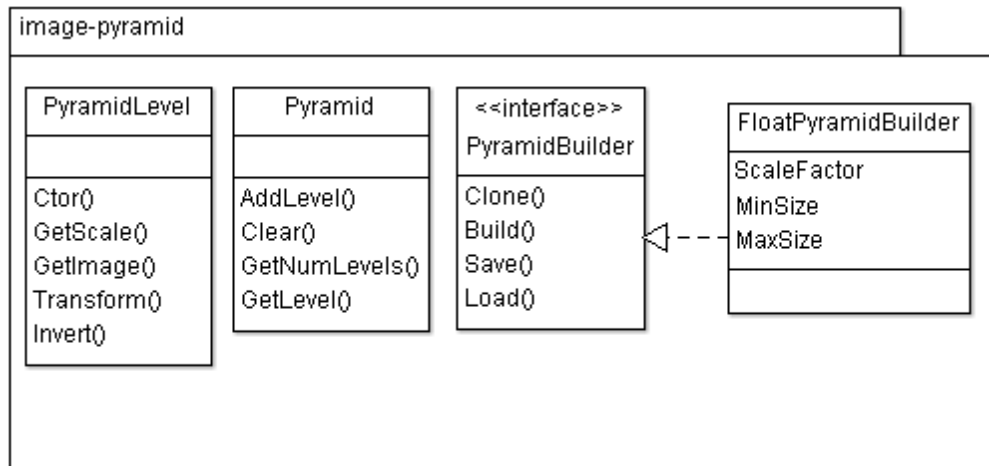


Pachetul "dataset" conține clase care modelează baza de date pentru antrenament, și implementează funcționalități de importare unor formate uzuale.



### 3.1. DIAGRAMA DE CLASE

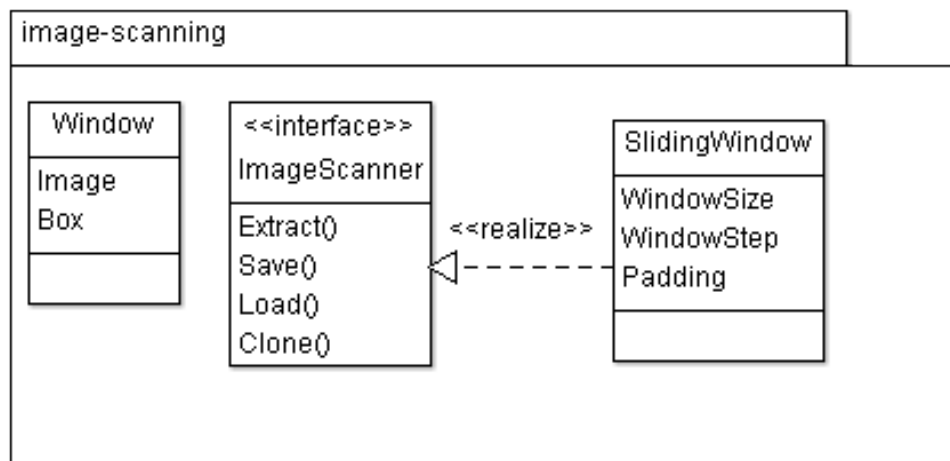
Pachetul "image-pyramid" conține interfețe și implementări care servesc la construcția piramidei de imagini.



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.

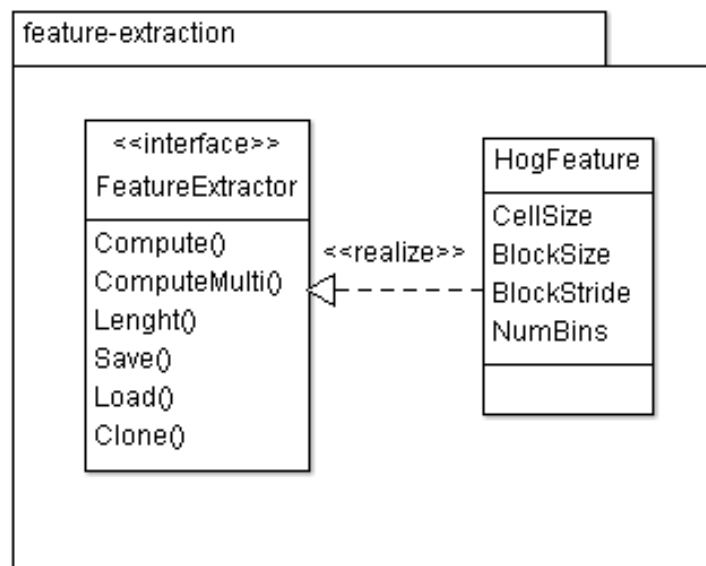
Pachetul "image-scanning" conține interfețe și implementări care servesc la scanarea imaginilor



