ACCEPTED CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION 2001

Rapid Object Detection using a Boosted Cascade of Simple Features

Paul Viola
viola@merl.com
Mitsubishi Electric Research Labs
201 Broadway, 8th FL
Cambridge, MA 02139

Michael Jones
mjones@crl.dec.com
Compaq CRL
One Cambridge Center
Cambridge, MA 02142

Viola-Jones Object Detection

By Fiseha B. Tesema, PhD

Outline

- Introduction to Face detection
- Application
- The Viola/Jones Face Detector

Face Detection

- Locate human face in images
- Basic idea: slide a window across image and evaluate a face model at every location.

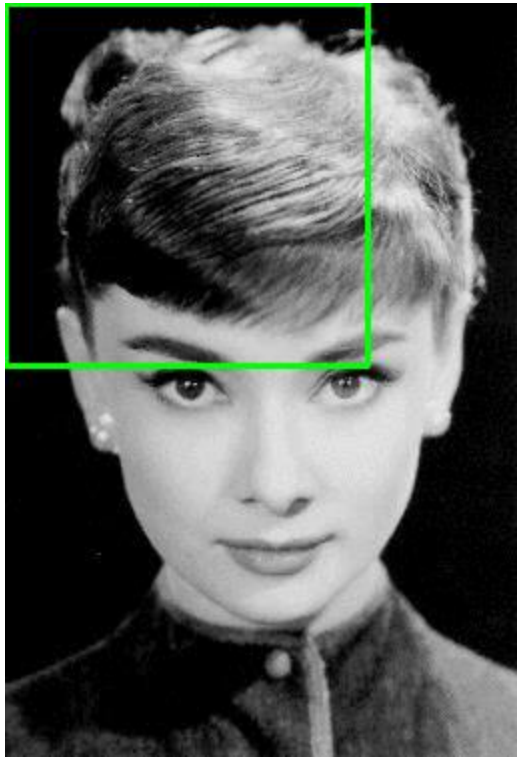


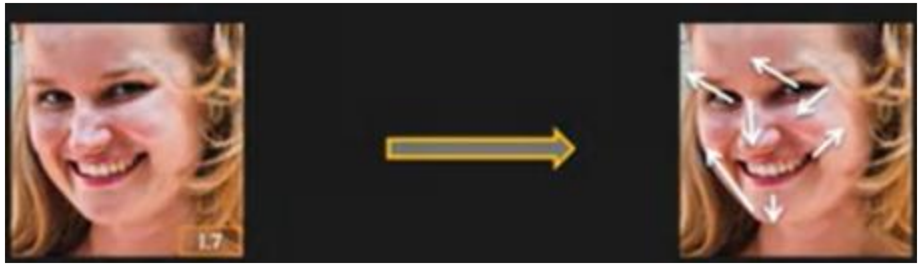
Figure 1: Example of the sliding a window approach, where we slide a window from left-to-right and top-to-bottom

Face Detection Framework

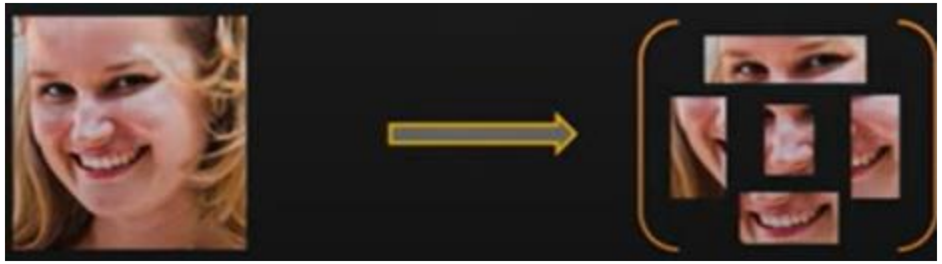
- For each window:
  - Features: which feature represent faces well?
  - Classifier: How to construct a face model and efficiently classify features as face or not?

What are Good Features for detection?

- Interest Points (Edges, Corners, SIFT)?

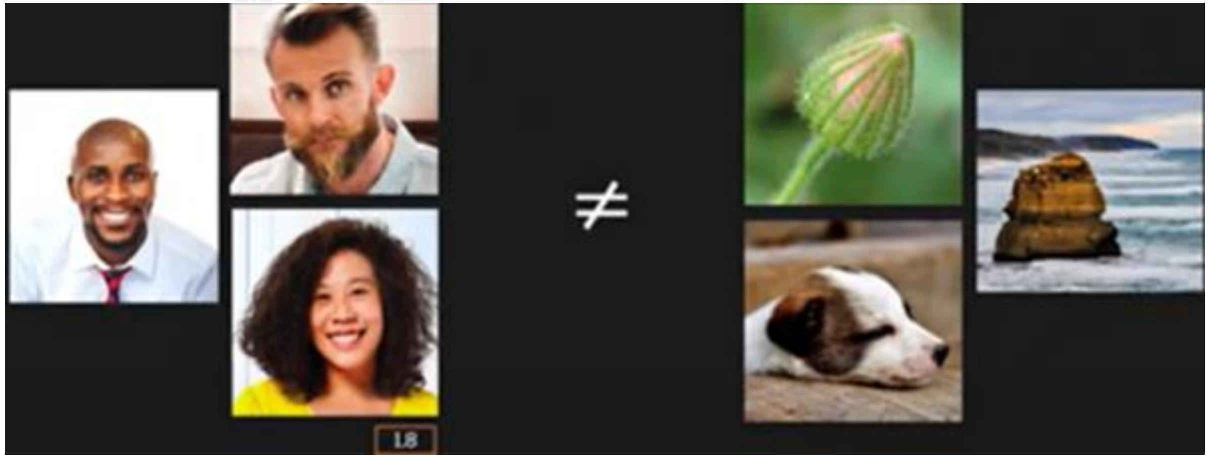


- Facial Components (templates)?



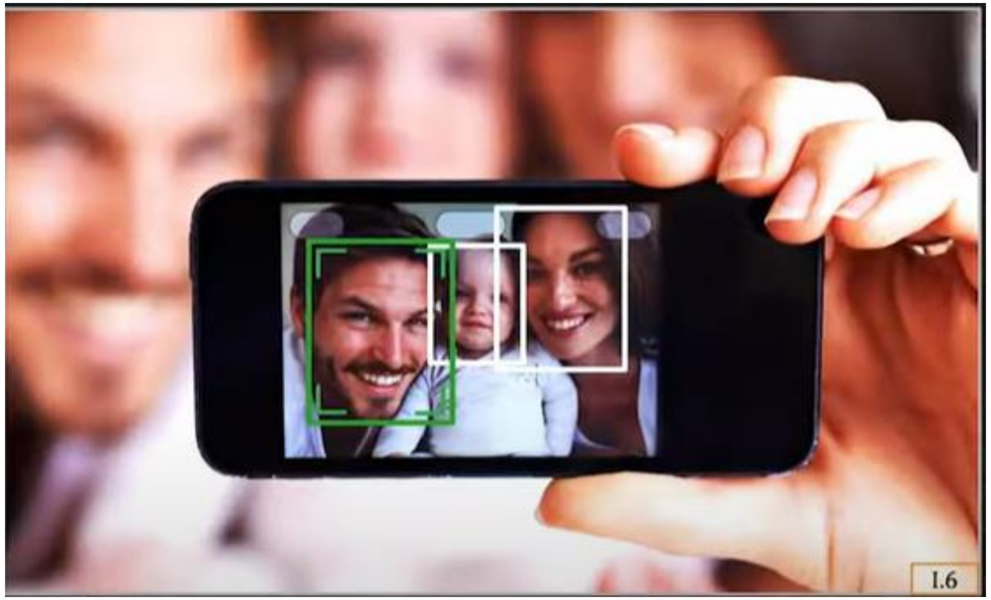
Characteristics of Good Features

- Discriminate Face/Non-Face



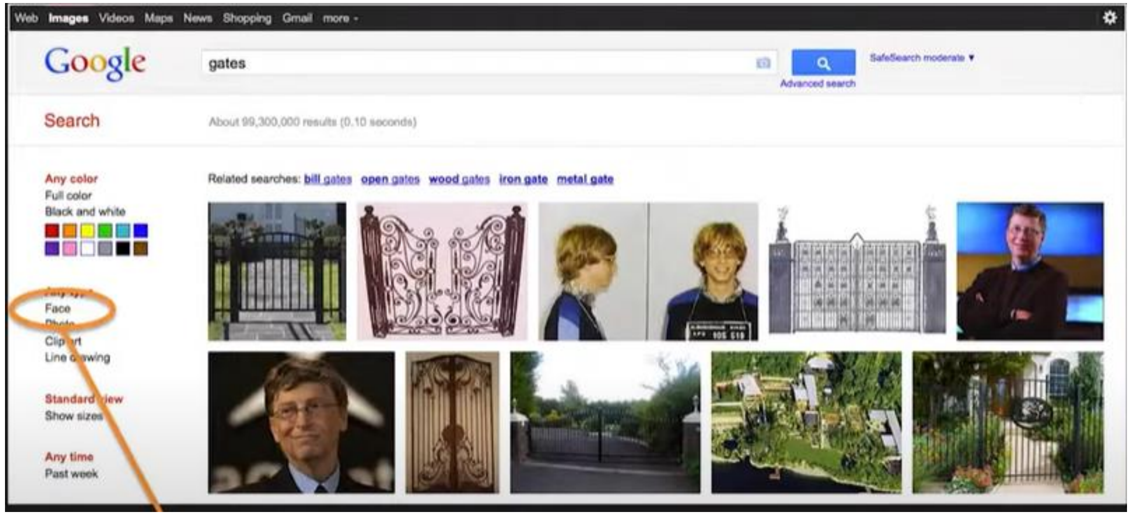
- Extremely Fast to Compute
  - Need to evaluate millions of windows in an image

