(Traditional) Object Detection

Daniil Osokin

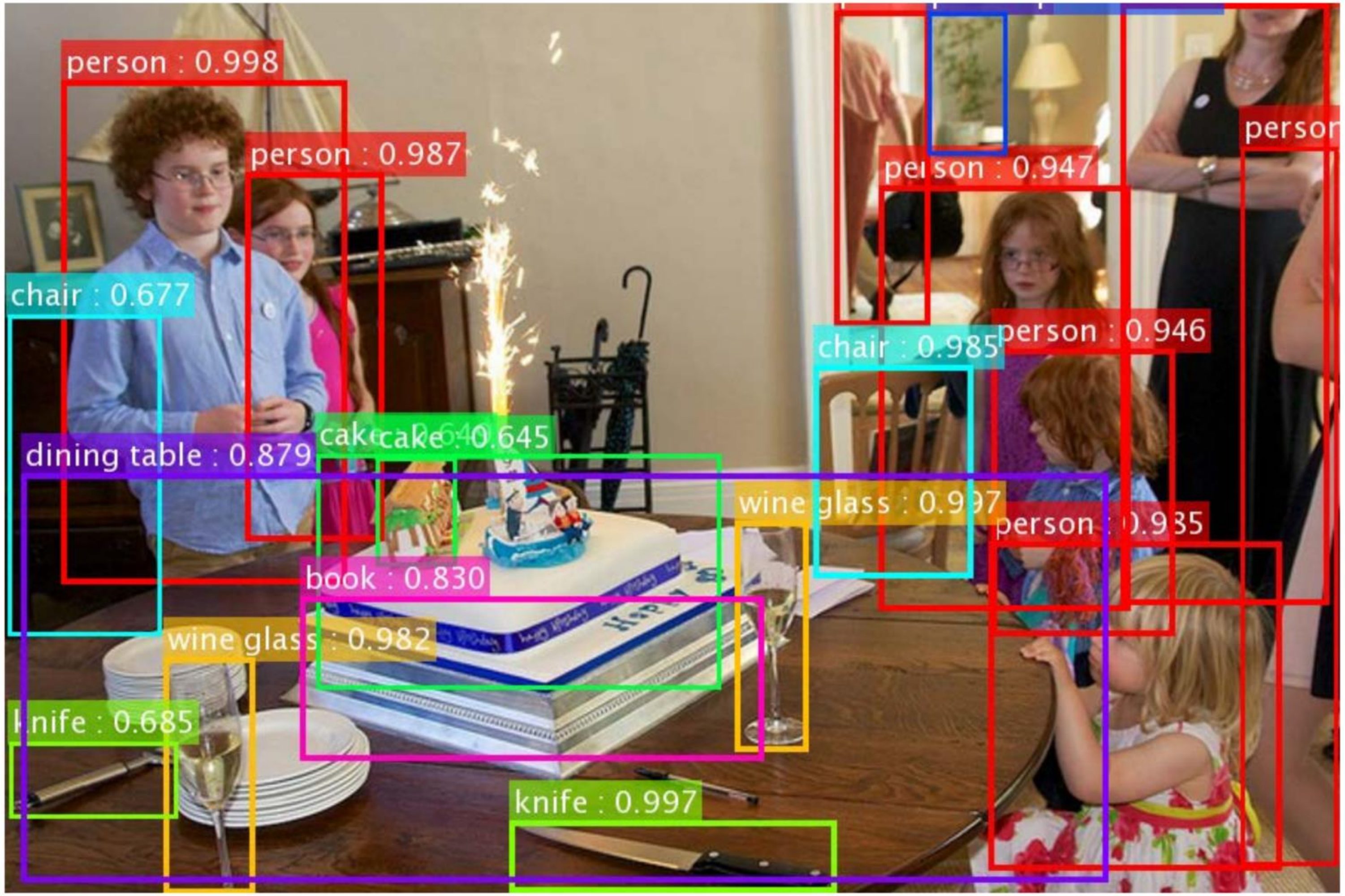
Intel, IOTG/VMC/ICV

daniil.osokin@intel.com

Internet of Things Group

Object Detection

1. Localization:
   - Make hypotheses about objects locations
2. Classification:
   - Give label for found objects



Images credit: https://chaosmail.github.io/deeplearning/2016/10/22/intro-to-deep-learning-for-computer-vision/

Challenges

- Large variance in object appearance
  - Rotation (in-plane, out-of-plane), pose (rigid/non-rigid objects), shape, occlusion, environmental conditions, etc.
- Objects can have different scales on the same image
- Real-time speed for real-world applications

Classification Task

Train dataset: set of pairs ($\mathcal{X}_i$, $\mathcal{Y}_i$), *i=1..N* (training samples)

- $\mathcal{X}_i$ - object, $\mathcal{X}_i = (x_i^1, x_i^2, \ldots, x_i^n) \in \mathbb{R}^n$ – features vector (descriptor of object)
- $\mathcal{Y}_i \in \{0, 1\}$ – label for each object (class of object)

Test dataset: $\mathcal{X}_i^{test}$, *no* labels given

The classification task is to predict labels for each object in test dataset (do classification)

Decision Tree

Given train dataset ($\mathcal{X}_i$, $\mathcal{Y}_i$), *i=1..N*, build a *tree* which reduces the classification error:

$$\sum_{i=1}^{N} |label_{gt} - label_{predicted}|$$

$\mathcal{X}_i = (x_i^1, x_i^2, ..., x_i^n)$

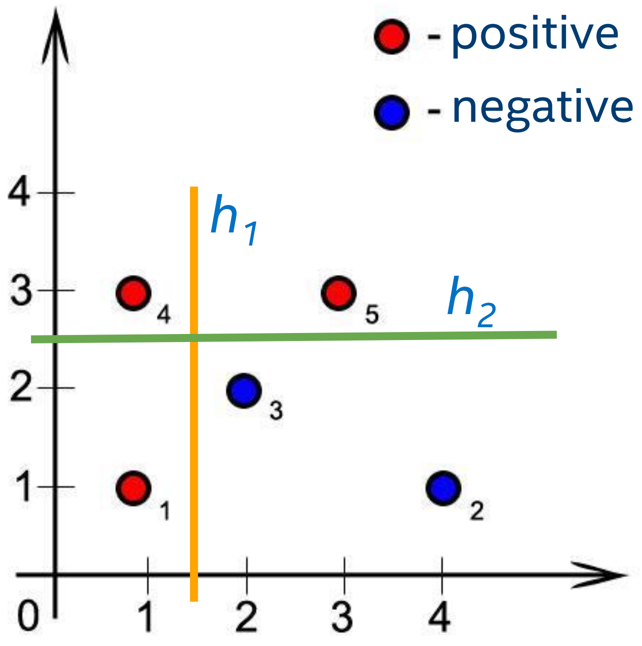
Decision tree (depth=1) examples:

Object Classification with Decision Tree

Decision tree (depth 1): $h_t(x) = \begin{cases} 1, & p^j x^j < p^j \theta^j \\ 0, & otherwise \end{cases}$ $\quad \theta^j - threshold \quad p^j - sign$

