

Concepte și aplicații în Vederea Artificială

Bogdan Alexe

bogdan.alexe@fmi.unibuc.ro

Cursul 12

anul III, Opțional Informatică, semestrul I, 2018-2019

Examen - evaluare

- În funcție de 2 aspecte:
 - teme de laborator (vor fi 5 de-a lungul semestrului);
 - lucrare finală de laborator (în săptămâna 14).
- puteți obține nota numai din teme dacă:
 - aveți cel puțin 3 teme peste nota 5;
 - nota finală = media celor mai mari 3 note (din 5 posibile) din teme.

SAU

- puteți obține nota numai din lucrarea finală de laborator (test pe calculator).

SAU

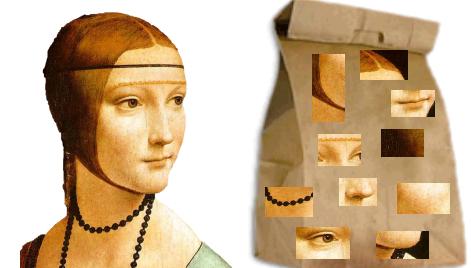
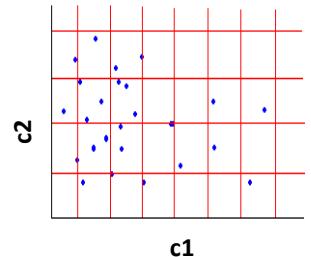
- puteți obține nota din media dintre lucrarea finală de laborator (trebuie să luați minim 5) și cea mai mare notă din temă (dacă vreti să o luăm în considerare)

Examen - evaluare

- Teme
 - tema 4 are termen limită în această săptămână;
 - vom afișa notele la primele 4 teme până luni, 7 ianuarie;
 - tema 5 are termen limită vineri, 11 ianuarie;
 - sperăm să afișăm notele la tema 5 până luni 14 ianuarie;
- Lucrare de laborator
 - miercuri, 16 ianuarie 2019, stabilim exact ora la primul curs din 2019.

Recapitulare – cursul trecut

- Clasificarea imaginilor
 - paradigma învățării supervizate
 - caracteristici și clasificatori
 - histograme de caracteristici
- Modelul Bag of Visual Words
 - clasificarea imaginilor folosind BoVW

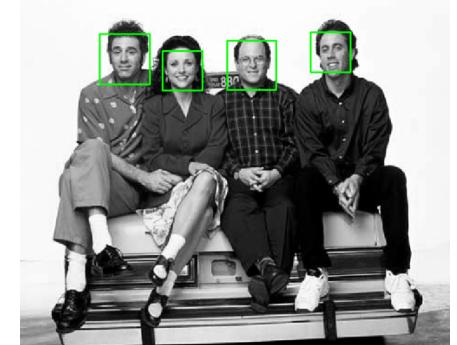


Cursul de azi

- Localizarea claselor de obiecte
 - metoda ferestrei glisante



- Detectare facială folosind metoda glisării ferestrei și histograme de gradienți orientați
(tema 5)



Localizarea obiectelor
la nivel de fereastră:
metoda ferestrei glisante
(sliding-window)

Recunoașterea claselor de obiecte la nivel generic

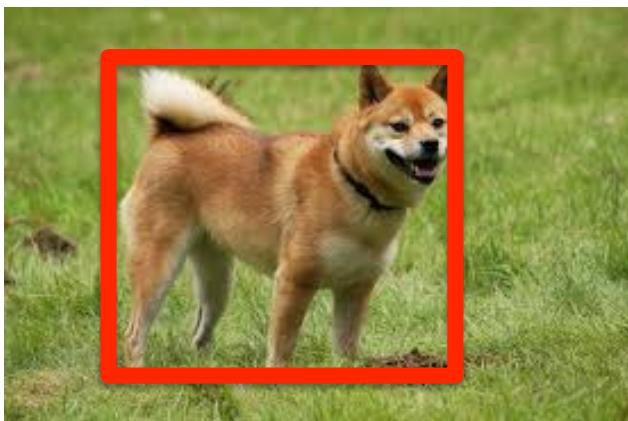
- Mașină



Vrem să recunoștem toate instanțierile unei clase de obiecte
Clasificare – DA/NU. Localizare – UNDE?

Recunoașterea claselor de obiecte la nivel generic

- Recunoaște orice câine în imagine

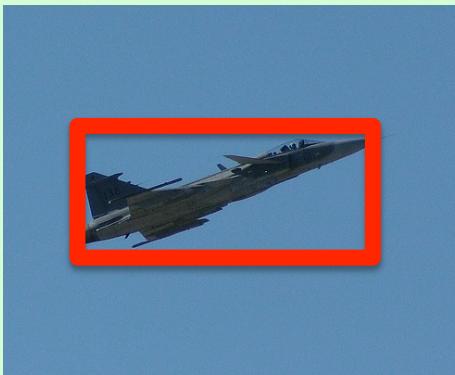


Recunoașterea claselor de obiecte la nivel generic

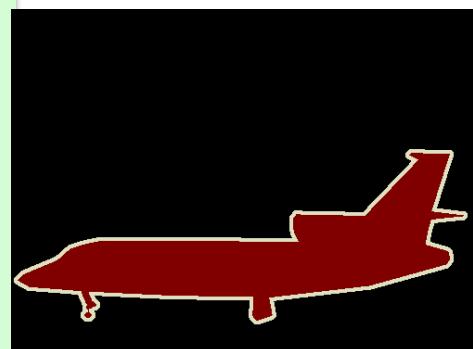
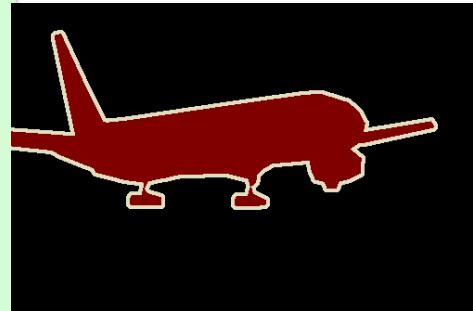
imagini inițiale



localizare - fereastră



localizare - contur



Dificultăți: robustețe



Iluminare



Postura obiectului



Mărime



Mascarea
obiectului

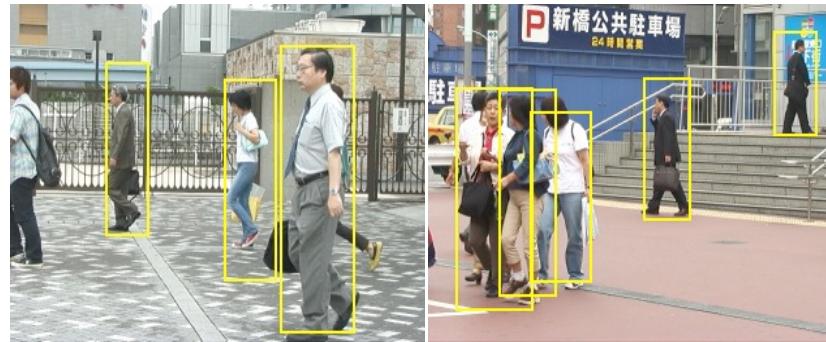


Variabilitate
înfățisare intra-
clasă



Poziționarea
camerei

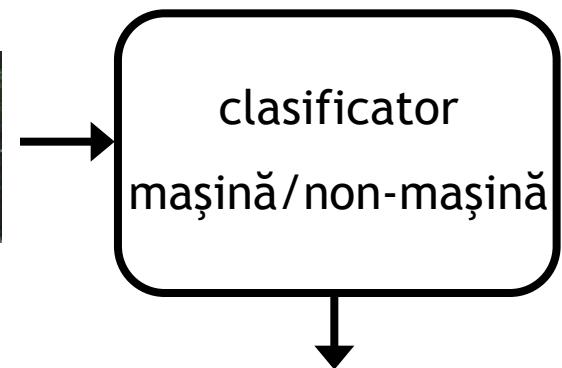
Dificultăți: robustețe



- Detectarea obiectelor în scene aglomerate
 - Învață variabilitatea obiectelor
 - Înfățisare, mărime, articulație
 - compensează pentru suprapunerea, mascarea obiectelor

Ideea principală: detectare via clasificare

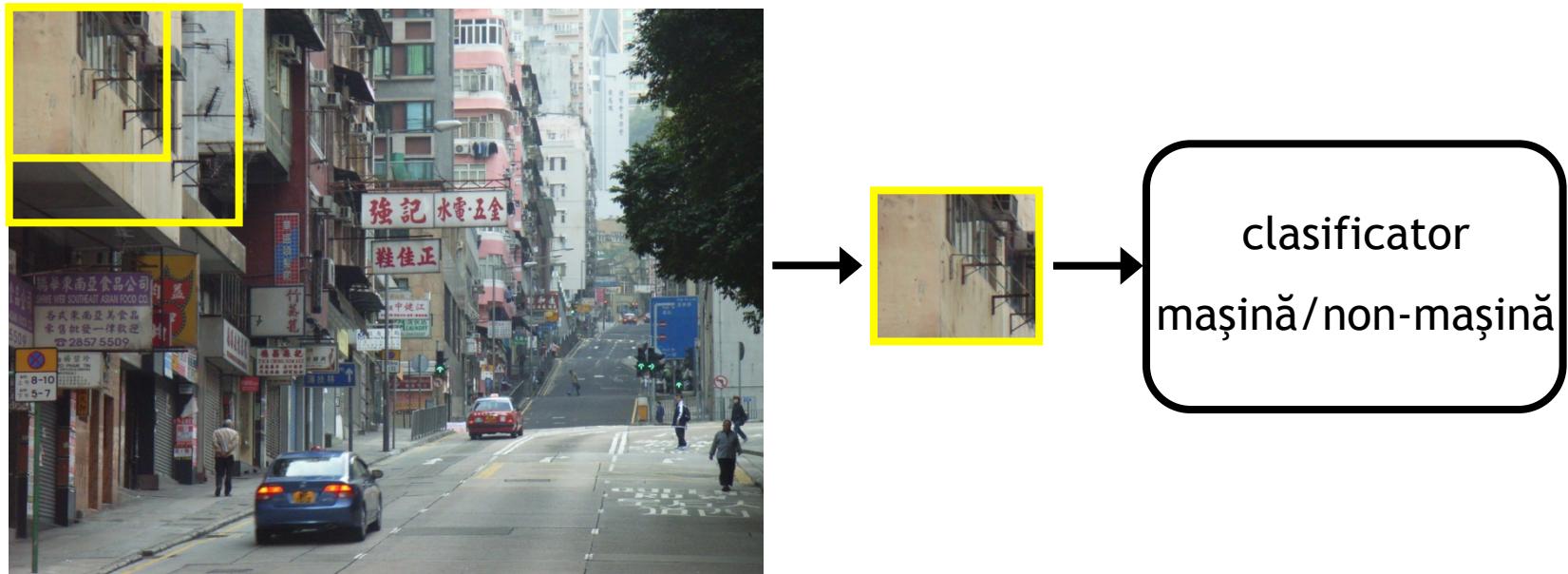
Componenta de bază: un clasificator binar



NDaște e mașină!

Ideea principală: detectare via clasificare

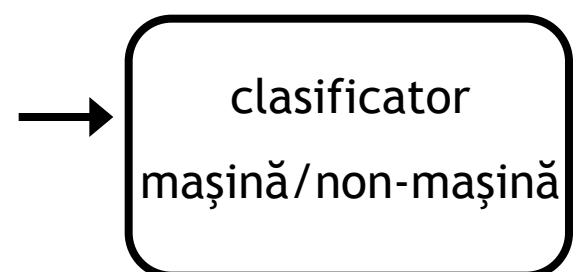
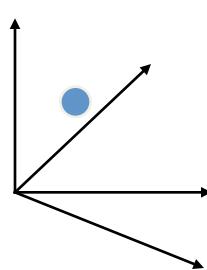
Dacă obiectul este într-o scenă aglomerată, glisează o fereastră căutând obiectul.