### 3.1. DIAGRAMA DE CLASE

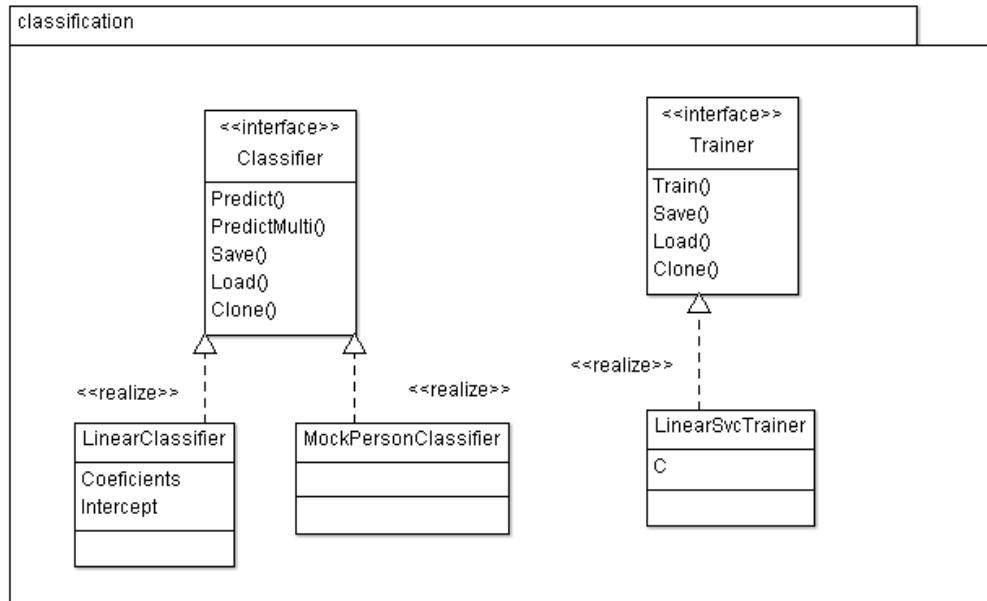
---

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

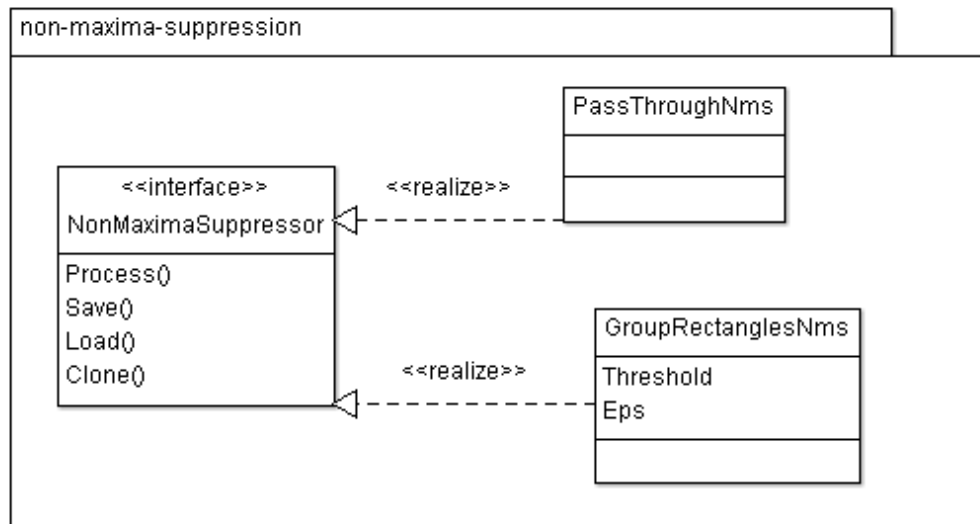


Pachetul "classification" conține interfețe și implementări care servesc la clasificare și antrenarea clasificatorilor.

### 3.1. DIAGRAMA DE CLASE



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

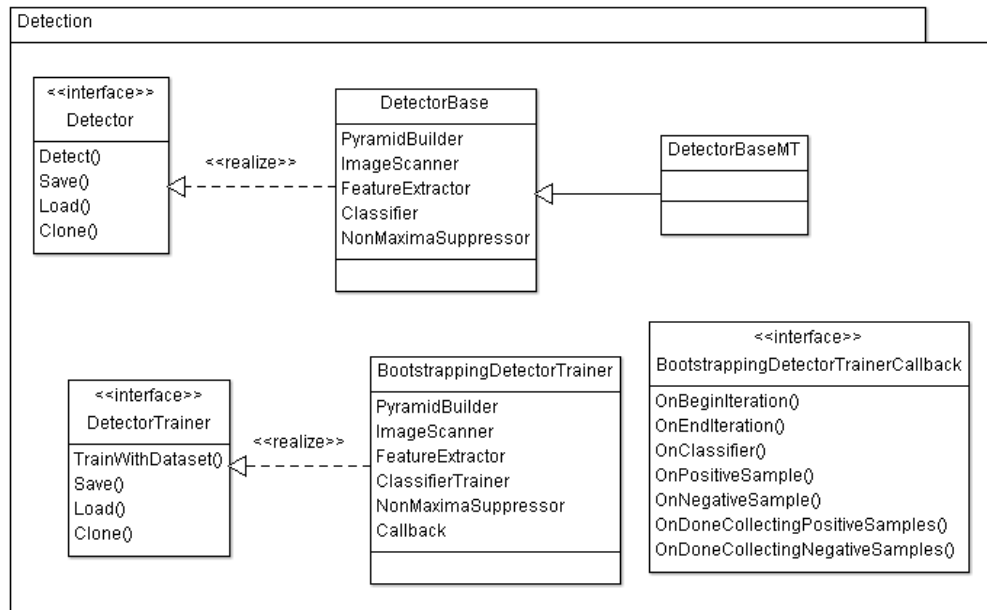


Pachetul "detection" conține interfețe și implementări care servesc la recunoașterea obiectelor în imagini și la antrenarea algoritmilor.



### 3.1. DIAGRAMA DE CLASE

---



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

## **3.2 Interoperabilitatea cu Python**

## **3.3 Serializarea**

## Capitolul 4

## Concluzii

# Bibliografie

- [DT05] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *In CVPR*, pages 886–893, 2005.
- [FGMR] Pedro F. Felzenszwalb, Ross B. Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part based models.
- [LBBH98] 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.
- [Low99] David G. Lowe. Object recognition from local scale-invariant features, 1999.
- [VJ01] Paul Viola and Michael Jones. Robust real-time object detection. In *International Journal of Computer Vision*, 2001.