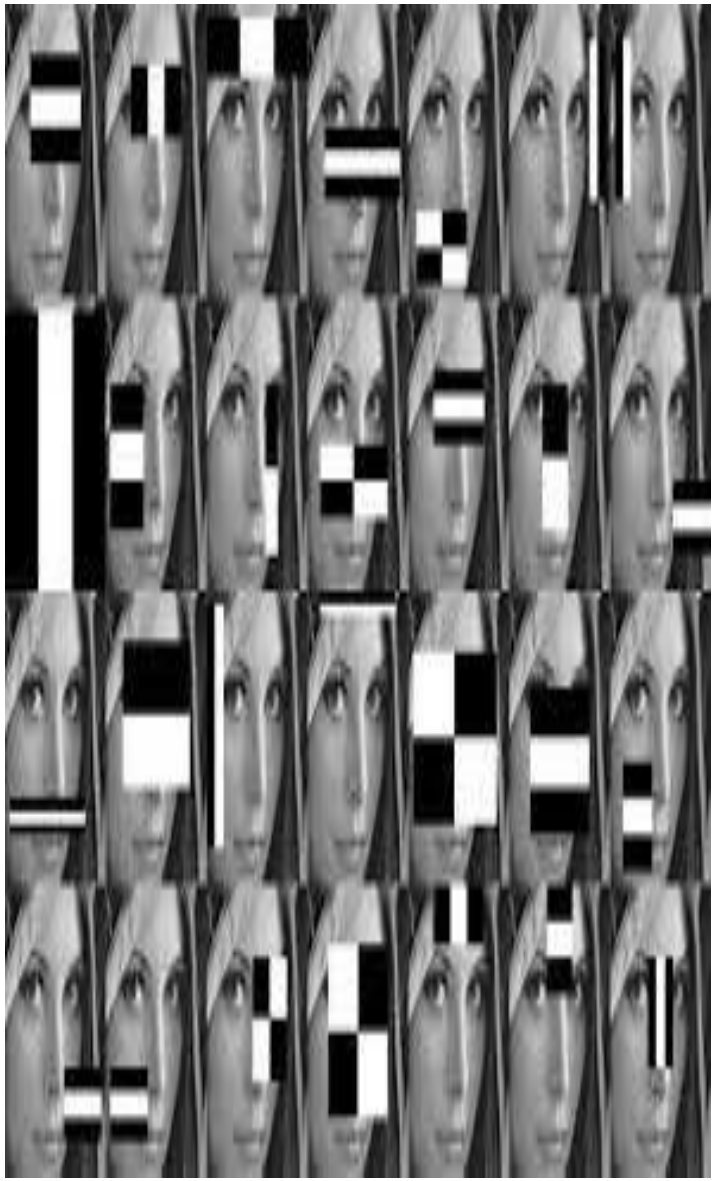


Lecture 16

Harr Cascades

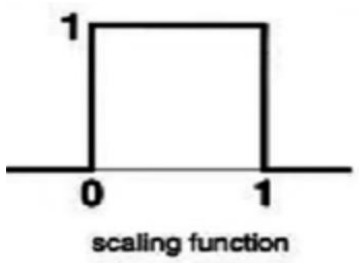
ECE 1390/2390

Haar wavelet

$$H_{2x2} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

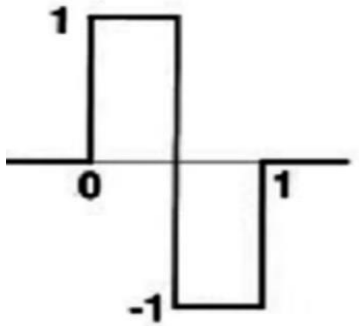
Integrator: Low-pass filter

Differentiator: High-pass filter



scaling function

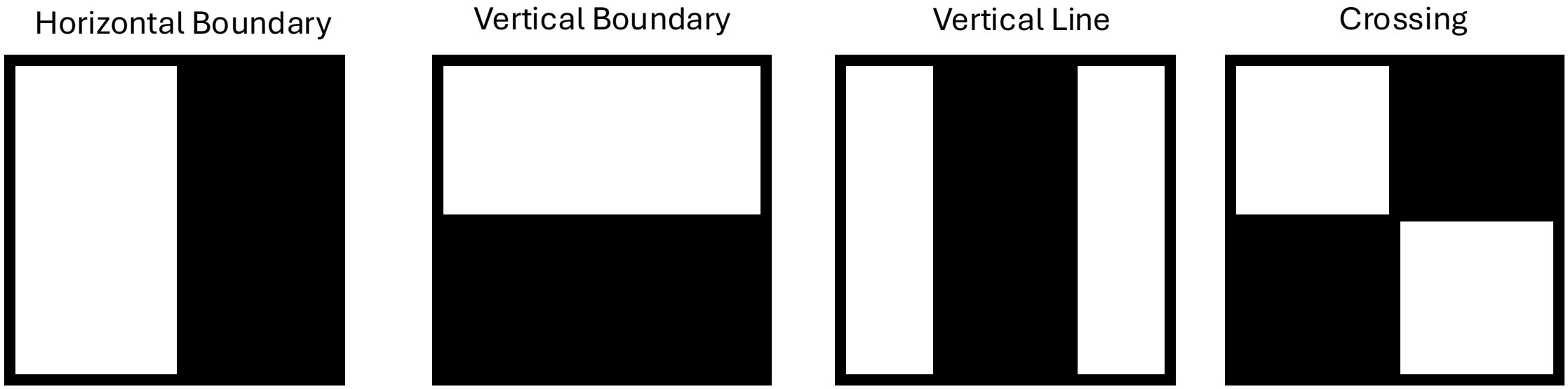
$$\phi(x) = \begin{cases} 1 & 0 \le x < 1 \\ 0 & else \end{cases}$$