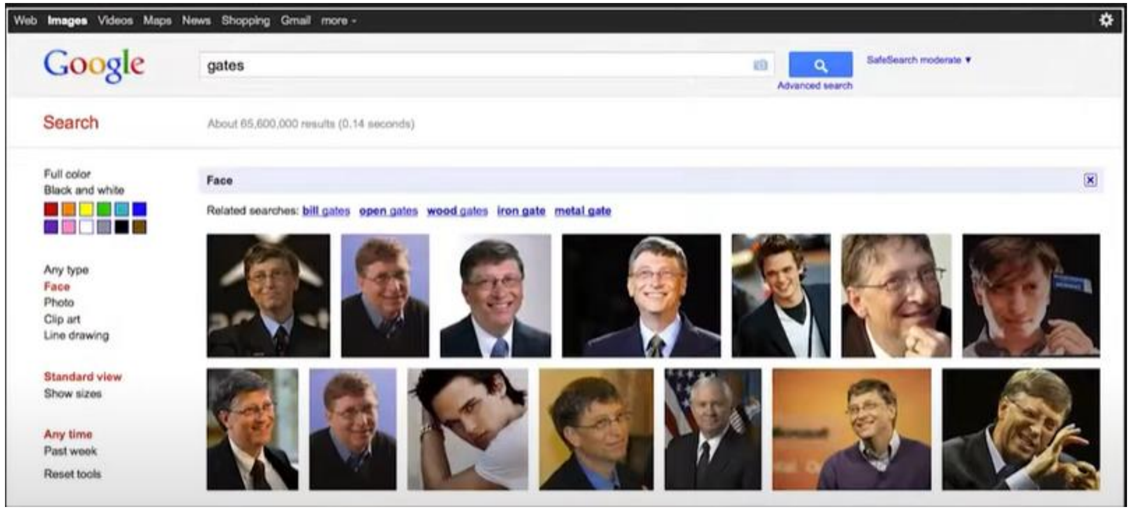
Application



- Automatic Selection of Camera settings (autofocus, exposure, color Balance, etc.)

Application



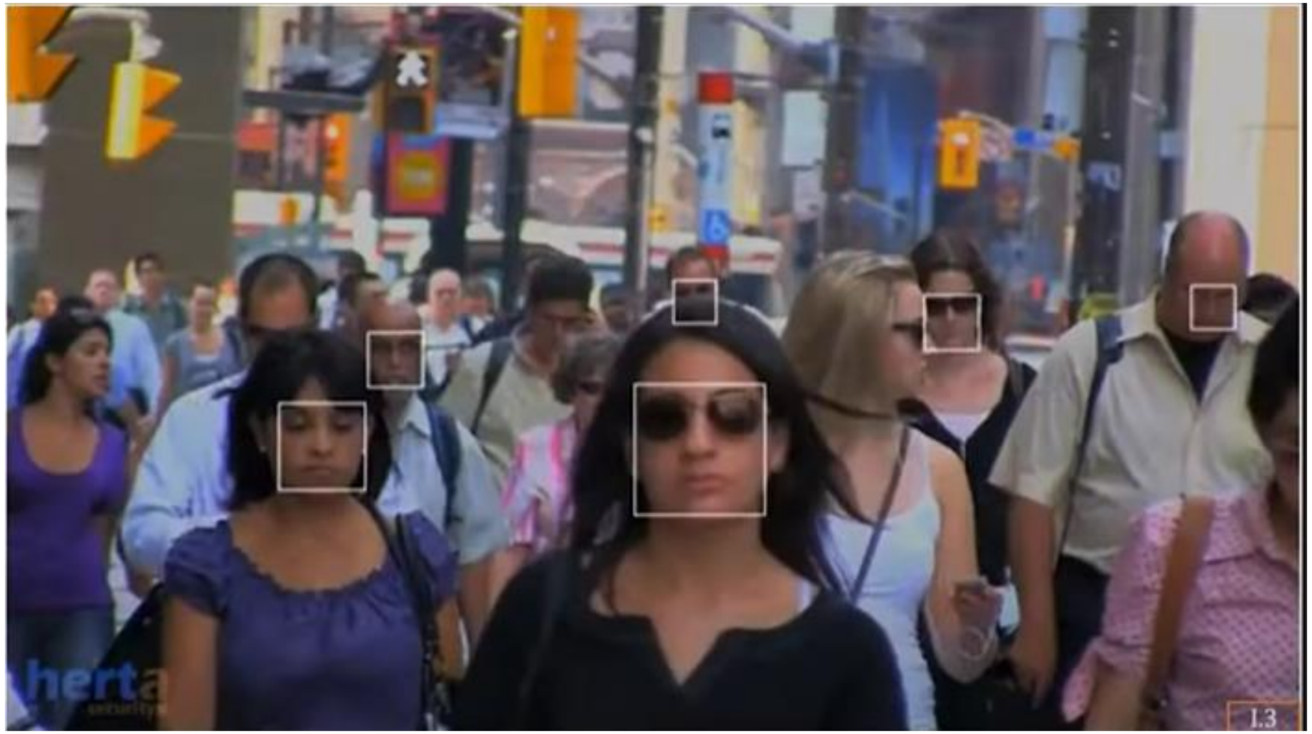


- Finding People using Search Engines

Application

- Intelligent Marketing

Application



- Biometrics, Surveillance, Monitoring

The Viola/Jones Face Detector

- A seminal approach to real-time object detection.
- Training is slow, but detection is very fast
- Key ideas:
  - **Haar-like Features-** simple rectangular features that achieves just above random
  - **Calculating the integral Image-** summed area table necessary for quick calculation
  - **AdaBoost Learning Algorithm-** creates a small set of only the **best features** to create more efficient classifiers.
  - **Cascade Filter-** discards negative windows early to focus more computational time on possible positive windows

Haar-like Features (Haar Features)

- Haar Wavelets were proposed by mathematician Alfred Haar in 1909 and are used in applications such as signal and image compression in electrical and computer engineering.
- To put simply: Haar Features are essentially collections of **pixels in rectangular shapes**.
- Haar features are conceptually similar to kernels in convolutional neural nets.

[https://en.wikipedia.org/wiki/Alfr%C3%A9d_Haar]

(a) (b)

Detect Edge (Edge Features)

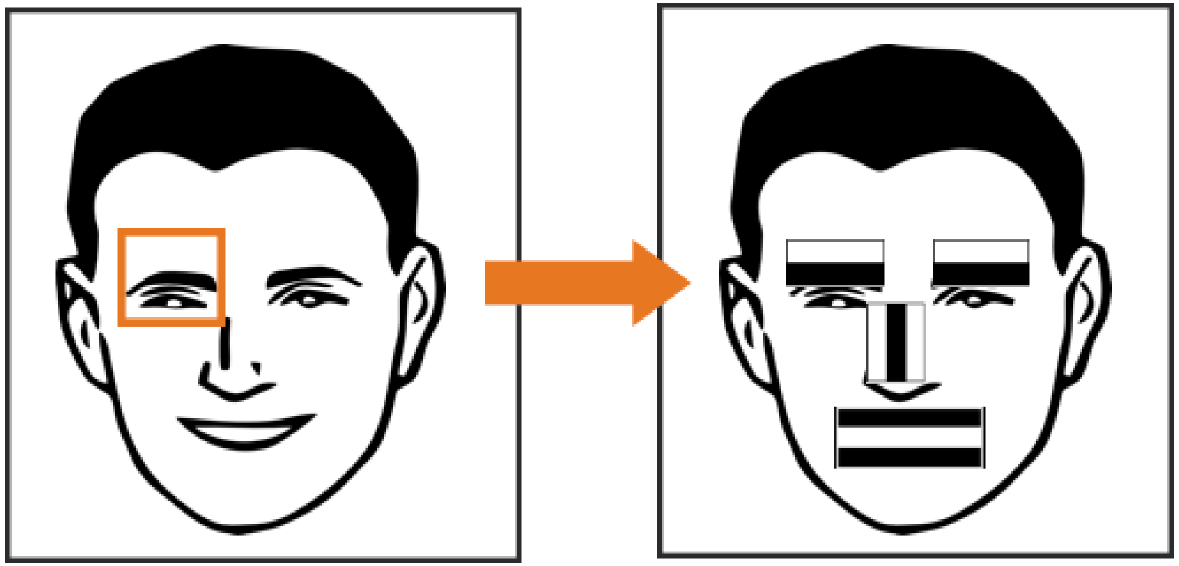
(c) (d)

