# Face

# Announcements

- Survey (12 students participate in the survey)
  - About lab session. Tutors, not enough engagement.
  - Starting lecture on time is an issue.
  - Project instead of exam.
  - Matlab or Python? (Matlab code only in the lecture)
  - Assignment is hard.
  - Lack of interaction.
  - Exercise after lecture.

# Review

- SIFT feature/interest point detector
- SIFT feature descriptor
- Image segmentation
  - K-means
  - Mean-shift

# SIFT: Overall Procedure at a High Level

1. Scale-space extrema detection

   Search over multiple scales and image locations.

2. Keypoint localization

   Fit a model to determine location and scale. Select keypoints based on a measure of stability.

3. Orientation assignment

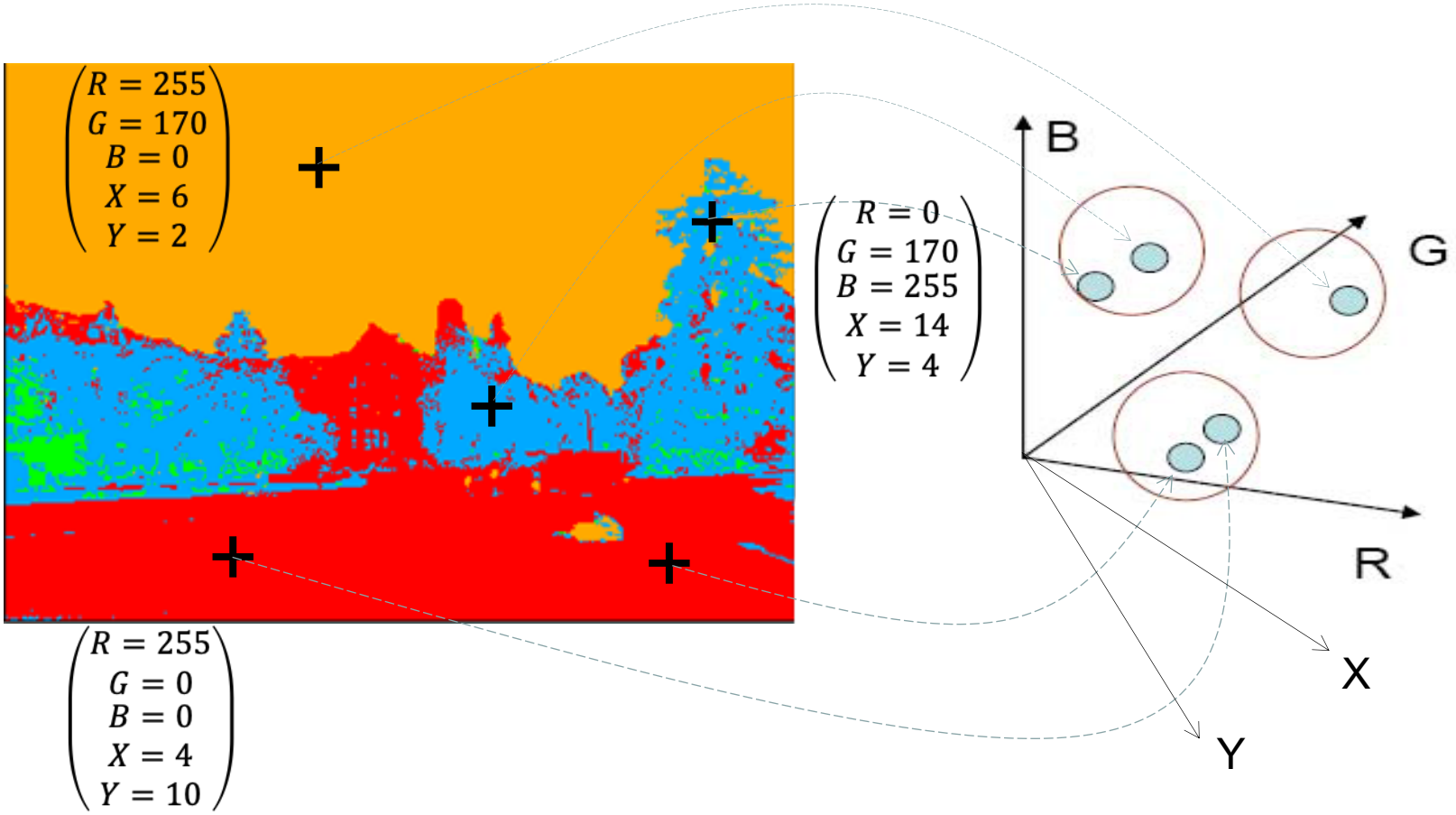
   Compute best orientation(s) for each key point region.

