Universitatea Transilvania din Brașov

Facultatea de Matematică și Informatică

Specializarea Informatică

Proiect de disertație

Tehnici de Machine Learning în procesarea și recunoașterea imaginilor

Autor:
Draghia Alin-Madalin

Profesor coordonator:
Lect. dr. Sasu Lucian Mircea

Brașov

Iulie 2014

Cuprins

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 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 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.

Figura 1.1: Volvo: sistemul de detectie a pietonilor

- Focalizarea automata a camerelor foto pe fețe

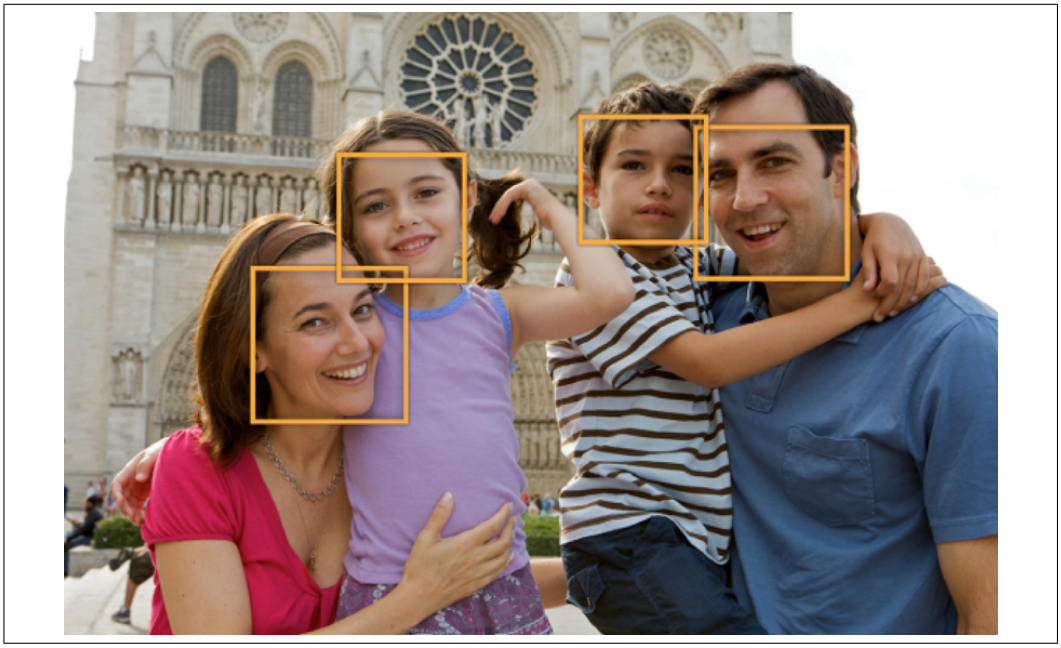


Figura 1.2: Camera foto: focus automat

- Analizarea traficului rutier

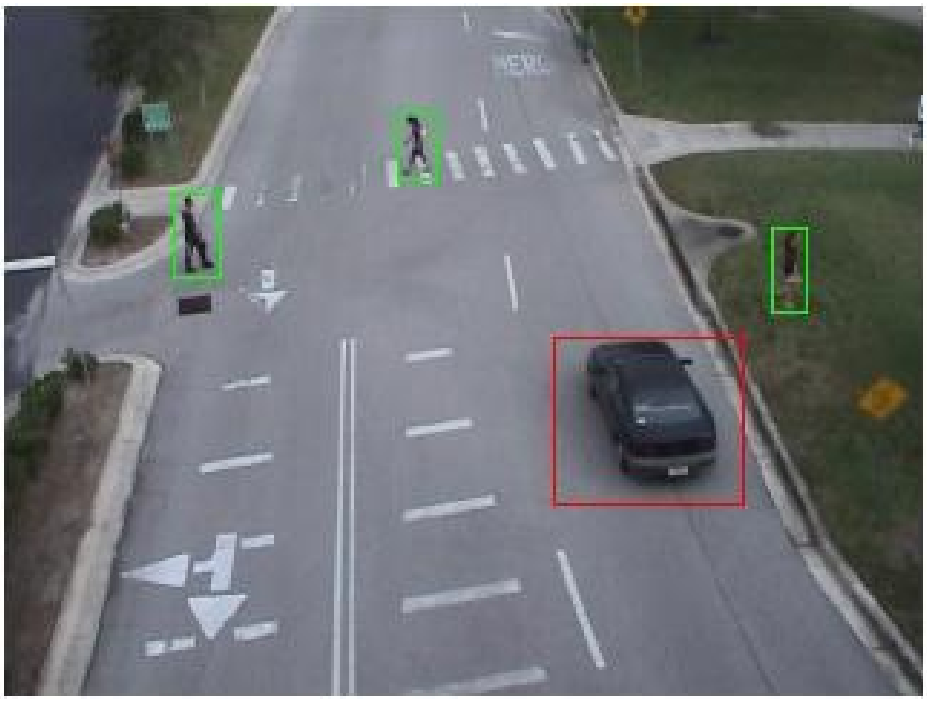


Figura 1.3: Analizarea traficului rutier

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 librarii 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 find imposibila. Nici codul sursa nu este ușor de urmări, acești algoritm sunt optimizați cu instrucțiuni de asamblare, cod pe mai multe fire de execuție și chiar cod care se executa pe procesorul grafic. Din aceasta

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 învață 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.

