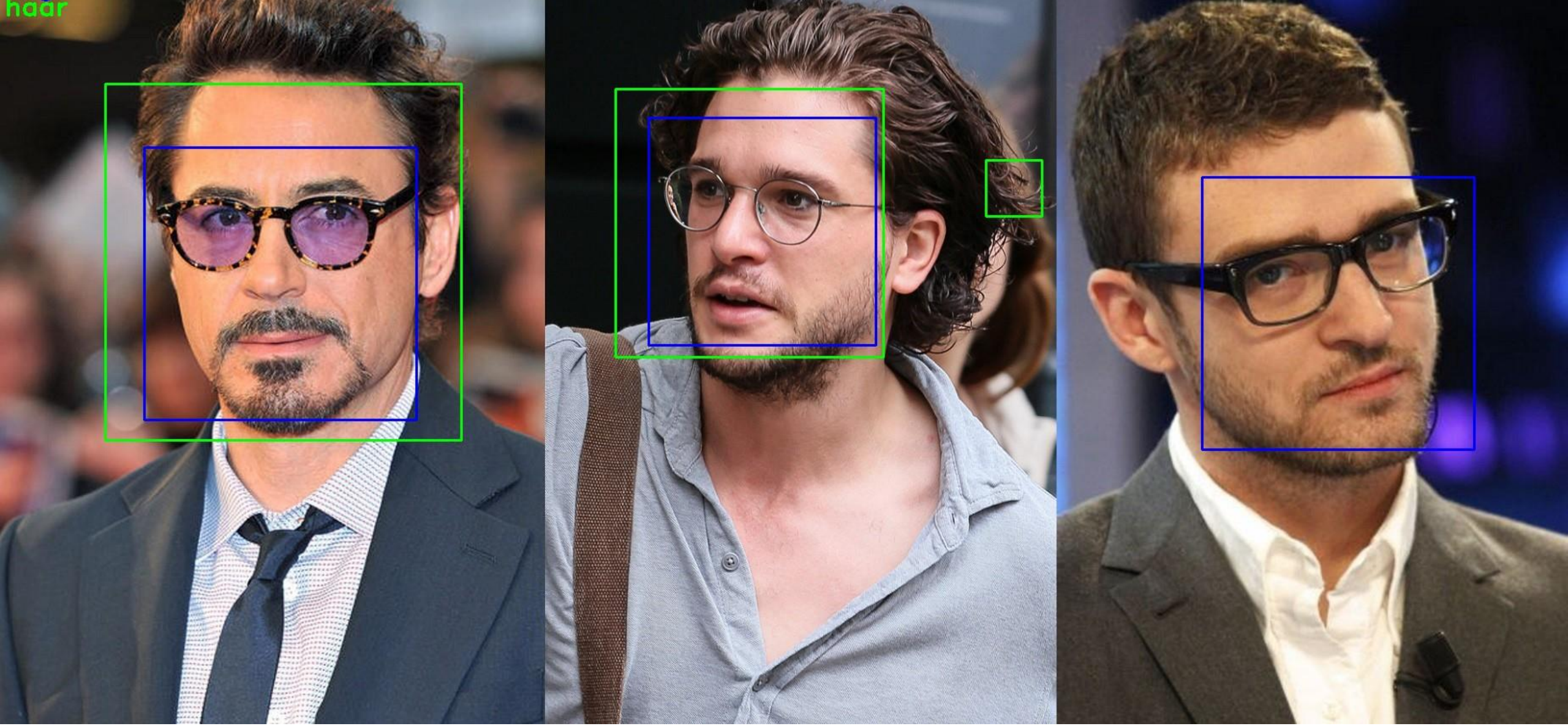


Face Detection

Face detection results produced by Rowley, Baluja, and Kanade (1998a)

Application face detection

Area	Applications
Information Security	Access Security (OS, Database) Data Privacy (e.g. medical records) User authentications (trading, on-line banking)
Access Management	Secure Access Authentications (restricted facilities) Permission based system Access log or Audit Trails
Biometrics	Personal identification (national IDs, passport, voter registration, driver licenses) Automated identity verification (boarder control)
Law Enforcement	Video Surveillance, Suspect identification Suspect tracking (investigation) Simulated aging Forensic Reconstruction of faces from remains, Tracing missing children.
Personal Security	Home video surveillance system Expression interpretation (driver monitoring system)
Entertainment	Home video game systems Photo camera applications

Introduction

- Before face recognition can be applied to a general image, the locations and sizes of any faces must first be found
- Face detection techniques can be classified as feature-based, template-based, or appearance-based (Yang, Kriegman, and Ahuja (2002))

Cara kerja

Figure: [8]