Negative – object with label = 0

Positive – object with label ≠ 0



h_1: $\theta^1 = 1.5$, $p^1 = 1$

h_2: $\theta^2 = 2.5$, $p^2 = -1$

Classifier predictions:

True Positive:
$label_{gt} = 1$
$label_{predicted} = 1$

True Negative:
$label_{gt} = 0$
$label_{predicted} = 0$

False Positive:
$label_{gt} = 0$
$label_{predicted} = 1$

False Negative:
$label_{gt} = 1$
$label_{predicted} = 0$

AdaBoost: Boosting with Decision Trees

1. Weighted train dataset: $(\mathcal{X}_i, \mathcal{Y}_i)$, $i=1..N$, weights $w_i = \frac{1}{N}$ for each object
2. For t = 1…T
   1. Train decision trees h_j, each makes split by x_i feature, compute error ε_j: $\varepsilon_j = \sum_{i=1}^{m} w_i * |h_j(x_i) - y_i|$
   2. Select tree with the lowest error $\varepsilon_j = \varepsilon$
   3. Update the weights of training samples: $w_i = w_i * \beta_t^{1-e_i}, \beta_t = \frac{\varepsilon}{1-\varepsilon}$

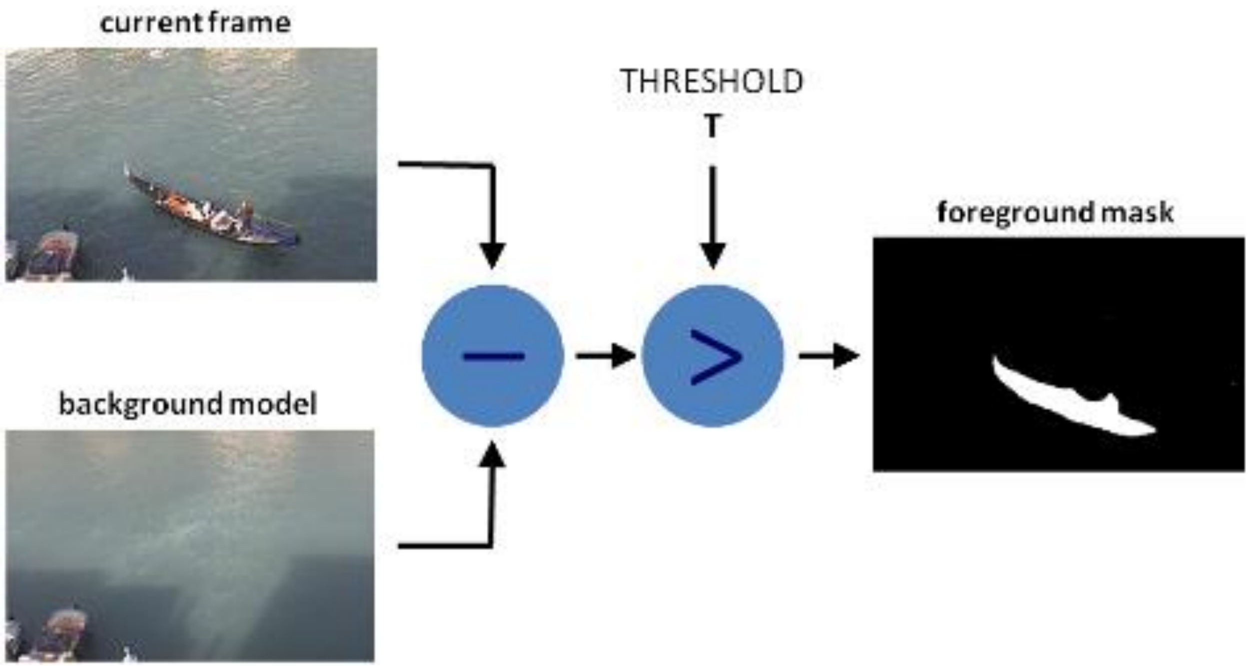
   $e_i = 0$ if $\mathcal{X}_i$ classified correctly, else $e_i = 1$

   4. Normalize weights
3. Final (strong) classifier: $H(\mathcal{X}) = \begin{cases} 1, & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t, \alpha_t = log\frac{1}{\beta_t} \\ 0, & \text{otherwise} \end{cases}$

For pedestrian detection typical T ~ 2000

Localization: Background Subtraction

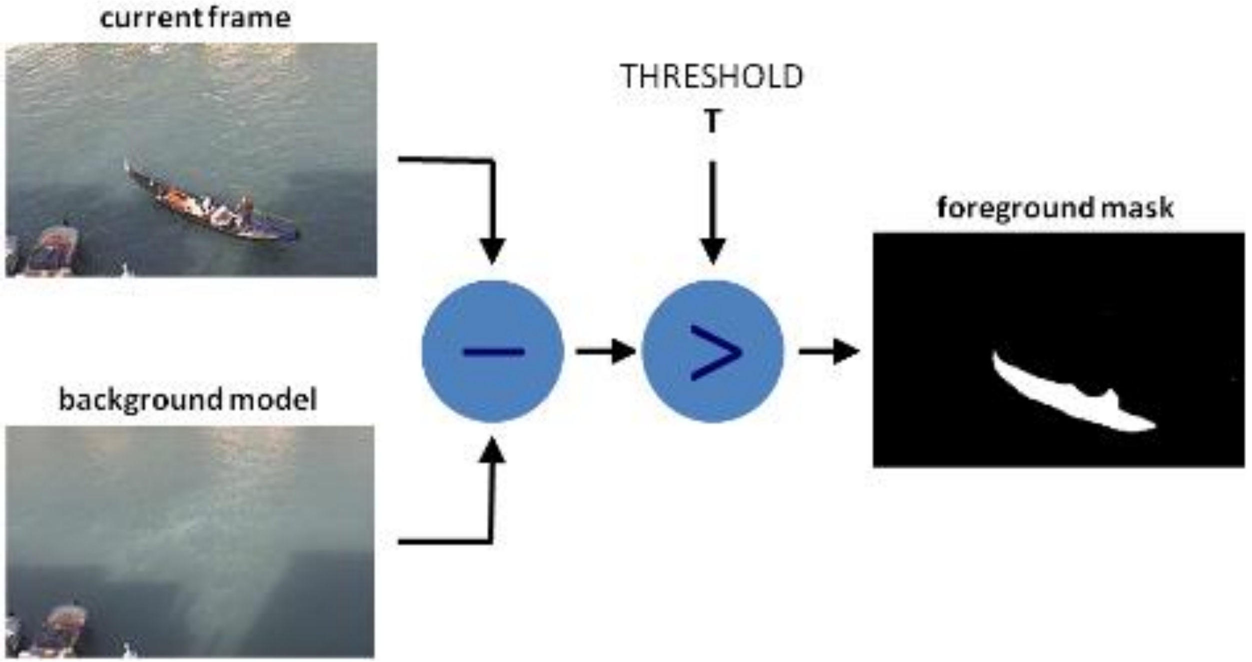
1. Given image of static background
2. Subtract from current frame the background, threshold to filter noise
3. Result mask contains *location hypotheses*



Images credit: https://chaosmail.github.io/deeplearning/2016/10/22/intro-to-deep-learning-for-computer-vision/

Localization: Background Subtraction

1. Given image of static background
2. Subtract from current frame the background, threshold to filter noise
3. Result mask contains *location hypotheses*



Q: When this does not work?