Detect Line (Line Features)

Example of some possible feature shapes

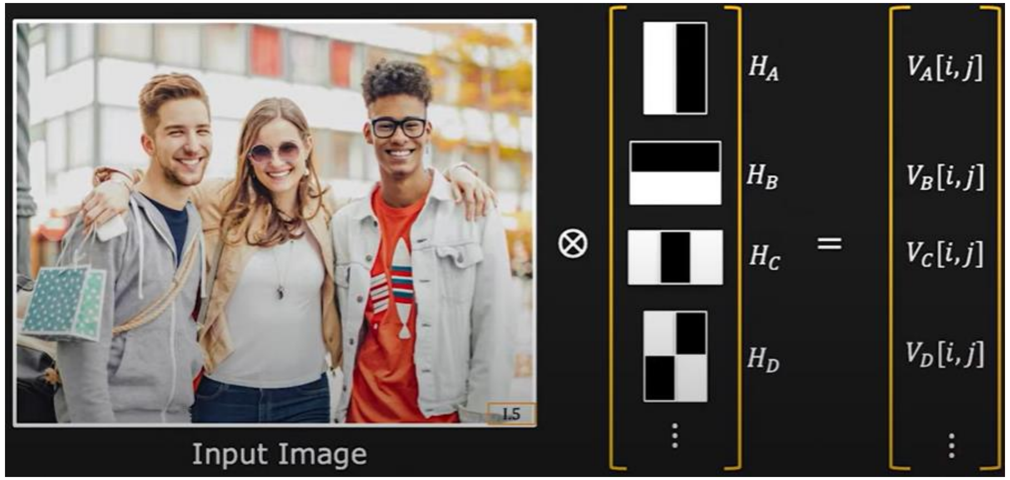
Haar-like Features (Haar Features)

- Edge Features:
  - E.g. eyebrow in an image will be darker and abruptly get lighter (skin)
- Line Features:
  - Naturally the shape of the lips region on your face go from dark to light to dark again



Haar-like feature extraction

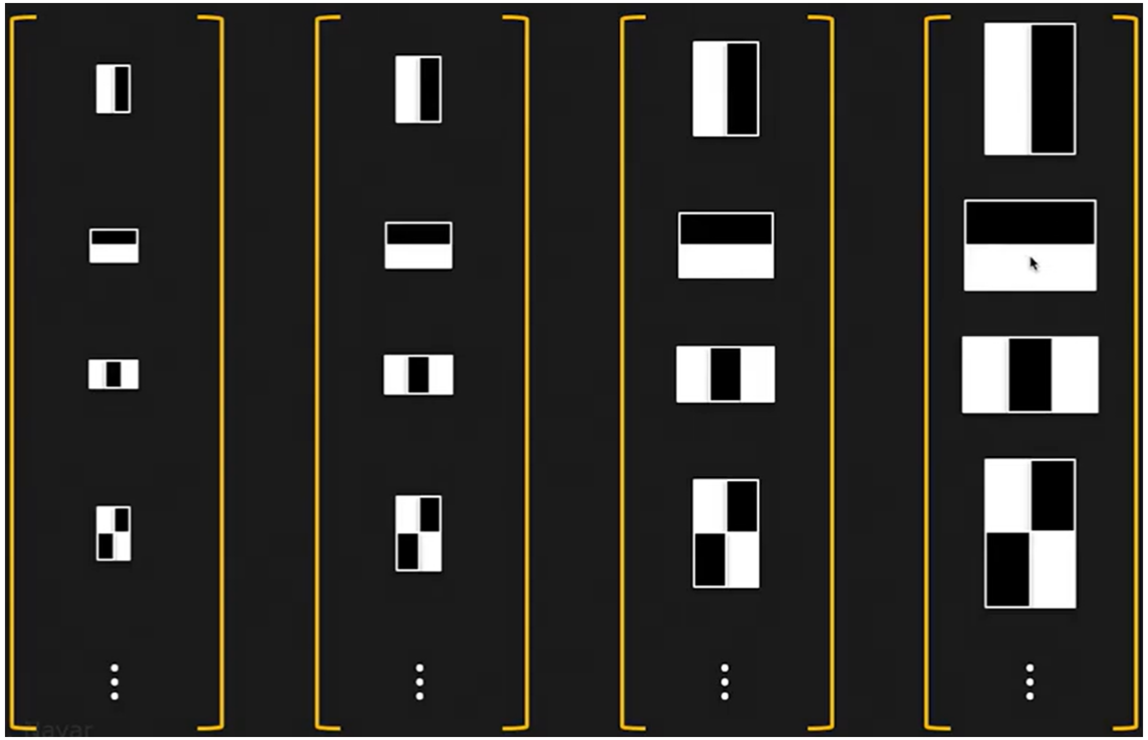
```
for each feature type:
   1. Move across the image each sub-window
   2. Calculate delta of the sum(unshaded) and sum(shaded)
   3. Use these values to train an AdaBoost variant model.
```



Haar Filters   Haar Features

Detecting Face of Different Size

- Compute Haar Features at different scales to detect face of different sizes



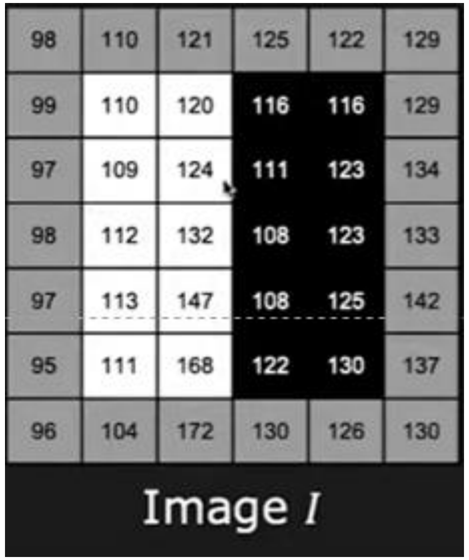
Haar Feature: Computation cost

- $Computation\ cost\ =\ (NXM - 1)$ additions per pixel per filter per scale
- Can we Do it Better?

$$V_A = \sum(pixels\ in\ white) - \sum(pixels\ in\ black)$$

Problem #1

- Summing up pixel values for all feature types in all images in the dataset can be very computationally expensive, especially depending on the resolution of the images.



$$V_A = \sum(\textit{pixels in white}) - \sum(\textit{pixels in black})$$

Integral Image

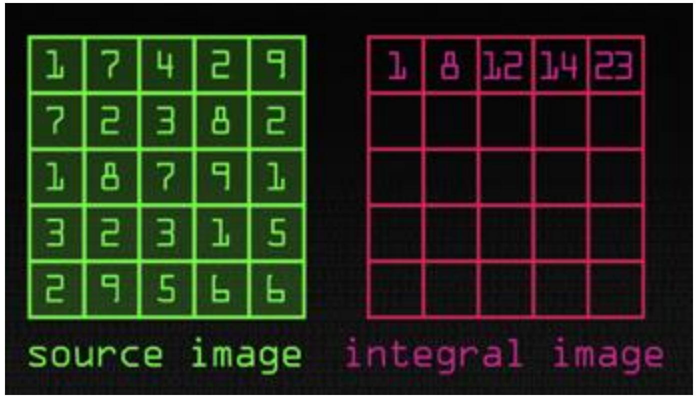
- To solve this, Viola and Jones introduced the concept of the Integral Image
  - Definition: The integral image at a pixel (x, y) contains the sum of all pixel values above and to the left of (x, y), inclusive:

$$II(x,y) = \sum_{i=0}^{x} \sum_{j=0}^{y} I(i,j)$$

where $I(i, j)$ is the intensity of the pixel at (i, j).