Se scrie o aplicate 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]

Librăria Boost

[Am rezervat o pagina]

Librăria scikit-learn

[Am rezervat o pagina]

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 generala a unui algoritm de recunoaștere. Vor fi prezentate în detaliu componentele din structura algoritmului. Se va prezenta o tehnica eficienta de antrenare a acestor algoritmi.

Capitolul 3 se prezintă în detaliu implementarea unei librarii 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 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 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 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 cu aceasta abordare ar fi ca nu se pot știi 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 a 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. 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.

O alta soluție, poate un pic mai naiva, dar în același timp foarte puternica este folosirea unei combinații de 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 permite ca in 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 și se opresc atunci când se ajunge la o dimensiune minima. 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 dimensiune imaginii sursa și L este nivelul piramidei pentru care dorim sa aflam dimensiune.

Se poate vizualiza piramida de imagini în figurile următoare:

[1] eng. sliding window

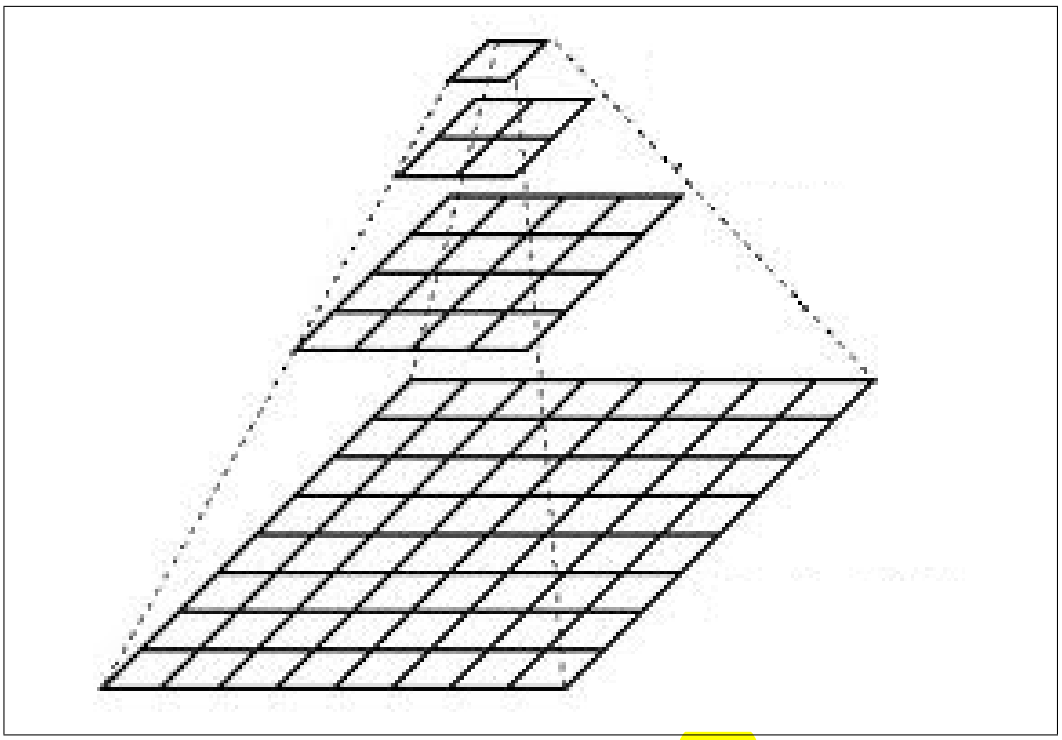


Figura 2.1: Piramida de imagini

Figura 2.2: Exemplu piramida de imagini

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)
  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, astfel eliminând 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 aceiași 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 2.3: Fereastra glisanta

Se observa ca și 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

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

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 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 2.4.