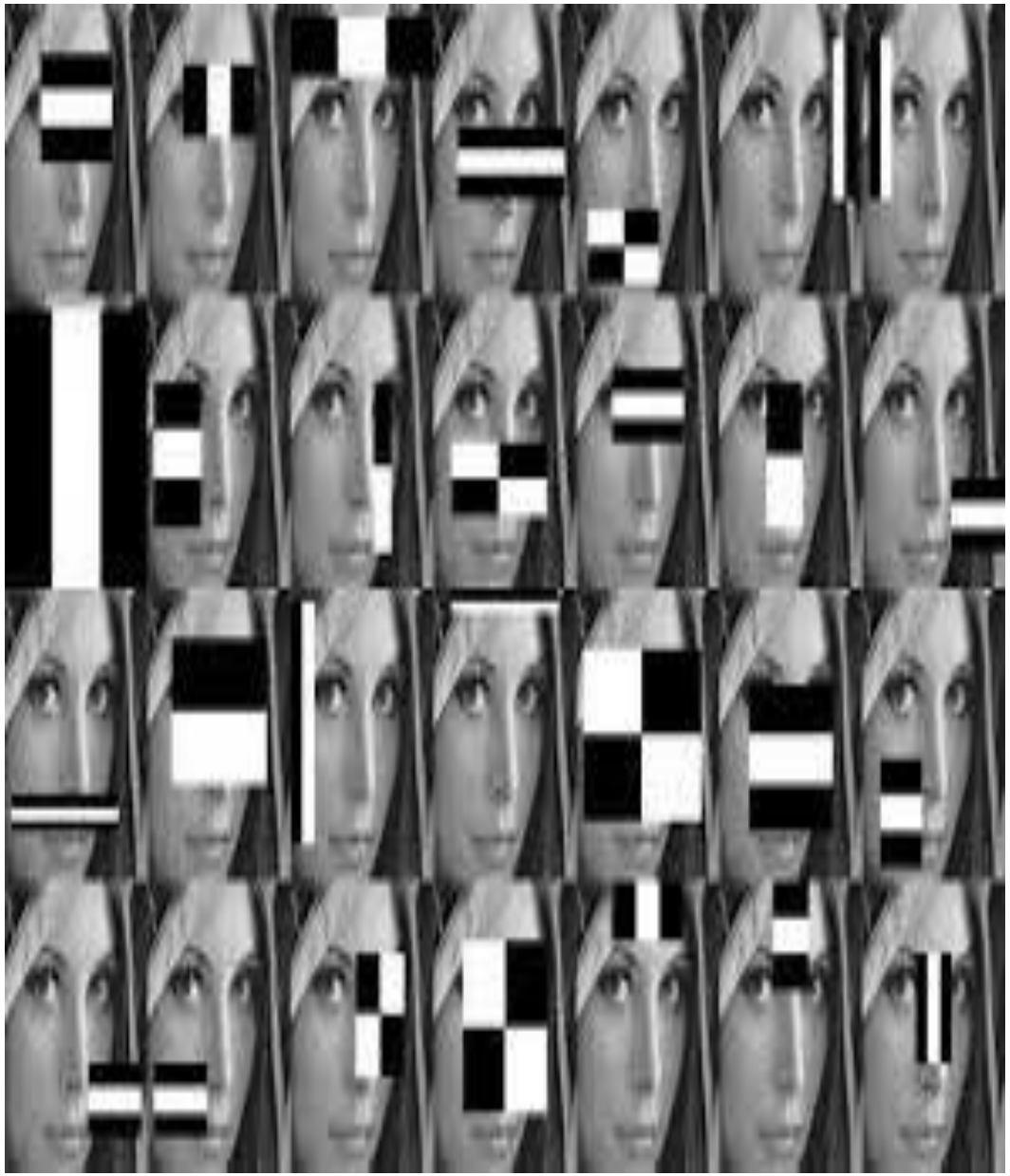


wavelet

$$\psi(x) = \begin{cases} 1 & 0 \le x < 1/2 \\ -1 & 1/2 \le x < 1 \\ 0 & else \end{cases}$$

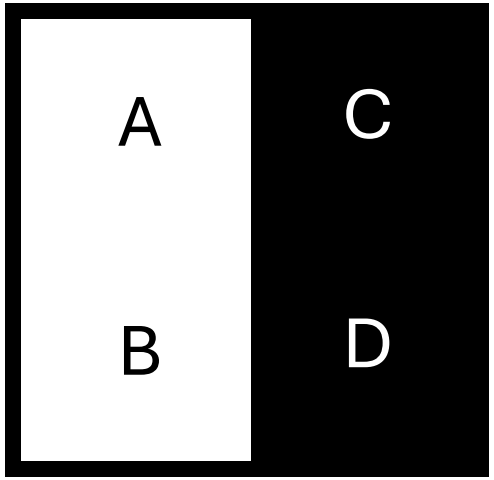
- Viola and Jones, "Rapid object detection using a boosted cascade of simple features", Computer Vision and Pattern Recognition, 2001
  - Haar-like feature extraction kernel (not actual Haar wavelets)
  - Series of convolution kernels to extract features
  - More features than pixels
    - E.g. 24 x 24 probe has 18,000 possible rectangle arrangements
  - `Uses only small subset of the possibilities



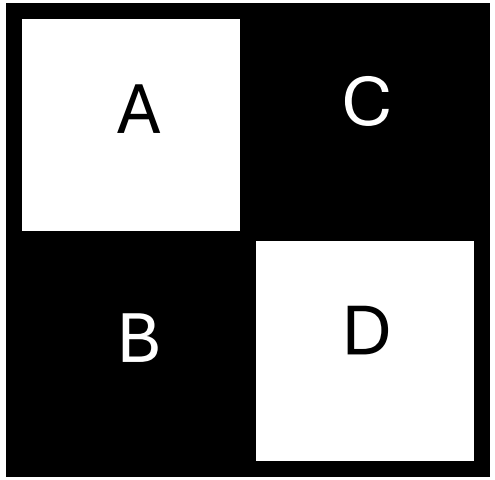


Feature Extraction:

Sum over rectangular windows



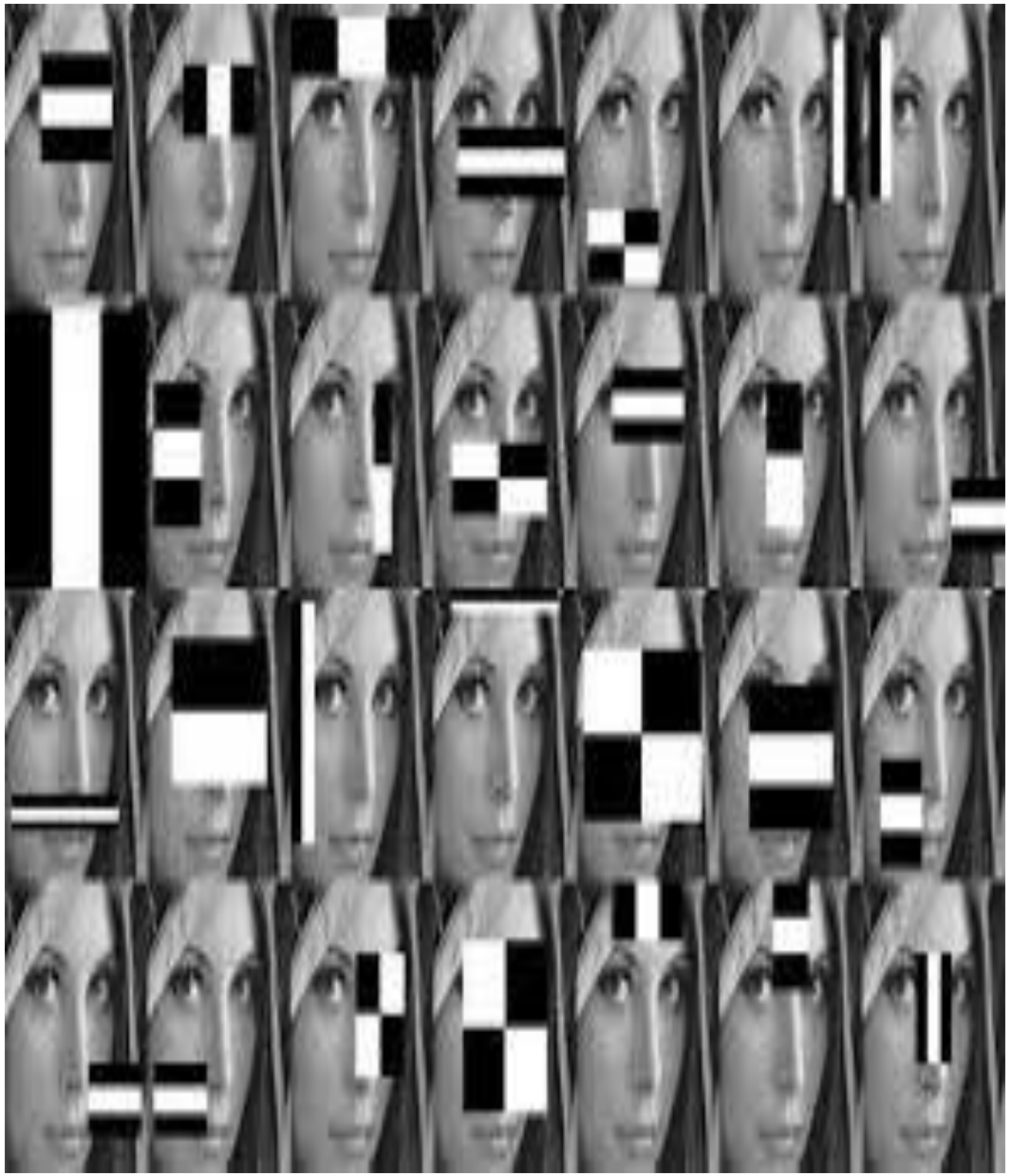
A+B – (C+D)



A+D – (B+C)

→ 28 features

Each feature is very weakly predictive



Adaboost (Adaptive boosting)

$$F = \sum_i f_i(x)$$

Boasted classifier

Weak binary (-1,1) classifiers

Yoav Freund and Robert Schapire (1995), *A desicion-theoretic generalization of on-line learning and an application to boosting*, Lecture Notes in Computer Science, Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 23–37

Positive Training Data

$$F = \sum_i \alpha_i \cdot f_i(x)$$

Boasted classifier

Weak binary (-1,1) classifiers

Training Data

$$\begin{bmatrix} 1 \\ -1 \\ \vdots \end{bmatrix} = \begin{bmatrix} f_0(P_0) & f_1(P_0) & f_2(P_0) & \\ f_0(P_1) & f_1(P_1) & f_2(P_2) & \cdots \\ & \vdots & & \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \vdots \end{bmatrix}$$