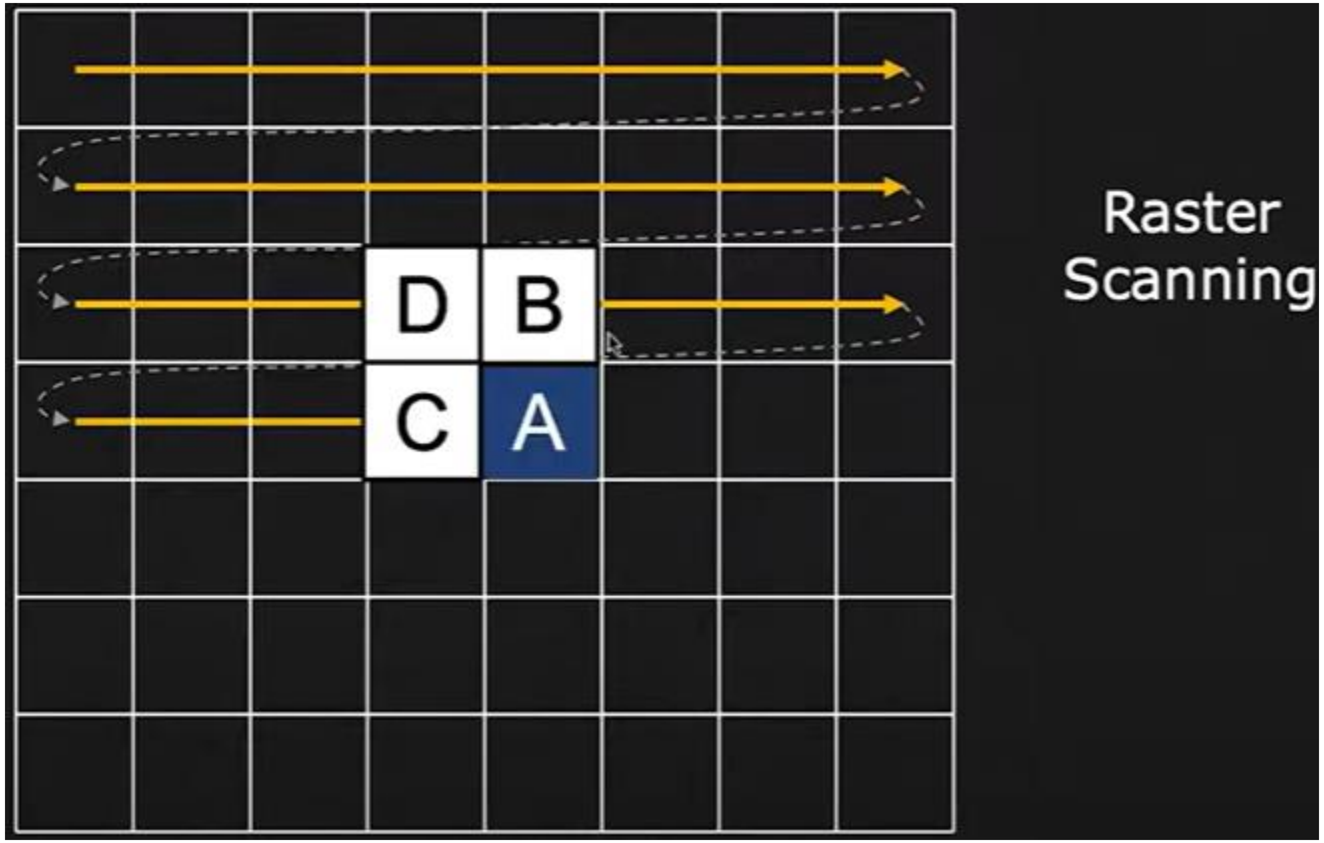
Integral Image

- To solve this, Viola and Jones introduced the concept of the Integral Image
  - A precomputed version of the source image
  - Store it in an intermediate form



Computing Integral Image

Computing Integral Image



- Let I_A and II_A be the values of Image and Integral Image, respectively, at pixel A.

$$\boxed{II_A = II_B + II_c - II_D + I_A}$$

Integral Image

- A table that holds the sum of all pixel values to the left and top of a given pixel, inclusive.

98	110	121	125	122	129
99	110	120	116	116	129
97	109	124	111	123	134
98	112	132	108	123	133
97	113	147	108	125	142
95	111	168	122	130	137
96	104	172	130	126	130

Image I

98	208	329	454	576	705
197	417	658	899	1137	1395
294	623	988	1340	1701	2093
392	833	1330	1790	2274	2799
489	1043	1687	2255	2864	3531
584	1249	2061	2751	3490	4294
680	1449	2433	3253	4118	5052

Integral Image II

Integral Image

- A table that holds the sum of all pixel values to the left and top of a given pixel, inclusive.

98	110	121	125	122	129
99	110	120	116	116	129
97	109	124	111	123	134
98	112	132	108	123	133
97	113	147	108	125	142
95	111	168	122	130	137
96	104	172	130	126	130

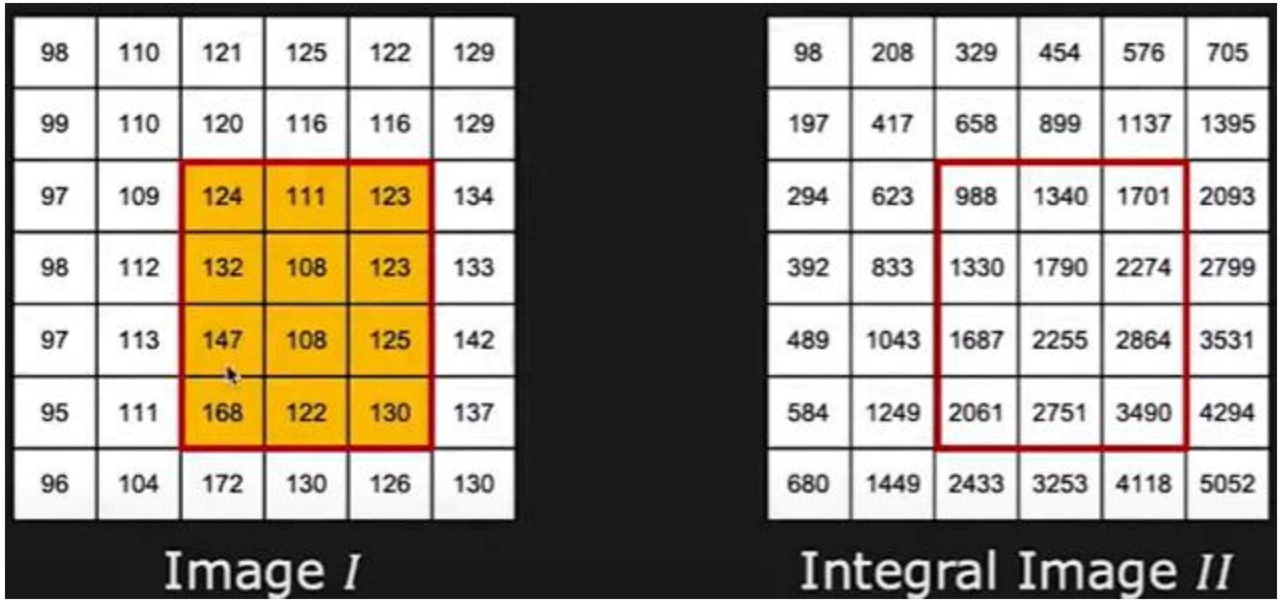
Image *I*

98	208	329	454	576	705
197	417	658	899	1137	1395
294	623	988	1340	1701	2093
392	833	1330	1790	2274	2799
489	1043	1687	2255	2864	3531
584	1249	2061	2751	3490	4294
680	1449	2433	3253	4118	5052

Integral Image *II*

Summation within a Rectangle

- Fast summation of arbitrary rectangle using integral images



98	110	121	125	122	129
99	110	120	116	116	129
97	109	124	111	123	134
98	112	132	108	123	133
97	113	147	108	125	142
95	111	168	122	130	137
96	104	172	130	126	130

Image I

98	208	329	454	576	705
197	417	658	899	1137	1395
294	623	988	1340	1701	2093
392	833	1330	1790	2274	2799
489	1043	1687	2255	2864	3531
584	1249	2061	2751	3490	4294
680	1449	2433	3253	4118	5052

Integral Image II

Summation within a Rectangle

- Fast summation of arbitrary rectangle using integral images

98	110	121	125	122	129
99	110	120	116	116	129
97	109	124	111	123	134
98	112	132	108	123	133
97	113	147	108	125	142
95	111	168	122	130	137
96	104	172	130	126	130

Image I

98	208	329	454	576	705
197	417	658	899	1137	1395
294	623	988	1340	1701	2093
392	833	1330	1790	2274	2799
489	1043	1687	2255	2864	3531
584	1249	2061	2751	3490	4294
680	1449	2433	3253	4118	5052