Ideea principală: detectare via clasificare

Pași:

1. Obținem exemple de învățare (pozitive + negative)
2. Definim caracteristici
3. Definim clasificator

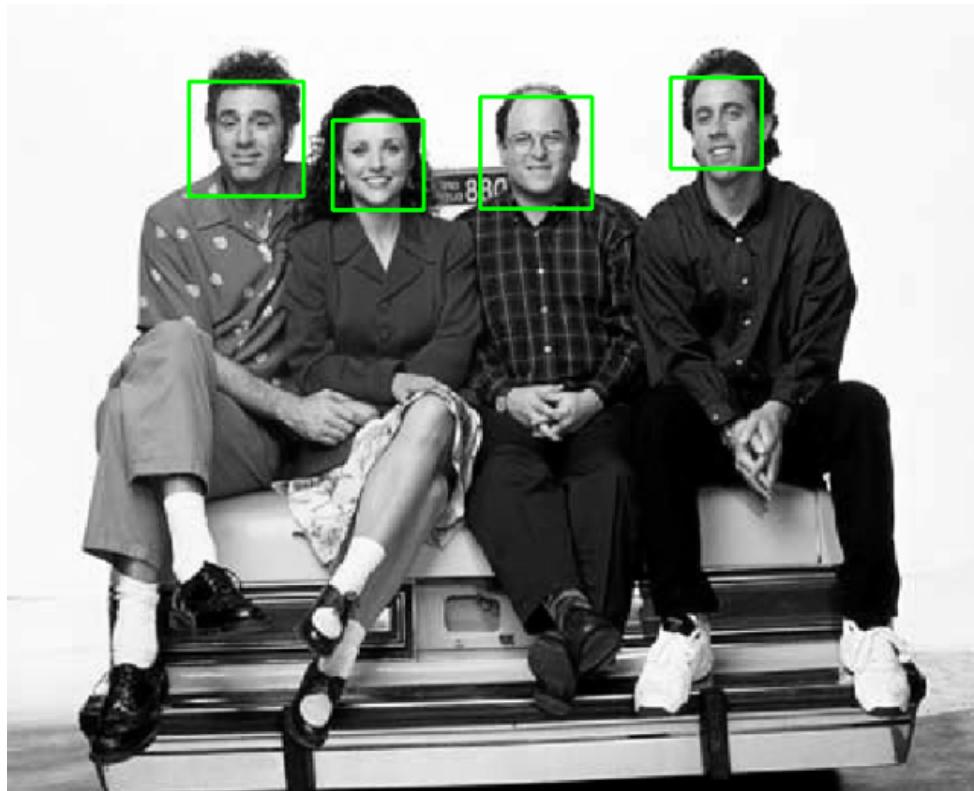


Extragere de
caracteristici

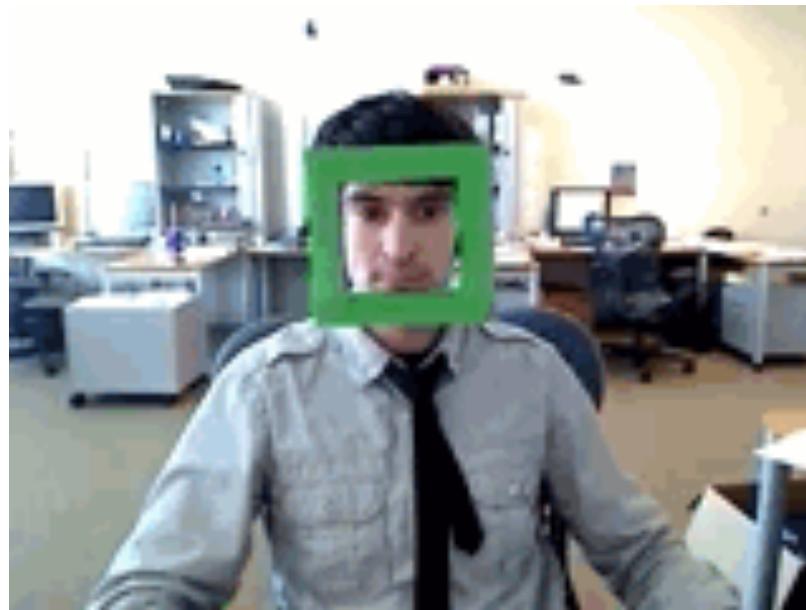
Ideea principală: detectare via clasificare

- Considerăm toate ferestrele dintr-o imagine
 - consideră ferestre poziționate în fiecare pixel, de mărimi diferite (+ orientări diferite)
- Pentru fiecare fereastră decide:
 - “Conține sau nu această fereastră o instantă a clasei de obiecte X?”

Detectare facială



Demo – cel mai bun sistem de detectare facială



Detectare facială și identificare



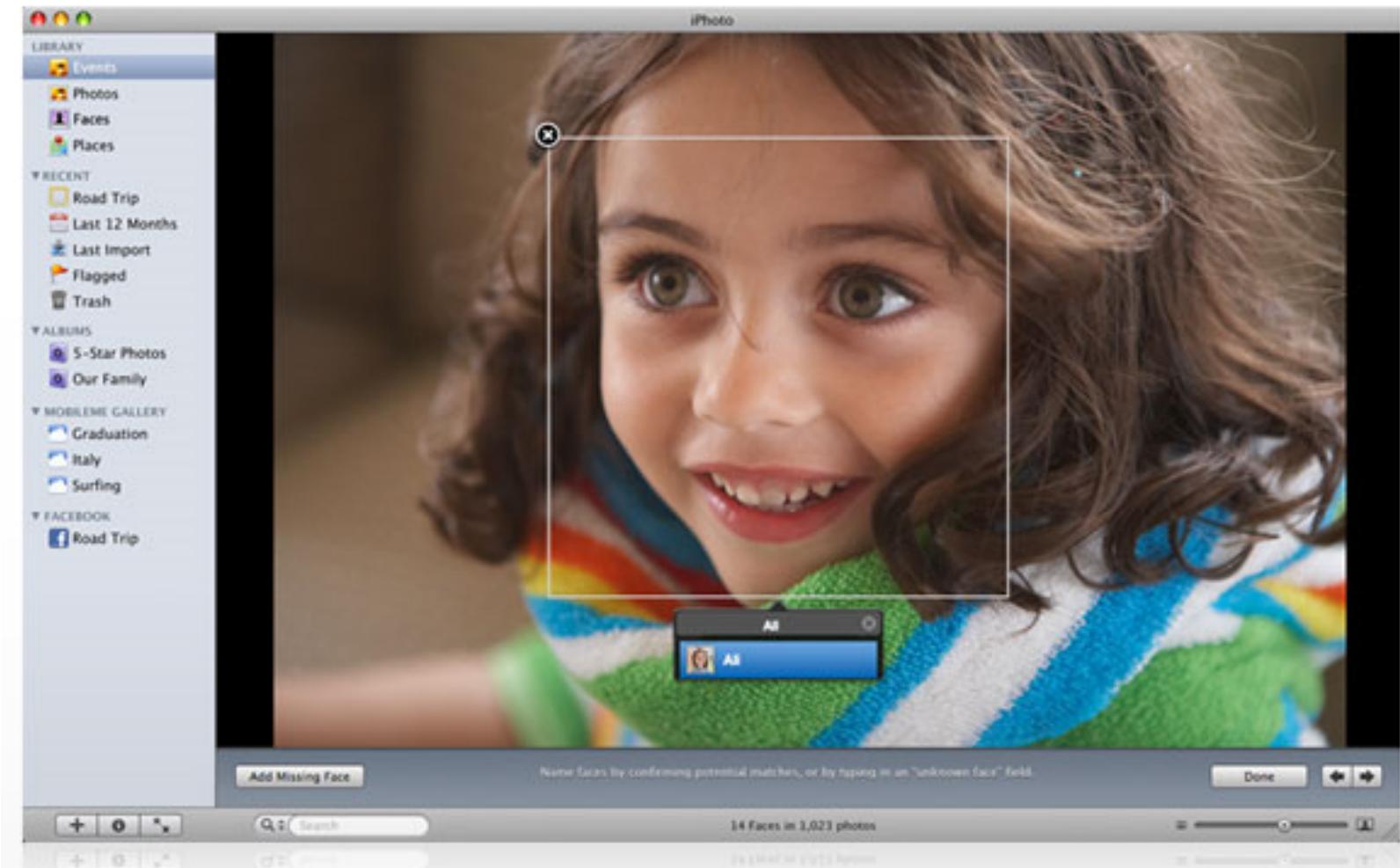
Detectare



Identificare

“Maria”

Aplicație: Apple iPhoto

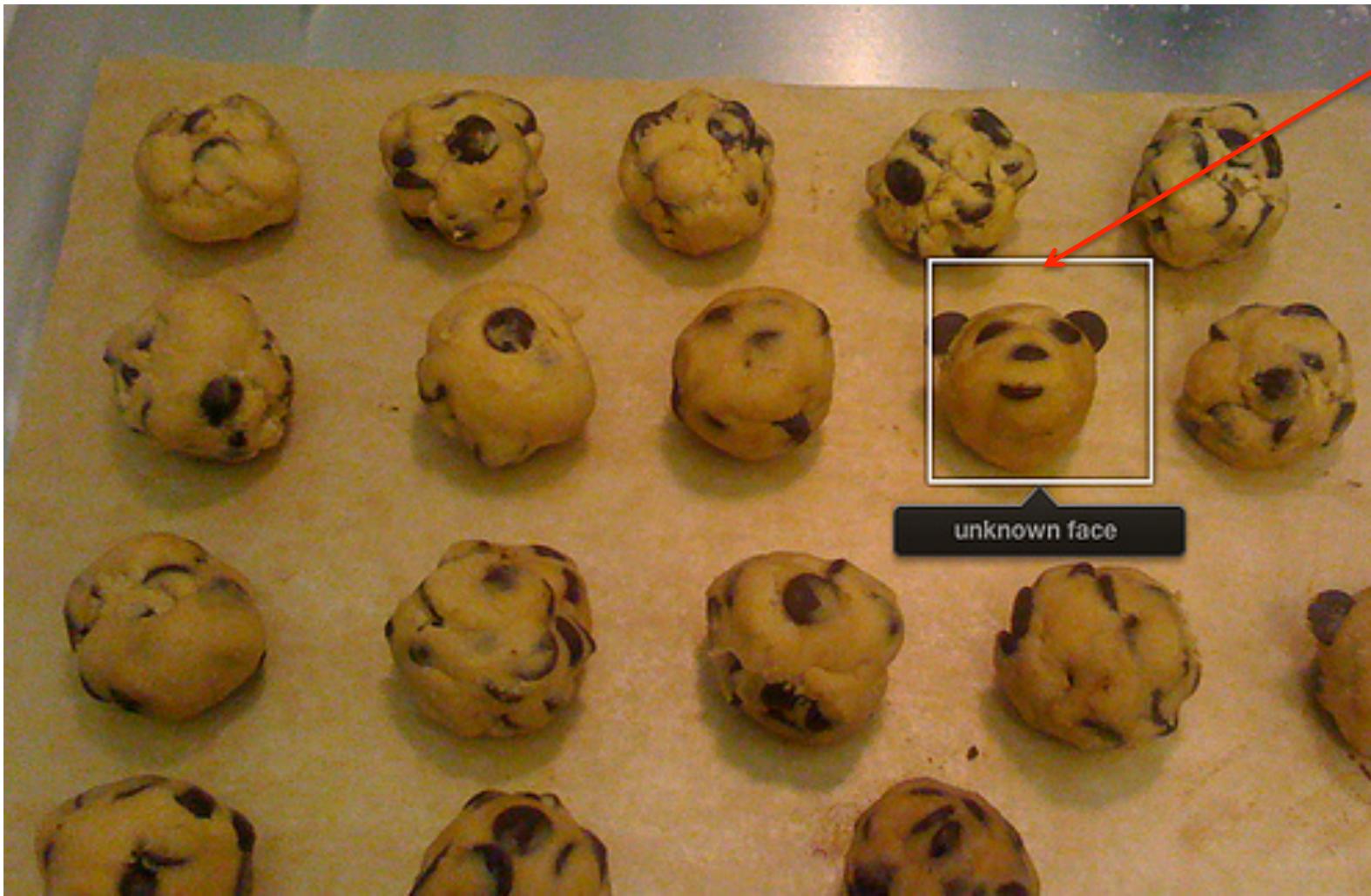


<http://www.apple.com/ilife/iphoto/>

Aplicație: Apple iPhoto

Exemplu
Fals Pozitiv

(detectorul
găsește o față
care nu există)



Reclamă Nikon

"Nikon S60 poate detecta până la 12 fețe."