Features [9]

1. Distance between the eyes
2. Width of the nose
3. Depth of the eye socket
4. Cheekbones
5. Jaw line
6. Chin

Feature-Based Methods

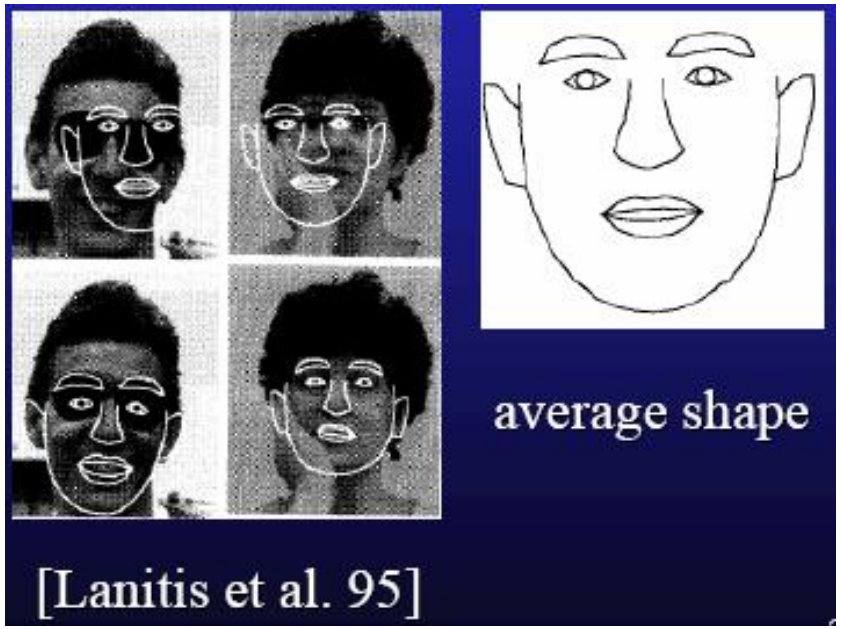
- Bottom-up approach: Detect facial features (eyes, nose, mouth, etc) first
- Facial features: edge, intensity, shape, texture, color, etc
- Aim to detect invariant features
- Group features into candidates and verify them

Feature-Based Methods: Summary

- Pros: Features are invariant to pose and orientation change
- Cons:
  - Difficult to locate facial features due to several corruption (illumination, noise, occlusion)
  - Difficult to detect features in complex background

Template Matching Methods

- Store a template
  - Predefined: based on edges or regions
- Deformable: based on facial contours (e.g., Snakes)
- Templates are hand-coded (not learned)
- Use correlation to locate faces



Template-Based Methods: Summary

- Pros:
  - Simple
- Cons:
  - Templates needs to be initialized near the face images
  - Difficult to enumerate templates for different poses (similar to knowledge-based methods)

Appearance-Based Methods: Classifiers

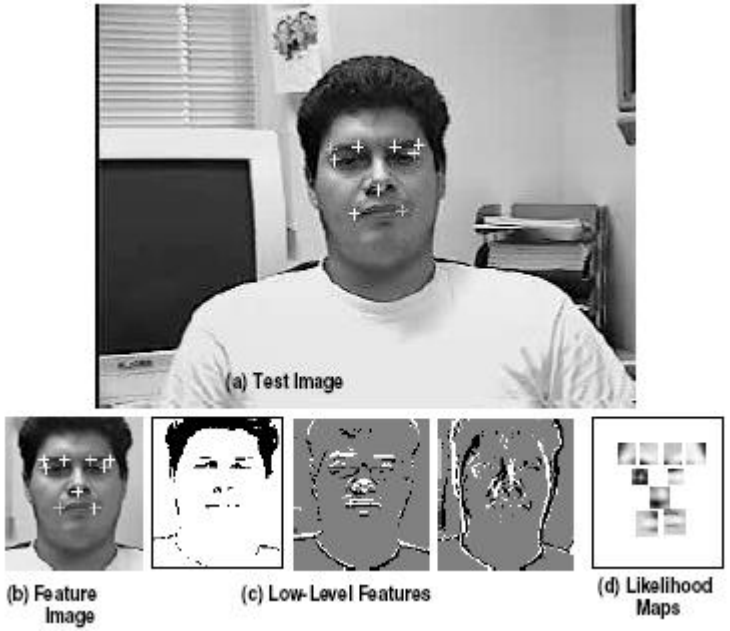
- Neural network
  - Multilayer Perceptrons
- Princiapl Component Analysis (PCA), Factor Analysis
- Support vector machine (SVM)
- Mixture of PCA, Mixture of factor analyzers
- Distribution Distribution-based method
- Naïve Bayes classifier
- Hidden Markov model
- Sparse network of winnows (SNoW)
- Kullback relative information
- Inductive learning: C4.5
- Adaboost