P

Integral Image II

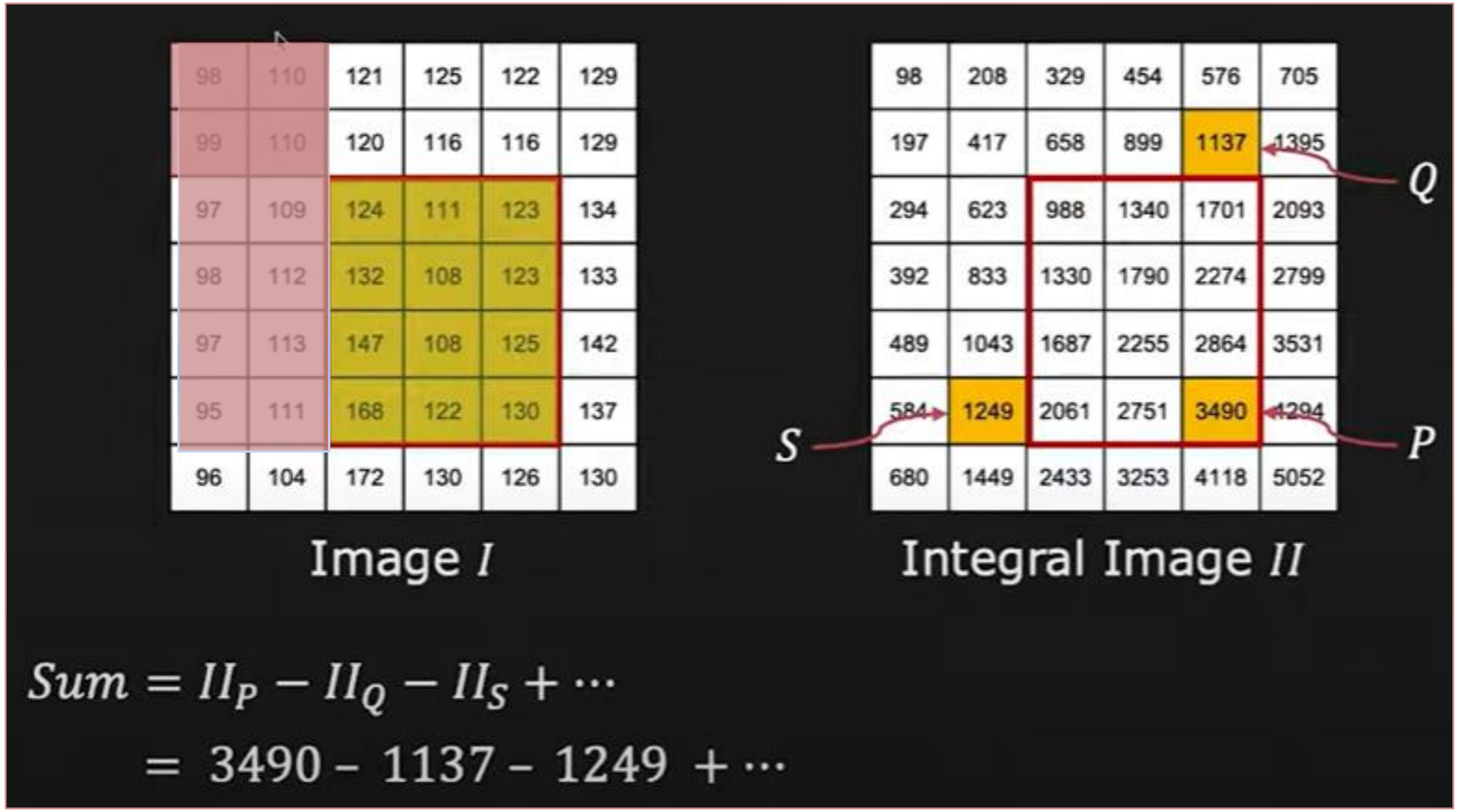
$$Sum = II_P + \cdots$$
$$= 3490 + \cdots$$

Summation within a Rectangle

- Fast summation of arbitrary rectangle using integral images

Summation within a Rectangle

- Fast summation of arbitrary rectangle using integral images



98	110	121	125	122	129
99	110	120	116	116	129
97	109	124	111	123	134
98	112	132	108	123	133
97	113	147	108	125	142
95	111	168	122	130	137
96	104	172	130	126	130

Image I

98	208	329	454	576	705
197	417	658	899	1137	1395
294	623	988	1340	1701	2093
392	833	1330	1790	2274	2799
489	1043	1687	2255	2864	3531
584	1249	2061	2751	3490	4294
680	1449	2433	3253	4118	5052

Integral Image II

$$Sum = II_P - II_Q - II_S + \cdots$$
$$= 3490 - 1137 - 1249 + \cdots$$

Summation within a Rectangle

- Fast summation of arbitrary rectangle using integral images

98	110	121	125	122	129
99	110	120	116	116	129
97	109	124	111	123	134
98	112	132	108	123	133
97	113	147	108	125	142
95	111	168	122	130	137
96	104	172	130	126	130

Image I

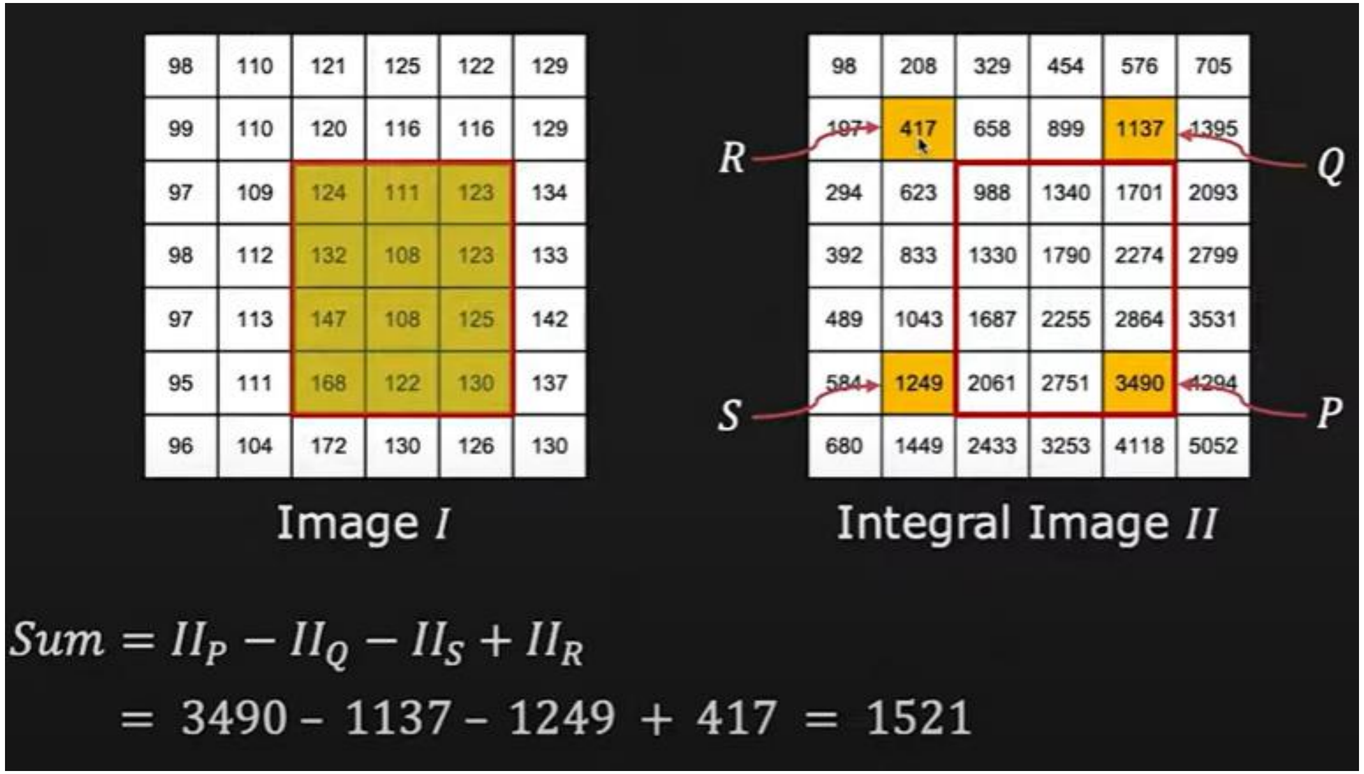
98	208	329	454	576	705
197	417	658	899	1137	1395
294	623	988	1340	1701	2093
392	833	1330	1790	2274	2799
489	1043	1687	2255	2864	3531
584	1249	2061	2751	3490	4294
680	1449	2433	3253	4118	5052

Integral Image II

$$Sum = II_P - II_Q - II_S + \cdots$$
$$= 3490 - 1137 - 1249 + \cdots$$

Summation within a Rectangle

- Fast summation of arbitrary rectangle using integral images



98	110	121	125	122	129
99	110	120	116	116	129
97	109	124	111	123	134
98	112	132	108	123	133
97	113	147	108	125	142
95	111	168	122	130	137
96	104	172	130	126	130

Image I

98	208	329	454	576	705
197	417	658	899	1137	1395
294	623	988	1340	1701	2093
392	833	1330	1790	2274	2799
489	1043	1687	2255	2864	3531
584	1249	2061	2751	3490	4294
680	1449	2433	3253	4118	5052

Integral Image II

$$Sum = II_P - II_Q - II_S + II_R$$
$$= 3490 - 1137 - 1249 + 417 = 1521$$

- Computational Cost: Only 3 additions

- The interesting point here is the cost is independent of the size of the rectangle.