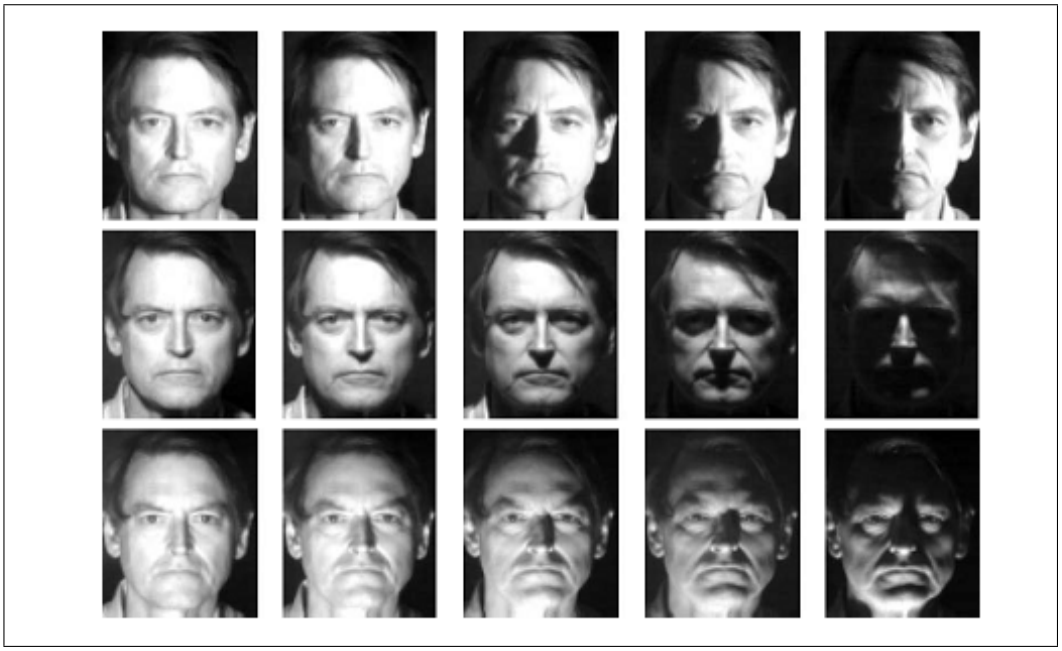


Figura 2.4: Efectul iluminarii

Exista modalități de a remedia efectul iluminării, cum ar fi egalizarea histogramei(fig. 2.5). O alta modalitate ar fi sa se folosească o reprezentare pe baza de gradienți care sunt invarianți la iluminare.