Theory of Our Algorithm

- The maximum likelihood(ML) test: $L(O) = \frac{P_F(O)}{P_N(O)}$
- In estimating $P_F(O)$ or $P_N(O)$, our assumption is that the permuted version of $O = (o_1, \cdots, o_n)$, $O' = (o_{s_1}, \cdots, o_{s_n})$ comes from a k_{th} order Markov process $X' = (X'_1, \cdots, X'_n) = (X_{s_1}, \cdots, X_{s_n})$ where $s_1, \cdots, s_n$ is a permuted sequence from $1, \cdots, n$.
- The optimal permutation maximizes the Kullback divergence(also known as cross entropy) between $P_F(O')$ and $P_N(O')$, $H_{F||N}(X') = \sum_{O'} P_F(O') \log \frac{P_F(O')}{P_N(O')}$

Facial Features Detection

- Region search

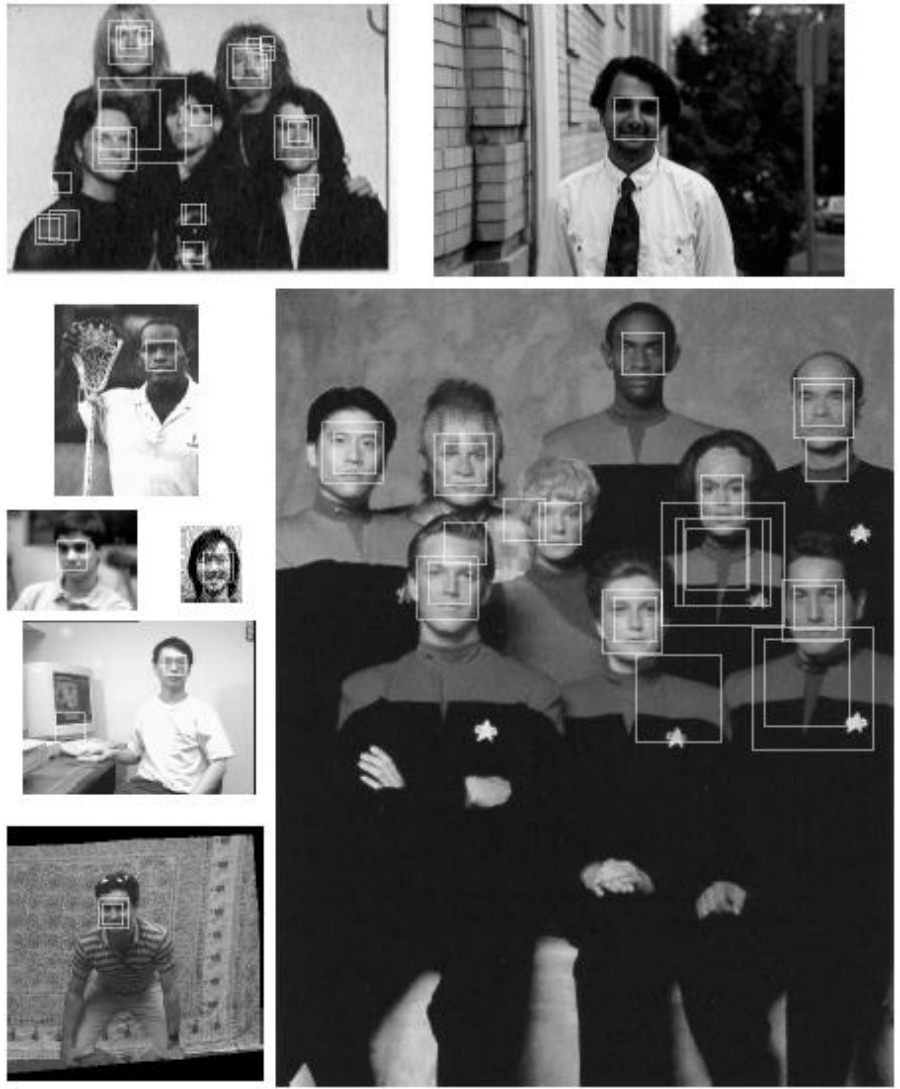


Face and Facial Feature Detection

- The algorithm is also used to detect 9 facial features: 2 outer mouth corners, 2 outer eye corners, 2 outer eye-brow corners, 2 inner eye-brow corners and the center of the nostrils.