De ce o problemă grea detectarea facială?

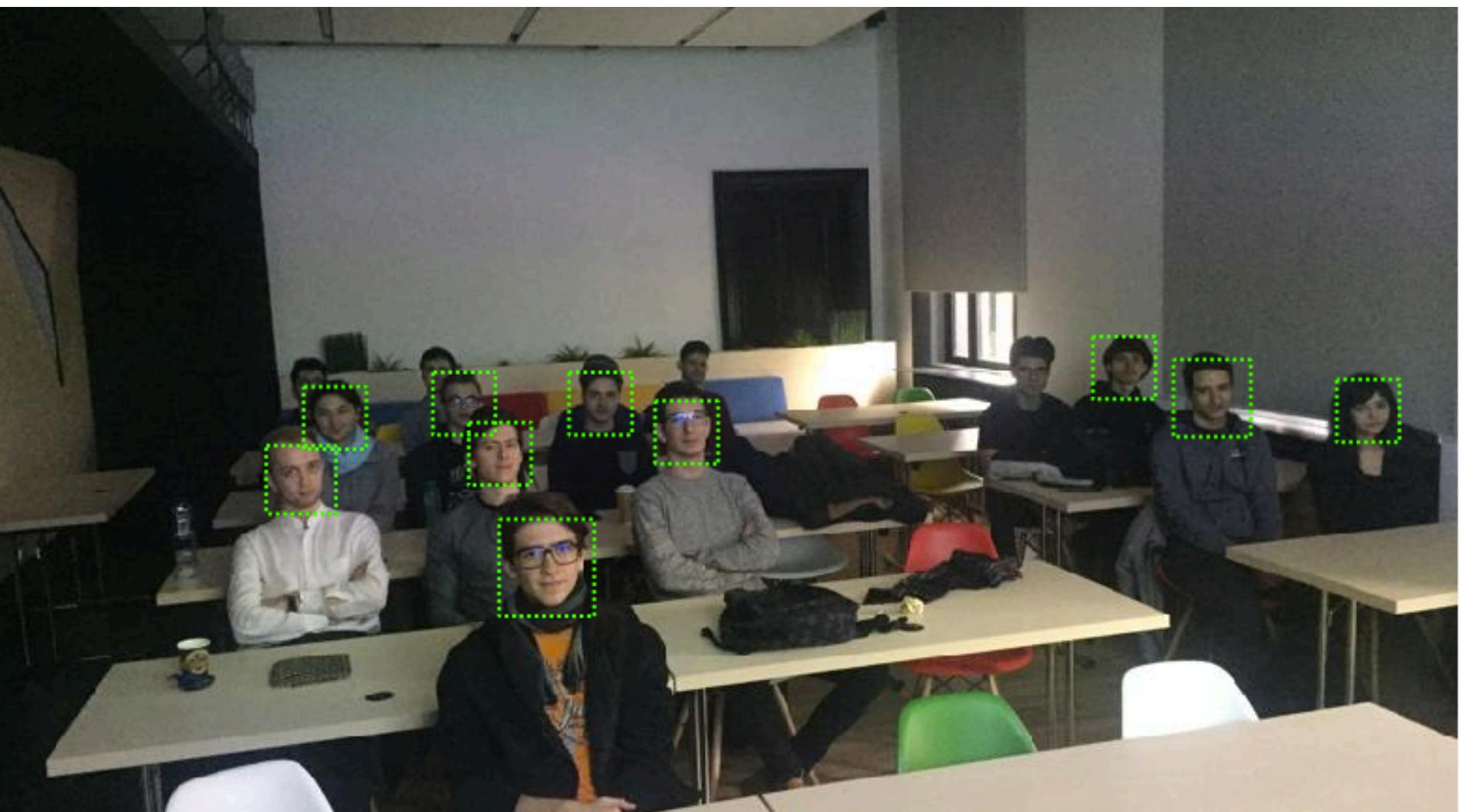
- **postura:** frontal, profil
- **prezență sau absența unor componente structurale:** caracteristici ca barbă, mustață, ochelari pot fi prezente sau nu în imagine + variabilitatea lor în formă culoare, mărime
- **expresii faciale**
- **mascare:** fețele pot fi mascate parțial de alte obiecte. Într-o imagine cu un grup de oameni, unele fețe pot masca alte fețe
- **condițiile în care este făcuta fotografia:** lumina + caracteristicile camerei afectează înfățisarea unei fețe

Abordări pentru detectarea facială

Există multe abordări de succes. Cele mai cunoscute:

1. detector bazat pe metoda glisării ferestrei și HOG (acest curs – tema 5)
2. detectorul Viola-Jones (implementat în Matlab)
3. detector bazat pe metoda vectorilor proprii (eigen-faces)
4. detector bazat pe rețele neuronale de tip deep (master)

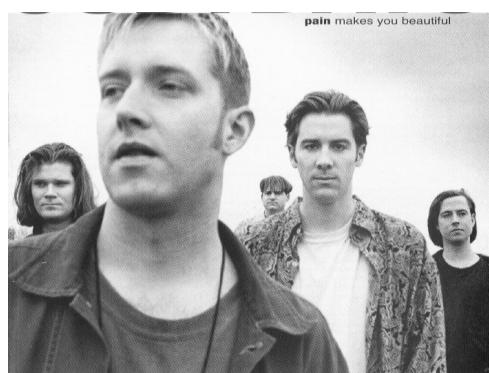






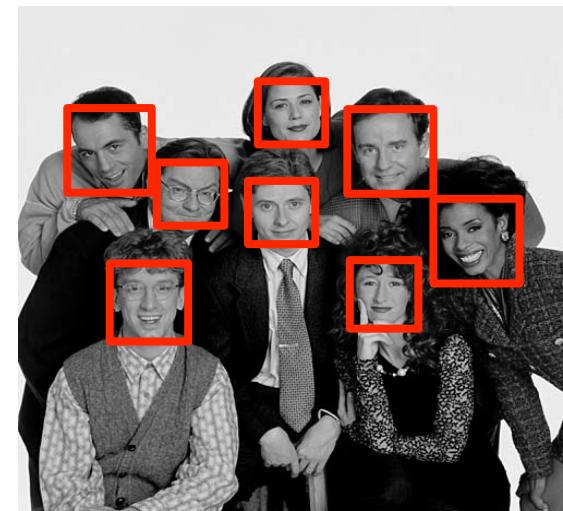
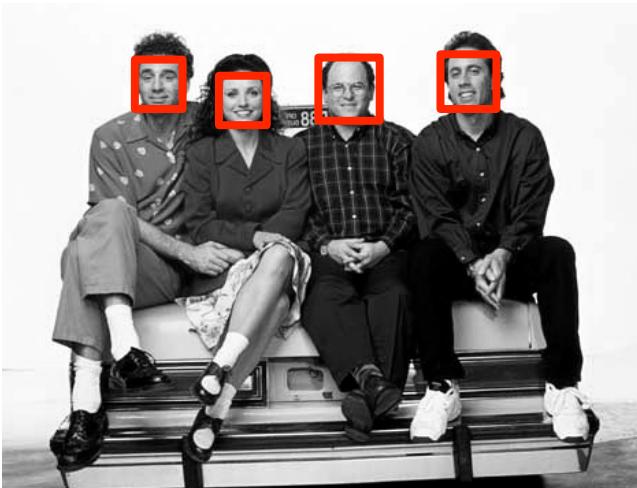
Detectare facială folosind metoda glisării ferestrei și histograme de gradienti orientați

Unde se află în imagine fețele umane?



Detectare facială folosind metoda glisării ferestrei și histograme de gradienți orientați

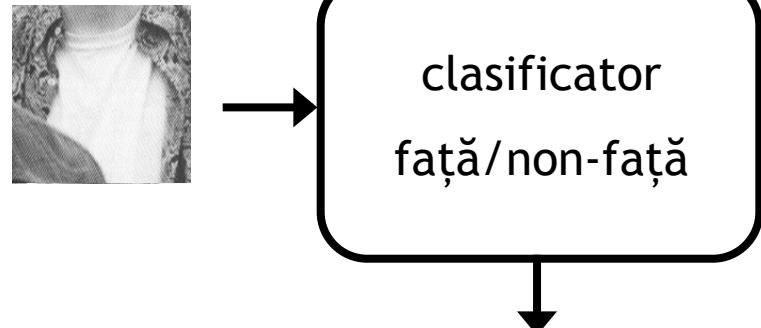
Unde se află în imagine fețele umane?



Localizare la nivel de fereastră dreptunghiulară

Metoda glisării ferestrei

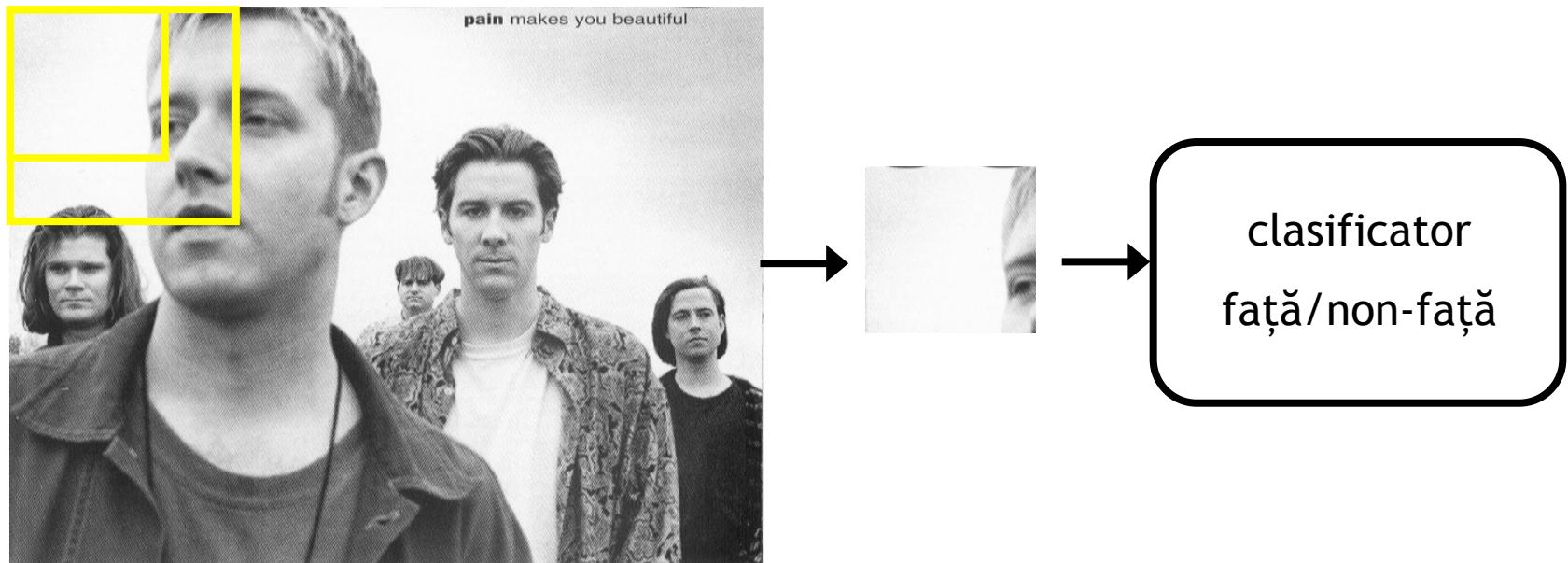
- Localizarea obiectelor la nivel de fereastră:
metoda ferestrei glisante (sliding-window)
 - detectare via clasificare: **clasificator binar** → conține sau nu fiecare fereastră din imagine instanță a clasei de obiecte X (față)?
 - consideră **ferestre poziționate în fiecare pixel**, de mărimi diferite



NDată e față.