Negative Training Data

Positive Training Data

$$F = \sum_i e^{\alpha_i} \cdot f_i(x)$$

Boasted classifier

Weak binary (-1,1) classifiers

Negative Training Data

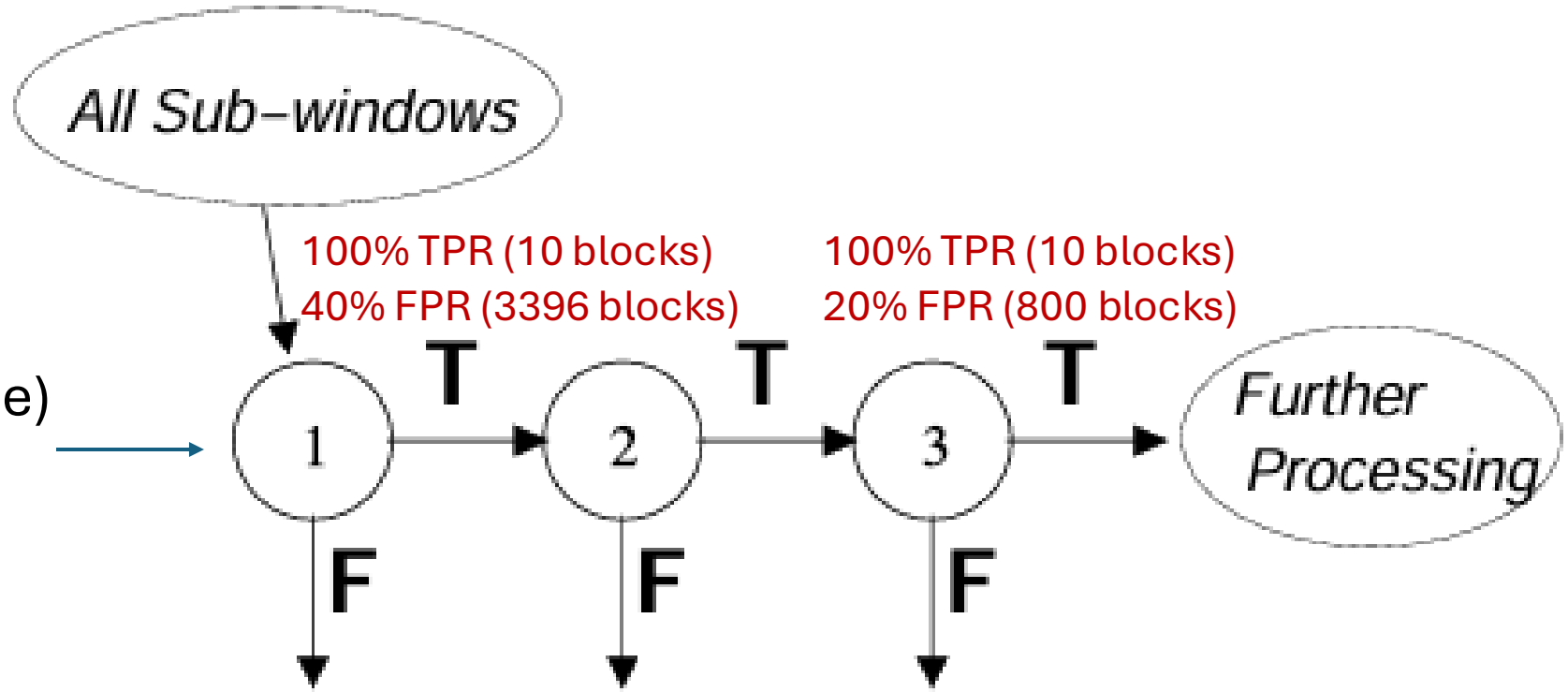
Training Data

$$\begin{bmatrix} 1 \\ -1 \\ \vdots \end{bmatrix} = \begin{bmatrix} f_0(P_0) & f_1(P_0) & f_2(P_0) & \cdots \\ f_0(P_1) & f_1(P_1) & f_2(P_2) & \\ & \vdots & & \end{bmatrix} \begin{bmatrix} e^{\alpha_0} \\ e^{\alpha_1} \\ e^{\alpha_2} \\ \vdots \end{bmatrix}$$

Ensure that coefficients are non-negative