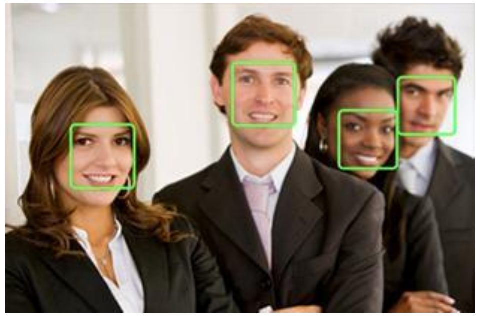
Images credit: https://chaosmail.github.io/deeplearning/2016/10/22/intro-to-deep-learning-for-computer-vision/

Localization: Sliding Window

1. Set fixed size window

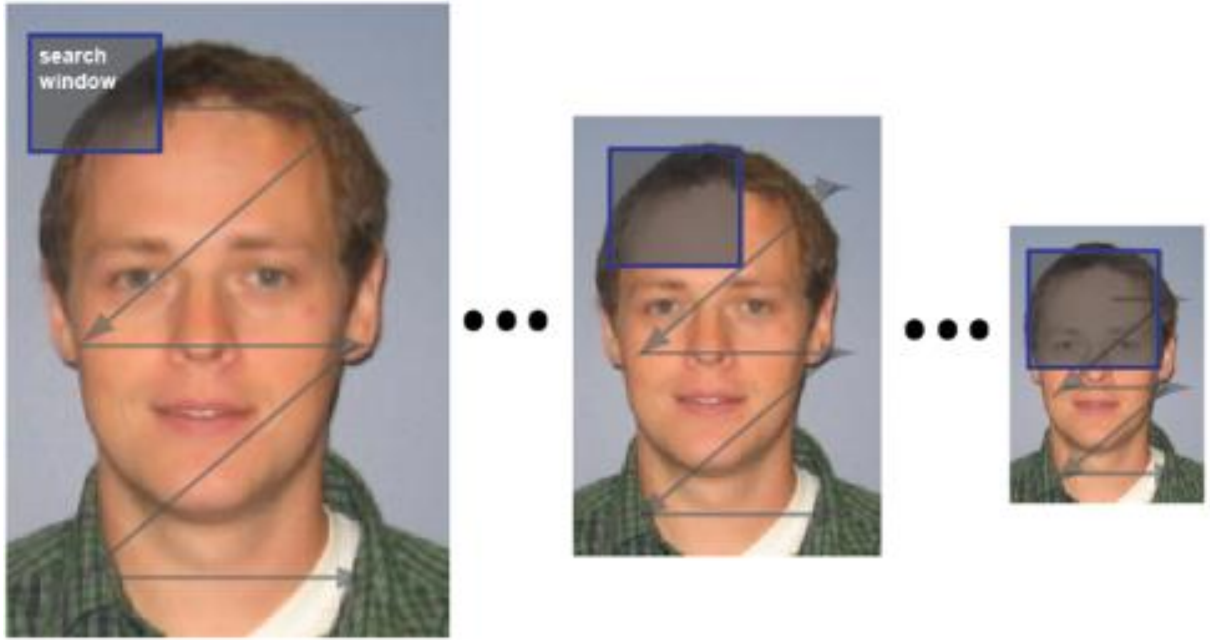
2. Slide this window (left to right, top to bottom) over the image. Each window position is a *location hypothesis*

3. To deal with various scales, slide window over image pyramid



Face: 24x24 pixels

Person: 60x120 pixels

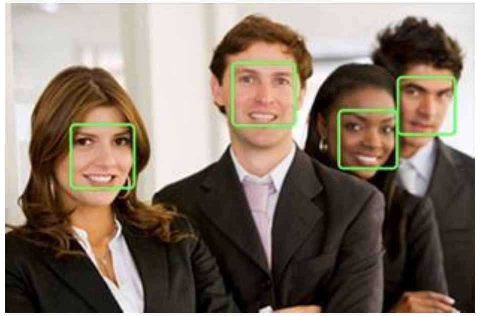


Images credit: http://mi.eng.cam.ac.uk/~cipolla/lectures/4F12/Slides/old/4F12-detection.pdf

Localization: Sliding Window

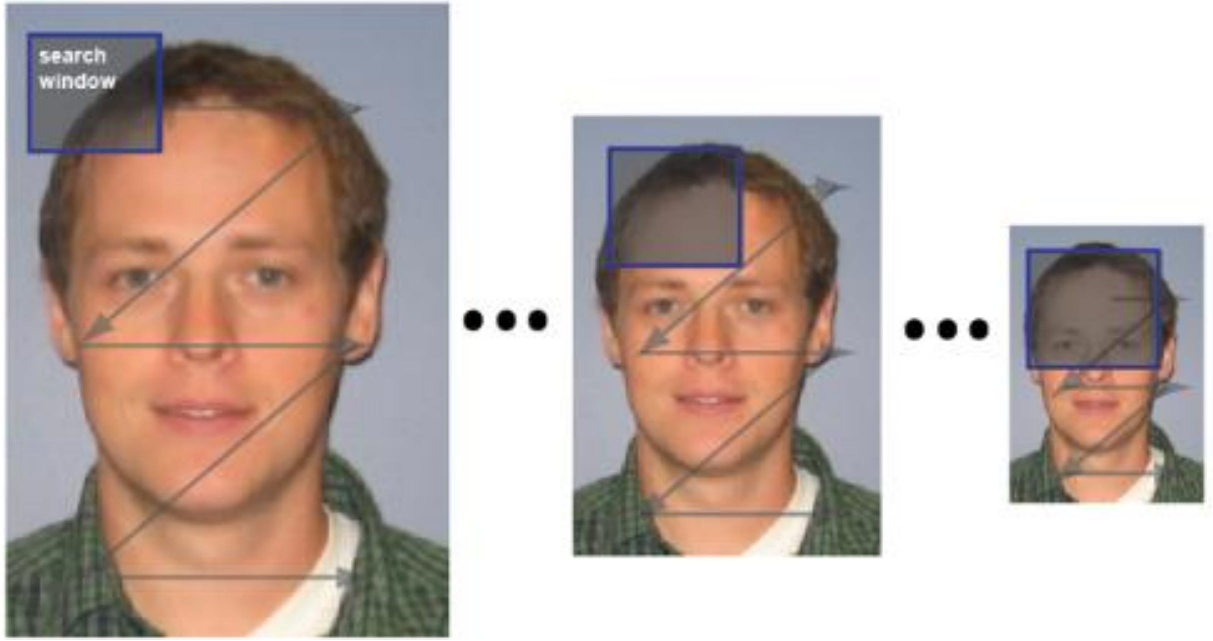
1. Set fixed size window
2. Slide this window (left to right, top to bottom) over the image. Each window position is a *location hypothesis*
3. To deal with various scales, slide window over image pyramid