Metoda glisării ferestrelor

- Localizarea obiectelor la nivel de fereastră:
metoda ferestrelor glisante (sliding-window)
 - detectare via clasificare: **clasificator binar** → conține sau nu fiecare fereastră din imagine instanță a clasei de obiecte X (față)?
 - consideră **ferestre poziționate în fiecare pixel**, de mărimi diferite



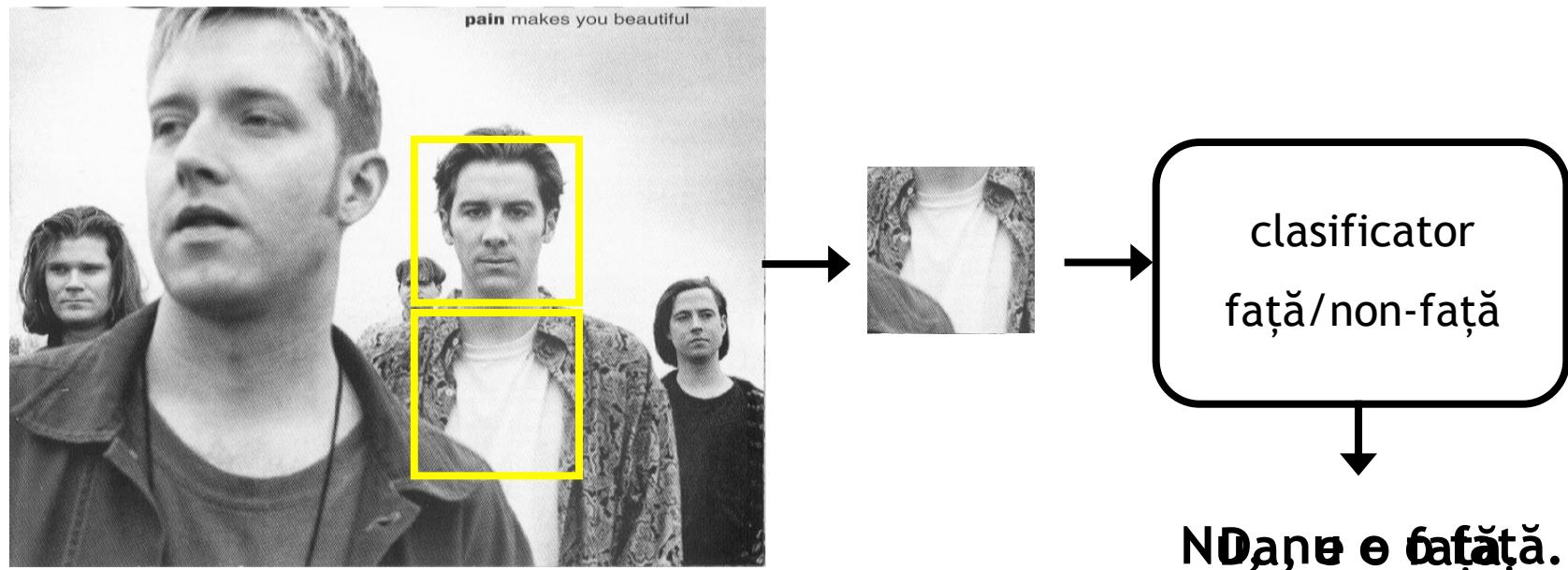
Implementarea în Matlab – tema 5

- scriptul ‘ruleazaProiect.m’ conține întreaga implementare a proiectului
- 14 funcții în total (din care 2 funcții sunt fișiere MEX – funcții codate în C pe care le putem apela în MATLAB. Aceste funcții sunt luate din librăria disponibilă aici: <http://www.vlfeat.org/>)
- 11 funcții complete în întregime
- 3 funcții ce trebuie completate la laborator:
 - obtineDescriptoriExemplePozitive.m
 - obtineDescriptoriExempleNegative.m
 - ruleazaDetectorFacial.m
 - optional: adăugați antrenarea cu exemple puternic negative
- realizați experimente pe baza implementării

Etape în construcția detectorului facial

1. Învățarea unui clasificator

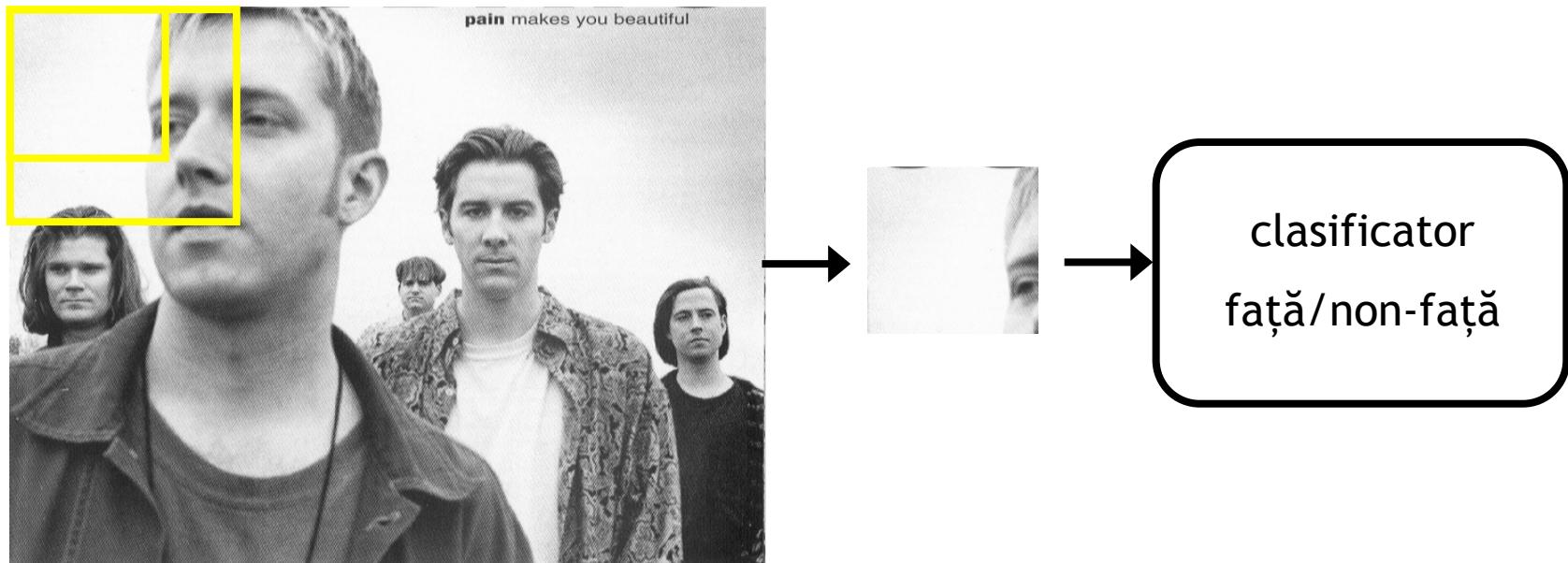
- **clasificator**: funcție care **asignează un scor** unei ferestre pe baza conținutului lor vizual (pixelii din interiorul ei)
- vrem să învățăm un clasificator care distinge ferestrele ce conțin fețe (**ideal, clasificatorul va asigna acestor ferestre un scor > 0**) de ferestrele ce nu conțin fețe (**ideal, clasificatorul va asigna acestor ferestre un scor < 0**)



Etape în construcția detectorului facial

2. Implementarea metodei glisării unei ferestre

- glisarea unei ferestre de la stânga la dreapta și de sus în jos
- clasificarea (asignarea unui scor) fiecărei ferestre pe baza clasificatorului învățat la etapa 1
- localizarea fețelor pe baza scorurilor ferestrelor



Etape în construcția detectorului facial

1. Învățarea unui clasificator

- clasificator: funcție care asignează un scor unei ferestre pe baza conținutului lor vizual (pixelii din interiorul ei)
- vrem să învățăm un clasificator care distinge ferestrele ce conțin fețe de ferestrele ce nu conțin fețe
- `obtineDescriptoriExemplePozitive.m` + `obtineDescriptoriExempleNegative.m`
- optional: antrenare cu exemple puternic negative (pe baza etapei 2)

2. Implementarea metodei glisării unei ferestre

- glisarea unei ferestre de la stânga la dreapta și de sus în jos
- clasificarea (asignarea unui scor) fiecărei ferestre pe baza clasificatorului învățat la etapa 1
- localizarea fețelor pe baza scorurilor ferestrelor
- `ruleazaDetectorFacial.m`

Învățarea unui clasificator

Ingrediente:

1. Multime de învățare/antrenare
 - exemple pozitive - ferestre care conțin fețe
 - exemple negative – ferestre care nu conțin fețe
2. Descrierea conținutului vizual al exemplelor de antrenare
 - descriptor pentru fiecare fereastră pe baza pixelilor din interior
 - descriptor = vector n-dimensional
3. Model de clasificator
 - cum arată funcția pe care o folosim la clasificare?
4. Învățarea parametrilor modelului
 - care sunt parametri optimi pentru care clasificatorul obține rezultate bune?

Mulțime de învățare/antrenare

- avem nevoie de exemple pozitive și negative pentru învățare
- o posibilitate (pentru calculul HOG): imagini în tonuri de gri (pentru imagini RGB folosiți `rgb2gray.m`)



- este nevoie de mii de exemple pentru a putea învăța un clasificator care surprinde variabilitatea fețelor
- o posibilitate: exemple pozitive și negative de dimensiuni 36 x 36 pixeli

Mulțime de învățare/antrenare

- **exemplu pozitive**

- ferestre (36x36 pixeli) ce conțin fețe
- se dau 6713 imagini (36x36) cropate cu fețe
- parametri.numarExempluPozitive = 6713;
- o imagine = o fereastră = 1 exemplu pozitiv



- **exemplu negative**

- ferestre (36x36 pixeli) ce nu conțin fețe
- le veți obține din imagini care nu conțin fețe
- se dau 274 de imagini ce nu conțin fețe
- diverse dimensiuni ($> 36 \times 36$ pixeli)
- o imagine 100x100 pixeli = mii de ferestre

36×36 pixeli = mii de exemple negative

- parametri.numarExempluNegative = 10000;
- $10000/274 \approx 37$ de ferestre din fiecare imagine



Descriptorul HOG al unei ferestre

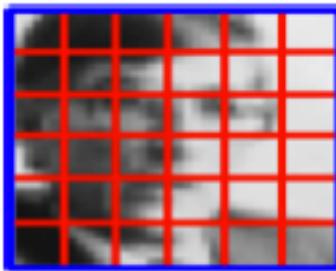
- un exemplu de învățare = o fereastră de dimensiuni 36×36 pixeli
- descriem conținutul vizual al unei ferestre printr-un descriptor HOG
- se împarte o imagine/fereastră în celule HOG: implicit fiecare celulă are dimensiunea 6×6 pixeli