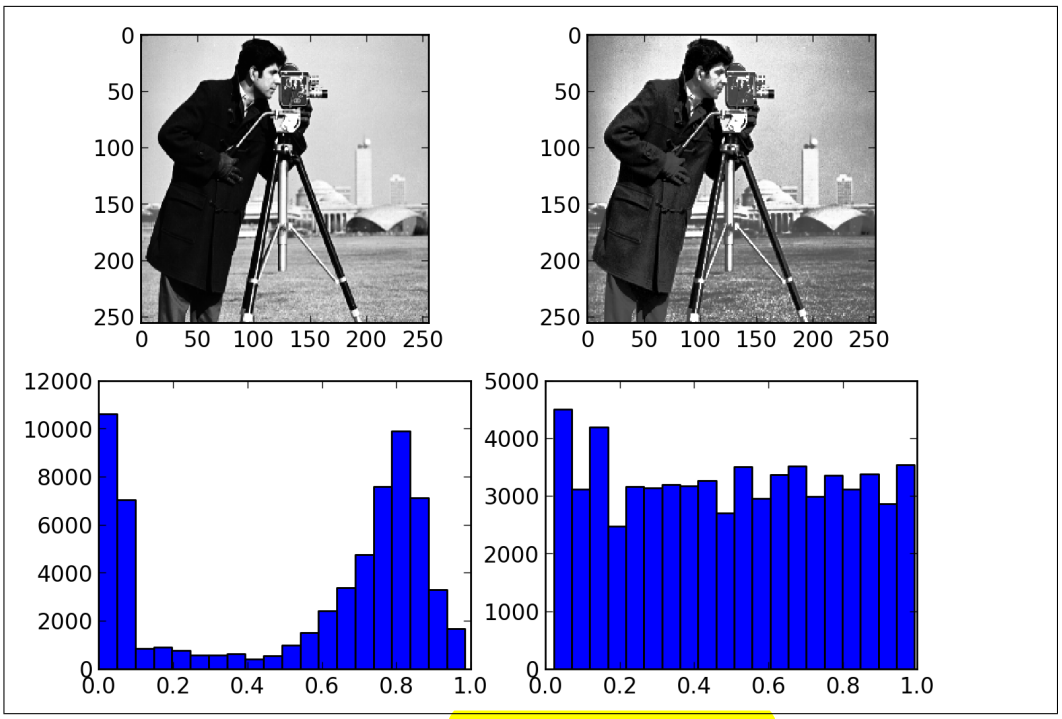


Figura 2.5: Egalizarea Histogramei

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

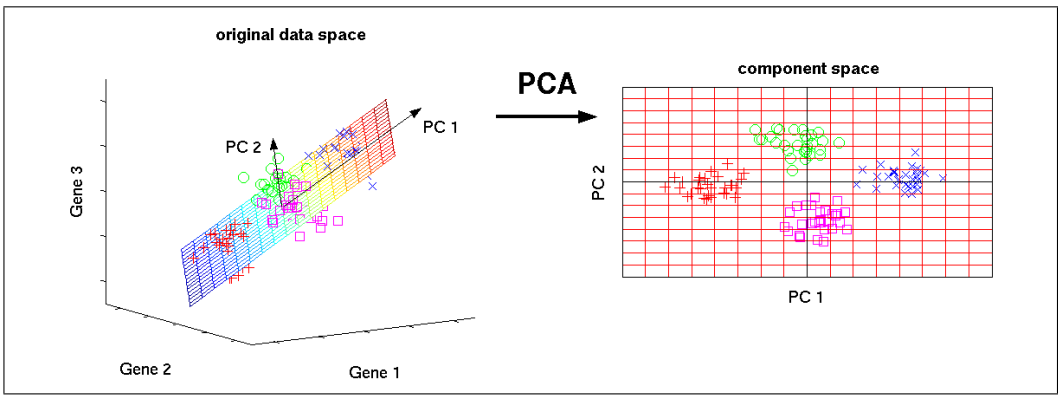


Figura 2.6: Analiza componentelor principale

[2]eng. PCA, principal component analisys

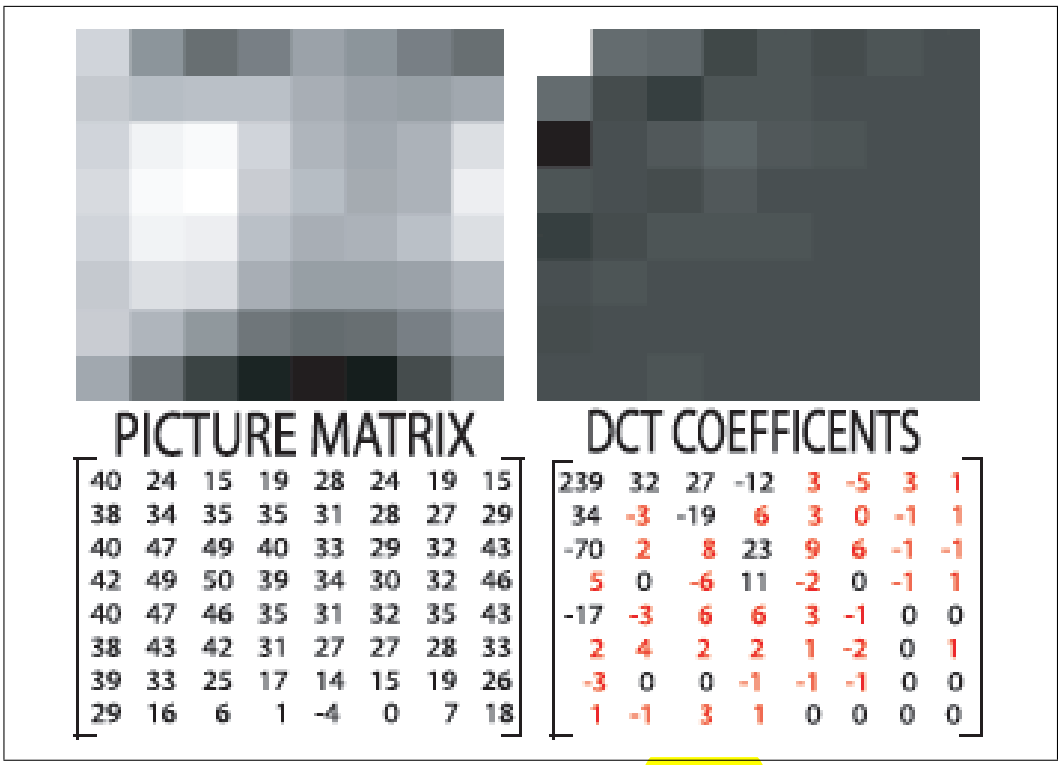


Figura 2.7: Transformata cosinus

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

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

[3]eng. Feature Engineering

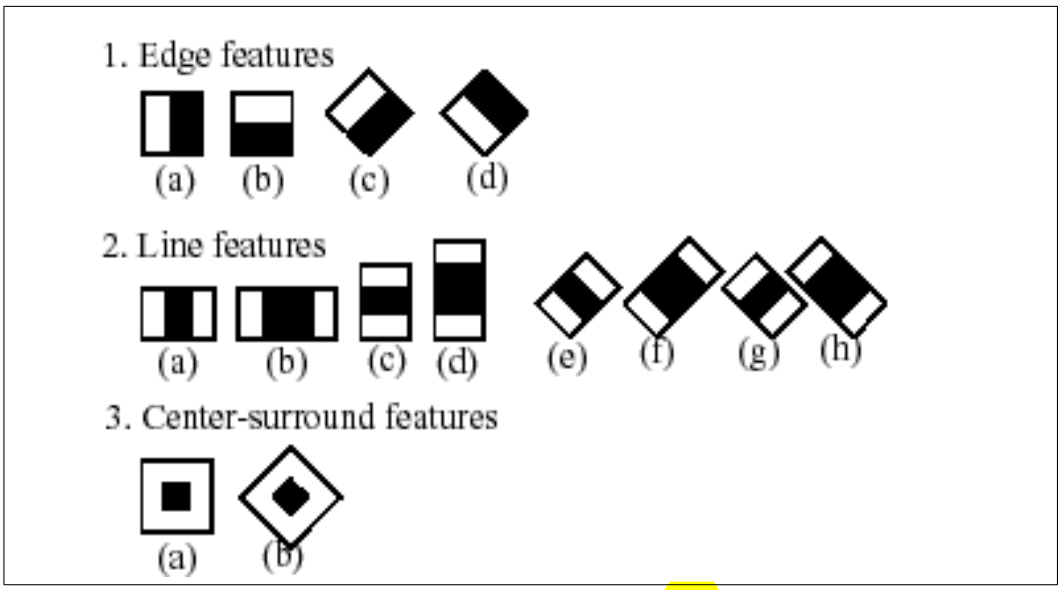


Figura 2.8: Trasaturi Haar

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

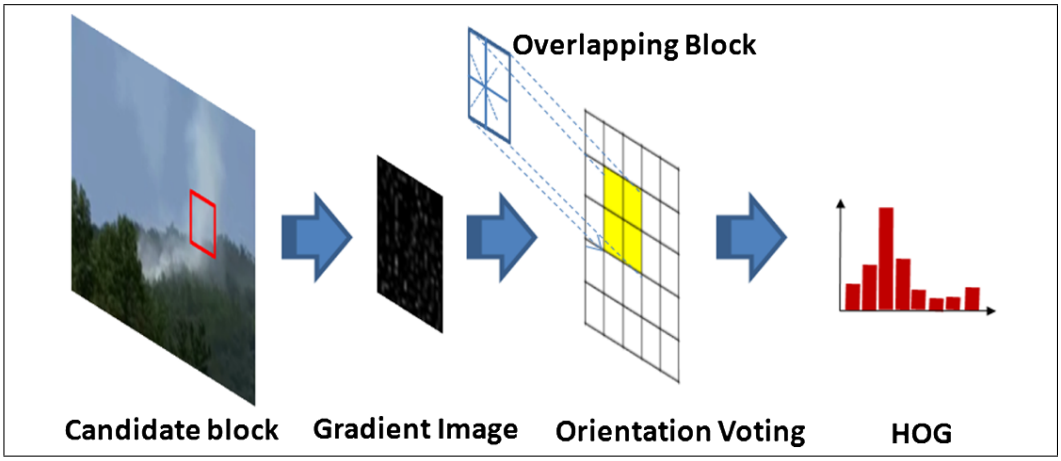


Figura 2.9: Trasaturi hog