Classifier Cascade

Most negatives are discarded in first stages, reducing computation



In original paper:

- 6000+ features with 38 stages
- 1, 10, 25, 25 and 50 features in the first five stages.

Training Haar models

Good working example:

https://github.com/bilardi/how-to-train-cascade

Not included in python install (need to install from source code)

https://github.com/opencv/opencv/apps/annotation

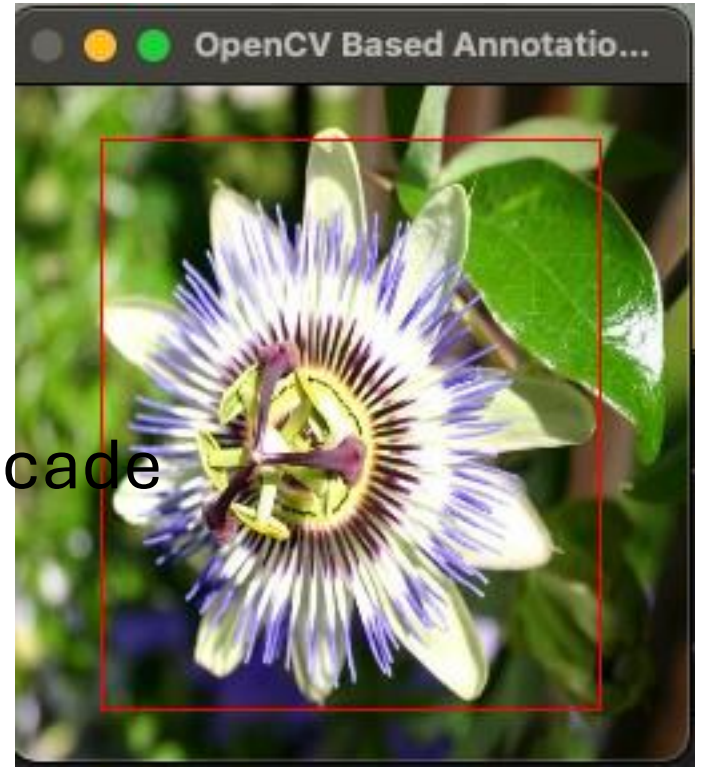
... /createsamples

... /traincascade

Training Haar models

Good working example:

https://github.com/bilardi/how-to-train-cascade



opencv_annotation

For interactive labeling of images using left mouse click followed by keystrokes

e.g. ./opencv/build/bin/opencv_annotation \
--images=/Users/theodorehuppert/Desktop/Haa/training/positives/ \
--annotations=/Users/theodorehuppert/Desktop/Haa/training/annotations.txt

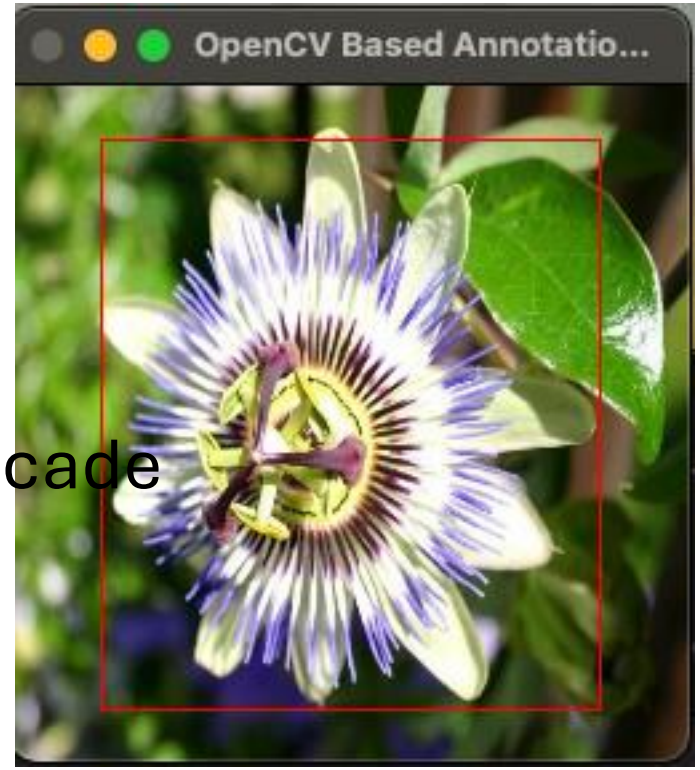
Note- if you don't give it the full pathnames, it doesn't do anything (but also gives no help info)
Must hit "C' (red box turns green) to add annotation. Each image can have multiple annotations

- Pressing c : confirm the annotation, turning the annotation green and confirming it is stored
- Pressing d : delete the last annotation from the list of annotations (easy for removing wrong annotations)
- Pressing n : continue to the next image
- Pressing ESC : this will exit the annotation software

Training Haar models

Good working example:

https://github.com/bilardi/how-to-train-cascade



opencv_annotation

For interactive labeling of images using left mouse click followed by keystrokes

e.g. ./opencv/build/bin/opencv_annotation \
--images=/Users/theodorehuppert/Desktop/Haa/training/positives/ \
--annotations=/Users/theodorehuppert/Desktop/Haa/training/annotations.txt

/Users/theodorehuppert/Desktop/Haa/training/positives/image_00001.jpg 1 28 12 198 230
/Users/theodorehuppert/Desktop/Haa/training/positives/image_00002.jpg 1 45 17 194 231
/Users/theodorehuppert/Desktop/Haa/training/positives/image_00003.jpg 1 22 68 221 179
/Users/theodorehuppert/Desktop/Haa/training/positives/image_00004.jpg 1 63 31 154 206
/Users/theodorehuppert/Desktop/Haa/training/positives/image_00005.jpg 1 76 49 162 173

Training Haar models

Good working example:

https://github.com/bilardi/how-to-train-cascade

opencv_annotation

Usage: ./opencv/build/bin/opencv_traincascade

-data: Folder (must exist) to store output data
-vec: Samples vector created by createsamples function
-bg: Negative samples list (TXT)
-numPos: Number of samples used in training for every stage
-numNeg: “

Useful options

-precalCalValBufSize: Preallocated memory (in Mb)
-precalCalIdxBufSize: Preallocated memory (in Mb)
-numThreads
-acceptanceRatioBreakValue: When to stop training model

Cascade Parameters

-numStages: number of stages (default=20)
-featureType: HAAR or LBP (local binary pattern)
-w: Width of samples (must match createsamples)
-h: Height of samples (must match createsamples)