ZOOM



imagine 36×36



împărțire în 6 celule HOG

- pentru fiecare celulă se calculează un descriptor (vi-l dăm noi la laborator)
- descriptorul unei ferestre = 36 celule * lungimea descriptorului unei celule

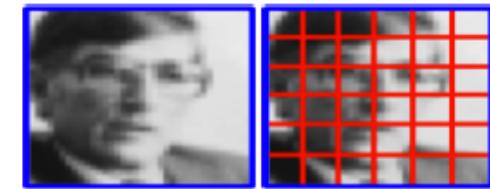
Descriptorul HOG al unei ferestre

Descriptor de lungime 31 calculat pentru o celulă astfel:

- se calculează magnitudinea și orientarea gradientului fiecărui pixel;
- fiecare orientare e asignată uneia dintre 18 orientări cu semn ($0-20^\circ, \dots, 340^\circ-360^\circ$);
- fiecare pixel contribuie la 4 celule (interpolare biliniară);
- obținem o histogramă h_{semn} de dimensiune 18;
- similar obținem o histogramă $h_{farasemn}$ de dimensiune 9 (orientări fara semn $0-20^\circ, 21^\circ-40^\circ, \dots, 161^\circ-180^\circ$);
- bloc = 4 celule; fiecare celulă face parte din 4 blocuri = 4 normalizări L2;
- 4 histograme h_{semn} și 4 histograme $h_{farasemn}$ în funcție de cele 4 normalizări
- facem medie celor 4 histograme h_{semn} și celor 4 $h_{farasemn}$
- obținem un descriptor de 27 de dimensiuni (media aproximează PCA=principal component analysis)
- adăugăm la cele 27 de dimensiuni + 4 coeficienți de normalizare (norma L1)
- total 31 de dimensiuni ([parametri.dimensiuneDescriptorCelula = 31](#))
- detalii aici: <http://www.vlfeat.org/api/hog.html#hog-overview>

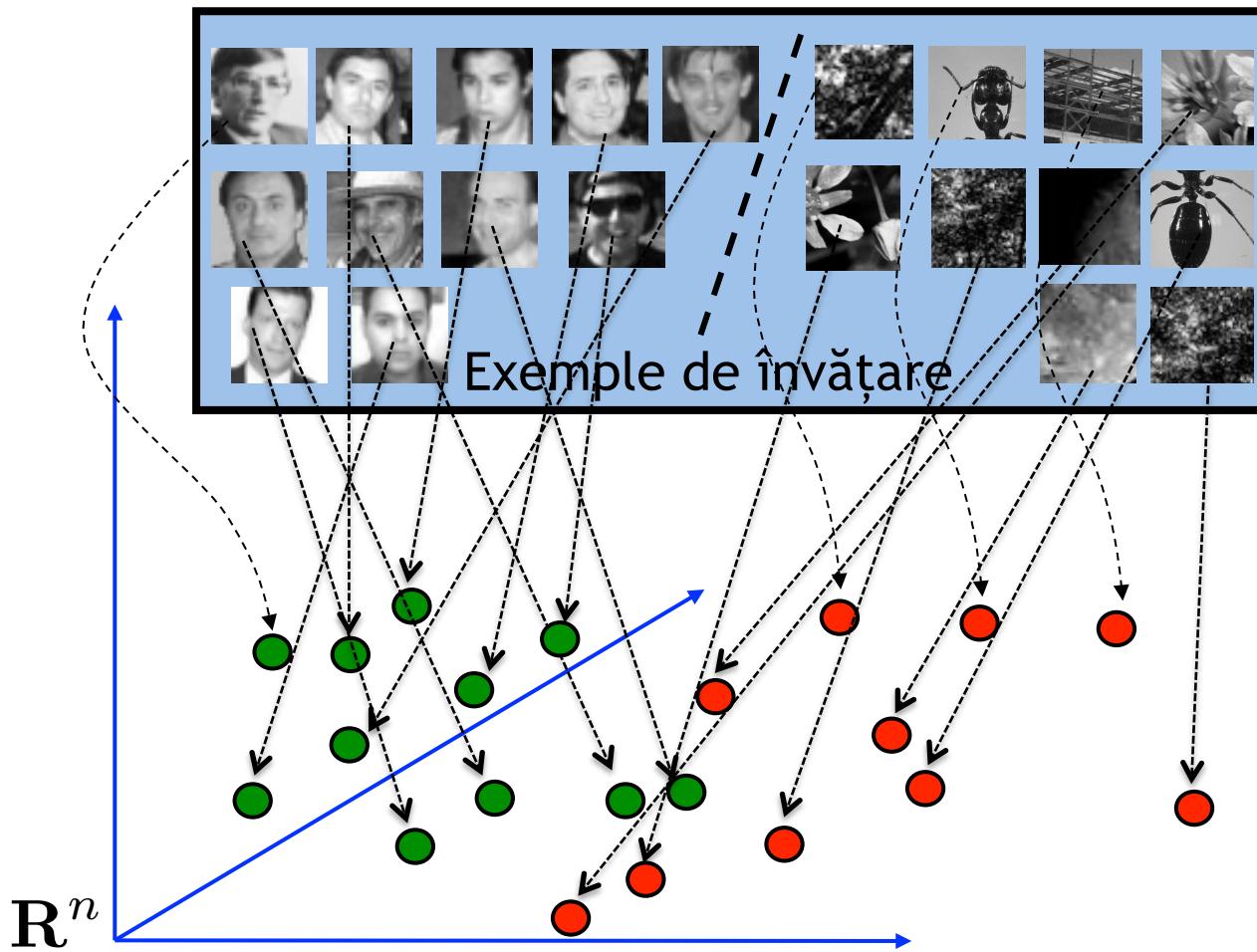
Descriptori pentru exemple de învățare

- Învățarea unui clasificator se realizează pe baza descriptorilor pentru exemple pozitive și exemple negative.
- pentru obținere descriptori exemple pozitive trebuie să codați funcția `obtineDescriptoriExemplePozitive.m` (**6713 exemple pozitive x 1116**)
- pentru obținere descriptori exemple negative trebuie să codați funcția `obtineDescriptoriExempleNegative.m` (**10000 exemple negative x 1116**)
- folosiți funcția `vl_hog` (fișier MEX): împarte o imagine de orice dimensiuni în celule HOG și calculează descriptorii celulelor:
 - 36x36 pixeli -> 6x6 celule HOG (6x6x31)
 - 100x100 pixeli -> 17x17 celule HOG (17x17x31)
- pentru imaginile negative trebuie să construiți voi exemplele negative (extragăți ferestre 36x36 pixeli din imagini negative) pe baza celulelor HOG



Descriptori pentru exemple de învățare

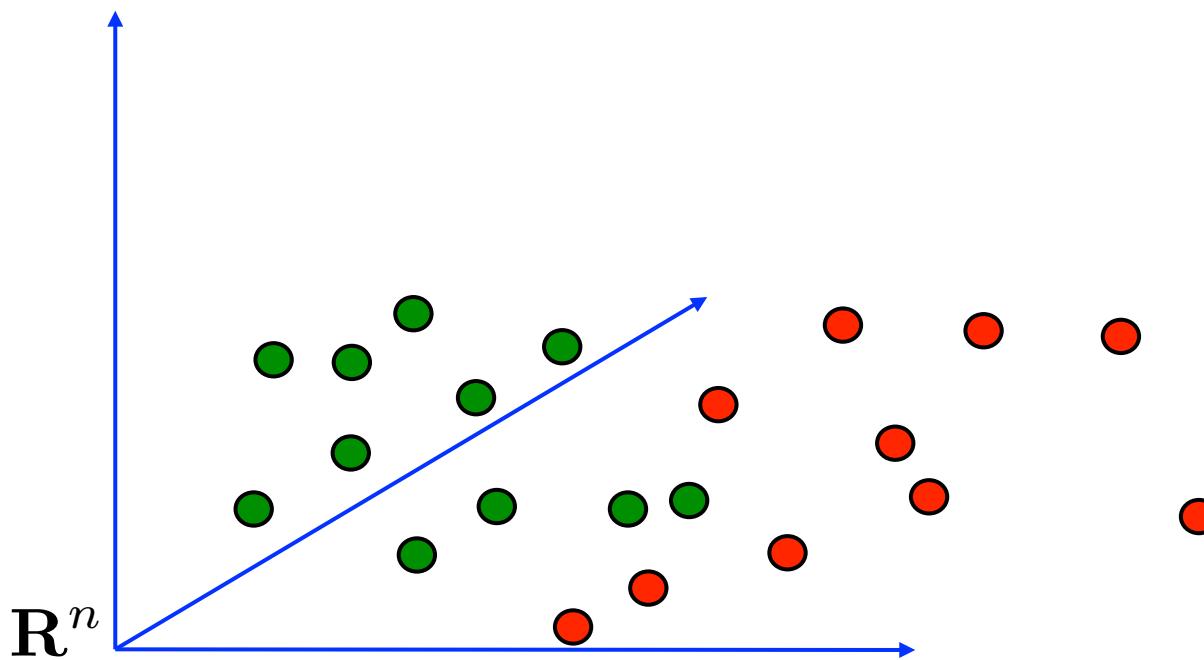
- Învățarea unui clasificator se realizează pe baza descriptorilor pentru **exemple pozitive** și **exemple negative**.



Spațiul descriptorilor

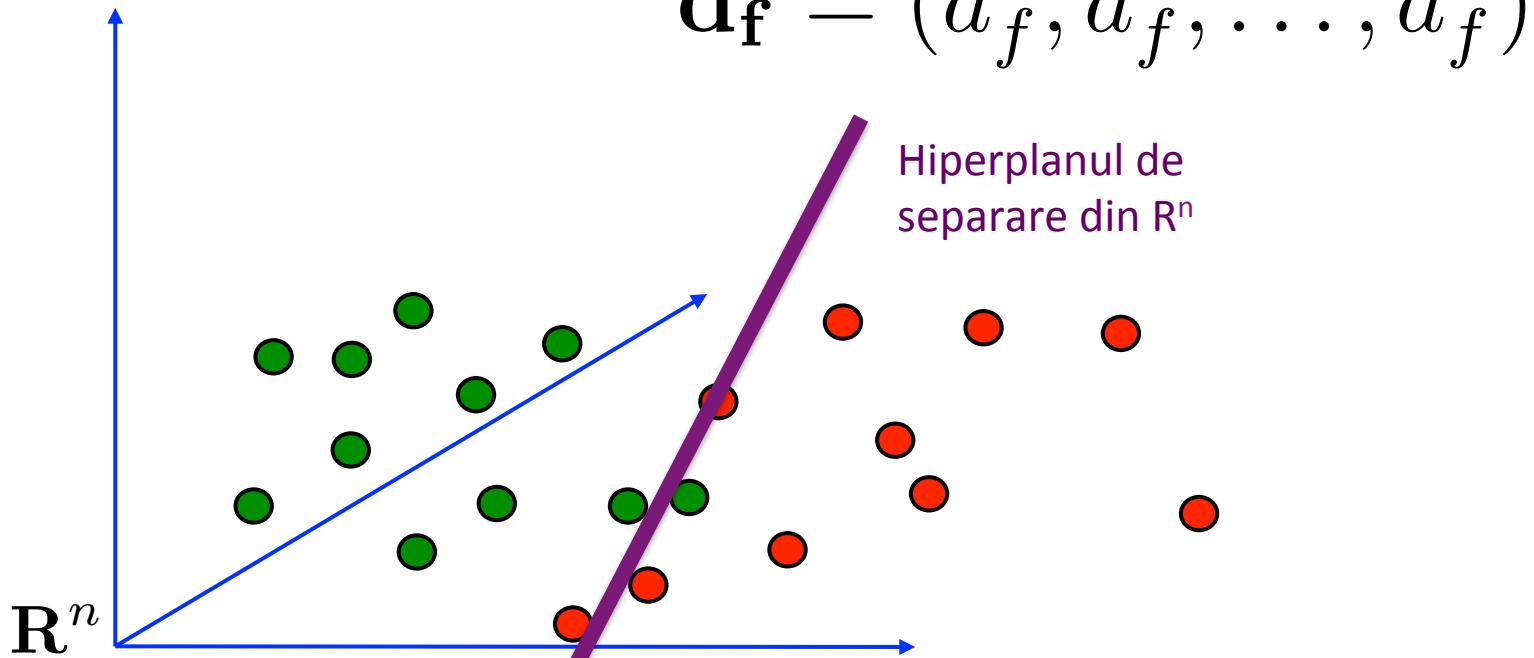
- descriptori pentru **exemple pozitive** și **exemple negative**
- fiecare descriptor \mathbf{d}_f este de dimensiune n

$$\mathbf{d}_f = (d_f^1, d_f^2, \dots, d_f^n)$$



Model de clasificator

- descriptori pentru **exemple pozitive** și **exemple negative**
- fiecare descriptor \mathbf{d}_f este de dimensiune n
- un posibil clasificator: clasificator liniar (hiperplan) care să separe cât mai bine **exemplile pozitive** de **exemplile negative**
- ecuația unui hiperplan în R^n : $\mathbf{w}^t \cdot \mathbf{d}_f + b = 0$
 $\mathbf{d}_f = (d_f^1, d_f^2, \dots, d_f^n)$