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

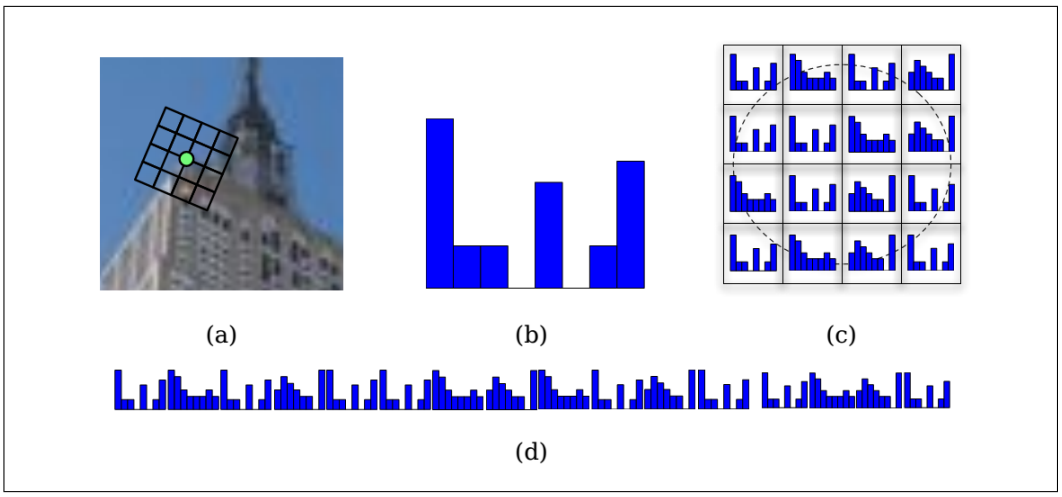


Figura 2.10: Descriptorul Sift

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 sa extragă trăsături mai puternice, un exemplu ar fi rețeaua neurala convolutionala.[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 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.[4] Învățarea automata, o ramura a inteligentei artificiale, este preocupata cu construcția și studiul 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 clasificatori 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.

Cel mai des întâlniți algoritmi de învățare în viziunea artificiala sunt: automatul cu vectori de suport[5], rețeaua neuronala.