Results

Detection Results



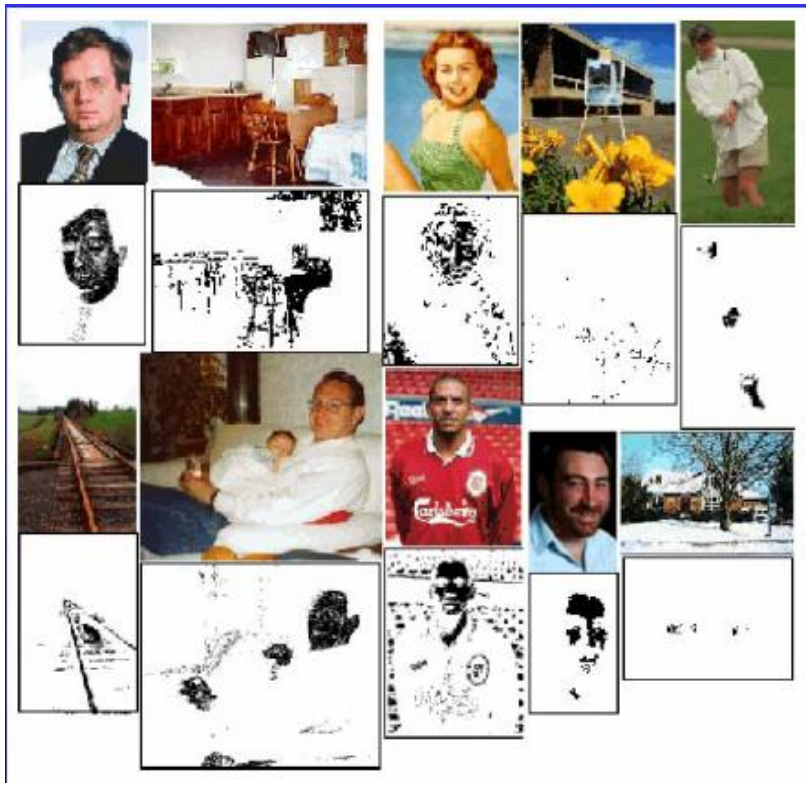
Side-View Face Detection

Appearance-Based Methods: Summary

- Pros:
  - Use powerful machine learning algorithms
  - Has demonstrated good empirical results
  - Fast and fairly robust
  - Extended to detect faces in different pose and orientation
- Cons:
  - Usually needs to search over space and scale
  - Need lots of positive and negative examples
  - Limited view-based approach

Color-Based Face Detector

- Pros:
  - Easy to implement
  - Effective and efficient in constrained environment
  - Insensitive to pose, expression, rotation variation
- Cons:
  - Sensitive to environment and lighting change
  - Noisy detection results (body parts, skin-tone line tone line regions)



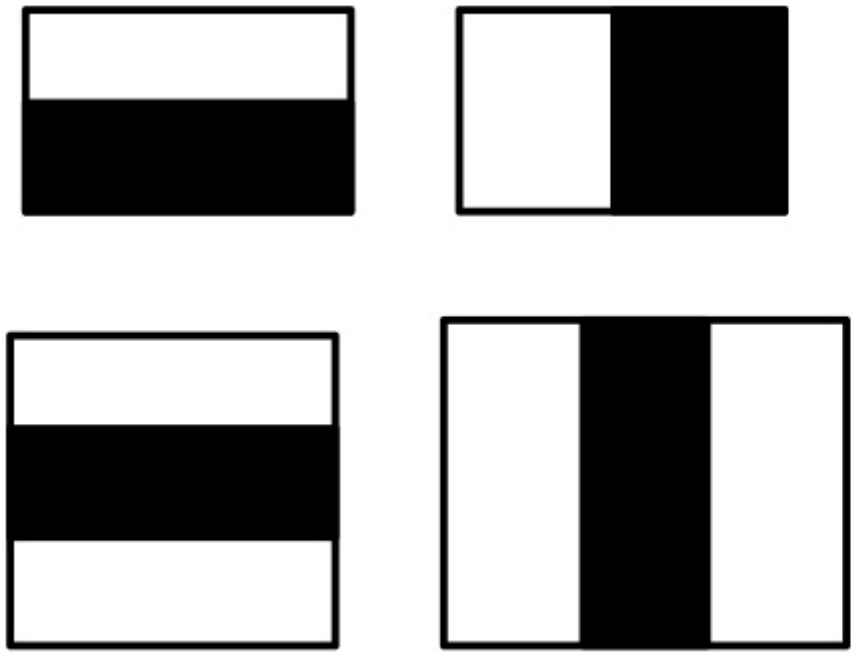
Face Detection using Haar Cascades [3], [10]

- Devised by **Paul Viola and Michael Jones** in 2001.
- Robust and very quick.
- 15 times quicker than any technique at the time of release.
- Could be operated in real-time.
- (95% accuracy at around 17 fps.)
- Feature extraction and feature evaluation.
  (Rectangular features are used)
- With a new image representation their calculation is very fast.
- Classifier training and feature selection using a method called AdaBoost. (A long and exhaustive training process)
- A degenerate decision tree of classifiers is formed.

