Widzenie Komputerowe - Recognition - Face detection

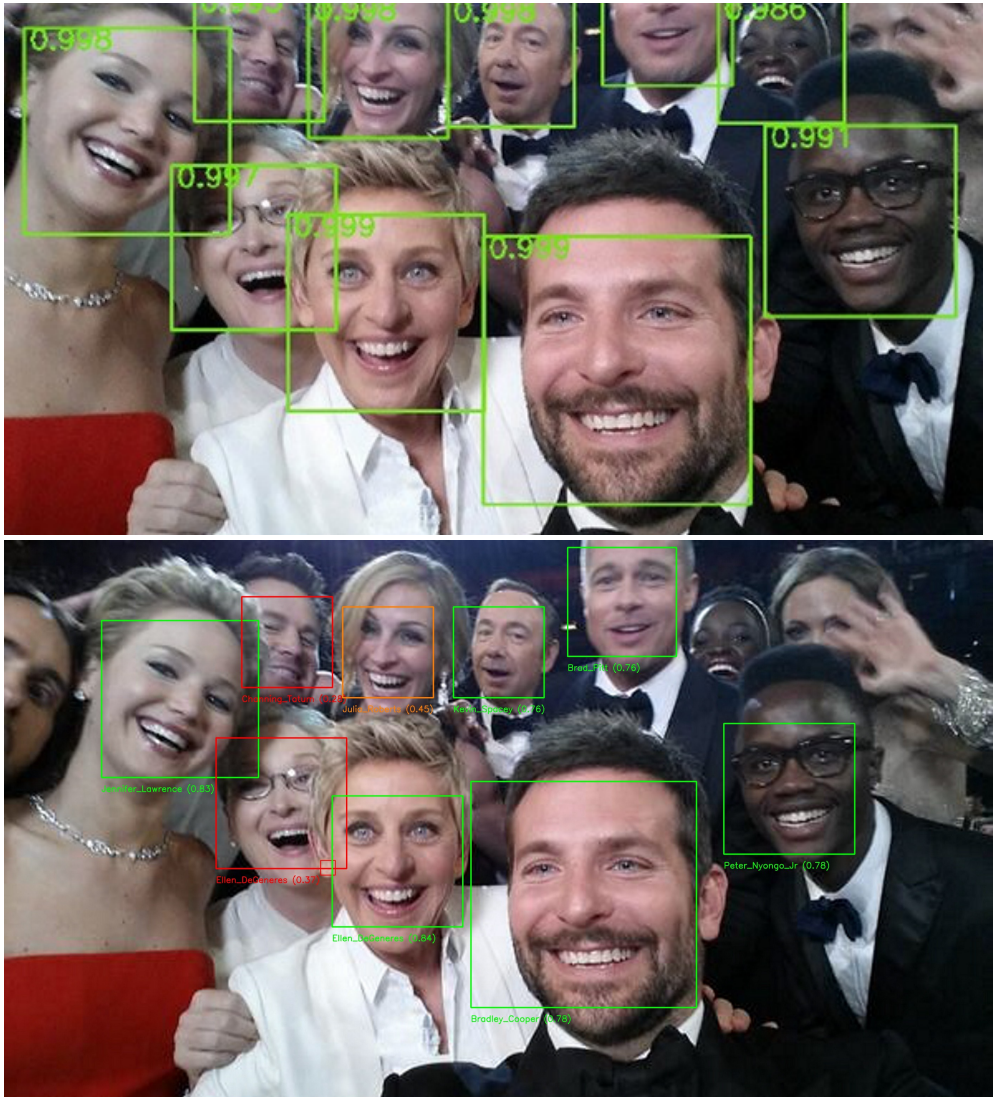
Wykład 9.

Magdalena Mazur-Milecka

Katedra Inżynierii Biomedycznej, WETI, PG

30 kwietnia 2020

Technologie Face Detection



- Detekcja twarzy - **face detection**
- Identyfikacja twarzy (tożsamości) - **face recognition**

Face ID, medium.com
Face recognition with OpenCV and dlib, www.pyimagesearch.com

Zastosowania

- biometria - rozpoznanie tożsamości
- systemy ochrony i nadzoru - rozpoznanie tożsamości, kontrola kierowców
- rozpoznanie emocji - analiza stanu zdrowia lub samopoczucia użytkownika
- rozrywka - efekty na komunikatorach wideo
- marketing - dostosowanie reklam, rekomendacji do płci, wieku itp, zakupy on-line
- fotografia - autofocus twarzy, detektor uśmiechu

Topowe technologie

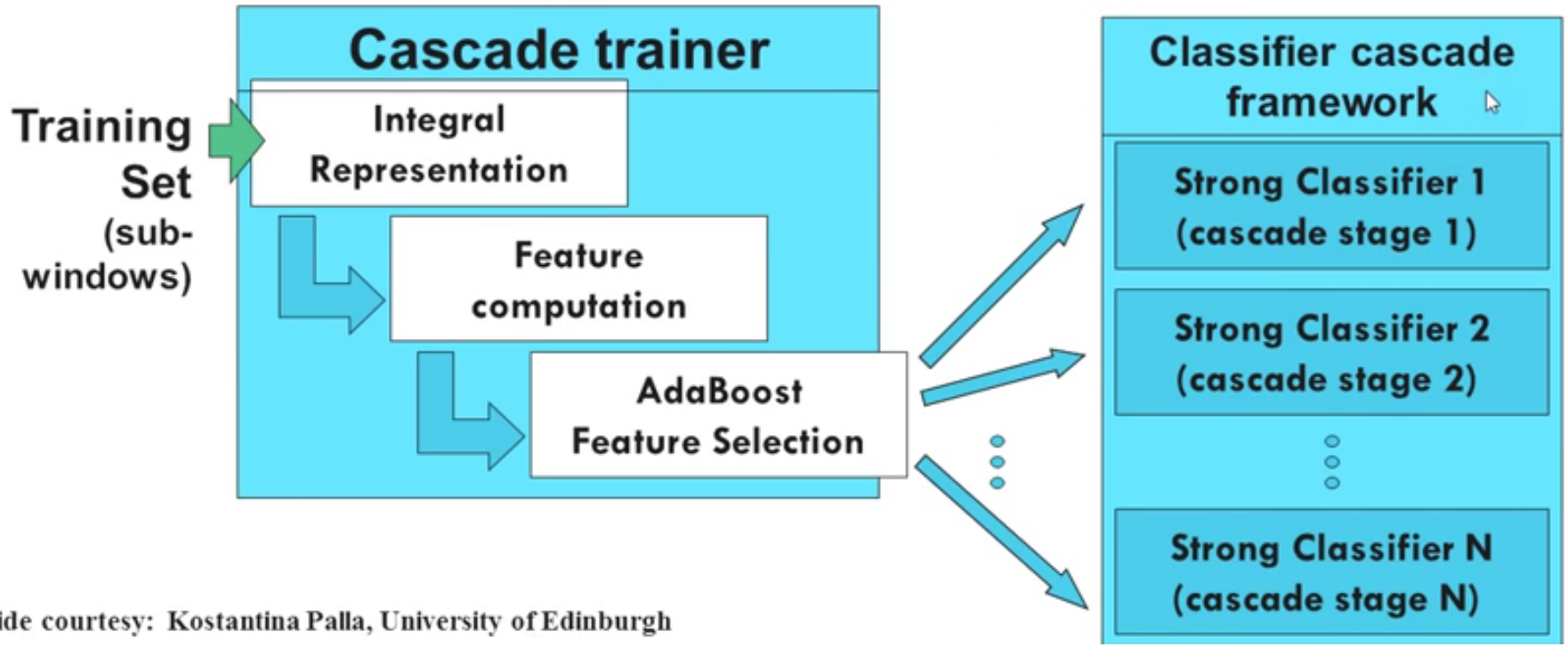
- DeepFace - Facebook 2014 (dokładność rozpoznania nieznanych twarzy 97,25 przypadków, podczas gdy współczynnik ten wynosi u ludzi 97,56)
- VGGFace - 2015
- FaceNet - Google 2015 - Google Photos
- OpenFace - open-source (Google, Intel) - 2016
- Rekognition - Amazon 2018
- i wiele, wiele innych

Spis treści:

Algorytm Viola Jones

Zagadnienia:

1. Wybór cech Haar'a,
2. Tworzenie obrazu integralnego (scałkowanego),
3. Trening AdaBoost,
4. Klasyfikator kaskadowy.



lide courtesy: Kostantina Palla, University of Edinburgh

Algorytm Viola Jones

Zagadnienia:

1. Wybór cech Haar'a,
2. Tworzenie obrazu integralnego (scałkowanego),
3. Trening AdaBoost,
4. Klasyfikator kaskadowy.

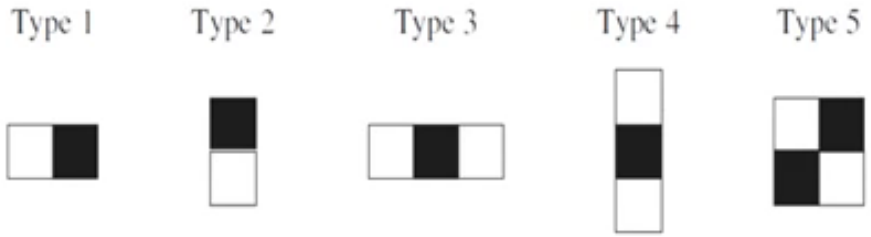
Algorytm Viola Jones może być wykorzystany do rozpoznawania różnych obiektów, jednak został stworzony w celu rozpoznawania twarzy.
Celem jest rozpoznanie twarzy od nie-twarzy. W taki sposób tworzony jest zbiór treningowy.

Cechy Haar'a

Cechy Haar'a - informacja o zmianie wartości kontrastu pomiędzy prostokątnymi grupami pikseli. Sąsiednie grupy o podobnej wariancji kontrastu (jasne lub ciemne) tworzą cechę Haar'a.

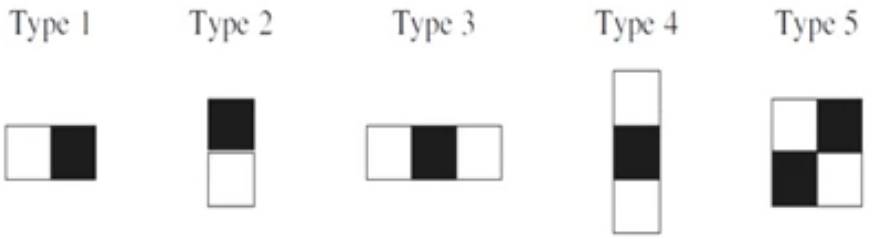
Cechy Haar'a

Cechy Haar'a - informacja o zmianie wartości kontrastu pomiędzy prostokątnymi grupami pikseli. Sąsiednie grupy o podobnej wariancji kontrastu (jasne lub ciemne) tworzą cechę Haar'a.



Cechy Haar'a

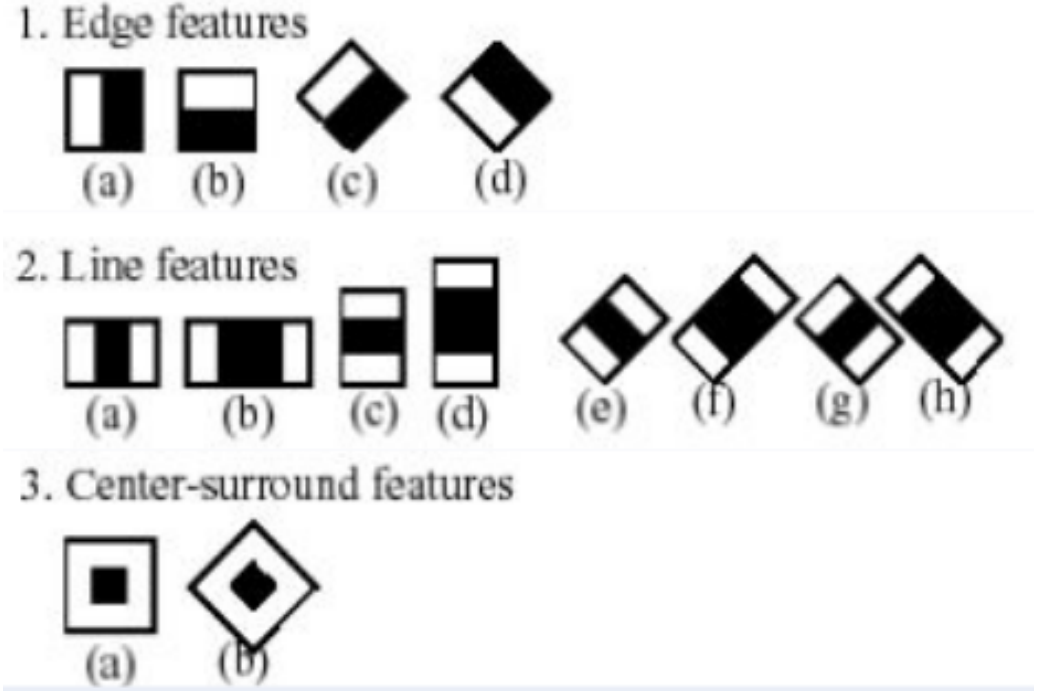
Cechy Haar'a - informacja o zmianie wartości kontrastu pomiędzy prostokątnymi grupami pikseli. Sąsiednie grupy o podobnej wariancji kontrastu (jasne lub ciemne) tworzą cechę Haar'a.