Haar Response Using Integral Image

98	110	121	125	122	129
99	110	120	116	116	129
97	109	124	111	123	134
98	112	132	108	123	133
97	113	147	108	125	142
95	111	168	122	130	137
96	104	172	130	126	130

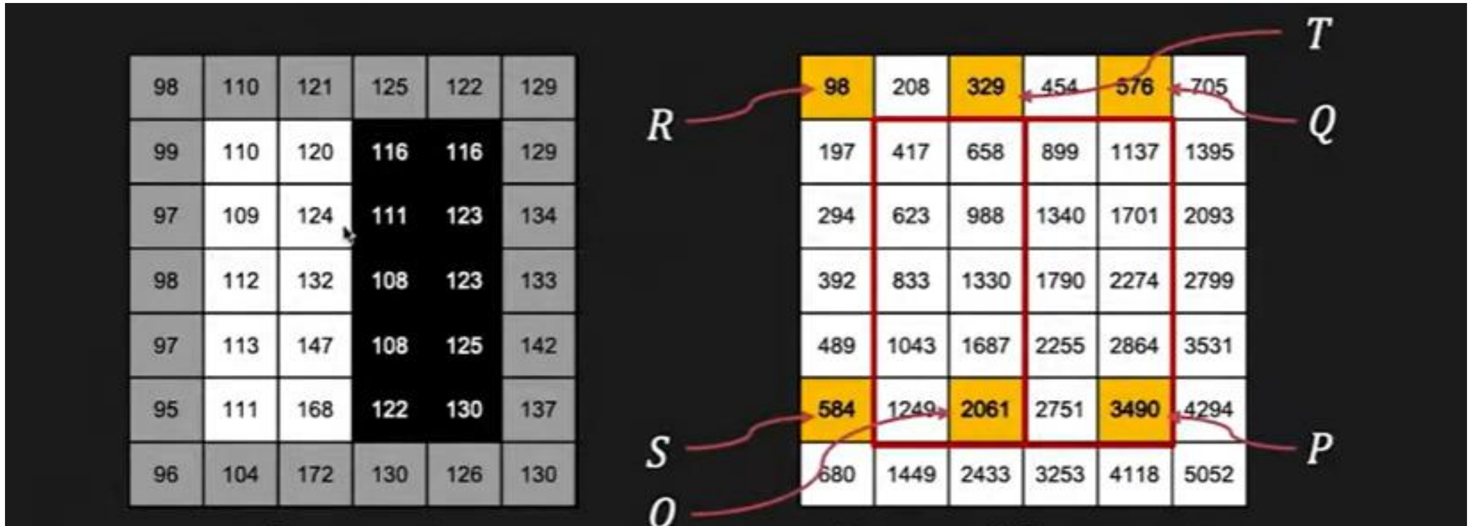
Image I

98	208	329	454	576	705
197	417	658	899	1137	1395
294	623	988	1340	1701	2093
392	833	1330	1790	2274	2799
489	1043	1687	2255	2864	3531
584	1249	2061	2751	3490	4294
680	1449	2433	3253	4118	5052

Integral Image II

$$V_A = \sum(pixels\ in\ white) - \sum(pixels\ in\ black)$$

Haar Response Using Integral Image



98	110	121	125	122	129
99	110	120	116	116	129
97	109	124	111	123	134
98	112	132	108	123	133
97	113	147	108	125	142
95	111	168	122	130	137
96	104	172	130	126	130

Image I

98	208	329	454	576	705
197	417	658	899	1137	1395
294	623	988	1340	1701	2093
392	833	1330	1790	2274	2799
489	1043	1687	2255	2864	3531
584	1249	2061	2751	3490	4294
680	1449	2433	3253	4118	5052

Integral Image II

$$V_A = \sum(\textit{pixel intensities in white}) - \sum(\textit{pixel intensities in black})$$

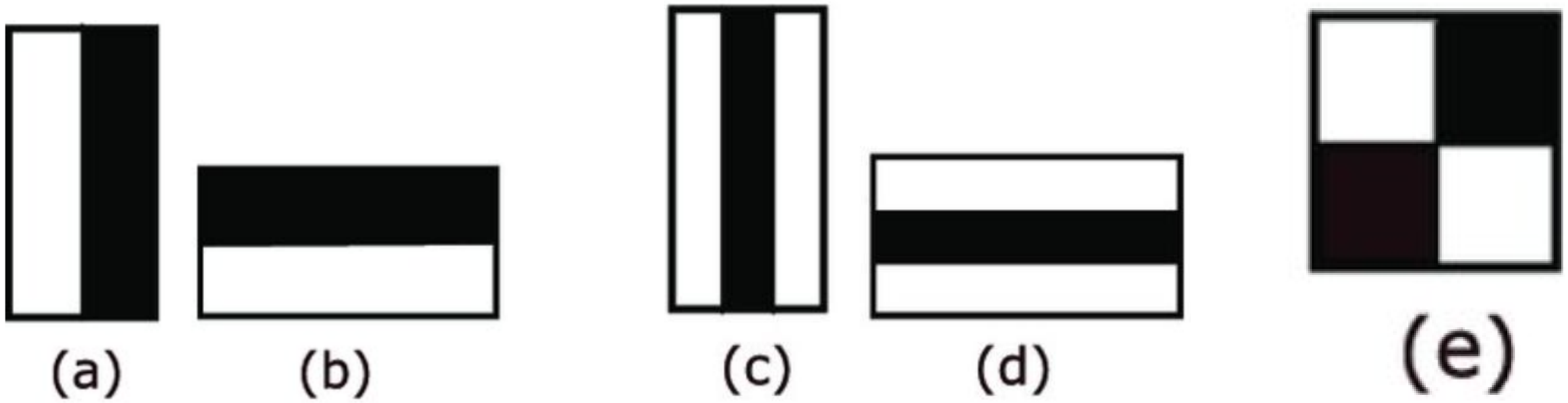
$$= (II_O - II_T + II_R - II_S) - (II_P - II_Q + II_T - II_O)$$

$$= (2061 - 329 + 98 - 584) - (3490 - 576 + 329 - 2061) = 64$$

Computational Cost: Only 7 additions

Feature Implementation

- Constant time evaluation using integral image
  - Edge features – 6 memory lookups
  - Line features – 8 memory lookups
  - Four- rectangle Features- 9 memory lookups