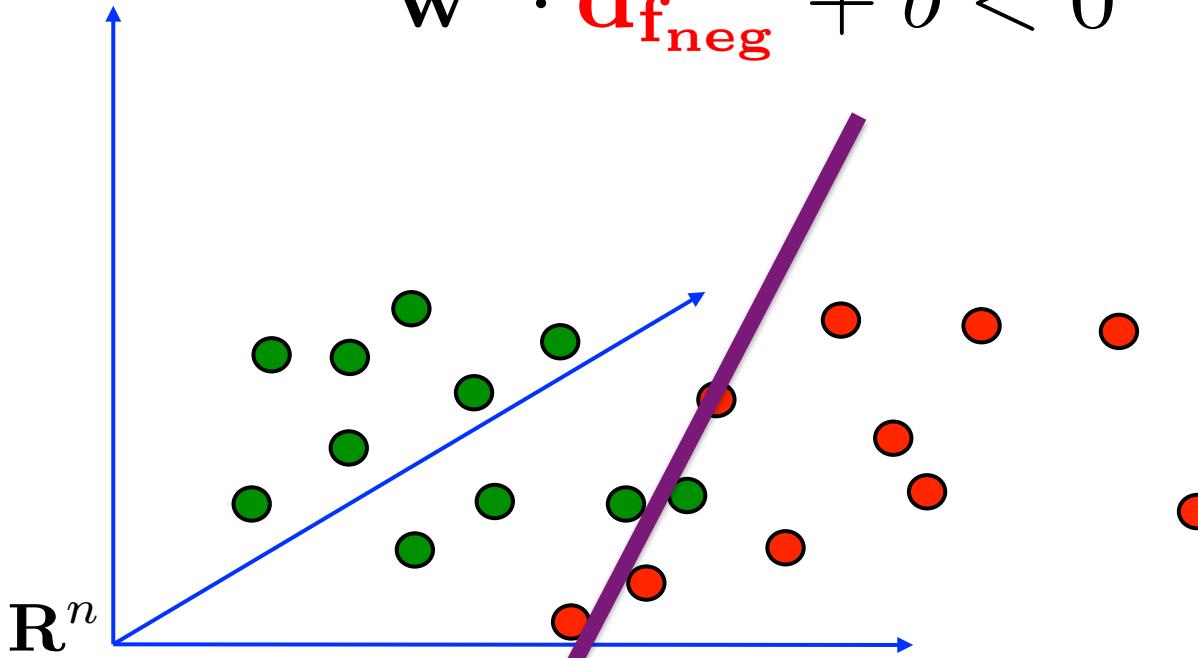
Model de clasificator

- regula de clasificare:
 - pentru **exemple pozitive** vrem să avem:

$$\mathbf{w}^t \cdot \mathbf{d}_{f_{\text{pos}}} + b > 0$$

- pentru exemple negative vrem să avem:

$$\mathbf{w}^t \cdot \mathbf{d}_{f_{\text{neg}}} + b < 0$$



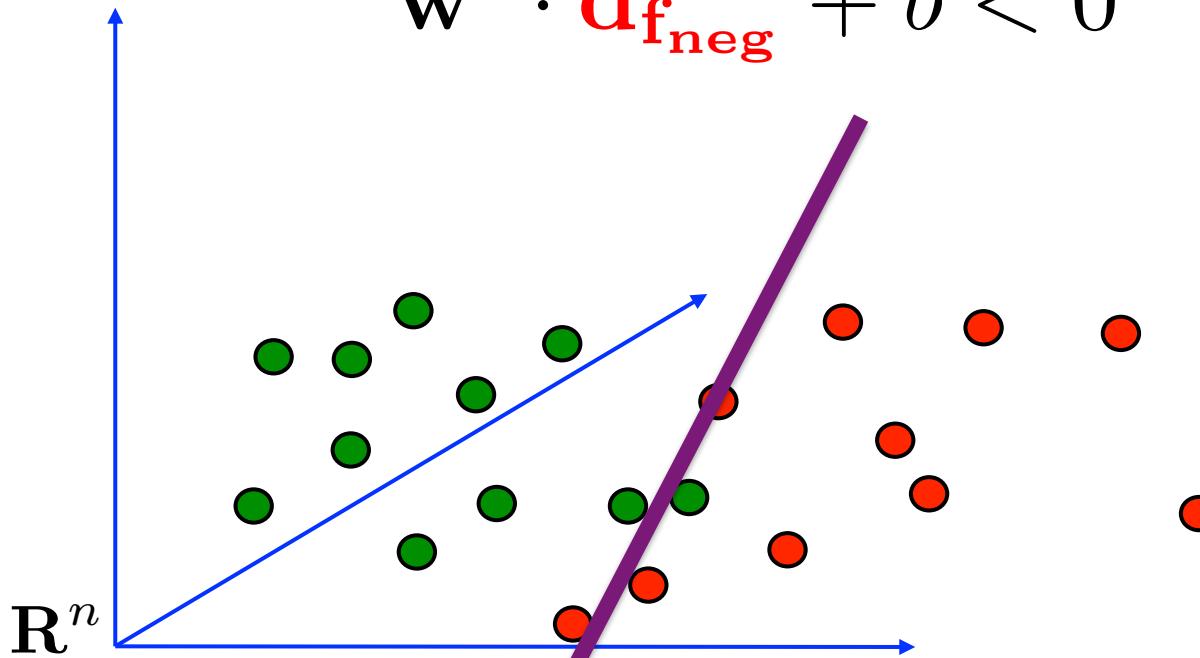
Învățarea parametrilor modelului

- învăță parametri (w, b) ai clasificatorului liniar (= hiperplan) astfel încât
 - pentru **exemple pozitive** vrem să avem:

$$w^t \cdot d_{f_{\text{pos}}} + b > 0$$

- pentru exemple negative vrem să avem:

$$w^t \cdot d_{f_{\text{neg}}} + b < 0$$



Vizualizare performanță clasificator

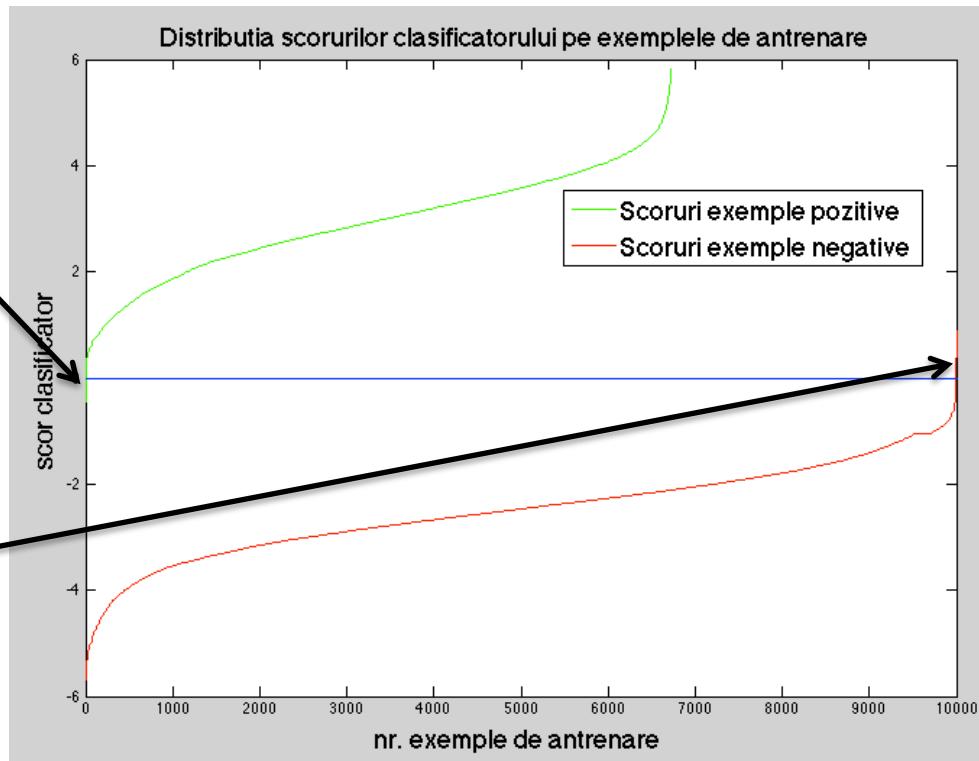
- învață parametrii (w, b) ai clasificatorului liniar (= hiperplan) astfel încât

$$w^t \cdot d_{f_{pos}} + b > 0$$

$$w^t \cdot d_{f_{pos}} + b < 0$$

scoruri < 0
pentru exemple pozitive

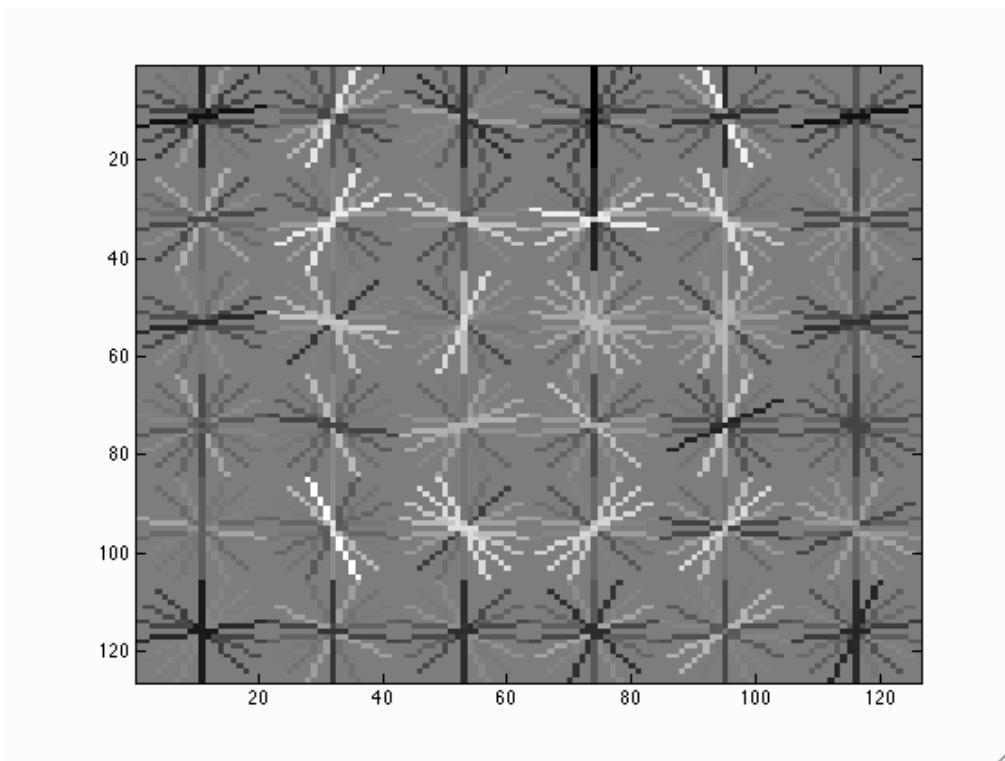
scoruri > 0
pentru exemple negative



- aproape toate exemplele pozitive au un scor > 0
- aproape toate exemplele negative au un scor < 0