Features [3], [10]

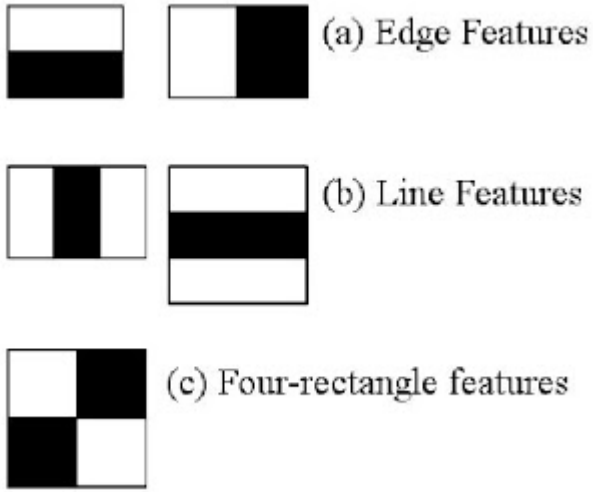
Four basic types:

- They are easy to calculate.
- The white areas are subtracted from the black ones.

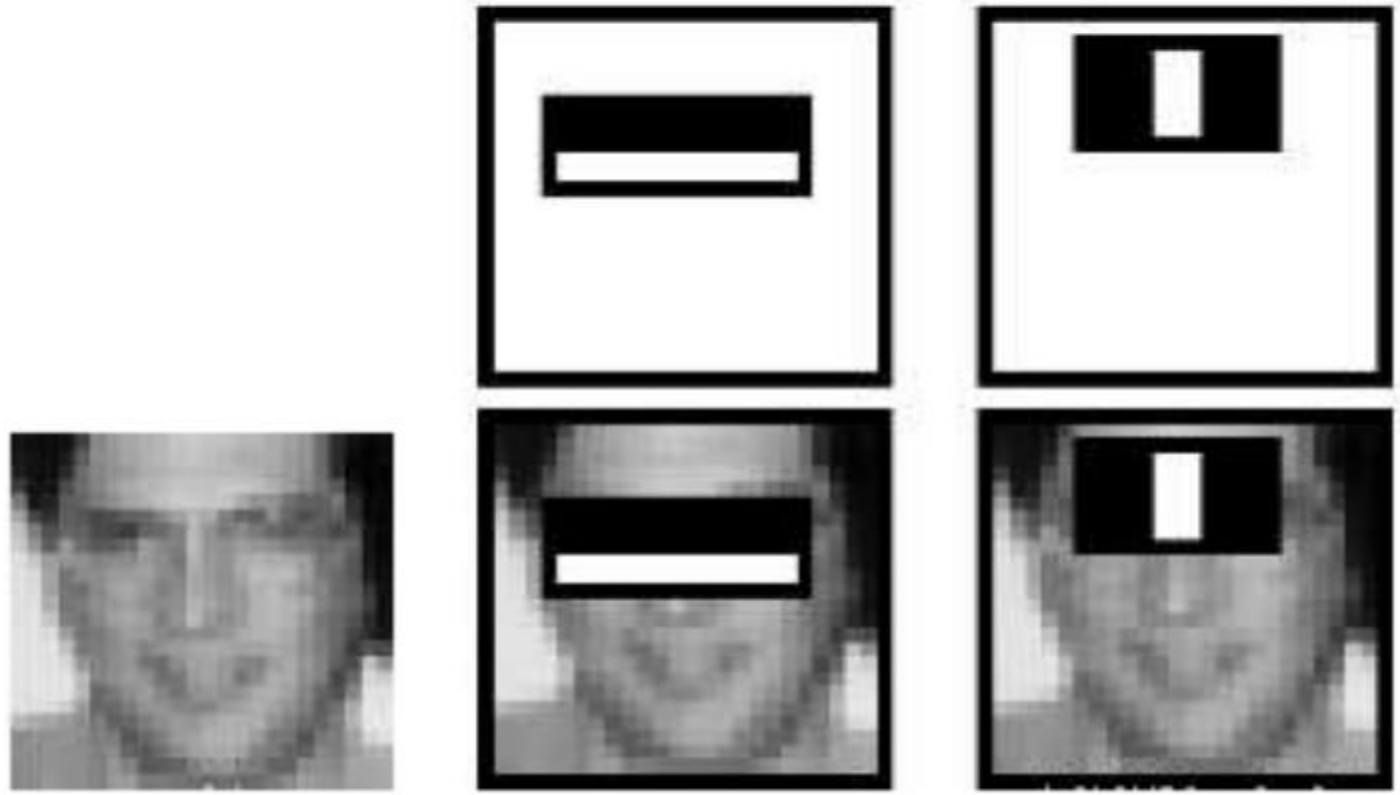


Edge Feature

Line Feature



Features Extraction [3], [10]



image

Challenges in Haar Cascades [10]