Q: How many sliding windows for person in 1280x720 image



Face: 24x24 pixels

Person: 60x120 pixels

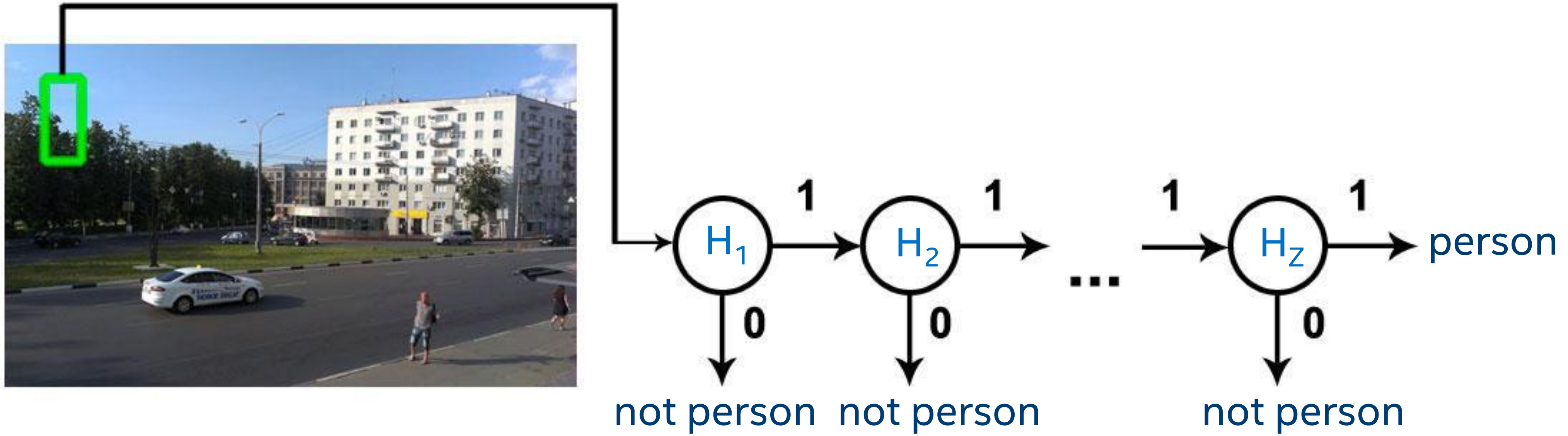


Images credit: http://mi.eng.cam.ac.uk/~cipolla/lectures/4F12/Slides/old/4F12-detection.pdf

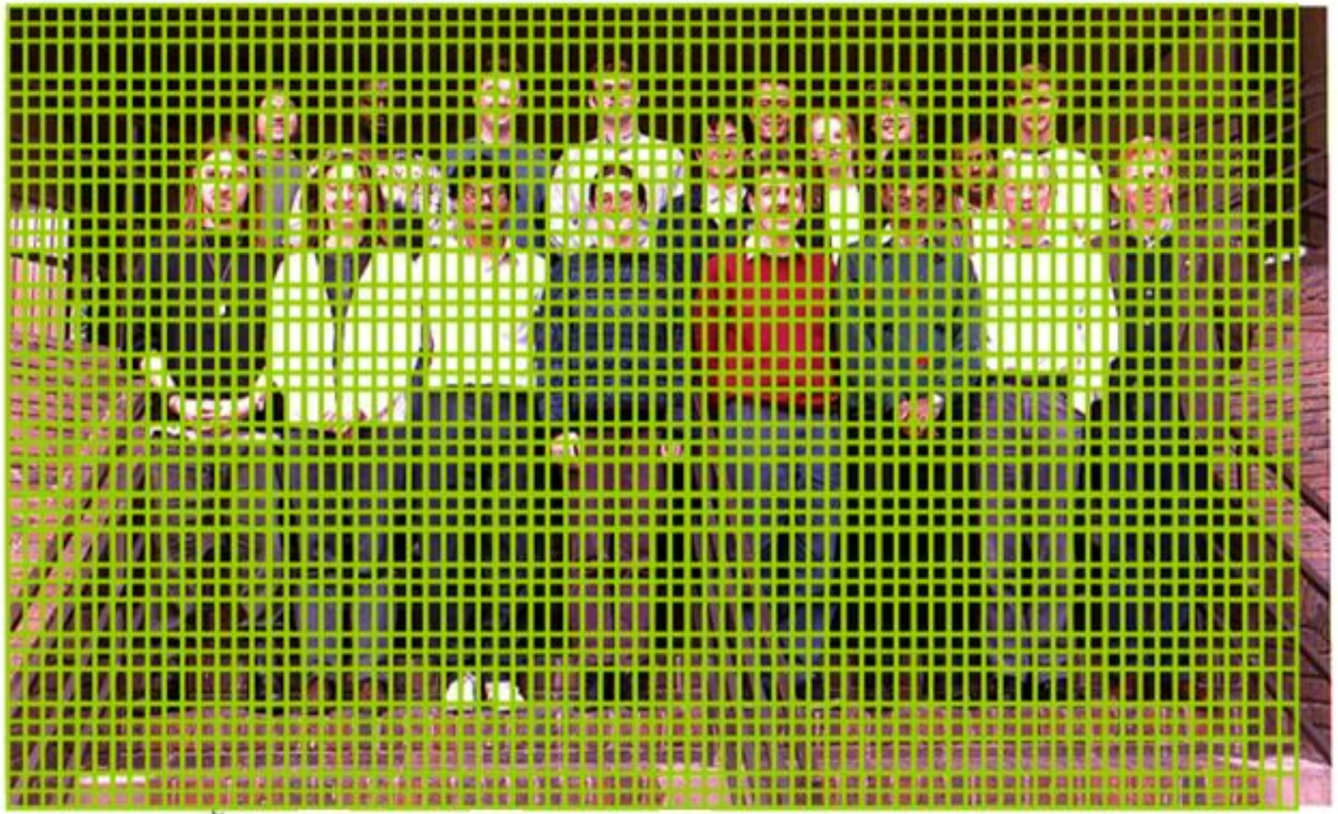
Cascade Classifier

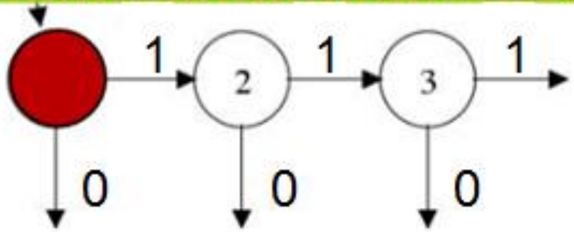
Some location hypotheses can be easily rejected, classified as negative, such as sky, road, uniform regions

No need to evaluate all classifiers in ensemble. Let's split it in *stages*.