[4] Machine Learning
[5] eng. Support Vector Machines

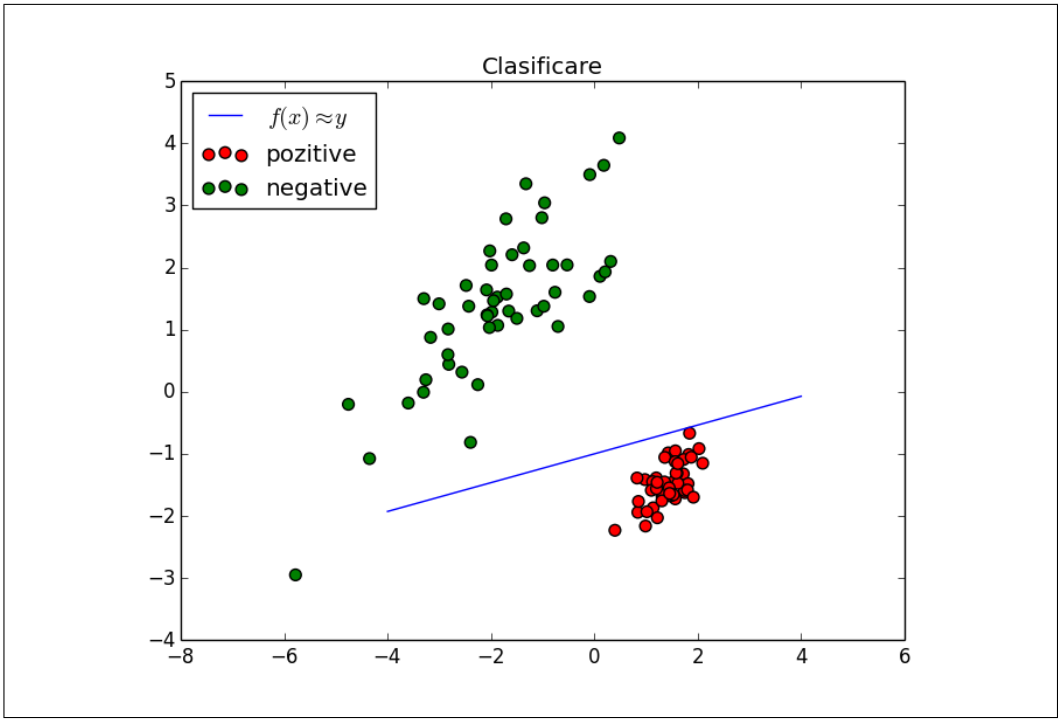


Figura 2.11: Clasificare

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 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

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

Implementare Librăriei

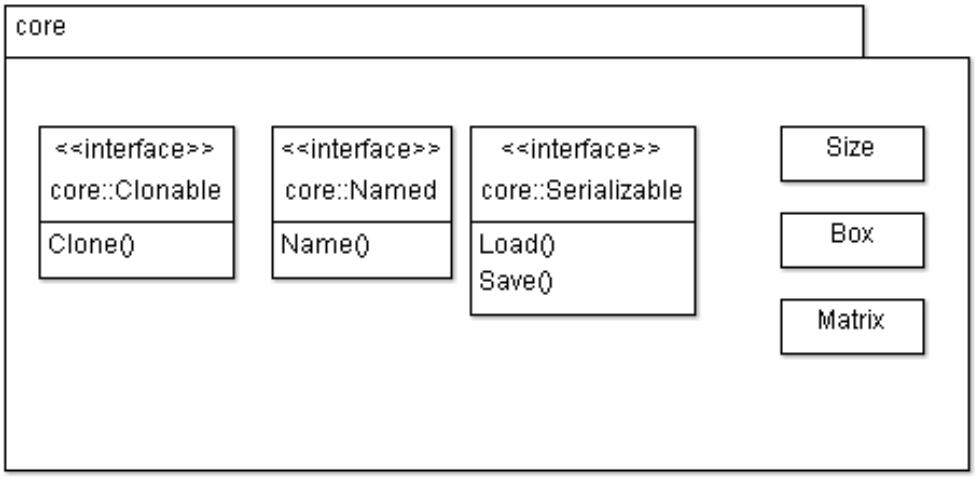
În acest capitol va fi prezentata implementarea librăriei software pentru dezvoltarea de algoritmi și aplicații de recunoaștere a obiectelor.

3.1 Diagrama de clase

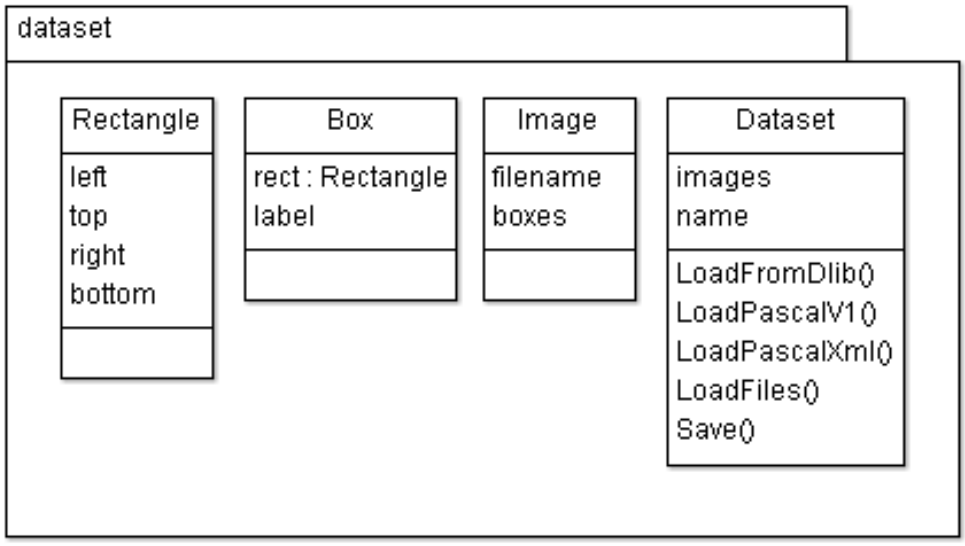
Librăria este împărțita în următoarele pachete:

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

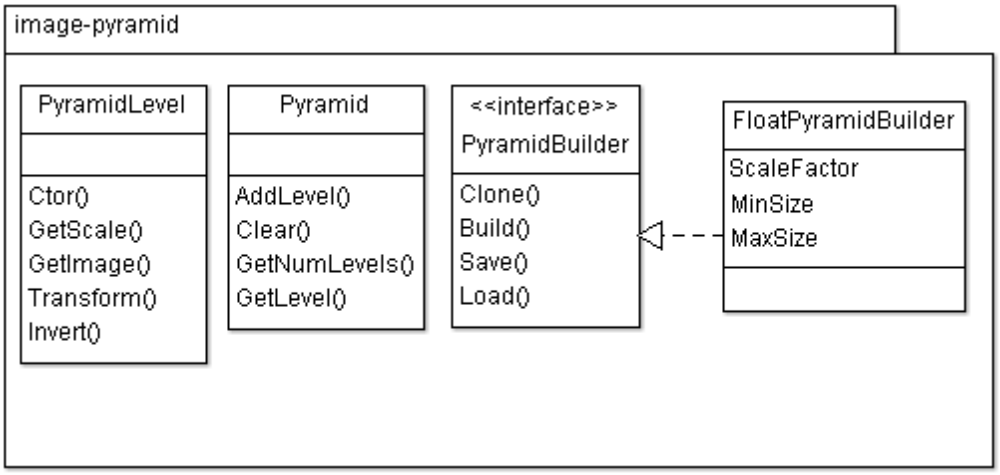
Pachetul "core" conține interfețe de baza ale librăriei.



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



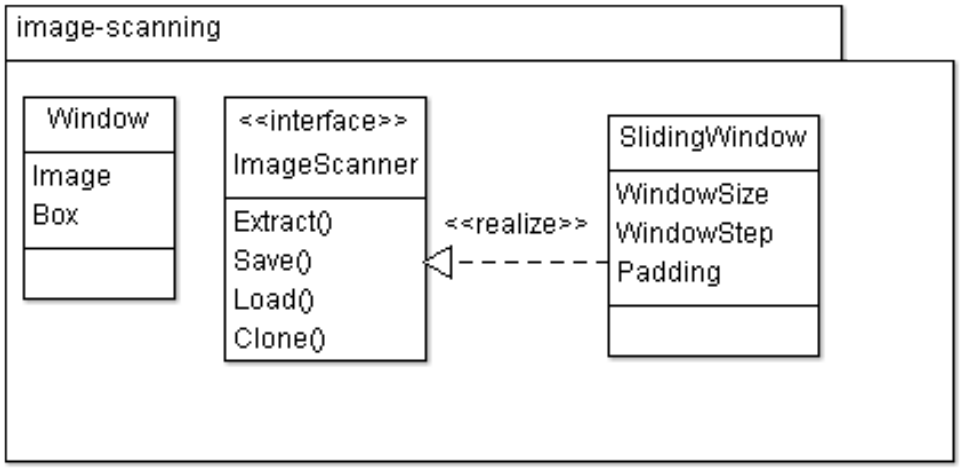
Pachetul "image-pyramid" conține interfețe și implementări care servesc la construcția piramidei de imagini.



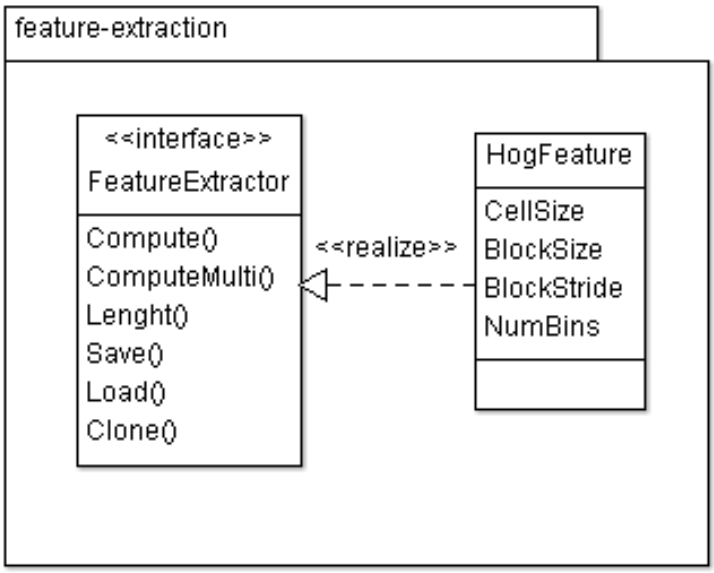
Clasa FloatPyramidBuilder construiește o piramida de imagini folosind ScaleFactor ca factor de scalare și MinSize, MaxSize ca criterii de terminare.