Cechy Haar'a

Cechy Haar'a - informacja o zmianie wartości kontrastu pomiędzy prostokątnymi grupami pikseli. Sąsiednie grupy o podobnej wariancji kontrastu (jasne lub ciemne) tworzą cechę Haar'a.

Cechy te można skalować.



Cechy Haar'a

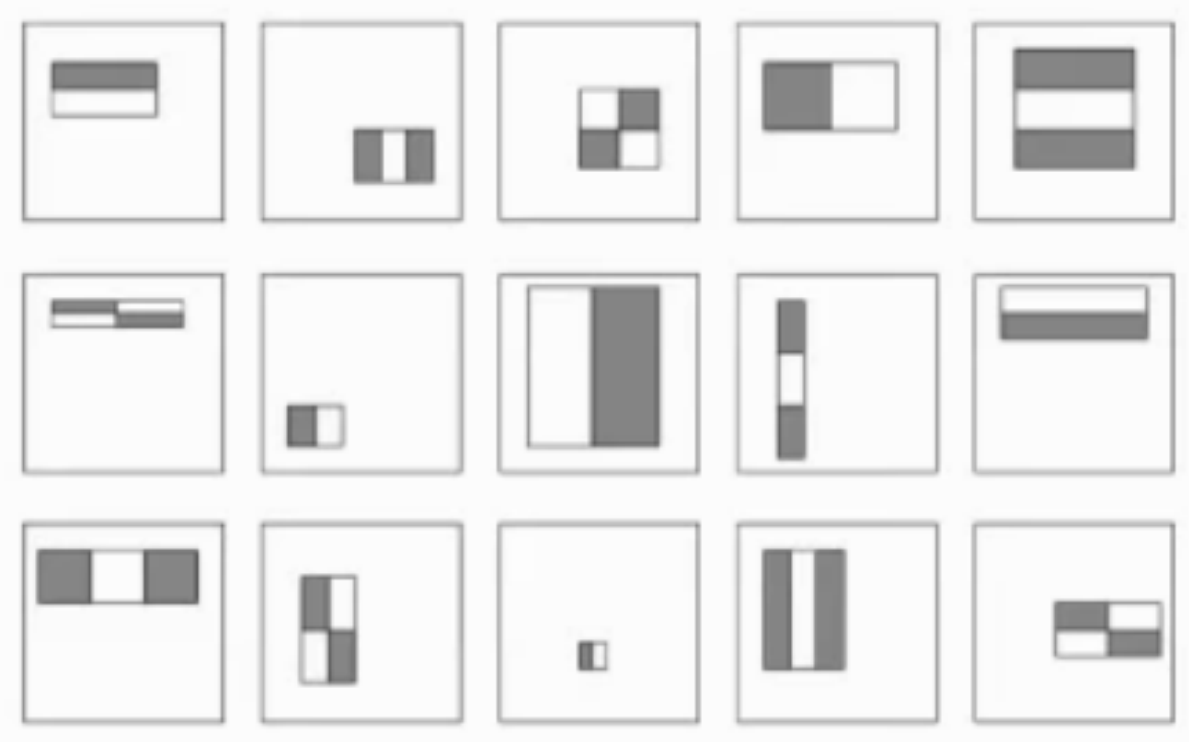
Cechy Haar'a - informacja o zmianie wartości kontrastu pomiędzy prostokątnymi grupami pikseli. Sąsiednie grupy o podobnej wariancji kontrastu (jasne lub ciemne) tworzą cechę Haar'a. Wartość każdej cechy to różnica sumy pikseli w miejscu białego prostokątna i sumy pikseli w miejscu czarnego prostokąta - wartość skalarna.

Cechy Haar'a

- Cechy są skalowane i obliczane w każdym miejscu obrazu,
- Algorytm Viola Jones używa startowego okna o wymiarach 24x24,

Cechy Haar'a

- Cechy są skalowane i obliczane w każdym miejscu obrazu,
- Algorytm Viola Jones używa startowego okna o wymiarach 24x24,
- Uwzględniając wszystkie typy cech Haar'a oraz ich położenia i skale, dla podstawowego okna obliczanych jest około 160 000 cech,



Cechy Haar'a

- Cechy są skalowane i obliczane w każdym miejscu obrazu,
- Algorytm Viola Jones używa startowego okna o wymiarach 24x24,
- Uwzględniając wszystkie typy cech Haar'a oraz ich położenia i skale, dla podstawowego okna obliczanych jest około 160 000 cech,
    1. Zbyt dużo danych, nie wszystkie niosą informację.

Cechy Haar'a

- Cechy są skalowane i obliczane w każdym miejscu obrazu,
- Algorytm Viola Jones używa startowego okna o wymiarach 24x24,
- Uwzględniając wszystkie typy cech Haar'a oraz ich położenia i skale, dla podstawowego okna obliczanych jest około 160 000 cech,
    1. Zbyt dużo danych, nie wszystkie niosą informację.
    2. Zbyt czasochłonne obliczenia.

Cechy Haar'a

- Cechy są skalowane i obliczane w każdym miejscu obrazu,
- Algorytm Viola Jones używa startowego okna o wymiarach 24x24,
- Uwzględniając wszystkie typy cech Haar'a oraz ich położenia i skale, dla podstawowego okna obliczanych jest około 160 000 cech,
    1. Zbyt dużo danych, nie wszystkie niosą informację. Rozwiązanie: kaskada Adaboost;
    2. Zbyt czasochłonne obliczenia.

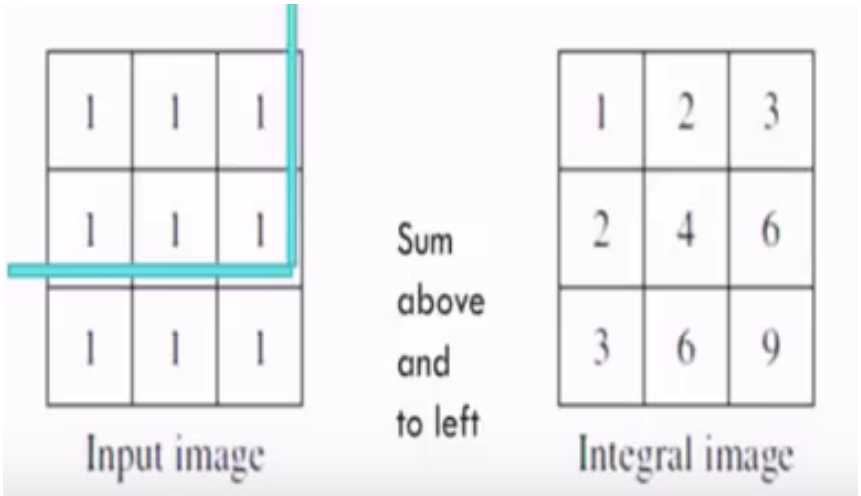
Cechy Haar'a

- Cechy są skalowane i obliczane w każdym miejscu obrazu,
- Algorytm Viola Jones używa startowego okna o wymiarach 24x24,
- Uwzględniając wszystkie typy cech Haar'a oraz ich położenia i skale, dla podstawowego okna obliczanych jest około 160 000 cech,
    1. Zbyt dużo danych, nie wszystkie niosą informację.
       Rozwiązanie: kaskada Adaboost;
    2. Zbyt czasochłonne obliczenia.
       Rozwiązanie: Obraz integralny.

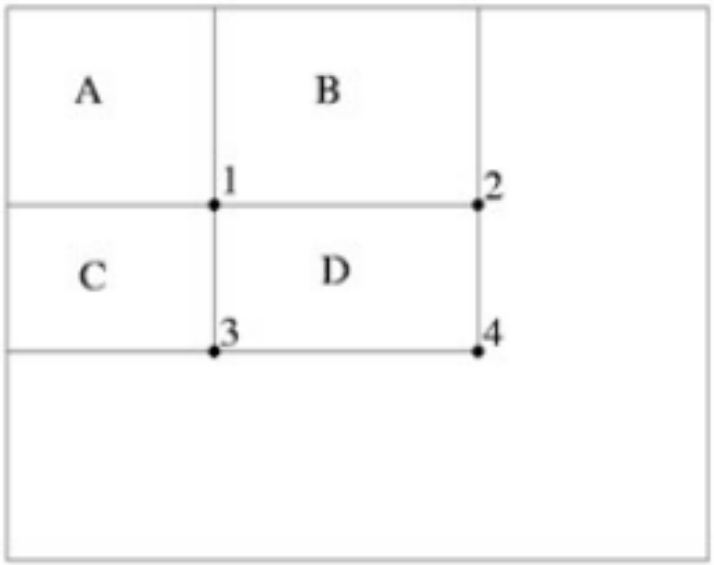
Obraz integralny

Prostokątne cechy są w prosty i szybki sposób obliczane przy pomocy obrazu integralnego.

Obraz integralny to macierz pikseli, które reprezentują sumę intensywności wszystkich poprzednich pikseli (po lewej i u góry).



Obraz integralny



$$D = 1+4-(2+3) = A+(A+B+C+D)-((A+B)+(A+C)) = D$$

Obraz integralny

1.

31	2	4	33	5	36
12	26	9	10	29	25
13	17	21	22	20	18
24	23	15	16	14	19
30	8	28	27	11	7
1	35	34	3	32	6

2.

31	33	37	70	75	111
43	71	84	127	161	222
56	101	135	200	254	333
80	148	197	278	346	444
110	186	263	371	450	555
111	222	333	444	555	666

$15 + 16 + 14 + 28 + 27 + 11 =$

$101 + 450 - 254 - 186 = 111$

Cechy Haar'a

- Cechy są skalowane i obliczane w każdym miejscu obrazu,
- Algorytm Viola Jones używa startowego okna o wymiarach 24x24,
- Uwzględniając wszystkie typy cech Haar'a oraz ich położenia i skale, dla podstawowego okna obliczanych jest około 160 000 cech,
    1. Zbyt dużo danych, nie wszystkie niosą informację.
       Rozwiązanie: kaskada Adaboost;
    2. Zbyt czasochłonne obliczenia.
       Rozwiązanie: Obraz integralny.

AdaBoost

Nie wszystkie cechy wnoszą ważną informację.

AdaBoost

Nie wszystkie cechy wnoszą ważną informację.

Kolejnym krokiem jest zbudowanie zbioru wzorców cech wyszukiwanych. Budowa klasyfikatora oparta jest na metodzie AdaBoost. Do uczenia, kaskada wymaga dużej liczby zdjęć z dokładnym wskazaniem obiektu.