Vizualizare template HOG învățat

- plotează media histogramelor h_{farasemn} (dimensiune $9 = 9$ orientări fără semn) cu intensitatea dată de valorile lui w corespunzătoare



Implementarea metodei glisării unei ferestre

1. Pentru o imagine test folosiți funcția `vl_hog` pentru a crea celulele și a calcula descriptorii HOG asociate celulelor



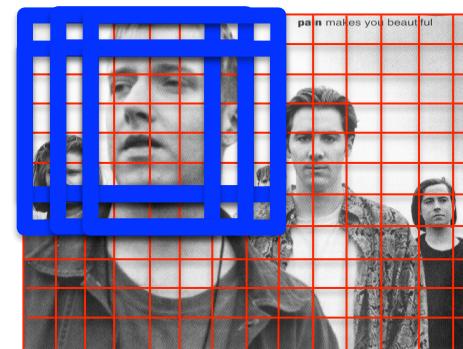
Implementarea metodei glisării unei ferestre

1. Pentru o imagine test calculați descriptorii HOG asociate celulelor
2. Evaluăți toate ferestrele pătratice f de dimensiune 36x36 pixeli
 - evaluarea clasificatorului: $\mathbf{w}^t \mathbf{d}_f + b$
3. Repetați 1+2 pentru mărimi diferite ale imaginii
 - localizarea fețelor din imagine de mărimi diferite



Implementarea metodei glisării unei ferestre

1. Pentru o imagine test calculați descriptorii HOG asociate celulelor
2. Evaluăți toate ferestrele pătratice f de dimensiune 36x36 pixeli
 - evaluarea clasificatorului: $\mathbf{w}^t \mathbf{d}_f + b$
3. Repetați 1+2 pentru mărimi diferite ale imaginii
 - localizarea fețelor din imagine de mărimi diferite



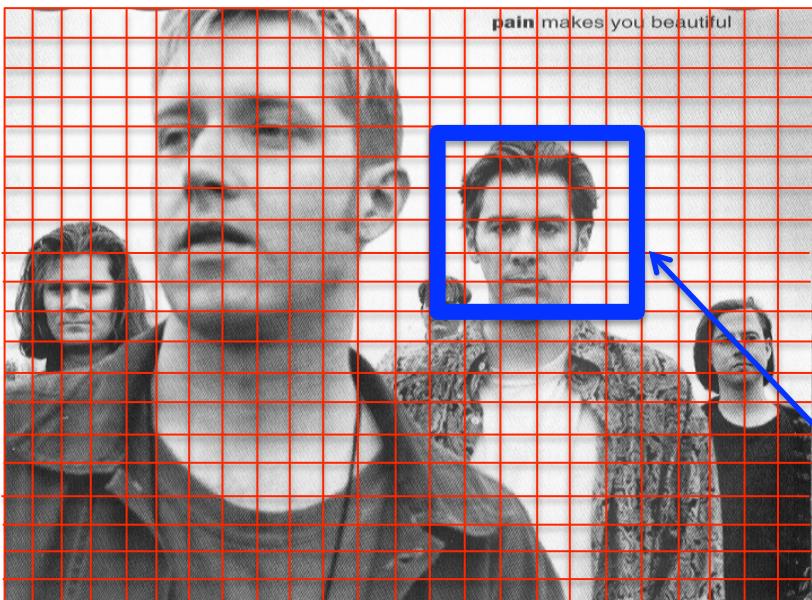
Implementarea metodei glisării unei ferestre

1. Pentru o imagine test calculați descriptorii HOG asociate celulelor
2. Evaluăți toate ferestrele pătratice f de dimensiune 36x36 pixeli
 - evaluarea clasificatorului: $\mathbf{w}^t \mathbf{d}_f + b$
3. Repetați 1+2 pentru mărimi diferite ale imaginii
 - localizarea fețelor din imagine de mărimi diferite
4. Maximele locale ale clasificatorului localizează fețele în imagine



Implementarea metodei glisării unei ferestre

1. Pentru o imagine test calculați descriptorii HOG asociate celulelor
2. Evaluăți toate ferestrele pătratice f de dimensiune 36x36 pixeli
 - evaluarea clasificatorului: $w^t d_f + b$
3. Repetați 1+2 pentru mărimi diferite ale imaginii
 - localizarea fețelor din imagine de mărimi diferite
4. Maximele locale ale clasificatorului localizează fețele în imagine



Maxim local
(fereastra cu scorul cel
mai mare)

Implementarea metodei glisării unei ferestre

1. Pentru o imagine test folosiți funcția `vl_hog` pentru a crea celulele și a calcula descriptorii HOG asociate celulelor
2. Evaluati toate ferestrele pătratice f de dimensiune 36x36 pixeli
 - evaluarea clasificatorului: $\mathbf{w}^t \mathbf{d}_f + b$
3. Repetați 1+2 pentru mărimi diferite ale imaginii
 - localizarea fețelor din imagine de mărimi diferite
4. Maximele locale ale clasificatorului localizează fețele în imagine

Codați toți acești pași în funcția `ruleazaDetectorFacial.m`

La pasul 4 folosiți funcția `eliminaNonMaximele.m` pentru a putea obține maximele locale

Puteți elimina foarte multe ferestre aplicând un threshold (implicit `parametri.threshold = 0`): toate ferestrele cu scorul < threshold să fie eliminate (eliminați toate ferestrele cu scor mic).

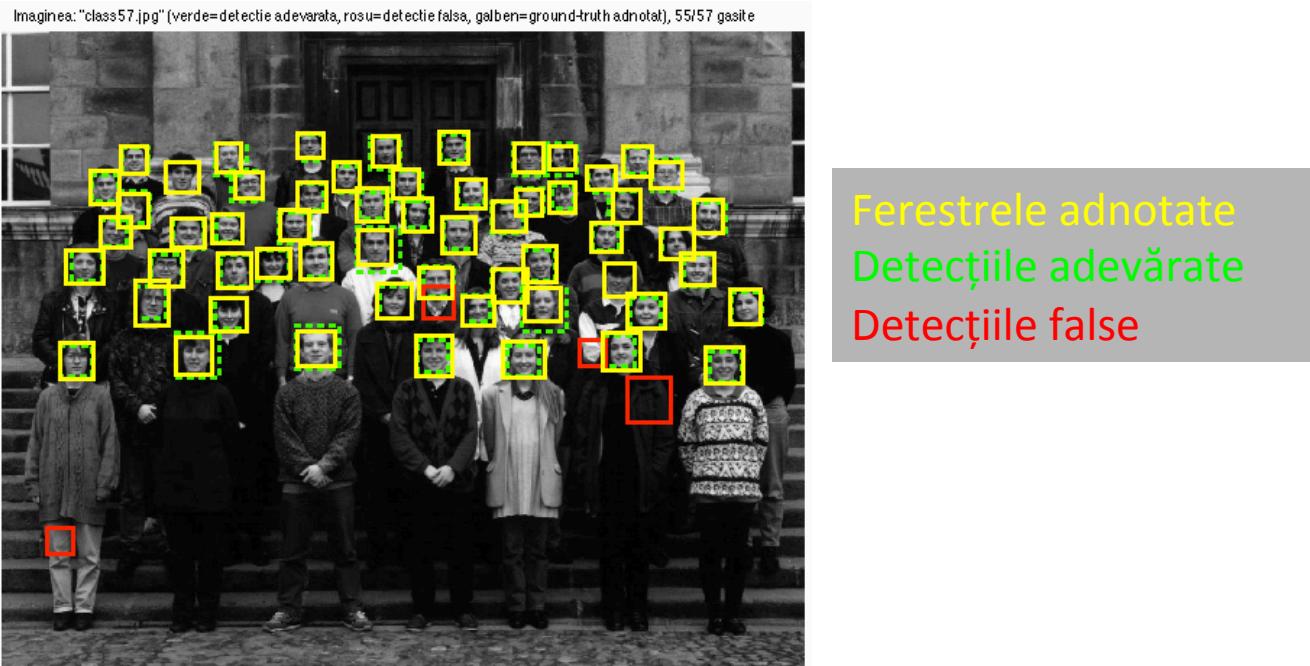
Optional: antrenare cu exemple puternic negative

1. Rulați funcția `ruleazaDetectorFacial.m` pe imaginile negative de antrenare (=274).
 - în aceste imagini toate ferestrele ar trebui să aibă un **scor < 0**
 - **nu există nici o față în aceste imagini**
2. Toate ferestrele din imagini negative pe care le obțineți cu **scor > 0** le adăugați exemplelor negative (implicit =10000).
 - trebuie să modificați funcția `ruleazaDetectorFacial.m` pentru a returna și descriptorul (exemplul de antrenare)
3. Reantrenați clasificatorul cu noile exemple negative adăugate de la 2

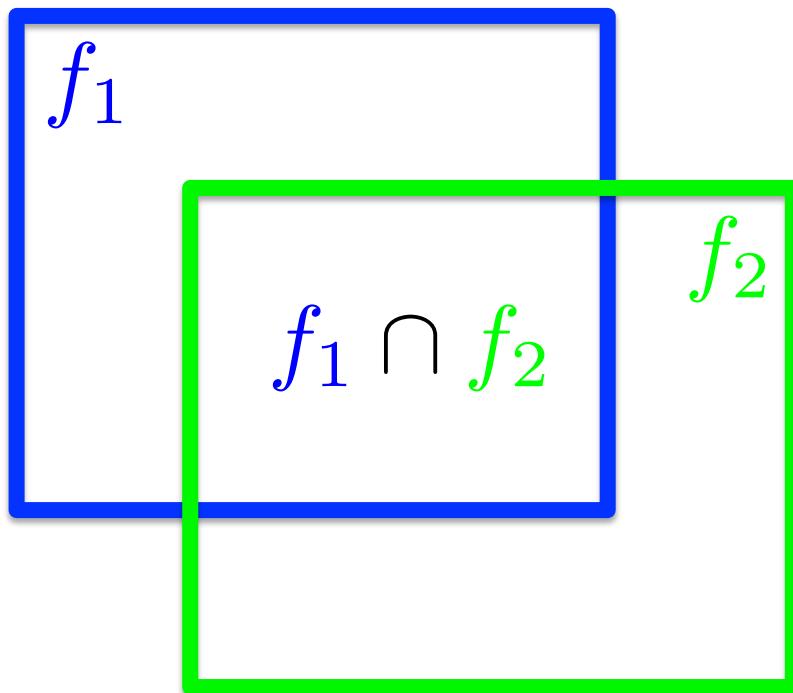
Evaluare – detectii adevărate și false

Cum evaluăm performanța unui detector facial?