• Variations in pose	Head positions, frontal view, profile view and head tilt, facial expressions.
• Illumination Changes	Light direction and intensity changes, cluttered background, low quality images.
• Camera Parameters	Resolution, color balance etc.
• Occlusion	Glasses, facial hair and makeup.

Advantages & Disadvantages [11]

Advantages:

- High detection accuracy.
- Low false positive rate.

Disadvantages:

- Computationally complex and slow.
- Longer training time.
- Less accurate on black faces.
- Limitation in difficult lightening conditions.
- Less robust to occlusion/obstacle.

Face detection technique

- Templates
- PCA (Principle Component Analysis)
- LDA (Liniear Discriminant Analysis)
- Neural Networks
- SVM (Support Vector Machine)
- ICA (Independent Component Analysis)

Hal lain dalam face detection

- Face contours
- Facial features (geometry)
- Motion detection
- Color

Human skin color based face detection

1. Filtering, untuk menemukan area warna kulit manusia. Dilakukan dengan cara :

   RGB -> YCbCr color format

$$Y = (0.299)\,R + (0.587)\,G + (0.114)\,B$$

$$Cb = (-0.169)\,R - (0.332)\,G + (0.500)\,B$$

$$Cr = (0.500)\,R + (-0.419)\,G + (-0.081)\,B$$

Figure 7 a) Original image b) YCbCr image

Human skin color based face detection

2. Temukan area yang berisi bagian wajah

Figure 8: a) Original image b)Face Detected image(shown using Rectangle)

Haar-cascade Detection in OpenCV

```
from __future__ import print_function
import cv2 as cv
import argparse

def detectAndDisplay(frame):
    frame_gray = cv.cvtColor(frame, cv.COLOR_BGR2GRAY)
    frame_gray = cv.equalizeHist(frame_gray)

    #-- Detect faces
    faces = face_cascade.detectMultiScale(frame_gray)
    for (x,y,w,h) in faces:
        center = (x + w//2, y + h//2)
        frame = cv.ellipse(frame, center, (w//2, h//2), 0, 0, 360, (255, 0, 255), 4)

        faceROI = frame_gray[y:y+h,x:x+w]
        #-- In each face, detect eyes
        eyes = eyes_cascade.detectMultiScale(faceROI)
        for (x2,y2,w2,h2) in eyes:
            eye_center = (x + x2 + w2//2, y + y2 + h2//2)
            radius = int(round((w2 + h2)*0.25))
            frame = cv.circle(frame, eye_center, radius, (255, 0, 0 ), 4)

    cv.imshow('Capture - Face detection', frame)

parser = argparse.ArgumentParser(description='Code for Cascade Classifier tutorial.')
parser.add_argument('--face_cascade', help='Path to face cascade.', default='data/haarcascades/haarcascade_frontalface_alt.xml')
parser.add_argument('--eyes_cascade', help='Path to eyes cascade.', default='data/haarcascades/haarcascade_eye_tree_eyeglasses.xml')
parser.add_argument('--camera', help='Camera divide number.', type=int, default=0)
args = parser.parse_args()

face_cascade_name = args.face_cascade
eyes_cascade_name = args.eyes_cascade

face_cascade = cv.CascadeClassifier()
eyes_cascade = cv.CascadeClassifier()

#-- 1. Load the cascades
if not face_cascade.load(cv.samples.findFile(face_cascade_name)):
    print('--(!)Error loading face cascade')
    exit(0)
if not eyes_cascade.load(cv.samples.findFile(eyes_cascade_name)):
    print('--(!)Error loading eyes cascade')
    exit(0)

camera_device = args.camera
#-- 2. Read the video stream
```

Algoritma Adaboost

1. Input the positive and negative training examples along with their labels $\{(\boldsymbol{x}_i, y_i)\}$, where $y_i = 1$ for positive (face) examples and $y_i = -1$ for negative examples.

2. Initialize all the weights to $w_{i,1} \leftarrow \frac{1}{N}$, where N is the number of training examples. (Viola and Jones (2004) use a separate N_1 and N_2 for positive and negative examples.)