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

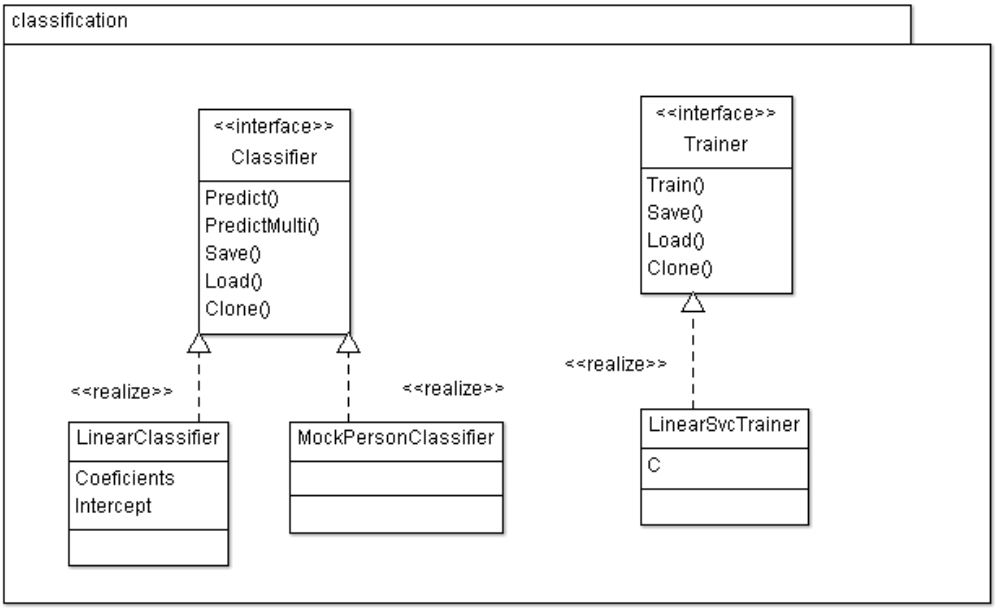
Pachetul "image-scanning" conține interfețe și implementări care servesc la scanarea imaginilor



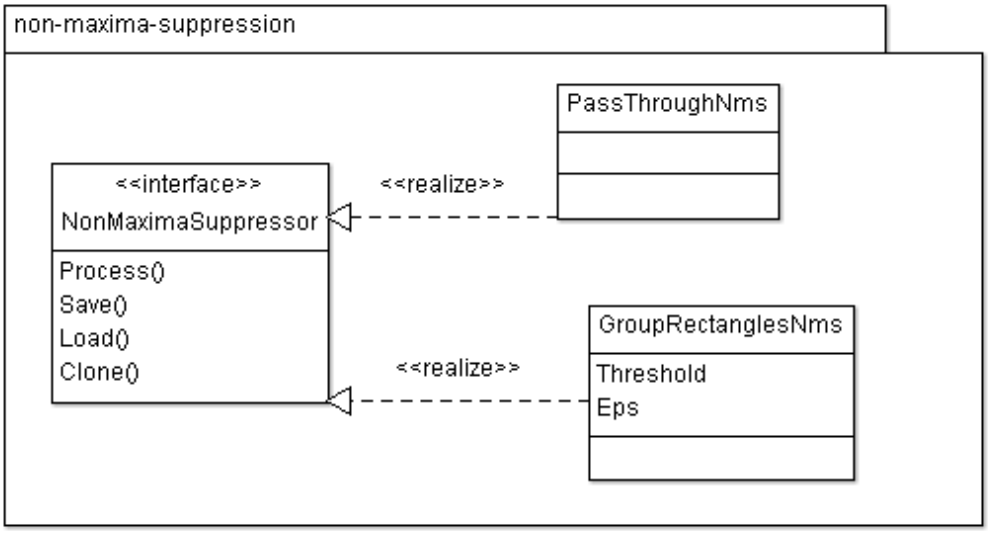
Pachetul "feature-extraction" conține interfețe și implementări care servesc la extragerea de trăsături din imagini.



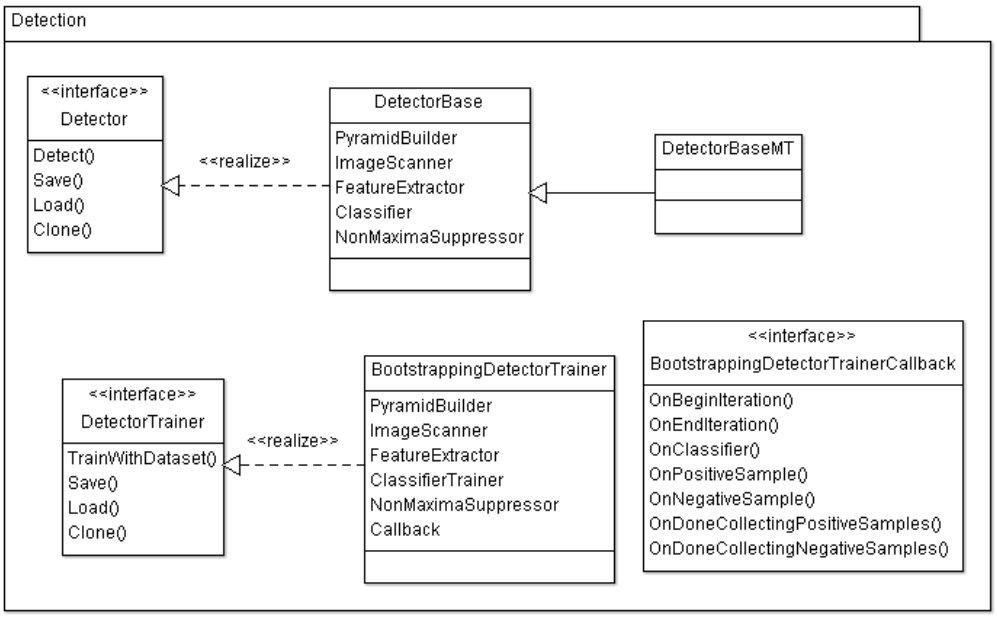
Pachetul "classification" conține interfețe și implementări care servesc la clasificare și antrenarea clasificatorilor.



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



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



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.