AdaBoost

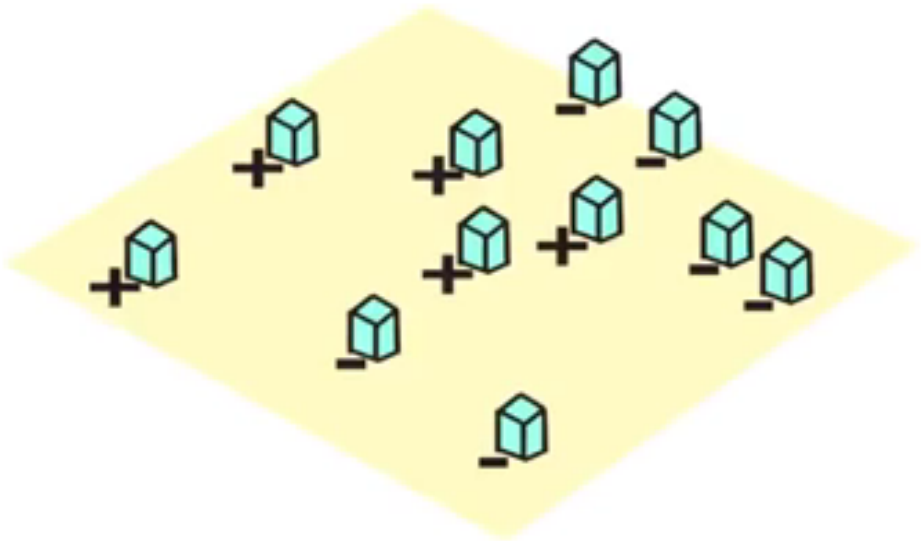
Boosting - algorytm uczenia maszynowego, polega na zastosowaniu słabych klasyfikatorów (trochę lepszy niż przypadek) - cech, w wielu rundach. W każdej rundzie wybierany jest słaby klasyfikator (cecha), który osiągnął wyniki lepsze od innych klasyfikatorów na przykładowym obrazie (wystarczy ponad połowa poprawnych odp), obliczane są dla niego wagi.
Klasyfikator cech AdaBoost jest liniową kombinacją wybranych słabych klasyfikatorów z ich wagami

$$F(x) = \alpha_1 f_1(x) + \alpha_2 f_2(x) + \alpha_3 f_3(x) + ...$$

Wynikiem słabych klasyfikatorów jest wartość 1 lub 0.

Algorytm AdaBoost

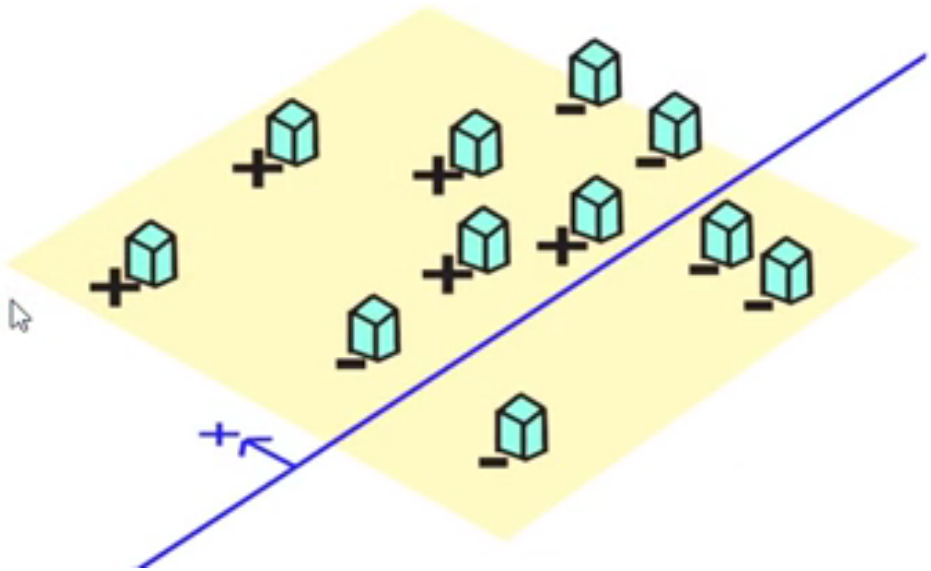
- Przypisanie jednakowych wag elementom zbioru treningowego



Qing Chen, Discover Lab, University of Ottawa

Algorytm AdaBoost

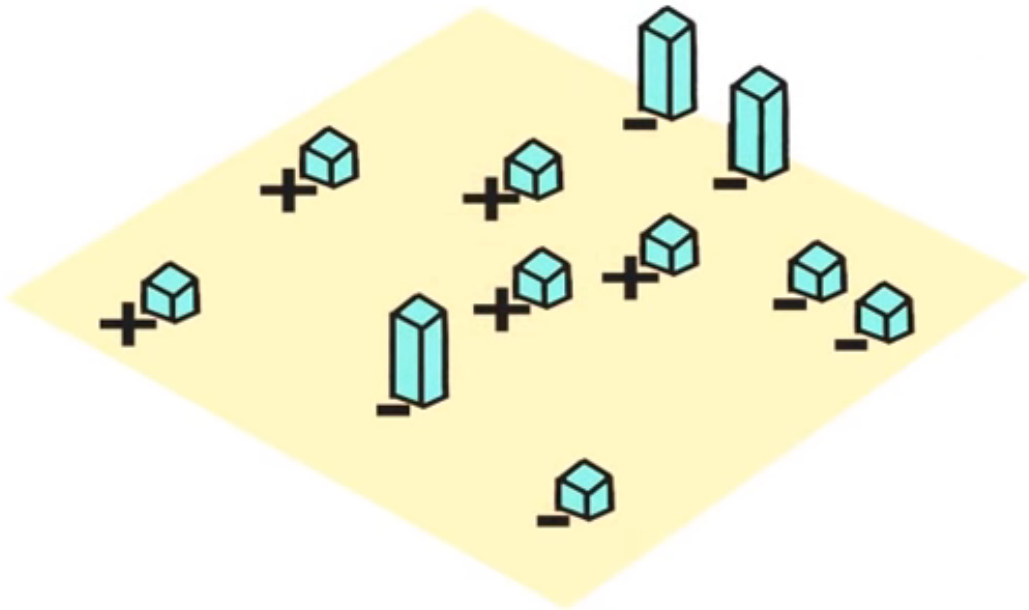
- Przypisanie jednakowych wag elementom zbioru treningowego
- Wybór klasyfikatora o najmniejszym błędzie



Qing Chen, Discover Lab, University of Ottawa

Algorytm AdaBoost

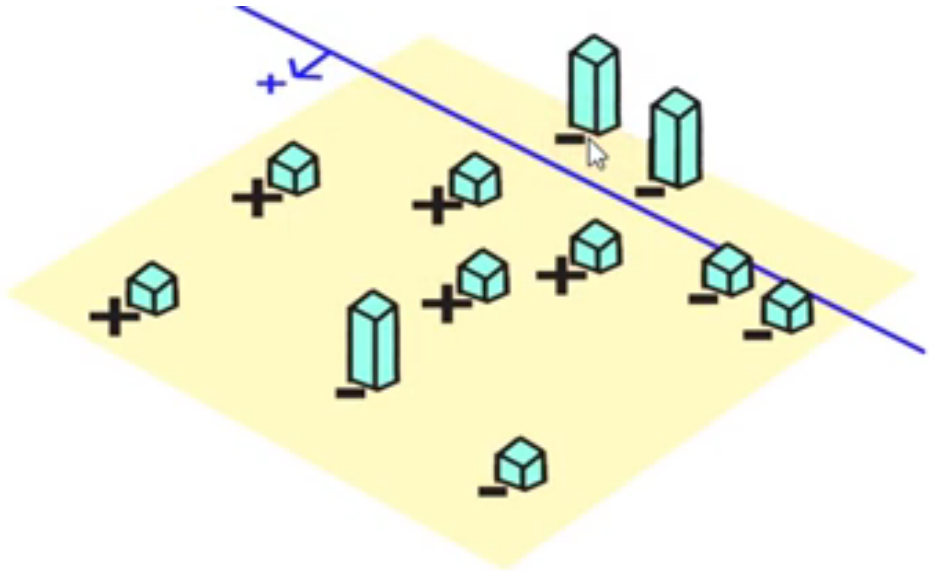
- Przypisanie jednakowych wag elementom zbioru treningowego
- Wybór klasyfikatora o najmniejszym błędzie
- Zwiększenie wag przykładów niepoprawnie sklasyfikowanych



Qing Chen, Discover Lab, University of Ottawa

Algorytm AdaBoost

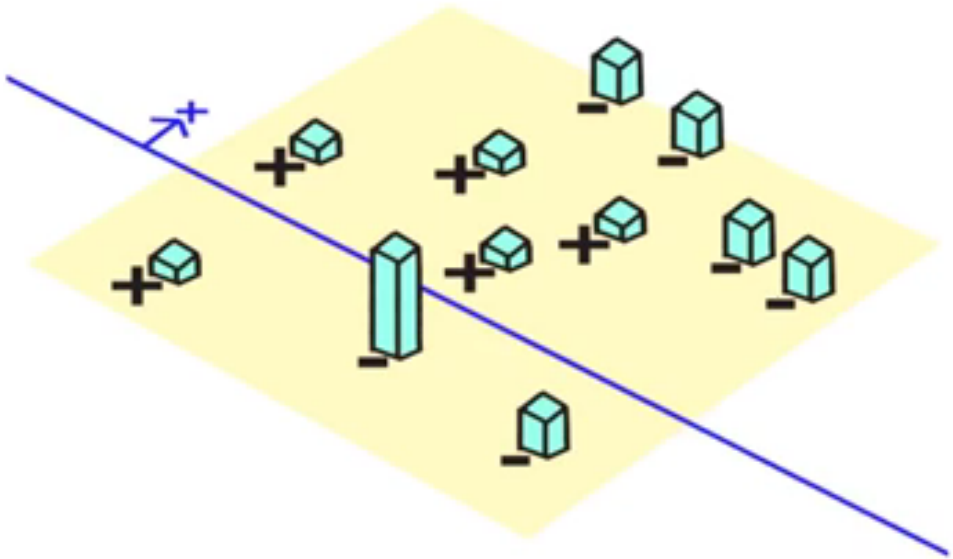
- Przypisanie jednakowych wag elementom zbioru treningowego
- Wybór klasyfikatora o najmniejszym błędzie
- Zwiększenie wag przykładów niepoprawnie sklasyfikowanych
- Następna epoka



Qing Chen, Discover Lab, University of Ottawa

Algorytm AdaBoost

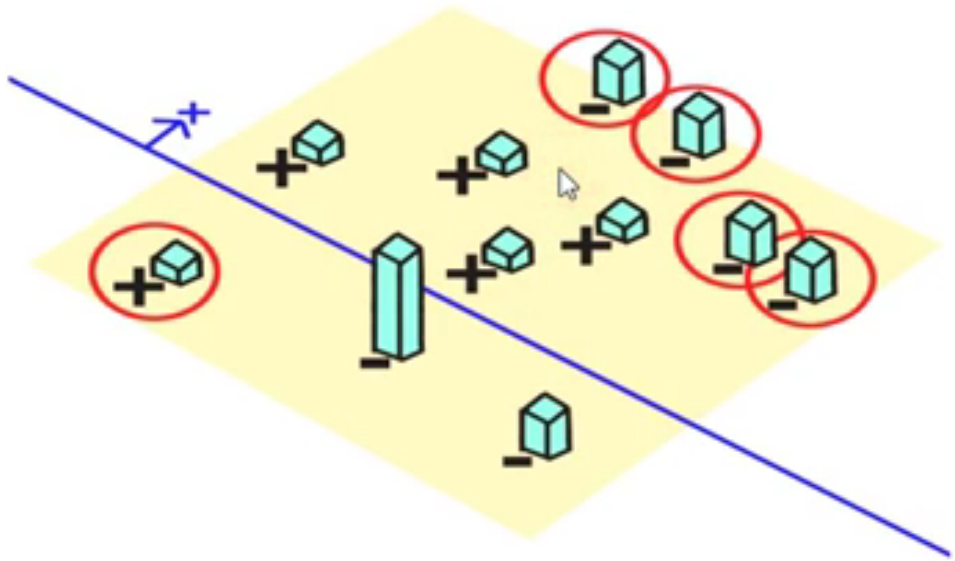
- Przypisanie jednakowych wag elementom zbioru treningowego
- Wybór klasyfikatora o najmniejszym błędzie
- Zwiększenie wag przykładów niepoprawnie sklasyfikowanych
- Następna epoka