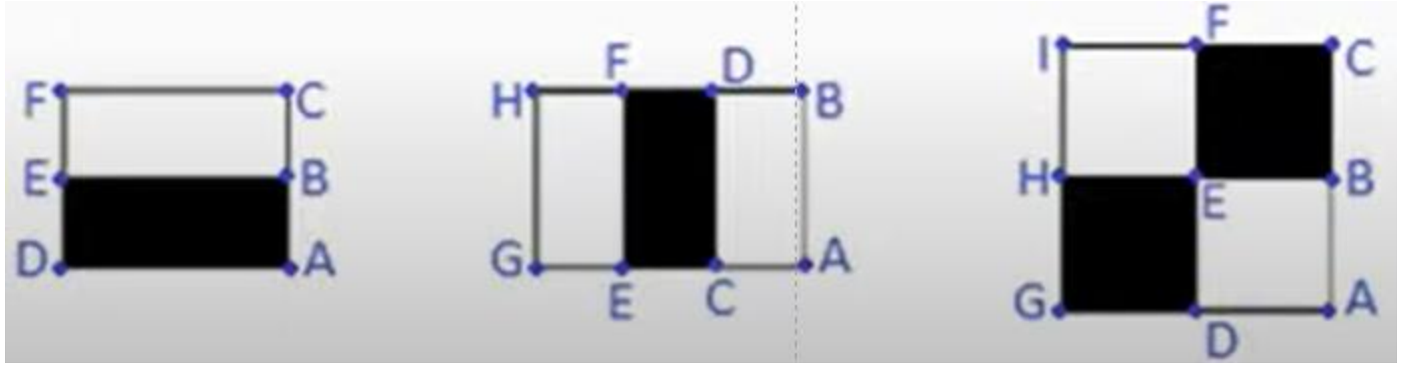
Edge features

Line features

Four- rectangle

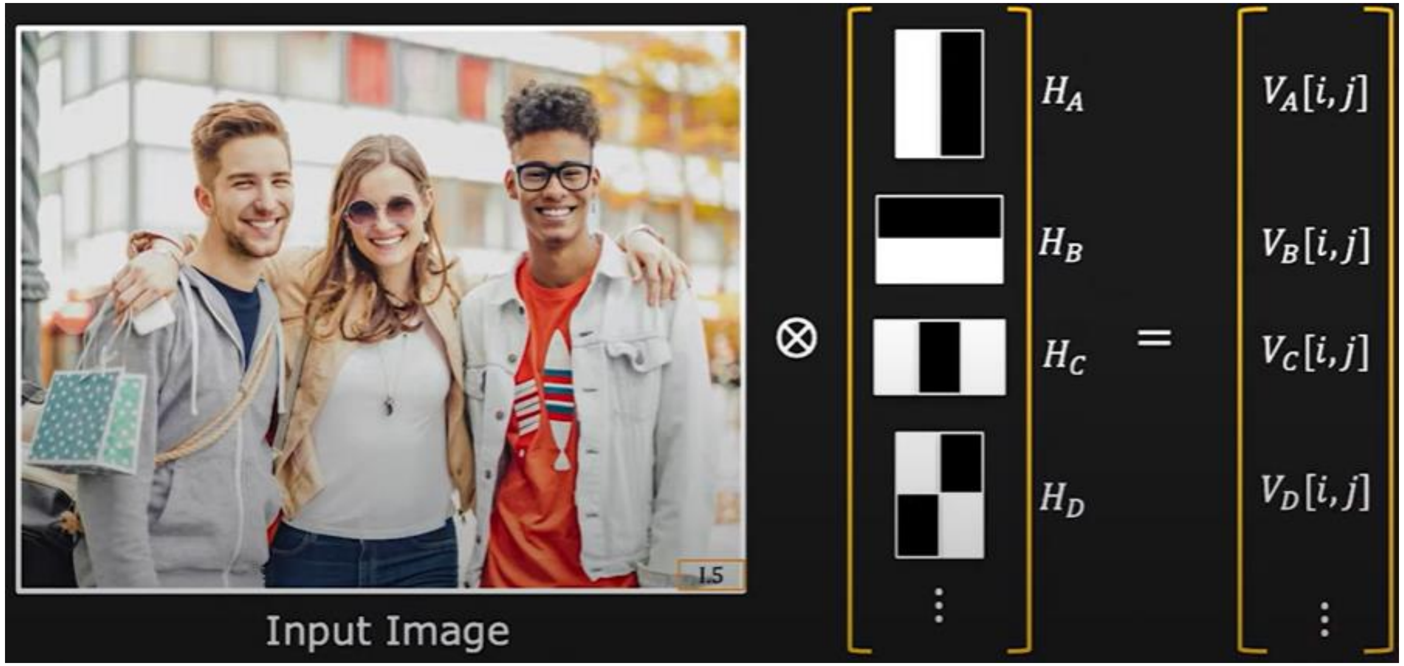
Haar Feature Calculations

- Using the integral image:
  - 2 rectangle: A -2B+C-D+2E-F
  - 3 rectangle: A-B-2C+2D+2E-2F-G-H
  - 4 rectangle: A-2B+C-2D+4E-2F+H-2I+J



Haar Features Using Integral Images

- Integral image needs to be computed once per test image.
- Allows fast computation of Haar features



Haar Filters

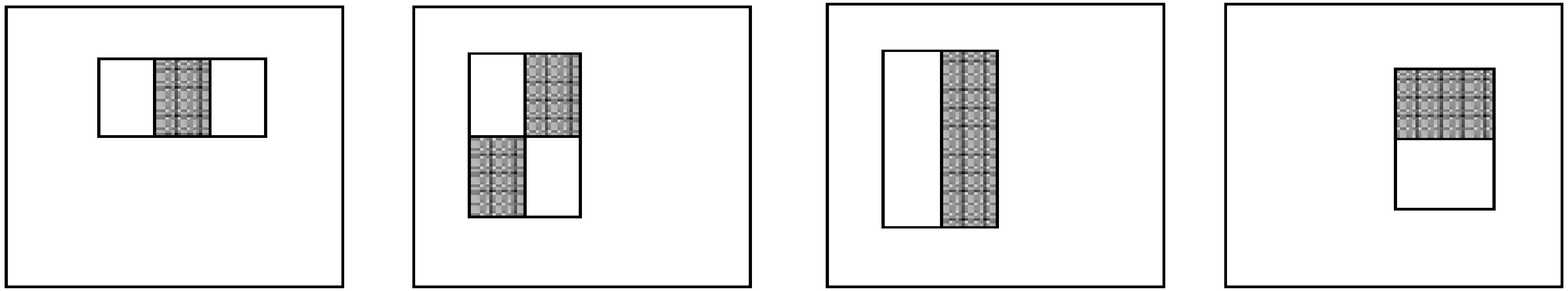
Haar Features

Problem #2

- There are over 160,000 possible feature combinations that can fit into a 24x24 pixel image, and over 250,000 for a 28x28 image.

Feature Selection

- Features are extracted from sub windows of a sample image.
  - The base size for a sub window is 24 by 24 pixels.
- Each of the four feature types are scaled and shifted across all possible combinations
  - In a 24 pixel by 24 pixel sub window there are ~160,000 possible features to be calculated.



Feature Selection

- Faces are complex and variable – we need a lot of features to capture all possible examples.
  - We can't possibly use all 160,000.
  - Can we create a good classifier using just a small subset of all possible features?
  - How to select such a subset?
- Boosting is a classification scheme that works by combining weak learners into a more accurate ensemble classifier
  - A weak learner need only do better than chance.
- Training consists of multiple boosting rounds.
- Feature selection: AdaBoost inherently performs feature selection by choosing only the most effective features during the training process.

Training process

1. Assign equal weights to all training examples (face and non-face images).
2. Iteratively select the best weak classifier (feature, threshold, polarity) that minimizes the weighted classification error.
3. Increase the weights of misclassified examples and decrease the weights of correctly classified examples. This forces subsequent weak classifiers to focus on the difficult examples.
4. Assign a weight to each selected weak classifier based on its accuracy.
5. The final strong classifier is a weighted linear combination of the selected weak classifiers.

Algorithm 1 AdaBoost Training for Viola-Jones

Require: Training set $(x_1, y_1), \ldots, (x_N, y_N)$ where $y_i \in \{-1, +1\}$
Require: M Haar-like features $f_1, \ldots, f_M$ ($M \approx 160{,}000$)
Require: T = number of boosting rounds
Ensure: Strong classifier $H(x)$

Initialize sample weights: $w_i^{(1)} = \frac{1}{N}$ for $i = 1, \ldots, N$
for $t = 1$ **to** T **do**
  Step 1: Train Weak Classifiers
  for $j = 1$ **to** M **do**
    Construct weak classifier $h_j(x)$ using feature f_j:

$$h_j(x) = \begin{cases} +1 & \text{if } f_j(x) > \theta_j \\ -1 & \text{otherwise} \end{cases}$$

    Choose θ_j to minimize weighted error:

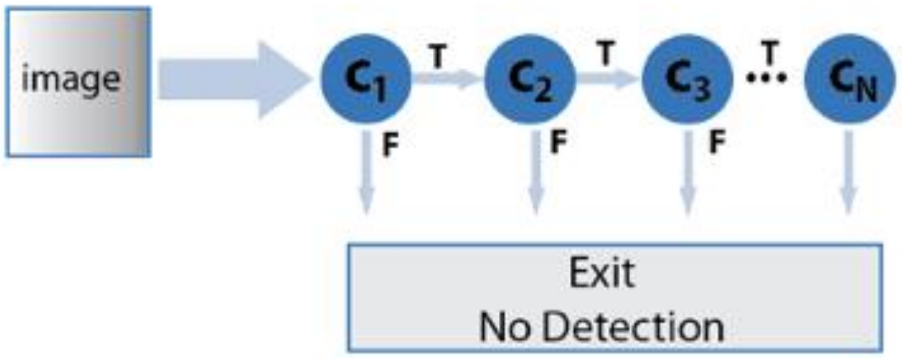
$$\epsilon_j = \sum_{i=1}^{N} w_i^{(t)} \cdot \mathbf{1}(h_j(x_i) \neq y_i)$$

  end for
  Step 2: Select Best Classifier
  Choose h_t with lowest error: $h_t = \arg\min_{h_j} \epsilon_j$
  Step 3: Compute Classifier Weight
  $\alpha_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$
  Step 4: Update Sample Weights
  for $i = 1$ **to** N **do**
    $w_i^{(t+1)} = w_i^{(t)} \cdot e^{-\alpha_t y_i h_t(x_i)}$
  end for
  Normalize weights: $w_i^{(t+1)} \leftarrow \frac{w_i^{(t+1)}}{\sum_{j=1}^{N} w_j^{(t+1)}}$
end for
Final Classifier:
$H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x) \right)$

Attentional Cascade of Classifier

- Evaluating a large number of features for every sub-window in an image is computationally expensive. The cascade structure is designed to quickly reject non-face regions while spending more computation on potential face regions.

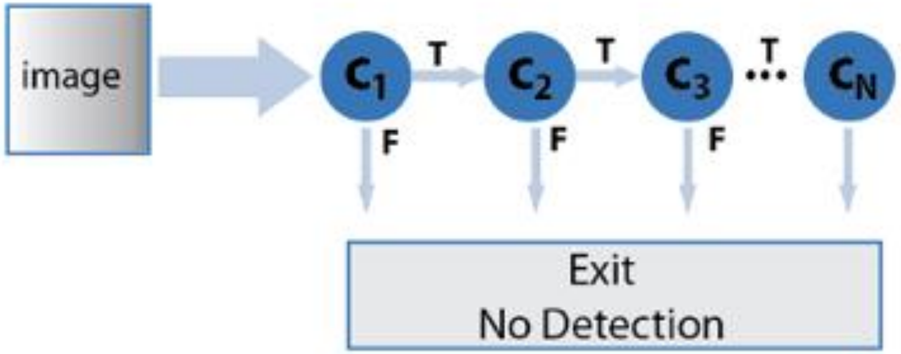
- The cascade consists of multiple stages, where each stage is a strong classifier composed of a small number of carefully selected features. The stages are ordered by increasing complexity (number of features).