3. For each training stage $j = 1 \ldots M$:

   (a) Renormalize the weights so that they sum up to 1 (divide them by their sum).

   (b) Select the best classifier $h_j(\boldsymbol{x}; f_j, \theta_j, s_j)$ by finding the one that minimizes the weighted classification error

   $$e_j = \sum_{i=0}^{N-1} w_{i,j} e_{i,j}, \tag{14.3}$$

   $$e_{i,j} = 1 - \delta(y_i, h_j(\boldsymbol{x}_i; f_j, \theta_j, s_j)). \tag{14.4}$$

   For any given f_j function, the optimal values of (θ_j, s_j) can be found in linear time using a variant of weighted median computation (Exercise 14.2).

   (c) Compute the modified error rate β_j and classifier weight α_j,

   $$\beta_j = \frac{e_j}{1 - e_j} \quad \text{and} \quad \alpha_j = -\log \beta_j. \tag{14.5}$$

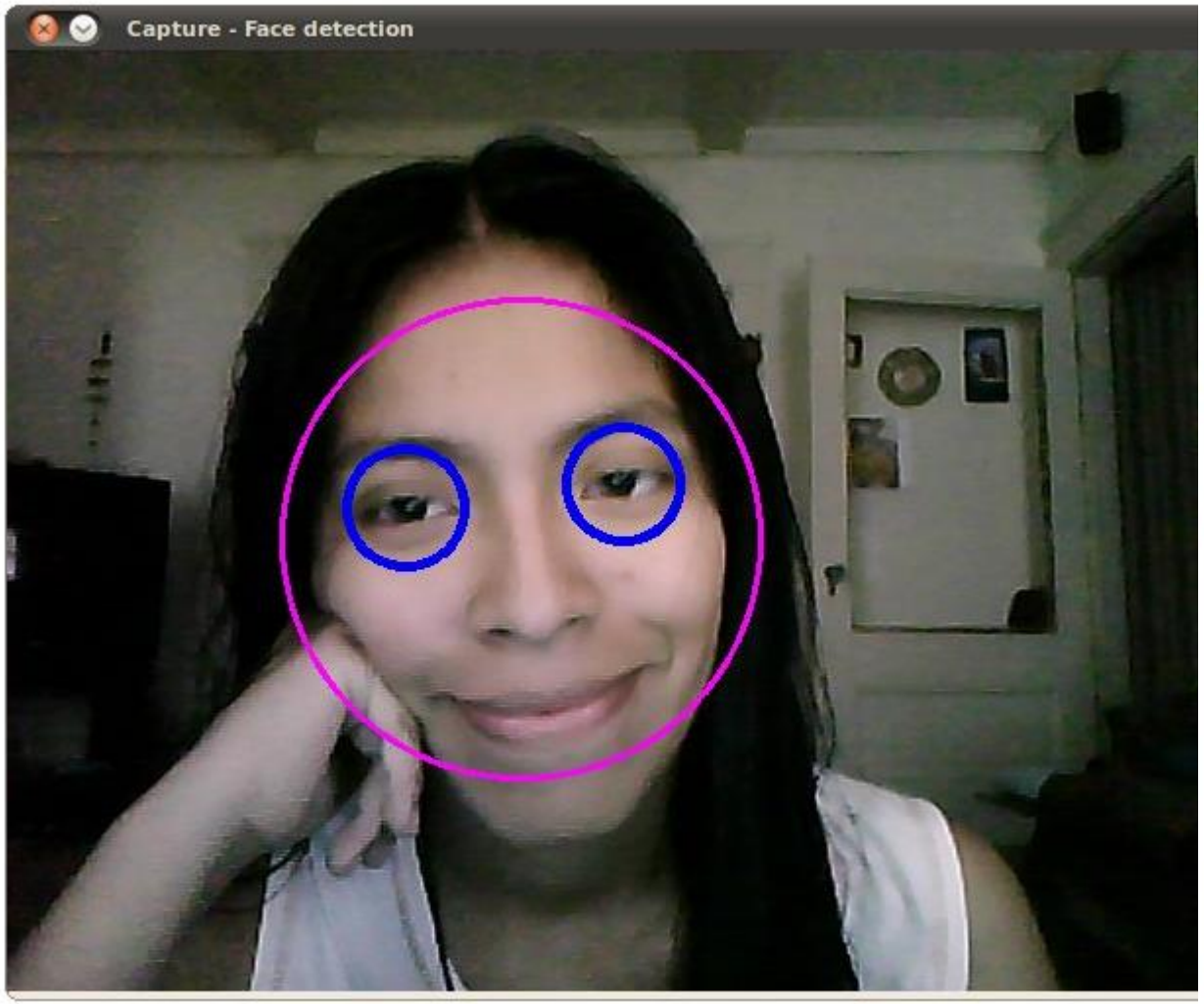
   (d) Update the weights according to the classification errors $e_{i,j}$