Qing Chen, Discover Lab, University of Ottawa

Algorytm AdaBoost

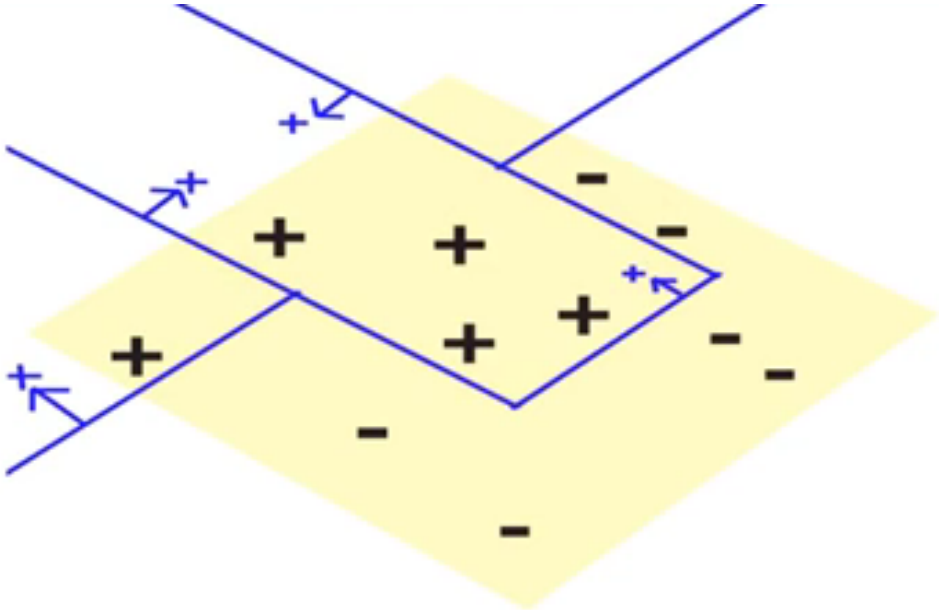
- Przypisanie jednakowych wag elementom zbioru treningowego
- Wybór klasyfikatora o najmniejszym błędzie
- Zwiększenie wag przykładów niepoprawnie sklasyfikowanych
- Następna epoka



Qing Chen, Discover Lab, University of Ottawa

Algorytm AdaBoost

- Przypisanie jednakowych wag elementom zbioru treningowego
- Wybór klasyfikatora o najmniejszym błędzie
- Zwiększenie wag przykładów niepoprawnie sklasyfikowanych
- Następna epoka
- Klasyfikator = suma ważonych słabych klasyfikatorów



$$F(x) = \alpha_1 f_1(x) + \alpha_2 f_2(x) + \alpha_3 f_3(x) + \ldots$$

Qing Chen, Discover Lab, University of Ottawa

Kaskada

- Statystycznie więcej jest okien o negatywnej odpowiedzi niż pozytywnej

Kaskada

- Statystycznie więcej jest okien o negatywnej odpowiedzi niż pozytywnej
- Czas obliczeń jest jednakowy dla wszystkich okien

Kaskada

- Statystycznie więcej jest okien o negatywnej odpowiedzi niż pozytywnej
- Czas obliczeń jest jednakowy dla wszystkich okien
- Bardziej wydajne działanie algorytmu polega na wcześniejszym odrzucaniu obrazów negatywnych i skupianiu się na regionach o większym prawdopodobieństwie odnalezienia twarzy

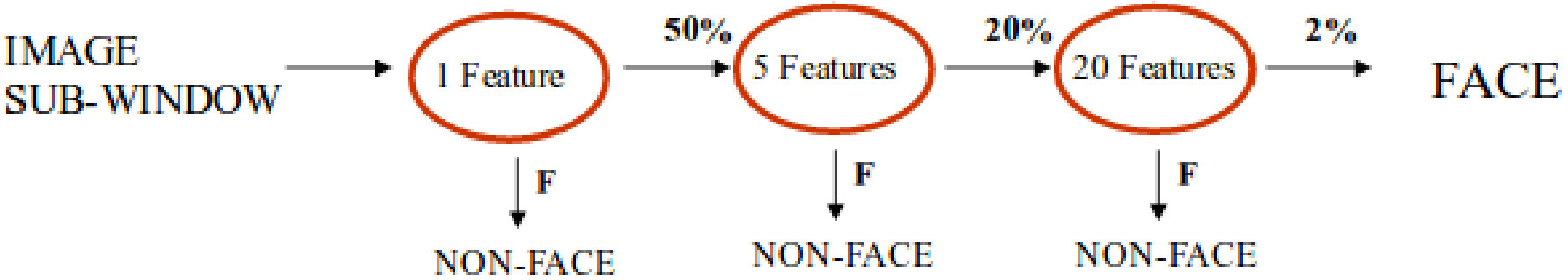
Kaskada

- Statystycznie więcej jest okien o negatywnej odpowiedzi niż pozytywnej
- Czas obliczeń jest jednakowy dla wszystkich okien
- Bardziej wydajne działanie algorytmu polega na wcześniejszym odrzucaniu obrazów negatywnych i skupianiu się na regionach o większym prawdopodobieństwie odnalezienia twarzy
- Klasyfikator składający się z 2 cech osiąga 100% detekcję w 50% False Positive - może działać jako pierwsza warstwa filtracji okien negatywnych

Kaskada

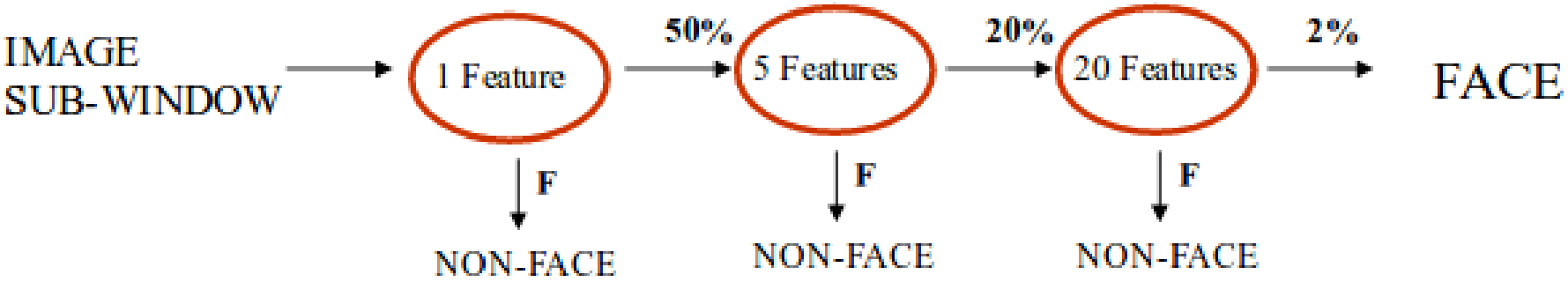
- Statystycznie więcej jest okien o negatywnej odpowiedzi niż pozytywnej
- Czas obliczeń jest jednakowy dla wszystkich okien
- Bardziej wydajne działanie algorytmu polega na wcześniejszym odrzucaniu obrazów negatywnych i skupianiu się na regionach o większym prawdopodobieństwie odnalezienia twarzy
- Klasyfikator składający się z 2 cech osiąga 100% detekcję w 50% False Positive - może działać jako pierwsza warstwa filtracji okien negatywnych
- Następne warstwy mogą składać się z kolejnych cech

Kaskada



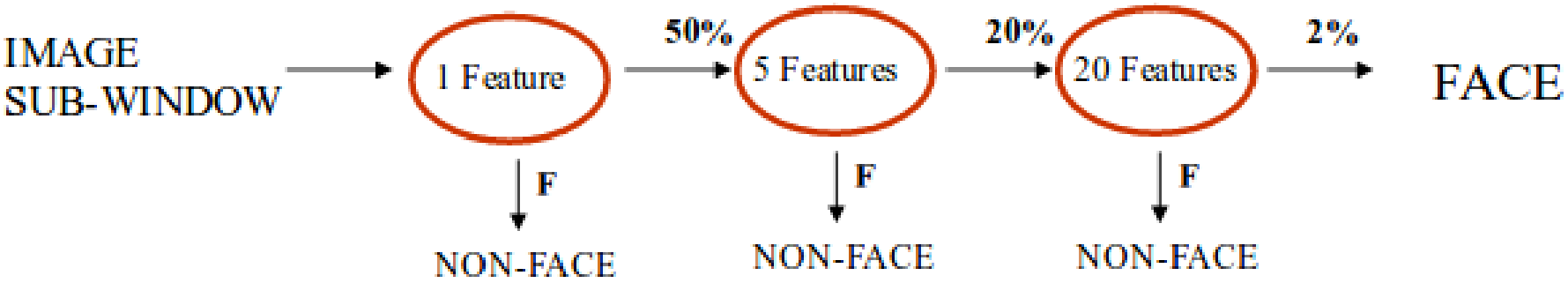
- Każdy etap jest zbudowany z silnego klasyfikatora,
- Wynikiem każdego z etapów jest decyzja, czy konkretne okno na pewno nie jest twarzą, lub czy możliwe, że jest,
- Okno jest automatycznie odrzucane w przypadku negatywnego wyniku któregokolwiek etapu

Trenowanie kaskady



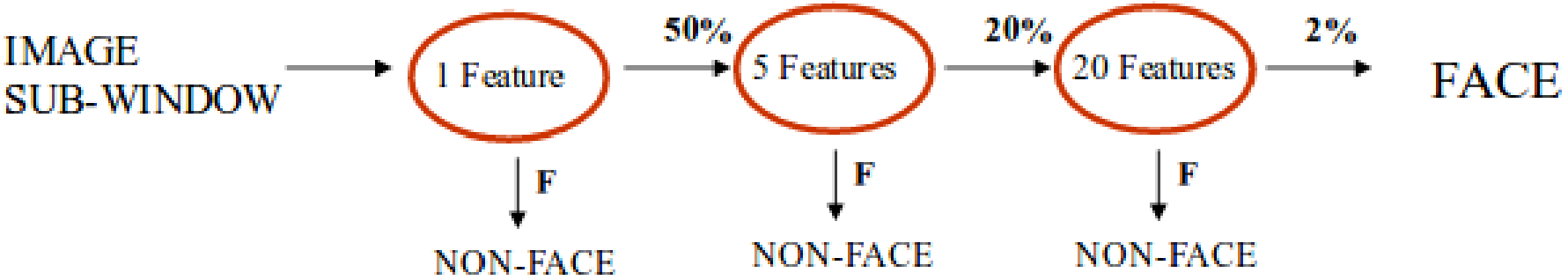
- Wybór:

Trenowanie kaskady



- Wybór:
  - Liczby etapów w kaskadzie

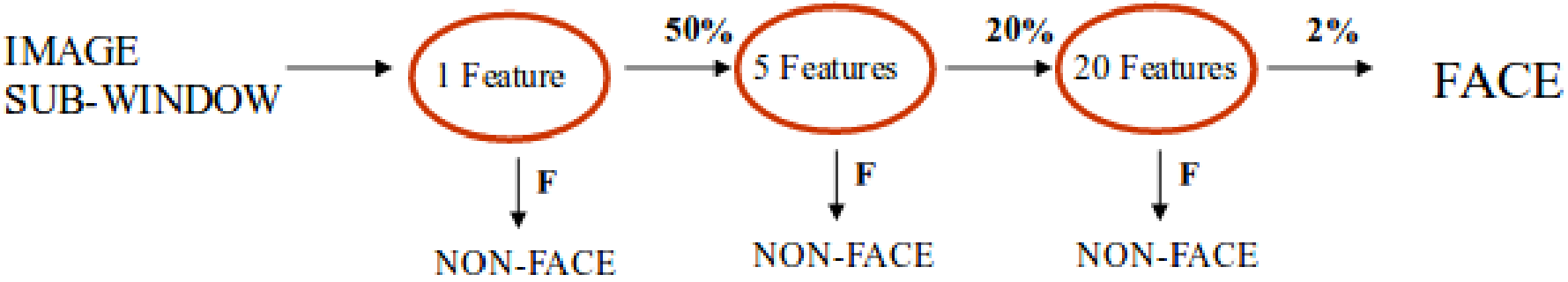
Trenowanie kaskady



- Wybór:
  - Liczby etapów w kaskadzie
  - Liczby cech każdego etapu

$$F(x) = \alpha_1 f_1(x) + \alpha_2 f_2(x) + \alpha_3 f_3(x) + ...$$

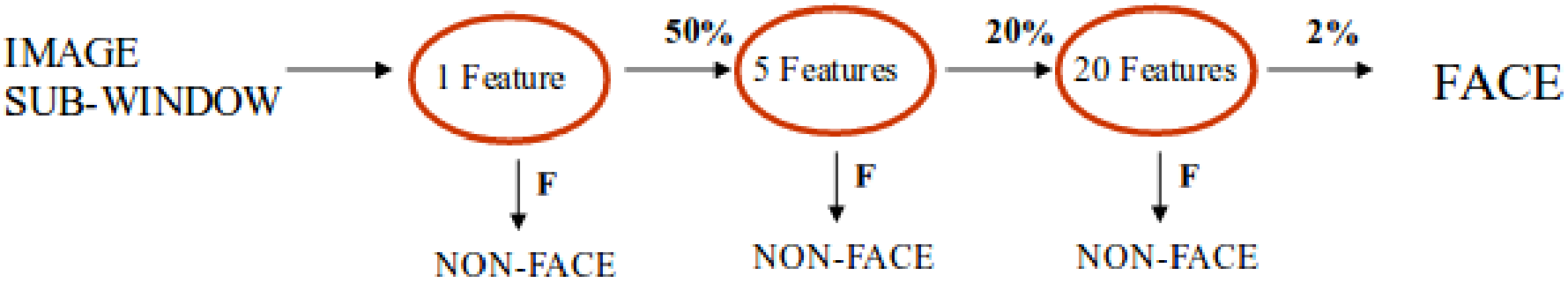
Trenowanie kaskady



- Wybór:
  - Liczby etapów w kaskadzie
  - Liczby cech każdego etapu
  - Progu każdego etapu

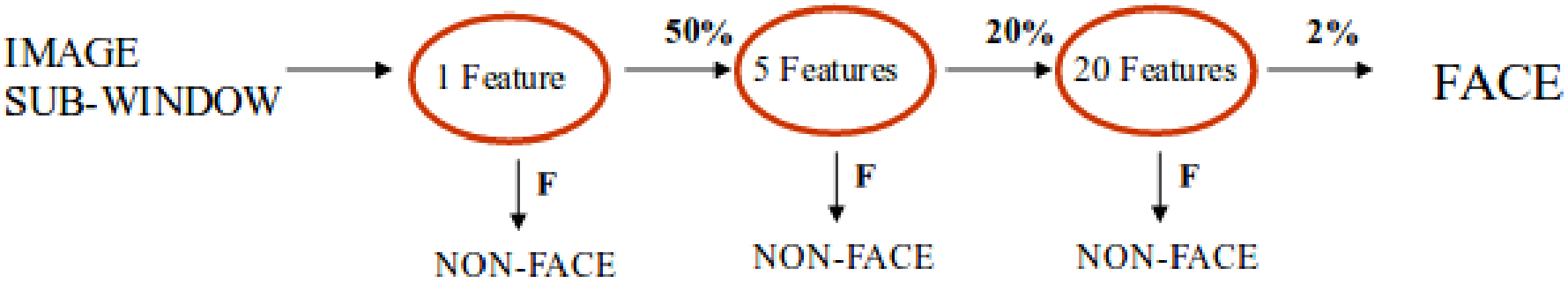
$$F(x) = \begin{cases} 1 & \text{gdy } \sum_{t-1}^{T} \alpha_t f_t(x) \geqslant \text{Próg} \\ 0 & \text{w przeciwnym wypadku} \end{cases}$$

Trenowanie kaskady



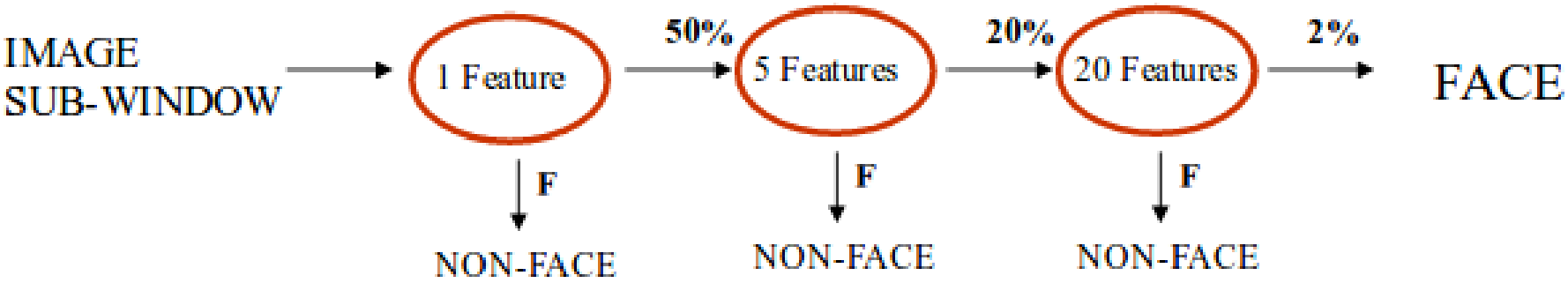
- Wybór:
  - Liczby etapów w kaskadzie
  - Liczby cech każdego etapu
  - Progu każdego etapu
- Problem optymalizacji:

Trenowanie kaskady



- Wybór:
  - Liczby etapów w kaskadzie
  - Liczby cech każdego etapu
  - Progu każdego etapu
- Problem optymalizacji:
  - Problem rzeczywisty: minimalizacja liczby wykorzystanych detektorów, parametrami optymalizowanymi są liczba etapów, detektorów (cech) i progi w poszczególnych etapach

Trenowanie kaskady



- Wybór:
  - Liczby etapów w kaskadzie
  - Liczby cech każdego etapu
  - Progu każdego etapu
- Problem optymalizacji:
  - Problem rzeczywisty: minimalizacja liczby wykorzystanych detektorów, parametrami optymalizowanymi są liczba etapów, detektorów (cech) i progi w poszczególnych etapach
  - Problem uproszczony: optymalizacja liczby fałszywych decyzji pozytywnych (FP) i wsółczynnik poprawnie wykrytych twarzy.

Trenowanie kaskady

Optymalizacja:

1. Wybór f_i - maksymalna akceptowalna liczba False Positive / etap

Trenowanie kaskady

Optymalizacja:

1. Wybór f_i - maksymalna akceptowalna liczba False Positive / etap
2. Wybór d_i - minimalna akceptowalna liczba True Positive / etap

Trenowanie kaskady

Optymalizacja:

1. Wybór f_i - maksymalna akceptowalna liczba False Positive / etap
2. Wybór d_i - minimalna akceptowalna liczba True Positive / etap
3. Wybór F_{calk} - całkowity współczynnik False Positive

Trenowanie kaskady

Optymalizacja:

1. Wybór f_i - maksymalna akceptowalna liczba False Positive / etap
2. Wybór d_i - minimalna akceptowalna liczba True Positive / etap
3. Wybór F_{calk} - całkowity współczynnik False Positive
4. Dopóki nie osiągnięto F_{calk}:

Trenowanie kaskady

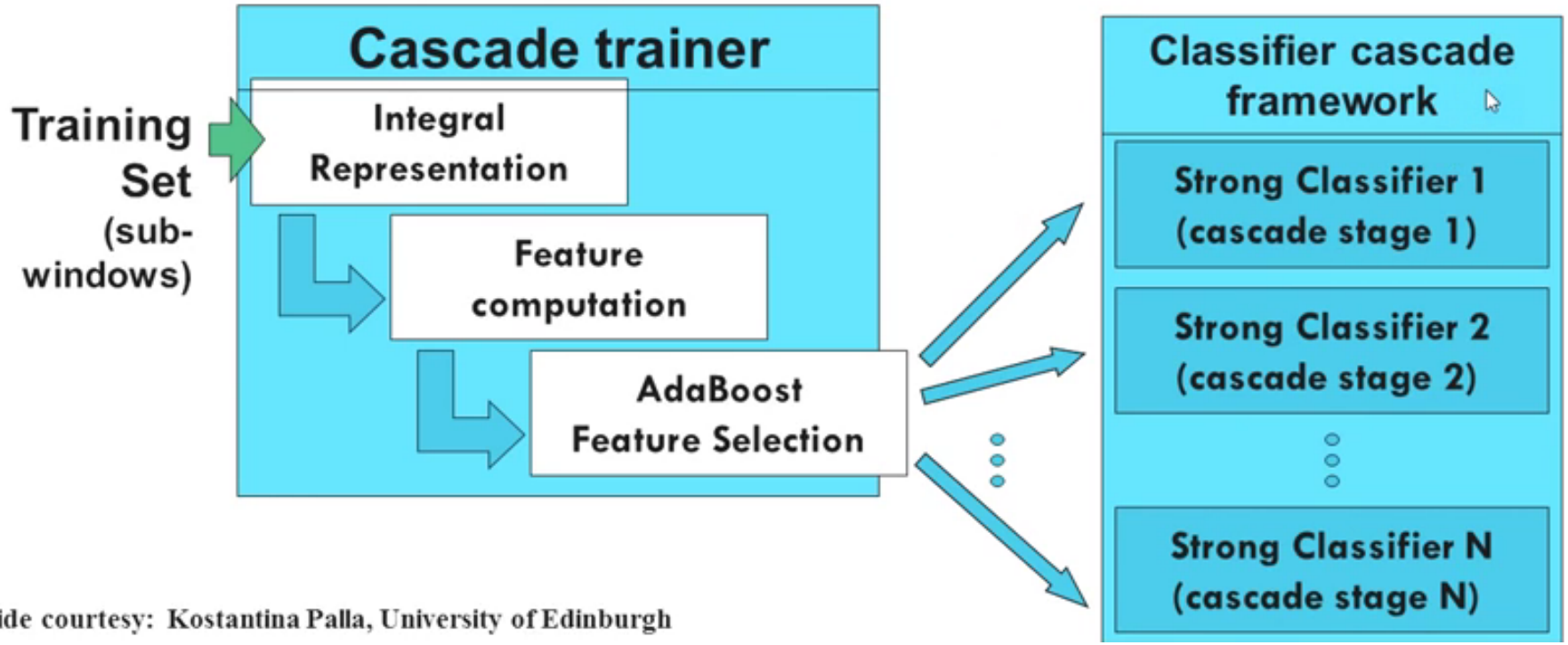
Optymalizacja:

1. Wybór f_i - maksymalna akceptowalna liczba False Positive / etap
2. Wybór d_i - minimalna akceptowalna liczba True Positive / etap
3. Wybór F_{calk} - całkowity współczynnik False Positive
4. Dopóki nie osiągnięto F_{calk}:
   - Dodanie nowego etapu:
     Dopóki nie osiągnięto f_i i d_i dla tego etapu:

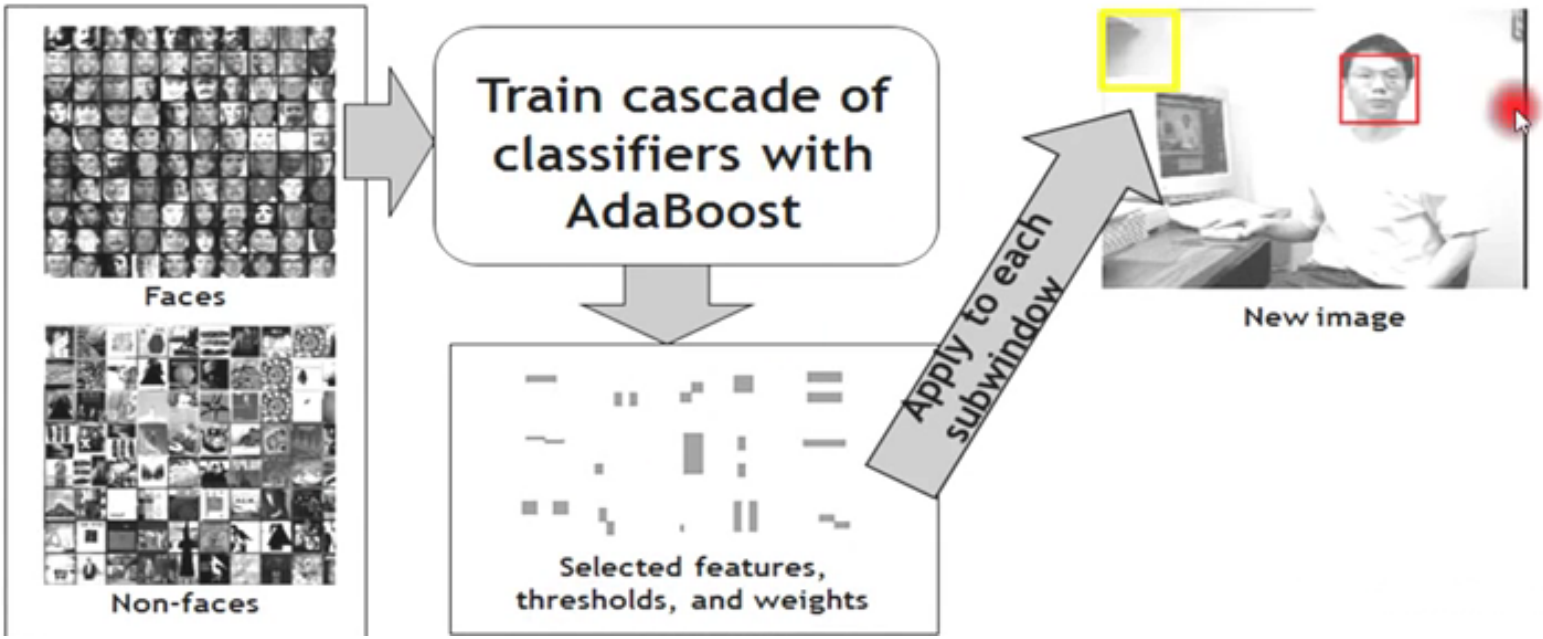
Trenowanie kaskady

Optymalizacja:

1. Wybór f_i - maksymalna akceptowalna liczba False Positive / etap
2. Wybór d_i - minimalna akceptowalna liczba True Positive / etap
3. Wybór F_{calk} - całkowity współczynnik False Positive
4. Dopóki nie osiągnięto F_{calk}:
   - Dodanie nowego etapu:
     Dopóki nie osiągnięto f_i i d_i dla tego etapu:
     - Dodanie cech i trenowanie nowego klasyfikatora



lide courtesy: Kostantina Palla, University of Edinburgh


Faces
Non-faces
Train cascade of classifiers with AdaBoost
Selected features, thresholds, and weights
Apply to each subwindow
New image

K. Grauman

Cechy algorytmu Viola Jones

Cechy algorytmu:

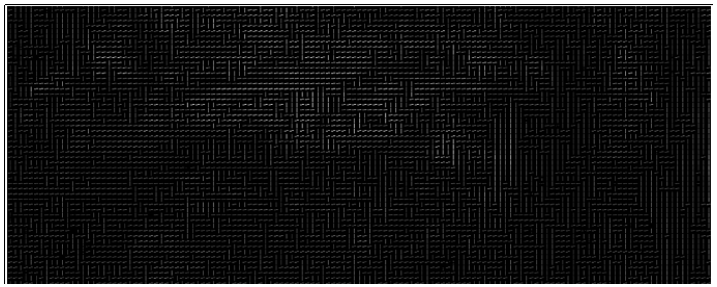
+ Działa w czasie rzeczywistym, prosta architektura,
+ Rozpoznaje twarze w różnych skalach,
- Trenowanie jest powolne, detekcja szybka,
- Brak informacji o teksturze i kształcie
- Zbiór treningowy musi zawierać obiekty i nie-obiekty,
– Twarze w obrazach treningowych muszą być dokładnie oznaczone,
– Twarze zwrócone przodem, niezasłonięte,
- Nie sprawdza się w detekcji ogólnych obiektów (ludzie).

Spis treści:

Histogram of Oriented Gradients (HoG)

Etapy algorytmu:

1. stworzenie obrazu gradientowego (zastąpienie każdego piksela przez gradient jego intensywności),



https://medium.com/@ageitgey

Histogram of Oriented Gradients (HoG)

Etapy algorytmu:

1. stworzenie obrazu gradientowego (zastąpienie każdego piksela przez gradient jego intensywności),
2. stworzene histogramu vektorów dla każdej komórki (np. 8x8 pikseli - redukcja 8x8 wektorów do 9 wartości - przeważnie tyle binów ma histogram),

https://medium.com/@ageitgey

Histogram of Oriented Gradients (HoG)

Etapy algorytmu:

1. stworzenie obrazu gradientowego (zastąpienie każdego piksela przez gradient jego intensywności),
2. stworzene histogramu vektorów dla każdej komórki (np. 8x8 pikseli - redukcja 8x8 wektorów do 9 wartości - przeważnie tyle binów ma histogram),
3. normalizacja - organizacja pikseli w bloki R-HOG (rectangular, np. 16x16) lub C-HOG (circular), znormalizowana grupa histogramów reprezentuje histogram bloku, który z kolei, jako jeden z wielu reprezentuje deskryptor

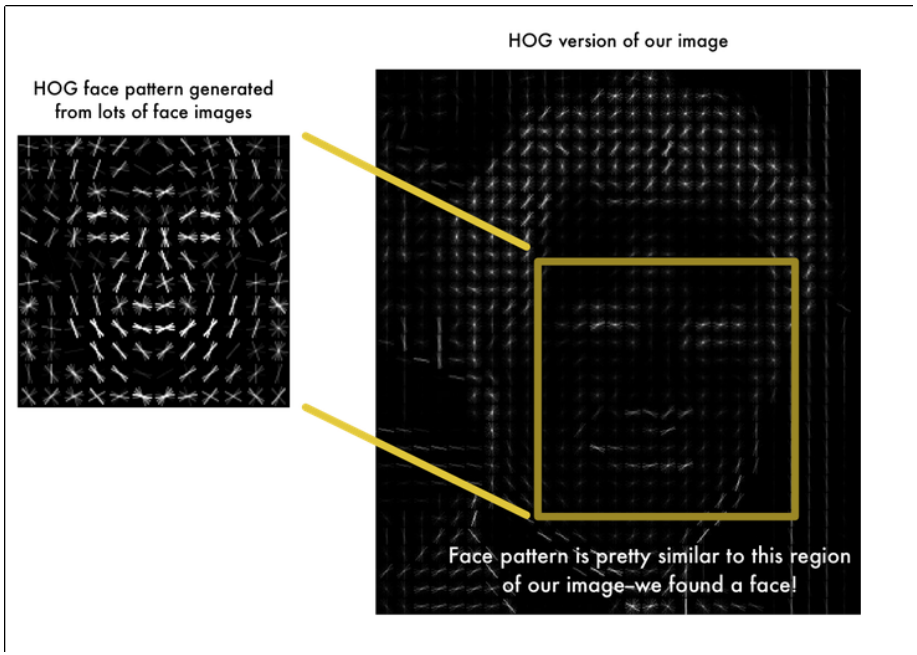
https://heartbeat.fritz.ai

Histogram of Oriented Gradients (HoG)

Etapy algorytmu:

1. stworzenie obrazu gradientowego (zastąpienie każdego piksela przez gradient jego intensywności),
2. stworzene histogramu vektorów dla każdej komórki (np. 8x8 pikseli - redukcja 8x8 wektorów do 9 wartości - przeważnie tyle binów ma histogram),
3. normalizacja - organizacja pikseli w bloki R-HOG (rectangular, np. 16x16) lub C-HOG (circular), znormalizowana grupa histogramów reprezentuje histogram bloku, który z kolei, jako jeden z wielu reprezentuje deskryptor
4. zastosowanie klasyfikatora (CNN lub SVM) do identyfikacji określonej twarzy.

HOG



Cechy:

- HOG to deskryptor cech, nie ma przypisanej metody klasyfikacji,
- często używany w parze z klasyfikatorem SVM,
- jest odporny na obroty, skalowanie, okluzje, zmiany koloru i oświetlenia (w ograniczonym stopniu),

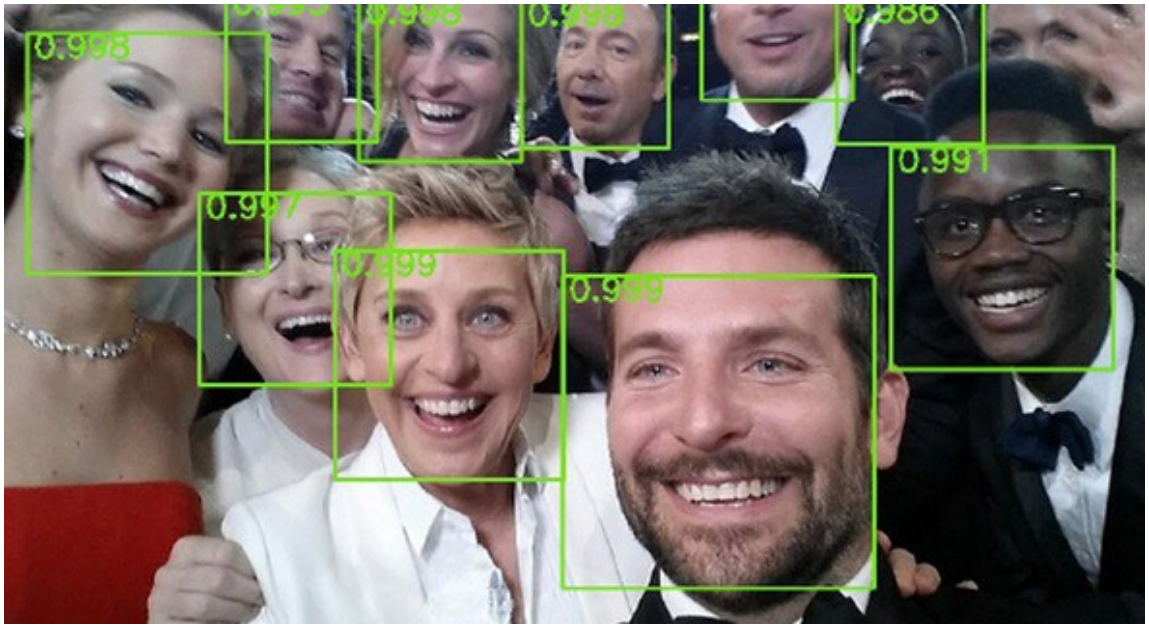
\+ szybki.

https://medium.com/@ageitgey

Spis treści:

1. Detektory, deskryptory
2. Ekstrakcja cech - krawędzie
3. Ekstrakcja cech - punkty
4. Ekstrakcja cech - linie - Transformacja Hough
5. Ekstrakcja cech - bloby
6. Detekcja twarzy
   - Algorytm Viola Jones (Kaskada Haara),
   - HoG,
7. **Detekcja i rozpoznanie twarzy**
   - Eigenfaces,

Rozpoznanie twarzy



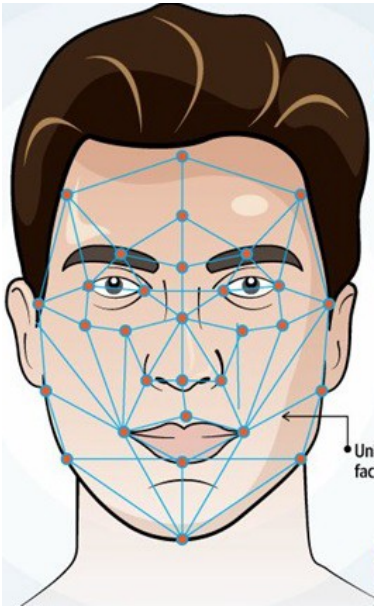
Kroki:

1. Detekcja twarzy

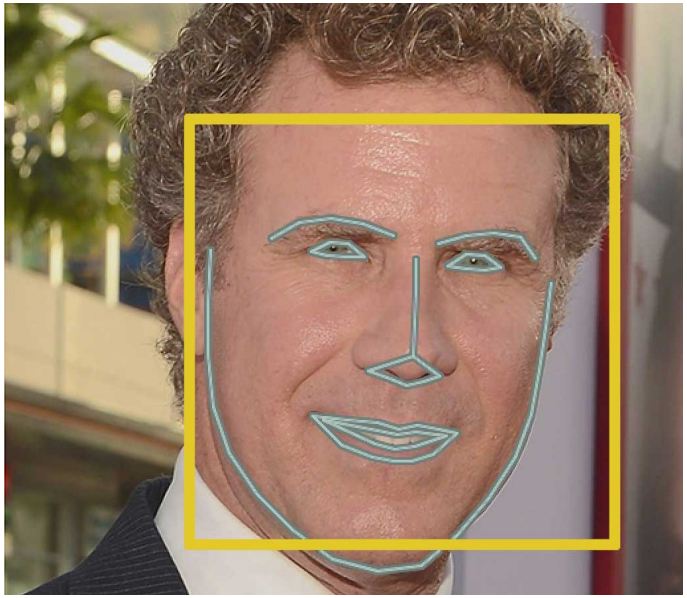
Face ID, medium.com

Rozpoznanie twarzy

Face ID



OpenFace

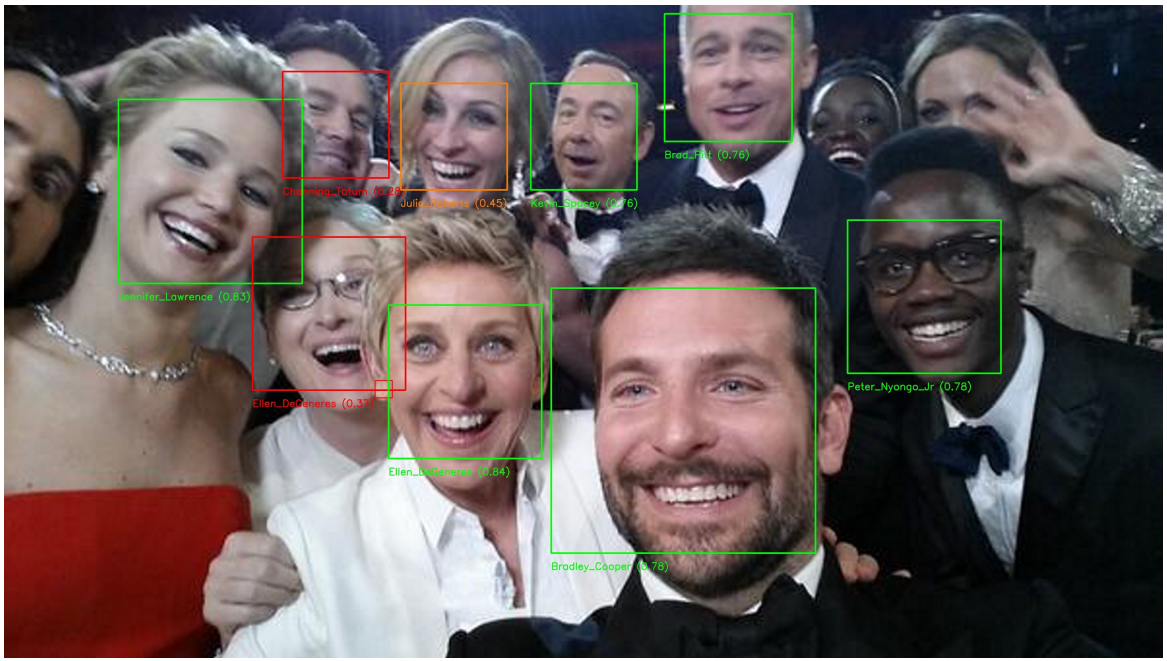


Kroki:

1. Detekcja twarzy
2. Ekstrakcja cech i ich parametryzacja (128 parametrów Google)

Face ID, medium.com
https://medium.com/@ageitgey

Rozpoznanie twarzy



Kroki:

1. Detekcja twarzy
2. Ekstrakcja cech i ich parametryzacja (128 parametrów Google)
3. Klasyfikacja

Face recognition with OpenCV and dlib, www.pyimagesearch.com

Rozpoznanie twarzy

Metody:

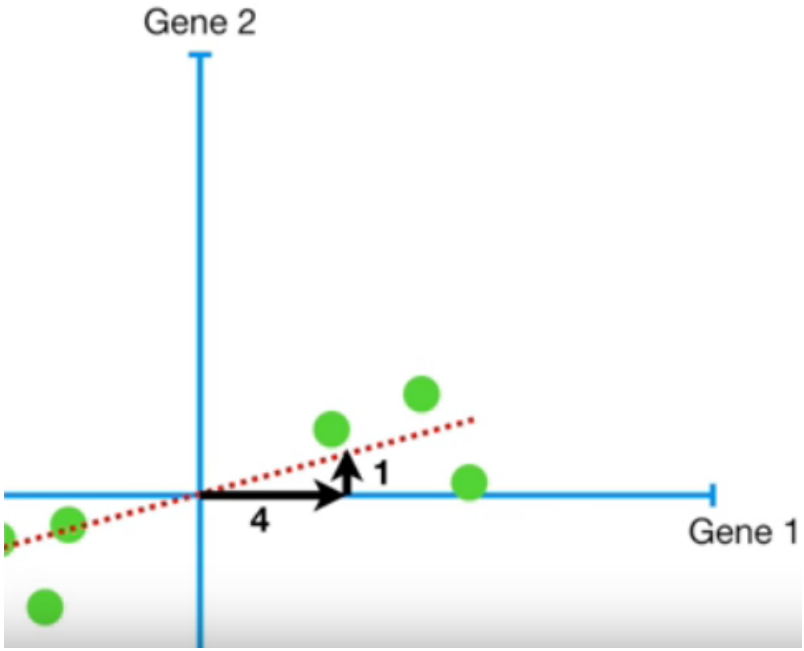
- Dopasowanie wzorców - funkcja rozpoznania między obrazem a wzorcem (w postaci np. modelu, pikseli) to korelacja lub miara odległości
  - Adaptative Appeareance Models
- Podejście statystyczne - analiza statystyczna cech obrazu, zadaniem jest ekstrakcja cech i ich analiza:
  - PCA - Eigenfaces
  - Discrete Cosine Transform (DCT)
  - Linear Discriminant Analysis (LDA Fisherfaces)
  - ICA
  - Transformata Gabora
- Sieci neuronowe - sieć sama wybiera istotne cechy obrazu

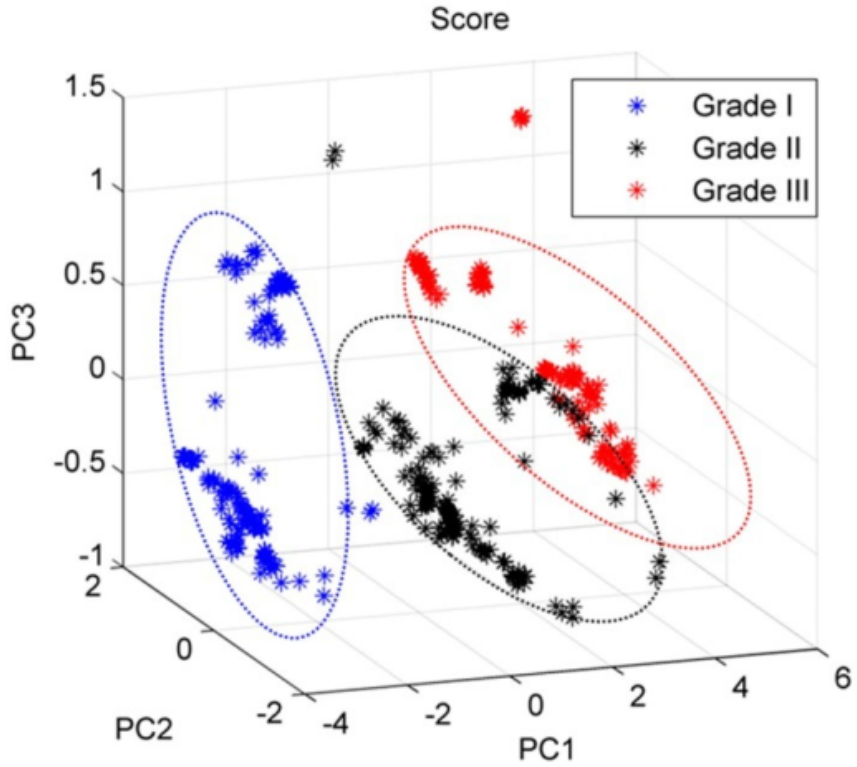
Eigenfaces

Eigenfaces - Algorytm twarzy własnych - metoda wykorzystuje analizę głównych składowych (PCA).

Eigenfaces

Eigenfaces - Algorytm twarzy własnych - metoda wykorzystuje analizę głównych składowych (PCA).





Zakłada, że obrazy twarzy mogą być przedstawione w przestrzeni ortogonalnych głównych składowych o mniejszej wymiarowości (Eigenface=Eigenvector), a każdy obraz może być przedstawiony jako liniowa kombinacja eigenfaces