Cascade Classifier: Example





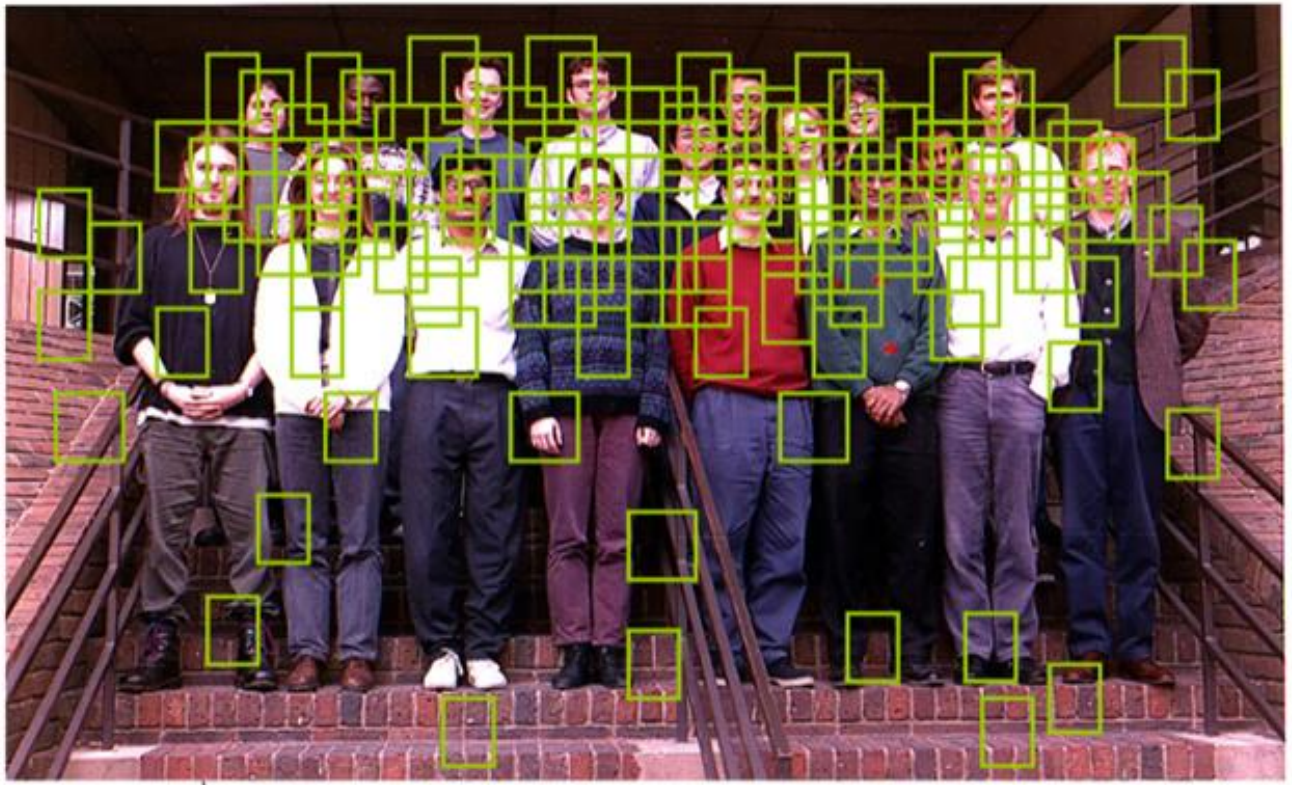
Face

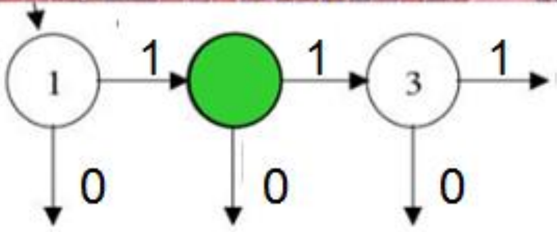
Not face

8192 hypotheses

Images credit: http://mi.eng.cam.ac.uk/~cipolla/lectures/4F12/Slides/old/4F12-detection.pdf

Cascade Classifier: Example





Face

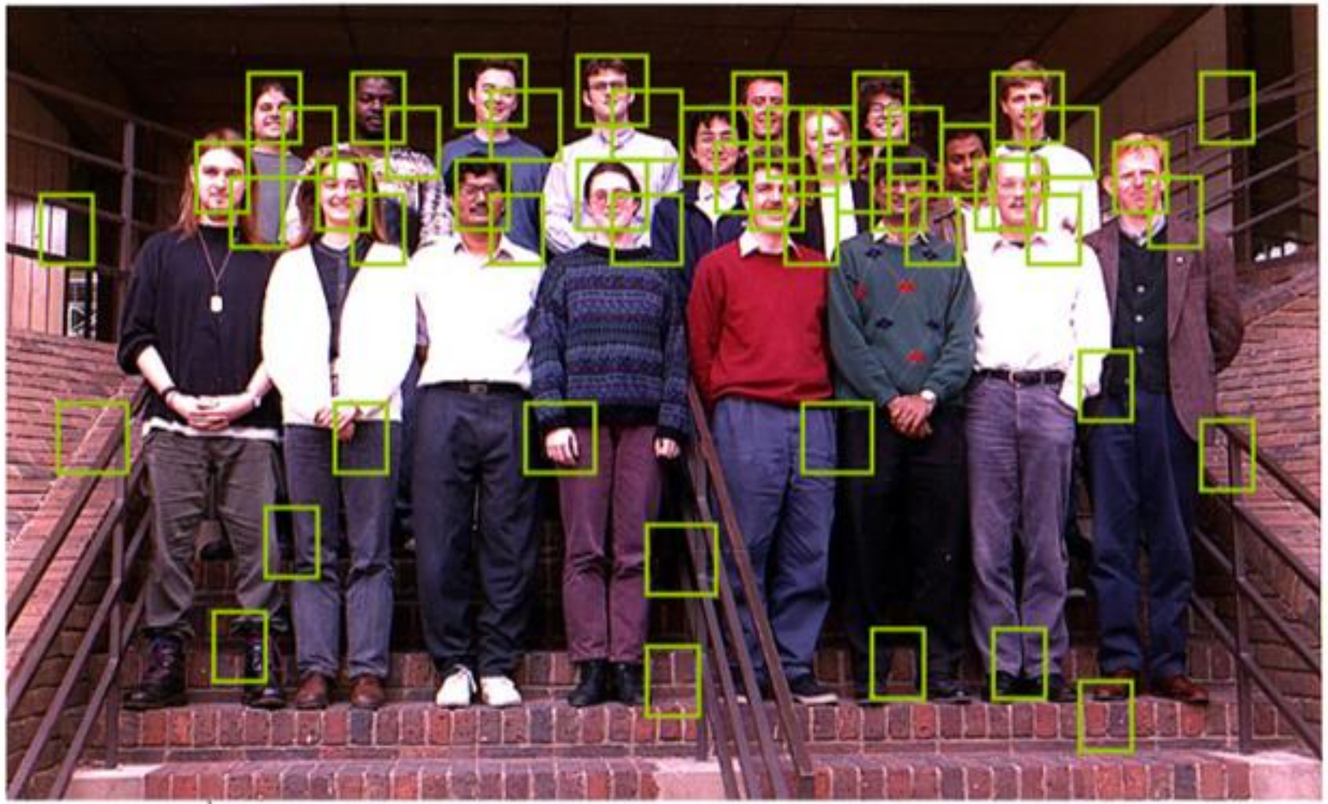
188

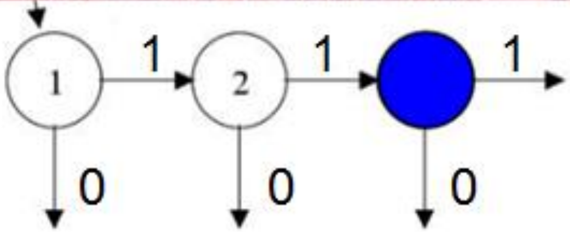
hypotheses

Not face

Images credit: http://mi.eng.cam.ac.uk/~cipolla/lectures/4F12/Slides/old/4F12-detection.pdf