Via Matlab

Inference

- A sub-window is passed through the first stage of the cascade.
- If the classifier at a stage rejects the sub-window as "non-face," the process stops for that sub-window, and it's discarded.
- If the classifier at a stage classifies the sub-window as a potential "face," it is passed on to the next stage in the cascade.
- A sub-window is classified as a face only if it passes through all the stages of the cascade.



Via Matlab

Advantages and Disadvantages

- Advantages
  - Detection is very fast
  - Less data needed for training than other ML models

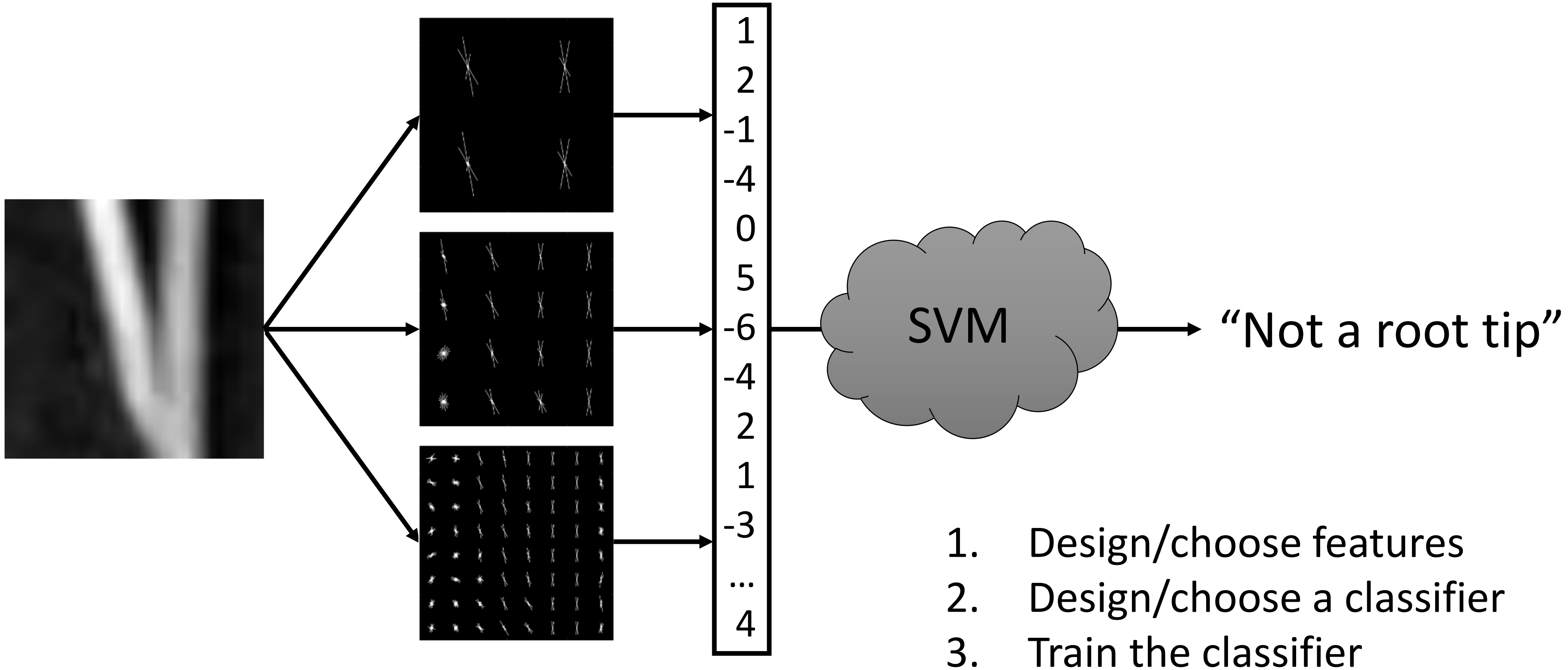
Advantages and Disadvantages

- Disadvantages
  - Training time is very slow
  - Restricted to binary classification
  - Mostly effective when face is in frontal view
  - May be sensitive to very high/low exposure (brightness)
  - High true detection rate, but also high false detection rate

Results

Learning in Vision

- The classic approach applies learned operations to user-defined features



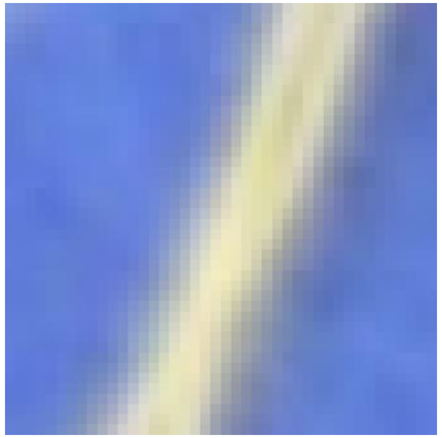
HoG, (Local Binary Patterns)LBP, etc

1. Design/choose features
2. Design/choose a classifier
3. Train the classifier

Learning in Vision

- Designing features can become a trial and error process

Tip

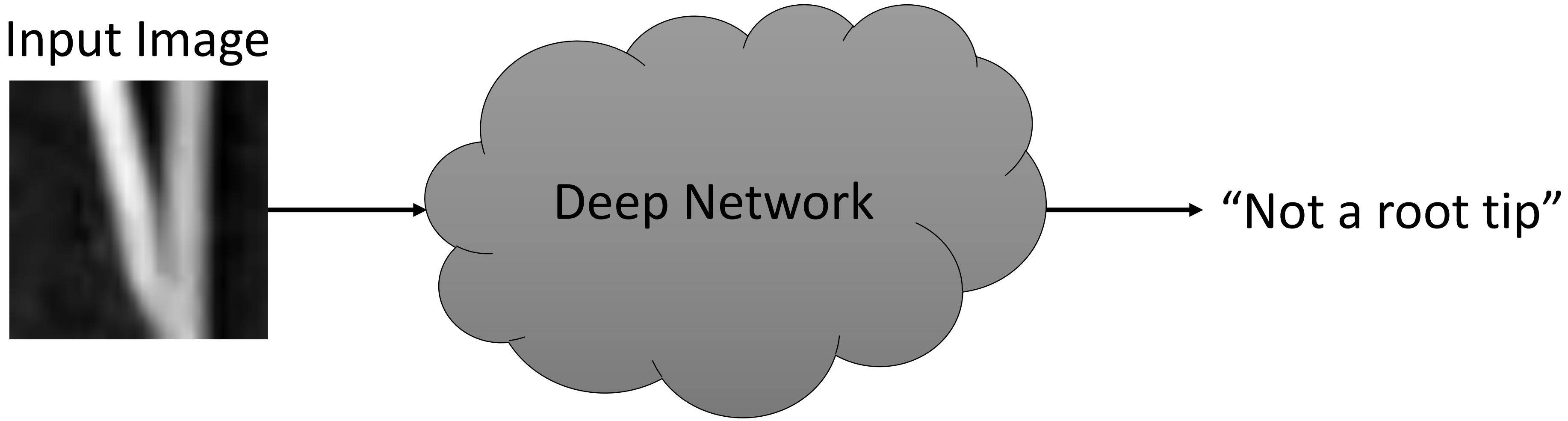


Crossover

- Learning will fail if the user limits it to the wrong features
- More recent approaches try reduce reliance on the user
    1. Bag of Words clusters the results of applying the user-defined set of feature detection operators to form a more generic visual vocabulary
    2. Viola-Jones selects from a much larger set of user-defined features

Deep Learning

- Deep learning does not use any pre-computed features
- Feature detection and classification are integrated
- Deep methods learn
  1. What features are needed to make classification possible
  2. How to do the classification given those features



References

- T. Hastie, R. Tibshirani, J. Friedman: The Elements of Statistical Learning, 2nd Edition, Springer, 2009.
- Y. Freund, R. E. Schapire: A decision-theoretic generalization of on-line learning and an application to boosting, Journal of Computer and System Sciences, 55(1):119-139, 1997.
- P. A. Viola, M. J. Jones: Robust Real-Time Face Detection, International Journal of Computer Vision 57(2): 137-154, 2004.
- J. Matas and J. Šochman: AdaBoost, Centre for Machine Perception, Technical University, Prague. http://cmp.felk.cvut.cz/~sochmj1/adaboost_talk.pdf

Next: CNNs and Deep Learning: an introduction