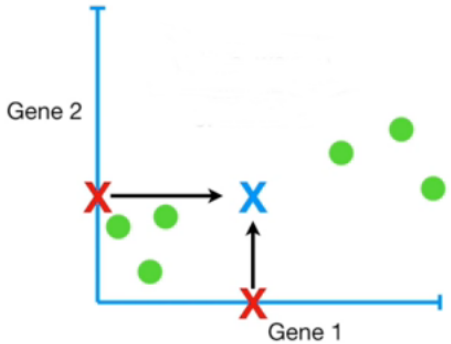
Tworzenie Eigenfaces

1. Zbiór treningowy

   $x_1, ..., x_M$

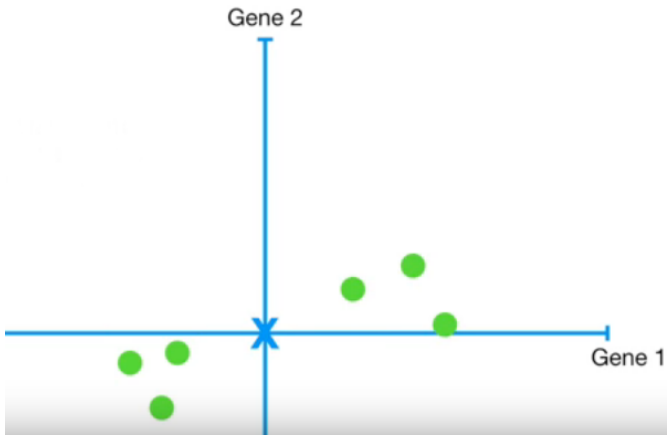
Tworzenie Eigenfaces

1. Zbiór treningowy $x_1, ..., x_M$
2. Obliczenie średniej twarzy μ



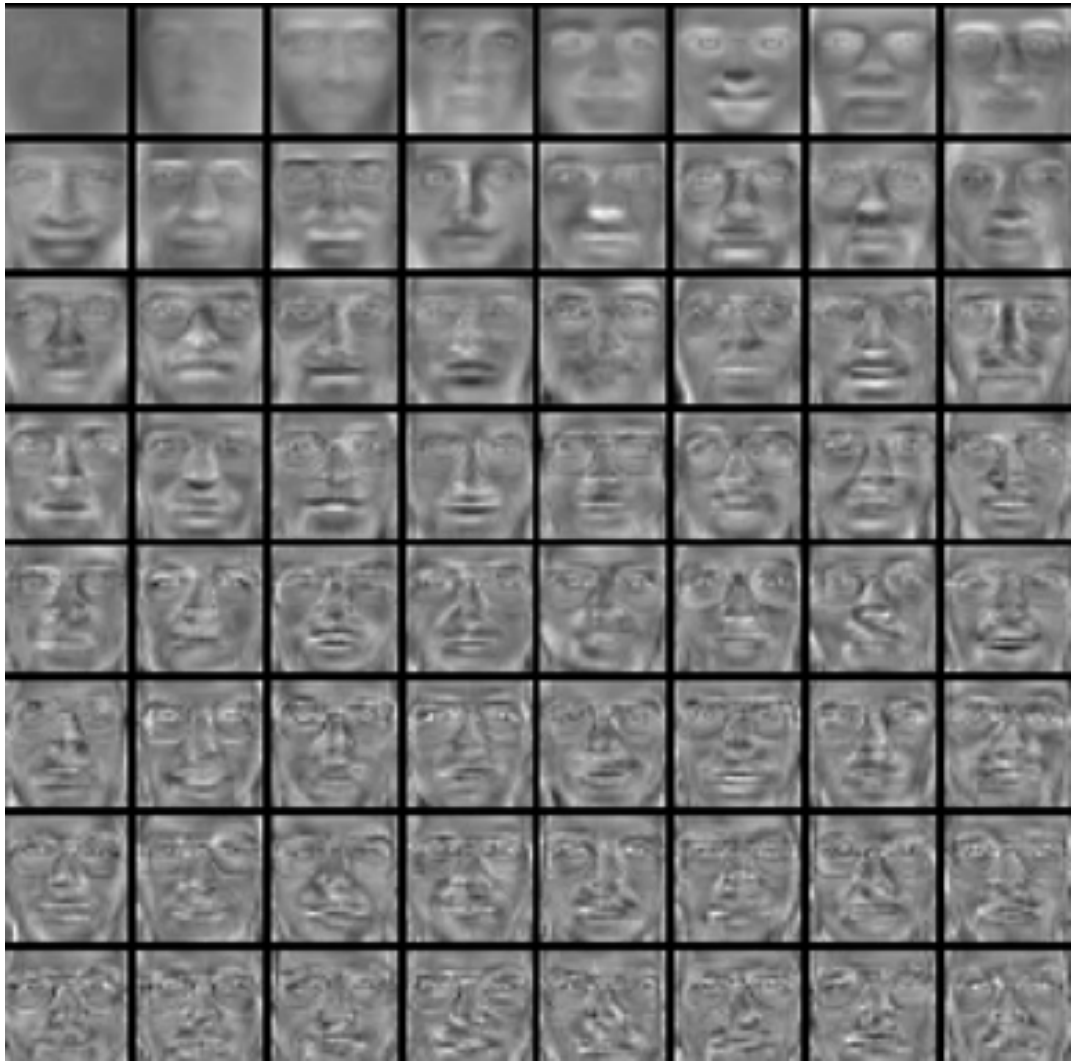
Tworzenie Eigenfaces

1. Zbiór treningowy $x_1, \ldots, x_M$
2. Obliczenie średniej twarzy μ
3. Odjęcie średniej twarzy od zbioru

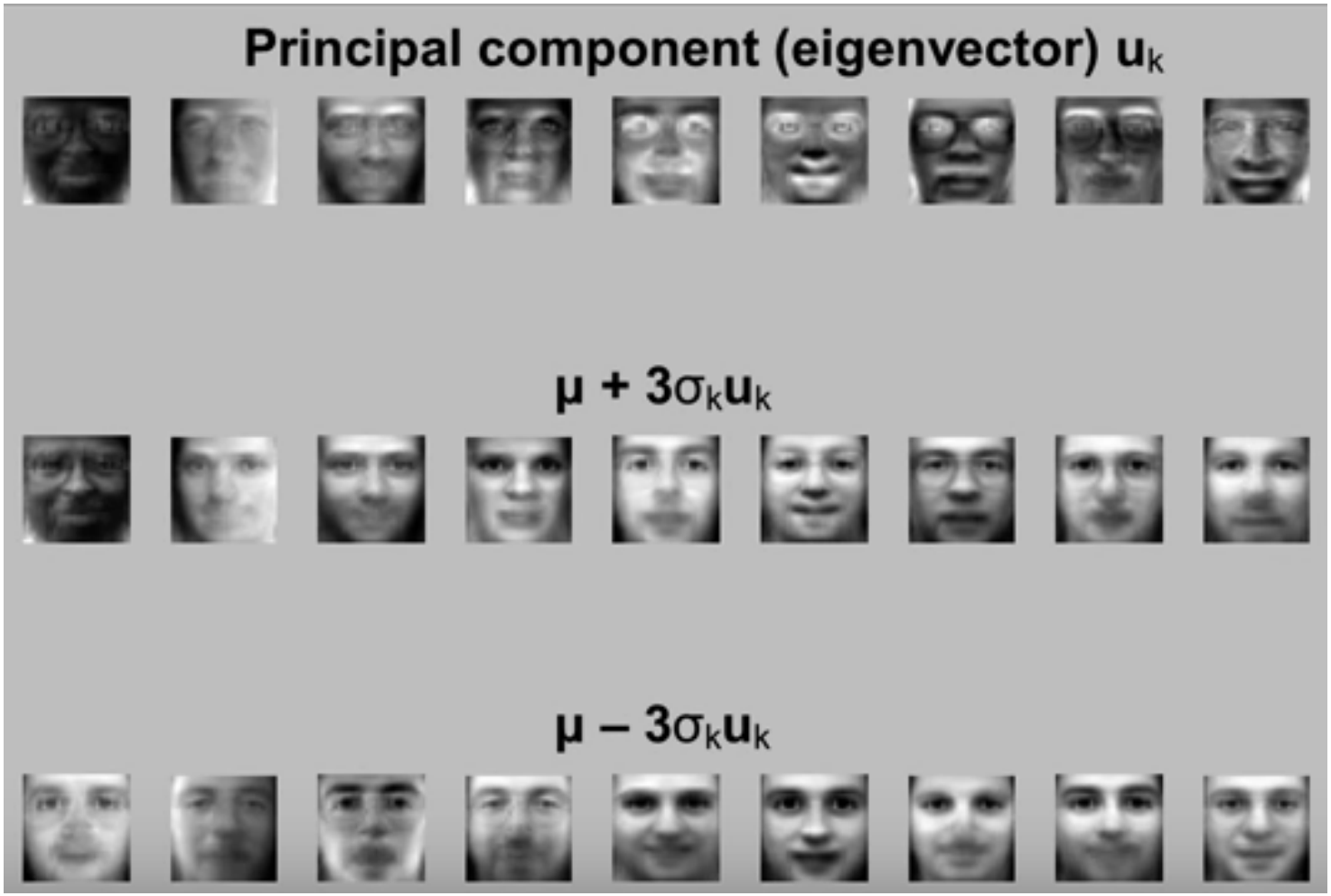


Tworzenie Eigenfaces

1. Zbiór treningowy $x_1, ..., x_M$
2. Obliczenie średniej twarzy μ
3. Odjęcie średniej twarzy od zbioru
4. Stworzenie Eigenfaces $u_1, ..., u_K$

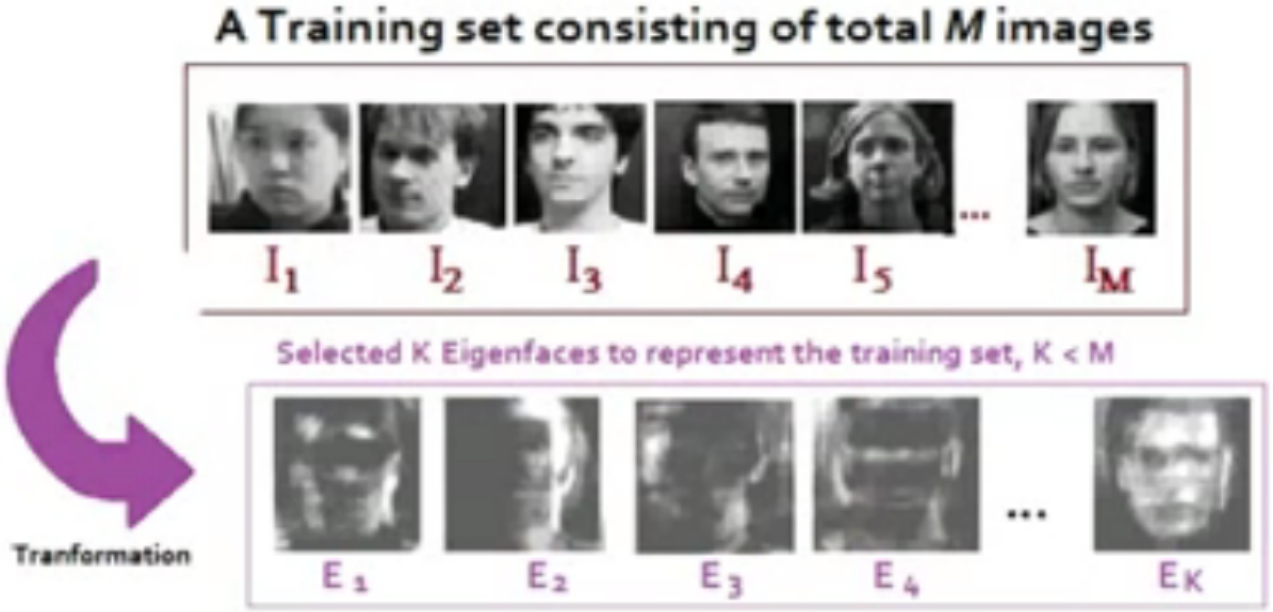


Eigenvectors



Introduction to Computer Vision, Udacity, GeorgiaTech

Tworzenie Eigenfaces

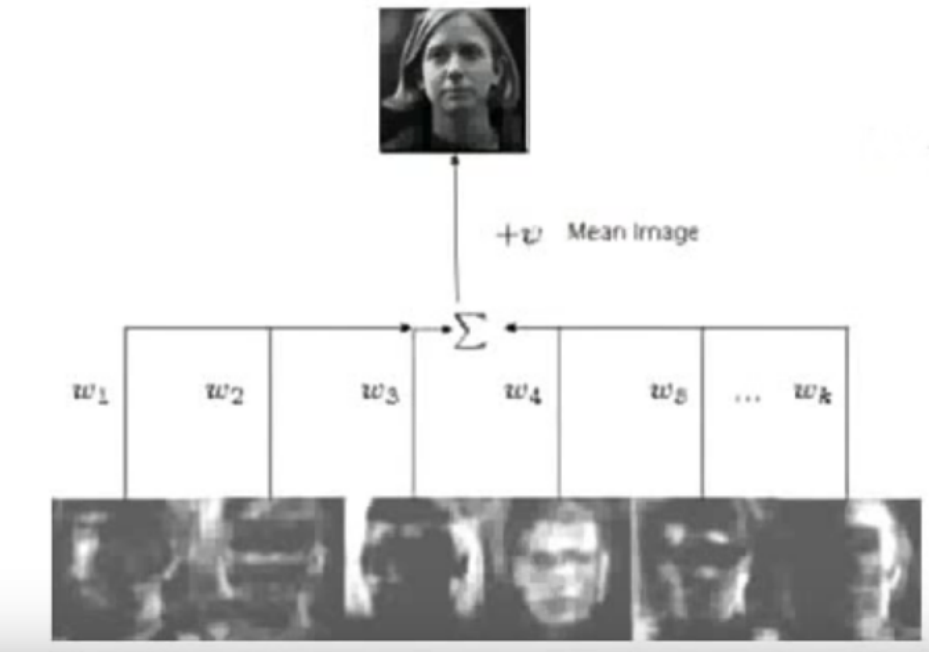


$$K \leqslant M$$

Eigenfaces to Eigenvectors macierzy kowariancji zbioru danych.

Eigenfaces

Każdy obraz twarzy może być przedstawiony jako liniowa kombinacja eigenvectorów (eigenfaces)



$$[w_1, .., w_K] = [u_1^T(x - \mu), ..., u_K^T(x - \mu)]$$

Eigenfaces

- Rozpoznanie twarzy:
  Sprawdzenie błędu $x - \mu$

Eigenfaces

- Rozpoznanie twarzy:
  Sprawdzenie błędu $x - \mu$
- Rozpoznanie konkretnej twarzy:
  Klasyfikacja do twarzy trenowanych w przestrzeni k-wymiarowej

Popularne detektory twarzy oparte na sieciach neuronowych

- DNN w OpenCV - oparty na Single-Shot-Multibox detector używa architektury ResNet-10:
  - \+ duża dokładność,
  - \+ działa w czasie rzeczywistym,
  - \+ rozpoznaje twarze w różnych skalach oraz orientacjach,
  - \+ działa podczas okluzji,
  - − brak kodu.
- CNN Face Detector w Dlib - używa Maximum-Margin Object Detector (MMOD):
  - szybkie trenowanie, nieduży zbiór treningowy,
  - działa szybko na GPU, bardzo wolno na CPU.
- face_recognition biblioteka Pythona - oparta na dlib, wygodniejsze użycie,
- YOLOFace,
- i wiele innych.