-bt: AdaBoost type. {DAB, RAB, LB, GAB}
-minHitRate: Desired positive hit rate for each stage
-maxFalseAlarmRate: Max false positives for each stage

-mode: BASIC- only vertical HAAR. ALL—also 45degree rotated

Training Haar models

Good working example:

https://github.com/bilardi/how-to-train-cascade

opencv_traincascade

Usage: ./opencv/build/bin/opencv_traincascade

-data: Folder (must exist) to store output data
-vec: Samples vector created by createsamples function
-bg: Negative samples list (TXT)
-numPos: Number of samples used in training for every stage
-numNeg: “

Useful options

-precalCalValBufSize: Preallocated memory (in Mb)
-precalCalIdxBufSize: Preallocated memory (in Mb)
-numThreads
-acceptanceRatioBreakValue: When to stop training model

Cascade Parameters

-numStages: number of stages (default=20)
-featureType: HAAR or LBP (local binary pattern)
-w: Width of samples (must match createsamples)
-h: Height of samples (must match createsamples)

-bt: AdaBoost type. {DAB, RAB, LB, GAB}
-minHitRate: Desired positive hit rate for each stage
-maxFalseAlarmRate: Max false positives for each stage

-mode: BASIC- only vertical HAAR. ALL—also 45degree rotated

- git clone https://github.com/bilardi/how-to-train-cascade
- cd how-to-train-cascade
- find ./positive_images -iname "*.jpg" > positives.txt
- find ./negative_images -iname "*.jpg" > negatives.txt

Note in the git repo, info.dat has already been created for you

- ../opencv/build/bin/opencv_createsamples -info info.dat -num 37 -w 80 -h 80 -vec samples.vec

- ../opencv/build/bin/opencv_traincascade -data classifier \
  -vec samples.vec -bg negatives.txt -numStages 50 \
  -minHitRate 0.999 -maxFalseAlarmRate 0.5 -numPos 37 \
  -numNeg 16 -w 80 -h 80 -precalcValBufSize 1024
  -precalcIdxBufSize 1024 -featureType HAAR -mode BASIC

Training Haar models

Good working example:

https://github.com/bilardi/how-to-train-cascade

Suggestions

- HAAR classifers work for “rigid, textured objects”. (e.g. faces onward, but faces side-view would need a different filter)
- Want to have around 1000-2000 positive images
- Want to have about 2:1 positive:negative images

- Total false discovery is (maxFalseAlarmRate [per stage])^number stages
- Total true positive rate is: (minHitRate [per stage])^number stages

  -minHitRate: Desired positive hit rate for each stage

  -maxFalseAlarmRate: Max false positives for each stage

Training Haar models

Useful list of a bunch of free (large) curated datasets:
https://imerit.net/blog/28-free-image-datasets-for-computer-vision-all-pbm/

https://image-net.org/. [requires registration with a .edu email]. (Basis for ResNet)

https://www.robots.ox.ac.uk/~vgg/data/

Trained models

https://github.com/opencv/opencv/data/haarcascades

eyes, front faces, profile faces, fullbody, upper body, lower body, smile, Russian license plates (???)

HOG (Histogram of oriented gradients)

https://github.com/lcit/HOG

Sobel filter

Low-pass filter

High-pass filter

$$\begin{vmatrix} 1 \\ 2 \\ 1 \end{vmatrix} * \underline{\begin{matrix} 1 & 0 & -1 \end{matrix}} = \begin{vmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{vmatrix}$$

1D Gaussian Filter

x - Derivative

Sobel - X

$$\begin{vmatrix} 1 \\ 0 \\ -1 \end{vmatrix} * \underline{\begin{matrix} 1 & 2 & 1 \end{matrix}} = \begin{vmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{vmatrix}$$