Cascade Classifier: Example





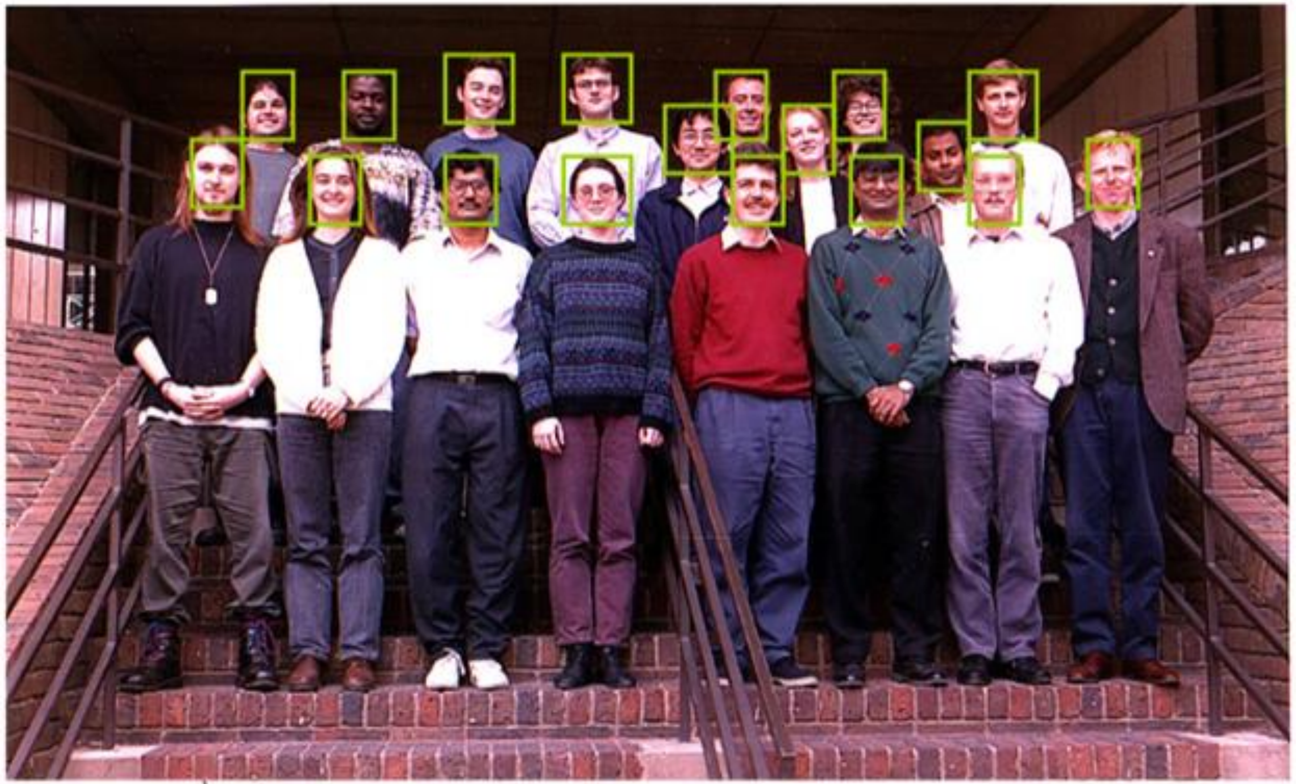
Face

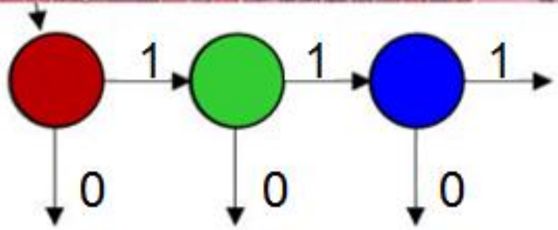
Not face

48
hypotheses

Images credit: http://mi.eng.cam.ac.uk/~cipolla/lectures/4F12/Slides/old/4F12-detection.pdf

Cascade Classifier: Example





Face

18

Faces

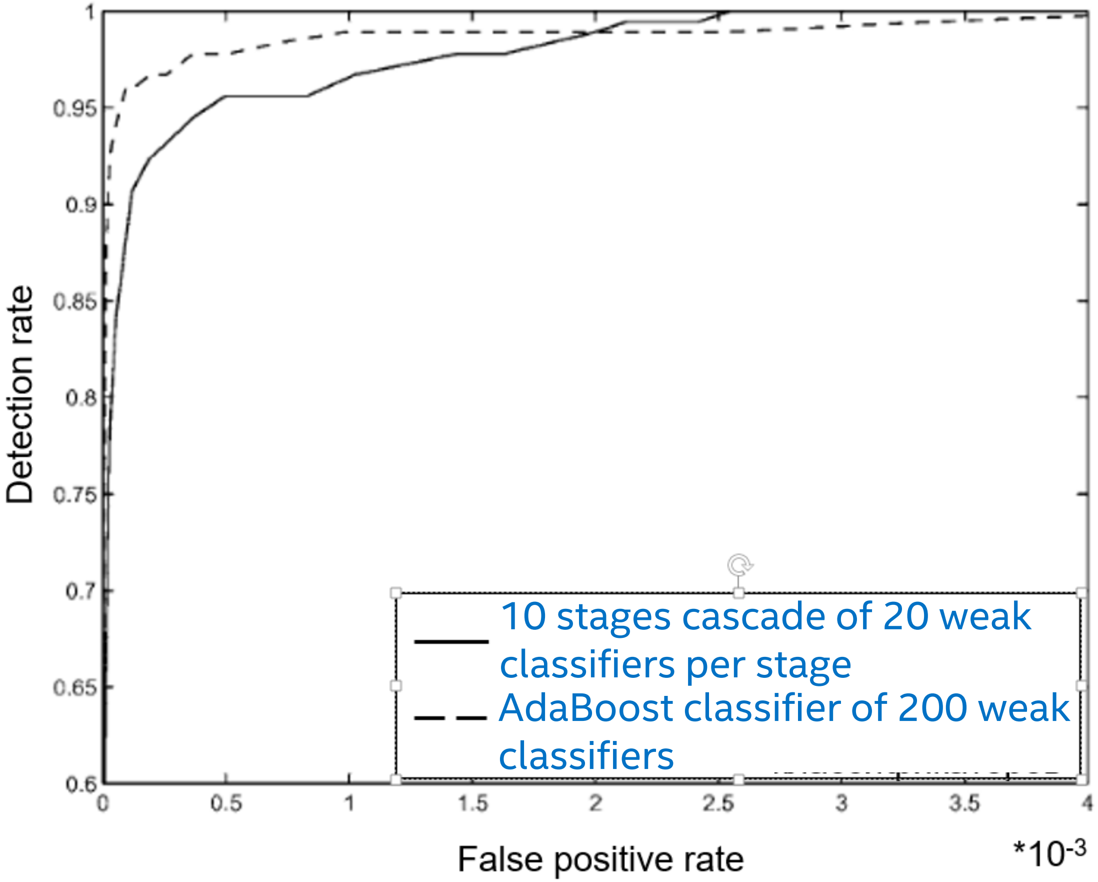
Not face

Images credit: http://mi.eng.cam.ac.uk/~cipolla/lectures/4F12/Slides/old/4F12-detection.pdf

Comparison of AdaBoost with Cascade Classifier (Viola & Jones, 2001)

Paul Viola, Microsoft research

Michael Jones, Merl



384x288 pixels input

Intel Pentium III 700 Mhz

@ 15 fps

Cascade boosts performance in ~10 times with comparable accuracy

Images credit: http://www.vision.caltech.edu/html-files/EE148-2005-Spring/pprs/viola04ijcv.pdf

Can you find a face in the coffe beans?