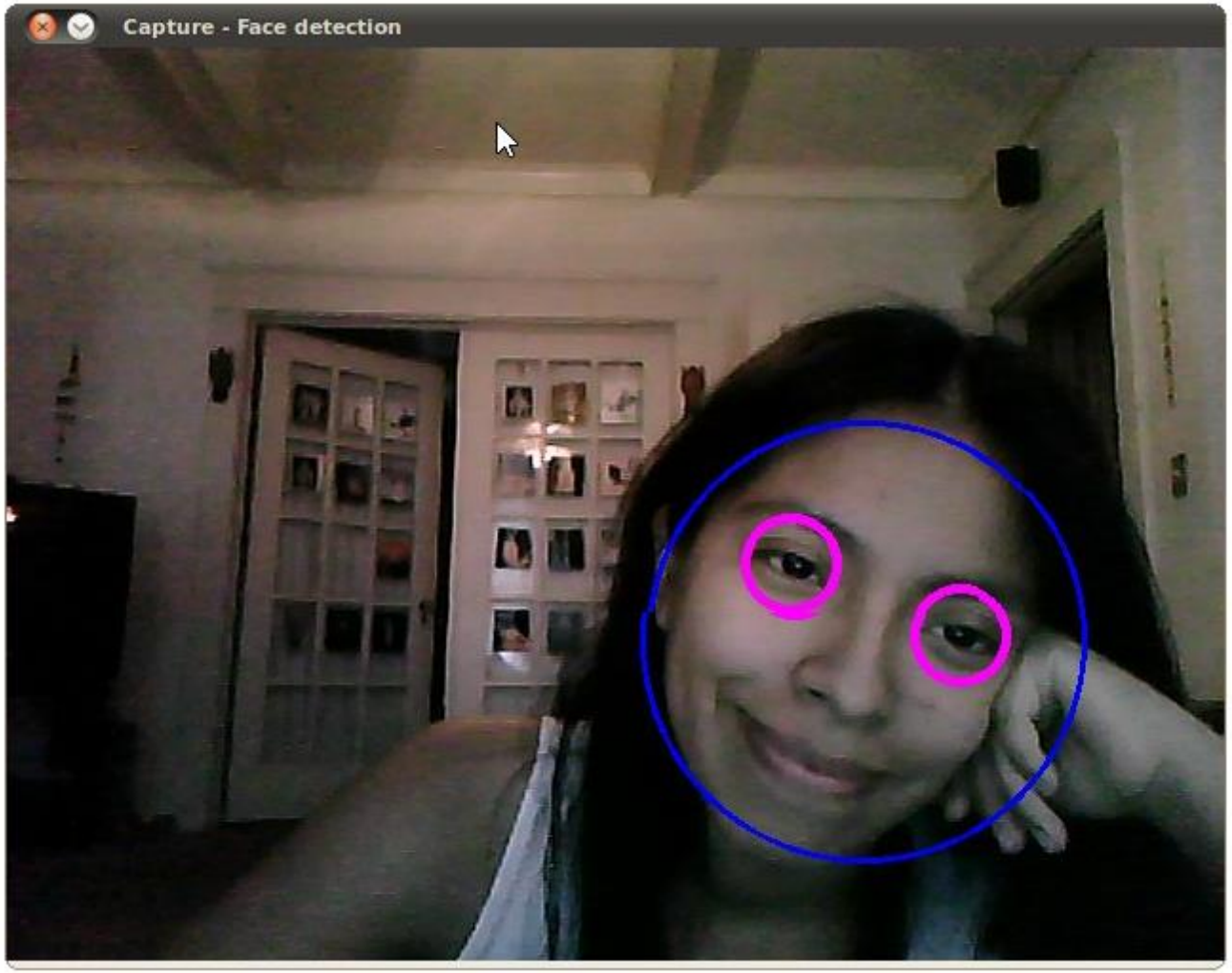
   $$w_{i,j+1} \leftarrow w_{i,j} \beta_j^{1-e_{i,j}}, \tag{14.6}$$

   i.e., downweight the training samples that were correctly classified in proportion to the overall classification error.

4. Set the final classifier to

   $$h(\boldsymbol{x}) = \text{sign}\left[\sum_{j=0}^{m-1} \alpha_j h_j(\boldsymbol{x})\right]. \tag{14.7}$$


Capture - Face detection


Capture - Face detection