y - Derivative

1D Gaussian Filter

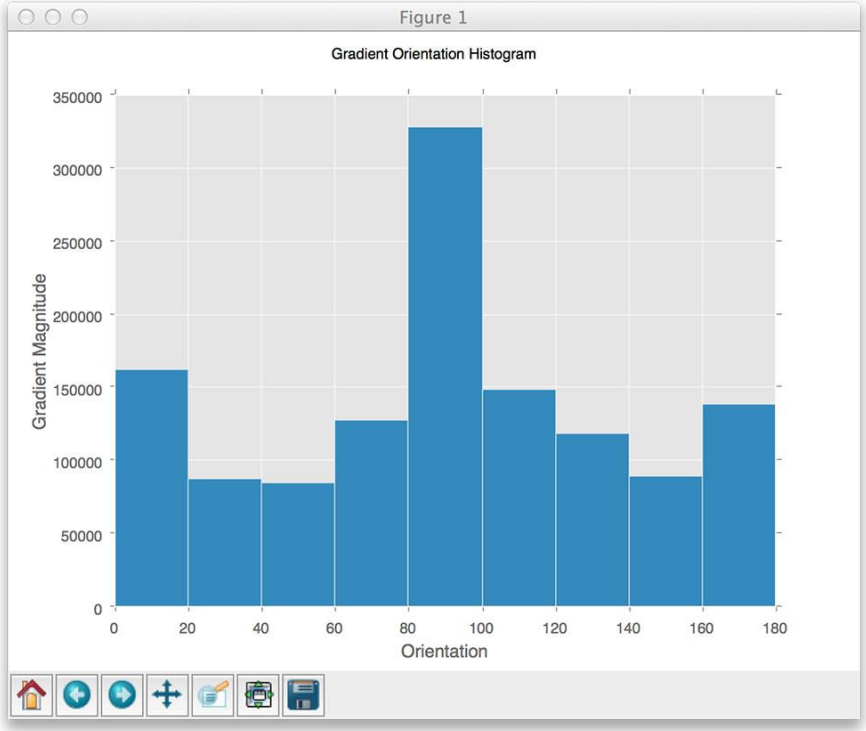
Sobel - Y

$$G = \sqrt{G_X{}^2 + G_Y{}^2}$$

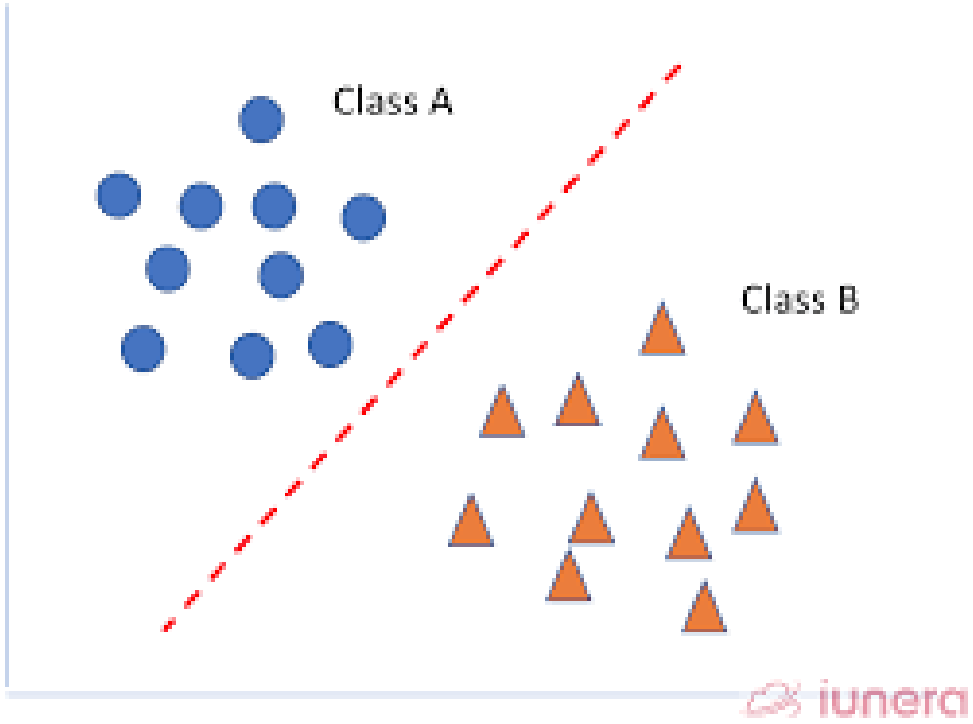
$$\theta = \tan^{-1}\left(\frac{GX}{GY}\right)$$

HOG (Histogram of oriented gradients)

Each block position provides
9 feature values (#bins used in this case)



HOG (Histogram of oriented gradients)



SVM Classifier

Given positive/negative samples (same as Haar cascade) , compute the HOG features for each sample → N features

+1 : Positive

0: Negative

Construct feature space matrix (X) <# features x examples> and label vector (Y = ∈ [0,1])

Find hyperplane (w) that minimizes

$$\arg\min_w \; w^T \cdot X - y = 0$$

HOG (Histogram of oriented gradients)

```
hog = cv2.HOGDescriptor()
hog.setSVMDetector(cv2.HOGDescriptor_getDefaultPeopleDetector())

# Detect people in the image
(rects, weights) = hog.detectMultiScale(imgray, winStride=(4, 4),
                                        padding=(8, 8), scale=1.05)
```