Images credit: http://mi.eng.cam.ac.uk/~cipolla/lectures/4F12/Slides/old/4F12-detection.pdf

Features

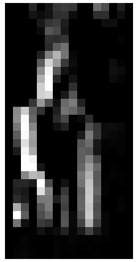
How to form feature vector?

pixels values

x = (50, 54, 52, 54, … , 186)

Feature vector, 512x1 пикселей

gradient values



[-1 0 1]

x = (0, 0, 38, 0, … , 0)

Feature vector, 512x1 пикселей

Features

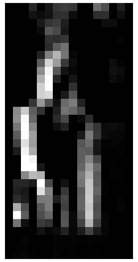
How to form feature vector?

pixels values

x = (50, 54, 52, 54, … , 186)

Feature vector, 512x1 пикселей

gradient values



[-1 0 1]

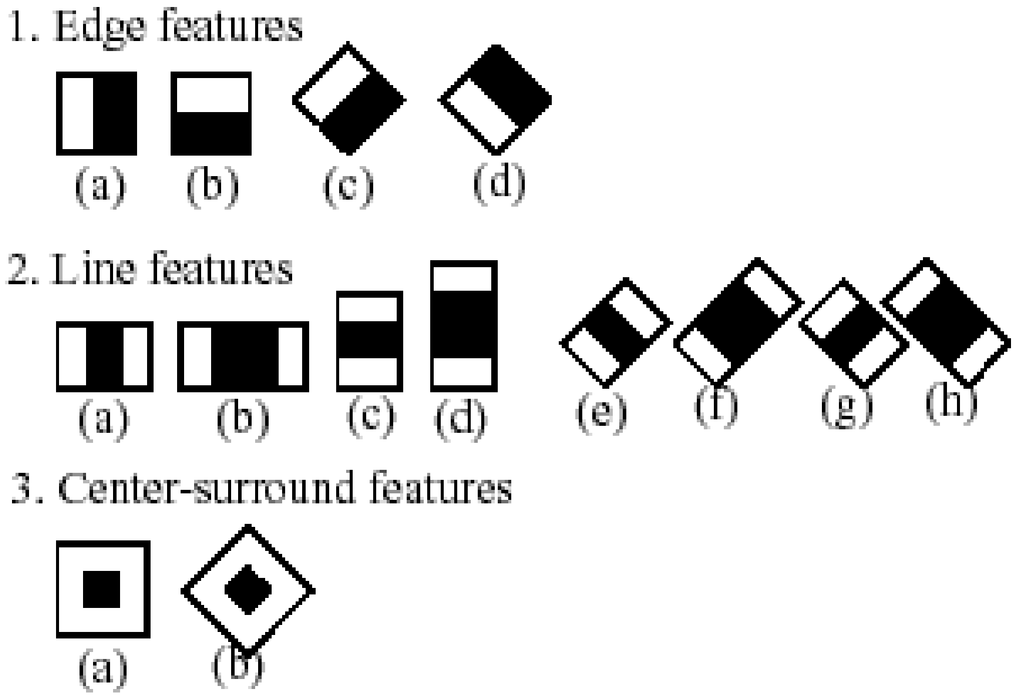
x = (0, 0, 38, 0, … , 0)

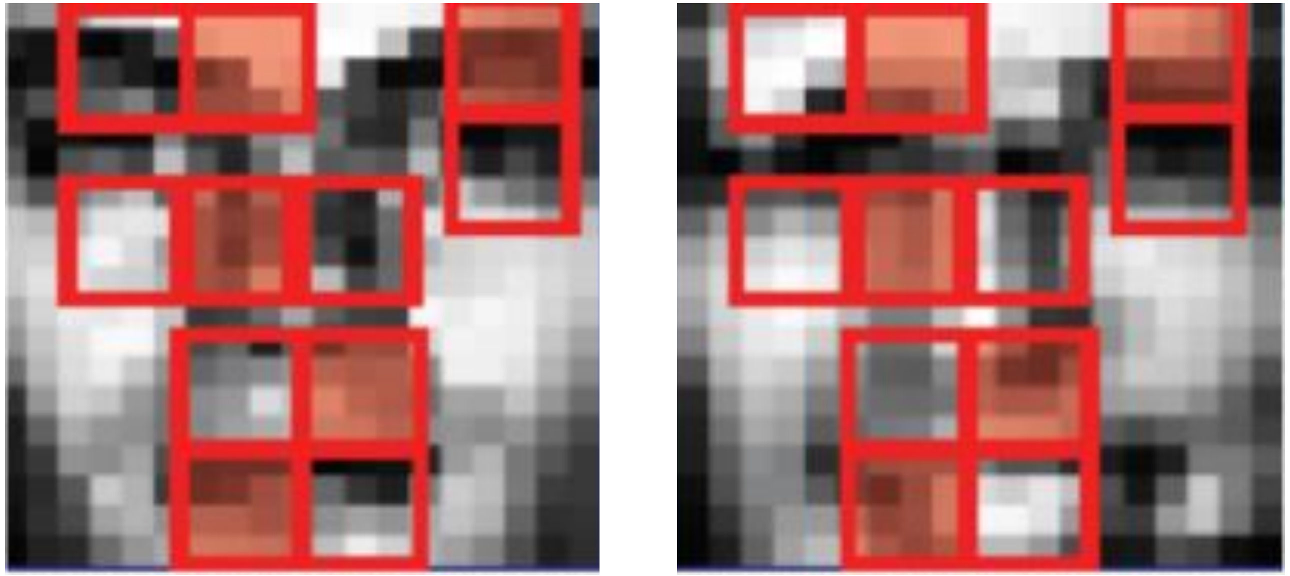
Feature vector, 512x1 пикселей

Which feature better?