- comparăm ceea ce am obținut (ferestrele de maxim local se numesc detectii) cu ceea ce trebuia să obținem (ferestre adnotate conținând fețe)
- detectii adevărate** = detectii care se suprapun (intersecție/reuniune) > 0.3 cu o fereastră adnotată
- detectii false** = detectii care nu se suprapun > 0.3 cu o fereastră adnotată sau există o altă detectie de scor mai mare care se suprapune > 0.3 cu fereastra adnotată (se penalizează multiple detectii ale aceleiași fețe)



Suprapunerea a două ferestre



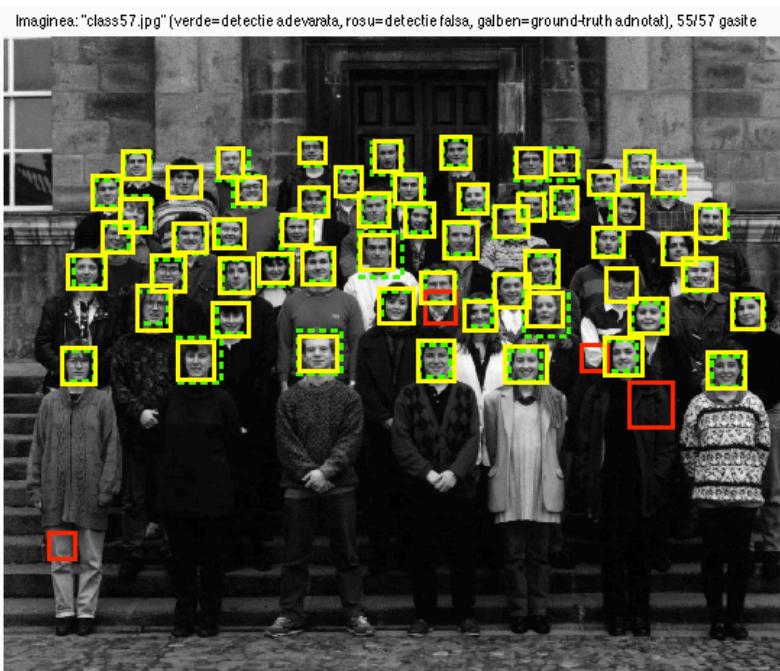
$$suprapunere(f_1, f_2) = \frac{aria(f_1 \cap f_2)}{aria(f_1 \cup f_2)}$$

$$suprapunere(f_1, f_2) = \frac{aria(f_1 \cap f_2)}{aria(f_1) + aria(f_2) - aria(f_1 \cap f_2)}$$

Evaluare – recall și precizie pe o imagine

Cum evaluăm performanța unui detector facial?

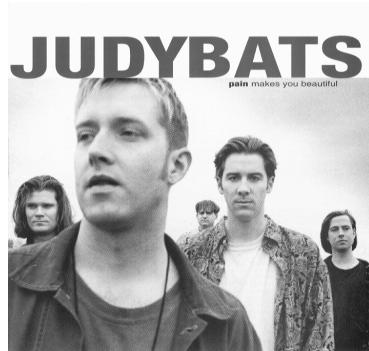
- comparăm ceea ce am obținut (ferestrele de maxim local se numesc detectii) cu ceea ce trebuia să obținem (ferestre adnotate conținând fețe)
- recall = câte fețe am găsit din cele adnotate =
$$\frac{\#\text{detectii adevarate}}{\#\text{ferestre adnotate}}$$
- precizie = câte detectii sunt adevărate =
$$\frac{\#\text{detectii adevarate}}{\#\text{detectii adevarate} + \#\text{detectii false}}$$



Ferestrele adnotate
Detectiile adevărate
Detectiile false
recall = $55/57 = 0.965$
precizie = $55/59 = 0.932$

Compararea a doi algoritmi

- se face pe numite baze de date pentru care se cunoaște ceea ce trebuia să întoarcă algoritmul de detectare facială (imaginile sunt adnotate)

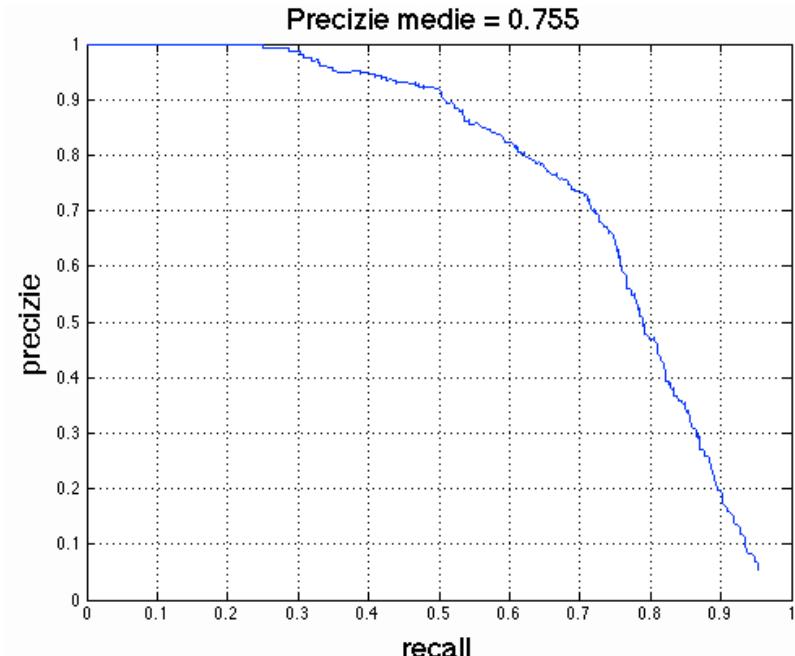


- se compară în funcție de precizie și recall sumarizate în graficul precizie-recall care oferă un număr: precizia medie a detectorului de fețe

Evaluare – recall și precizie pe toate imaginile

Se construiește graficul precizie-recall în felul următor:

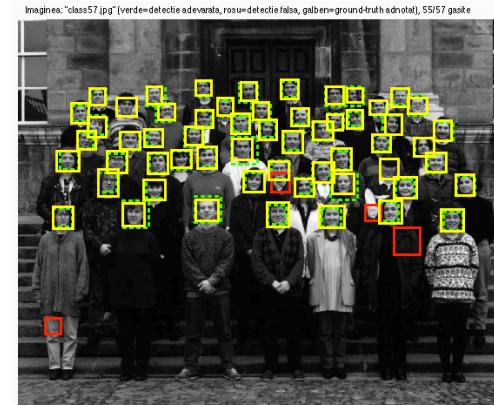
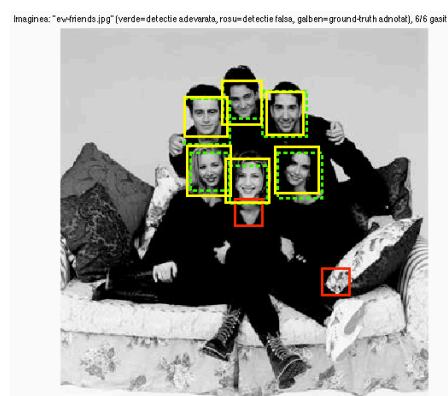
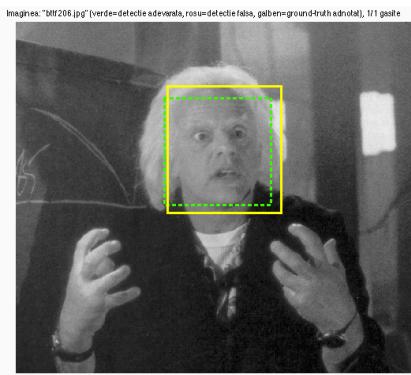
1. se rulează detectorul facial pe toate imaginile și se obține pentru fiecare imagine o mulțime de detectii;
2. se stabilește pentru fiecare imagine (pentru care avem adnotări) care sunt detectiile adevărate și cele false;
3. se ordonează descrescător toate detectiile în funcție de scorul lor;
4. pentru un threshold care descrește de la scorul cel mai mare la scorul cel mai mic se calculează precizia și recall-ul pentru toate detectiile cu scorul $>$ threshold



Valorea precizie medie sumarizează graficul precizie-recall prin aria de sub grafic
Pe baza acestei valori evaluăm performanța unui algoritm

Vizualizare rezultate

- Pentru seturi de date cu adnotări (CMU+MIT) folosiți funcția vizualizeazaDetectiiInImagineCuAdnotari.m ([parametri.existaAdnotari = 1](#))



- Pentru seturi de date fără adnotări (imaginele de la curs) folosiți funcția vizualizeazaDetectiiInImagineFaraAdnotari.m ([parametri.existaAdnotari = 0](#))



Experimente

Realizați experimente (vreți să obțineți o valoare precizie medie cât mai bună) după ce ați codat cele 3 funcții:

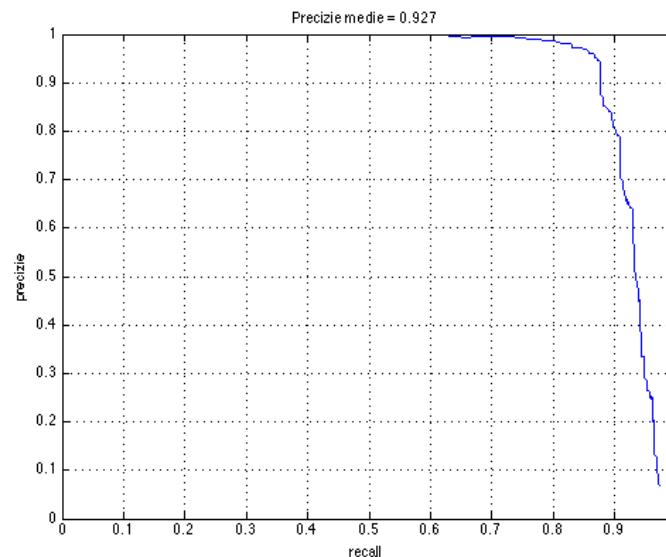
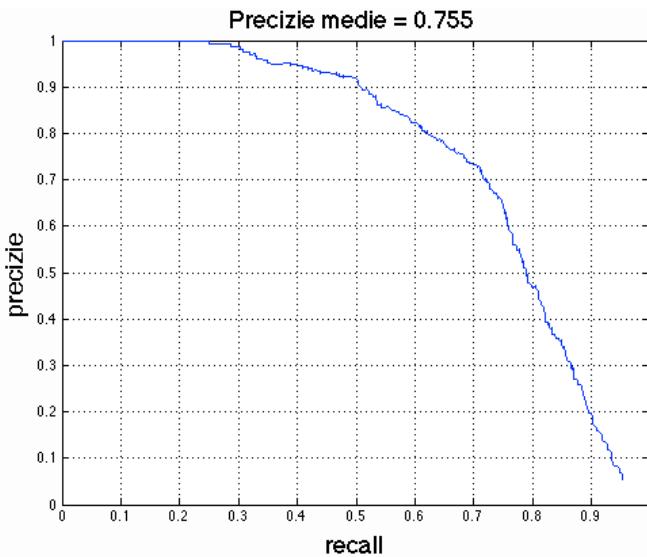
- `obtineDescriptoriExemplePozitive.m`
- `obtineDescriptoriExempleNegative.m`
- `ruleazaDetectorFacial.m`
- opțional: adăugați antrenarea cu exemple puternic negative

Parametri:

- `dimensiuneFereastra` = 36 (fix)
- `dimensiuneCelulaHOG` = 6 (implicit) – încercați alte valori
- `dimensiuneDescriptorCelula` = 31 (fix)
- `numarExemplePozitive` = 6713 (implicit) – creșteți numărul exemplelor
- `numarExempleNegative` = 10000 (implicit) – încercați alte valori
- `threshold` = 0 (implicit) – încercați alte valori (valori mari: recall mic, precizia mare; valori mici: recall mare, precizia mică)
- salvați `w, b` (parametri.w și parametri.b)

Compararea a doi algoritmi

- se face pe numite baze de date pentru care se cunoaște ceea ce trebuia să întoarcă algoritmul de detectare facială (imaginile sunt adnotate)
- se compară în funcție de precizie, recall sumarizate în graficul precizie-recall care oferă un precizia medie



- algoritmul din dreapta este mai bun decât algoritmul din stânga

Demo

<https://www.sighthound.com/products/cloud>

versus

algoritmul de la laborator