Features: Haar-like

Features should be fast to compute at any scale

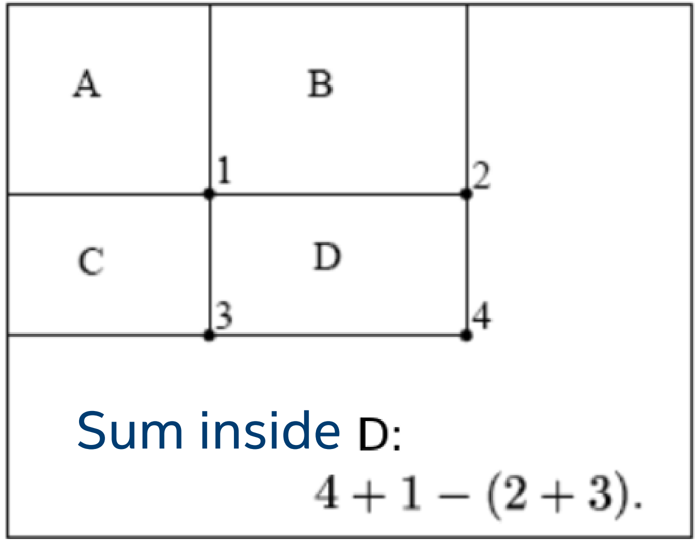




Images credit: https://github.com/itseez-academy/itseez-summer-school-2015-lectures/blob/master/slides/Day%203%20--%20Alexander%20Bovyrin%20--%20Object%20Detection.pdf

Integral Image for Fast Computation

$$ii(x,y) = \sum_{x' \le x, y' \le y} i(x', y'),$$



1	1	1	1
1	1	1	1
1	1	1	1
1	1	1	1

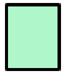
Input image

→

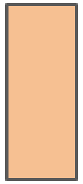
0	0	0	0	0
0	1	2	3	4
0	2	4	6	8
0	3	6	9	12
0	4	8	12	16

Integral image

Sum inside:

 = (4 + 0) - (3 + 0) = 1

 = (12 + 0) - (0 + 8) = 4

 = (12 + 1) - (4 + 3) = 6

Features: Histogram of Oriented Gradients (HoG)

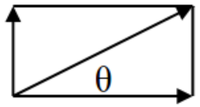
1. Compute horizontal, vertical gradients for each pixel inside the image
2. Compute magnitude, orientation of gradients
3. Divide image (size 64x128) by blocks 16x16 pixels: 2x2 cells by 8x8 pixels (7 * 15 blocks in total)
4. For each block compute histogram of gradient orientation

$$\begin{bmatrix} -1 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$$

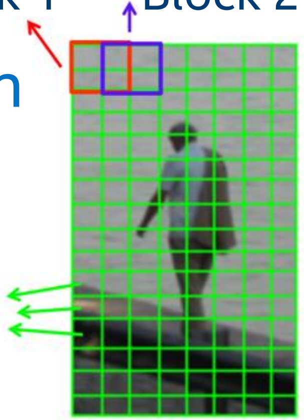
1. Kernels of gradients

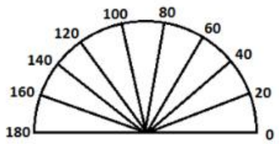
$$s = \sqrt{s_x^2 + s_y^2}$$

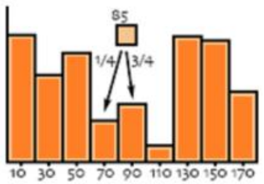
$$\theta = \arctan\left(\frac{s_y}{s_x}\right)$$



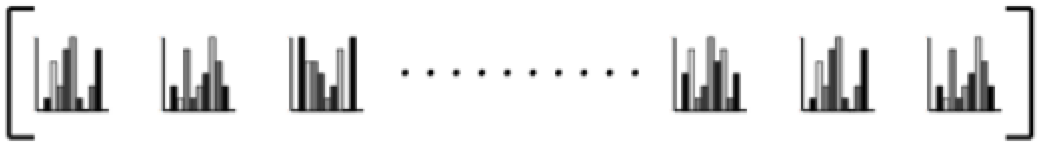
2. Magnitude, orientation (angle)







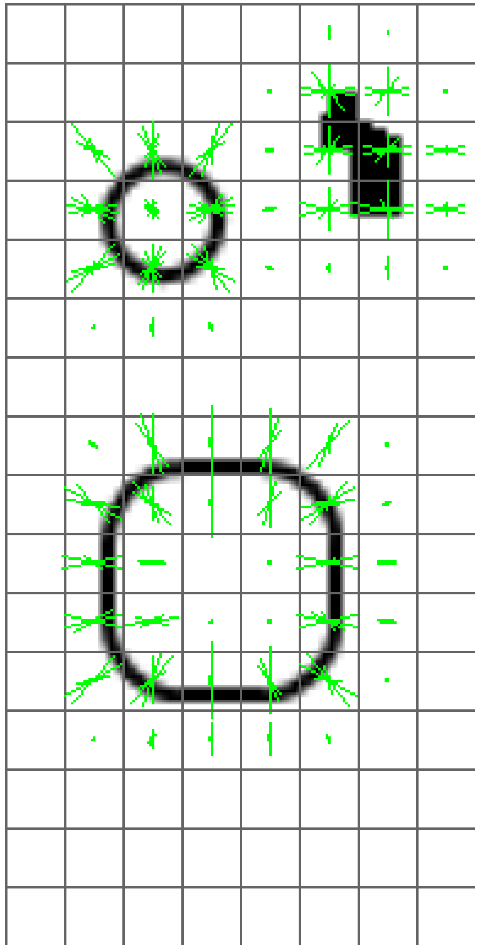
5. Normalize blocks
6. Concatenate blocks histograms into feature vector

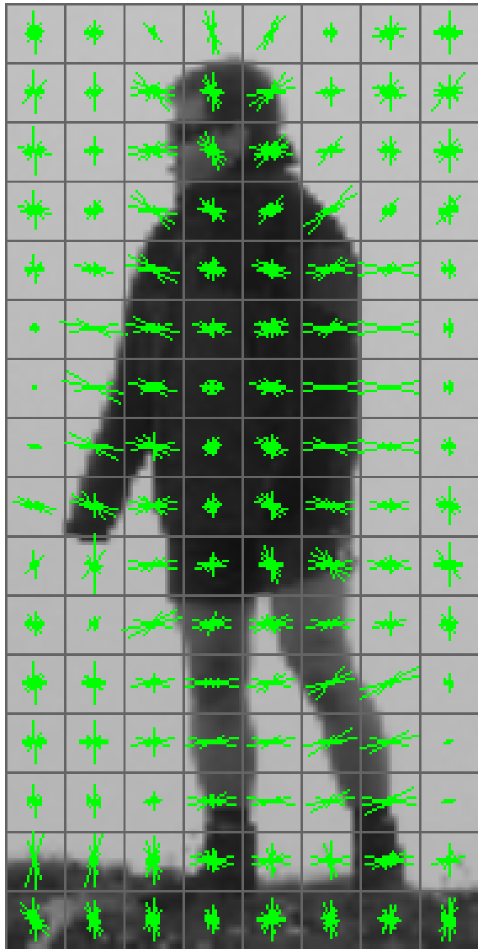


$\in \mathbb{R}^{3780}$

Images credit: http://crcv.ucf.edu/courses/CAP5415/Fall2012/Lecture-6a-Hog.pdf

Features: Histogram of Oriented Gradients (HoG)





Magnitude of gradient in specific direction is visualized in green

Images credit: http://crcv.ucf.edu/courses/CAP5415/Fall2012/Lecture-6a-Hog.pdf

Summary

Classification:

- Decision Tree
- AdaBoost
- Cascade Classifier

Localization:

- Sliding window
- Image pyramid

Features:

- Haar-like
- Histogram of Oriented Gradients (HoG)

Q&A

daniil.osokin@intel.com