4. Keypoint description

   Use local image gradients at selected scale and rotation to describe each key point region.

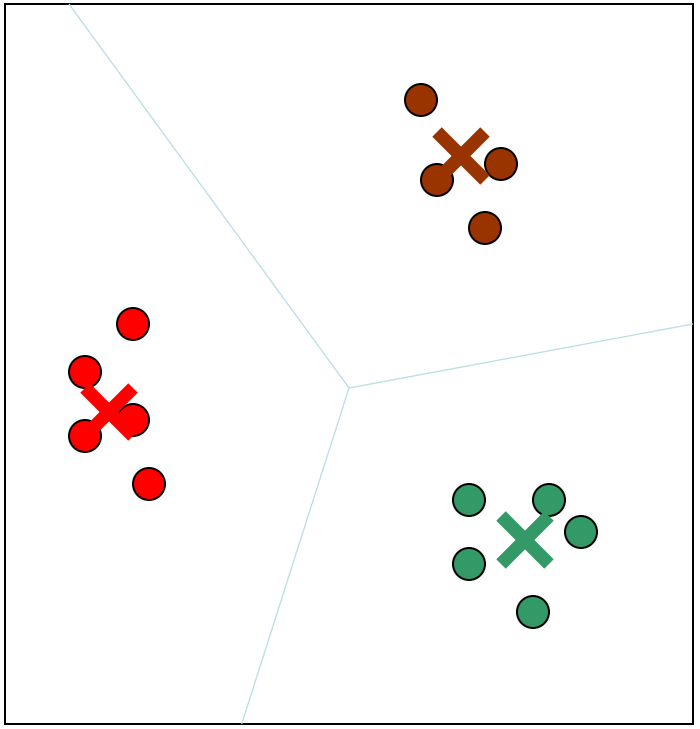
# Segmentation as clustering

- Cluster similar pixels together (RGB+ XY)



# Re-cap: K-means clustering

- Start with 3 random positions of the cluster centers.
- By computing the distance, *assign* each data point to the closest center.
- Re-compute the centers after the assignments.
- Iterate until no points are reassigned.



Iteration = 3

# Recap: K-means clustering

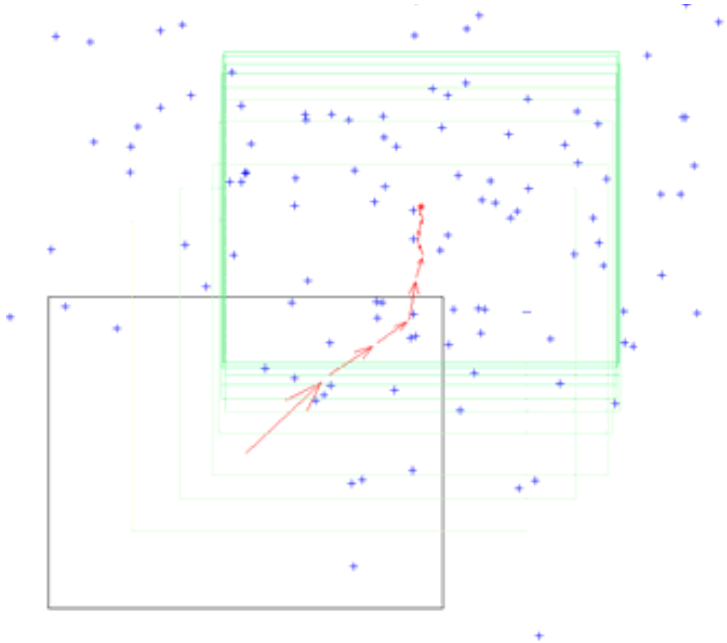
1. Select a value of K
2. Select a feature vector for every pixel (color, texture, position, or combination of these etc.)
3. Define a similarity measure between feature vectors (Usually Euclidean distance)
4. Apply the K-means algorithm to all the feature vectors



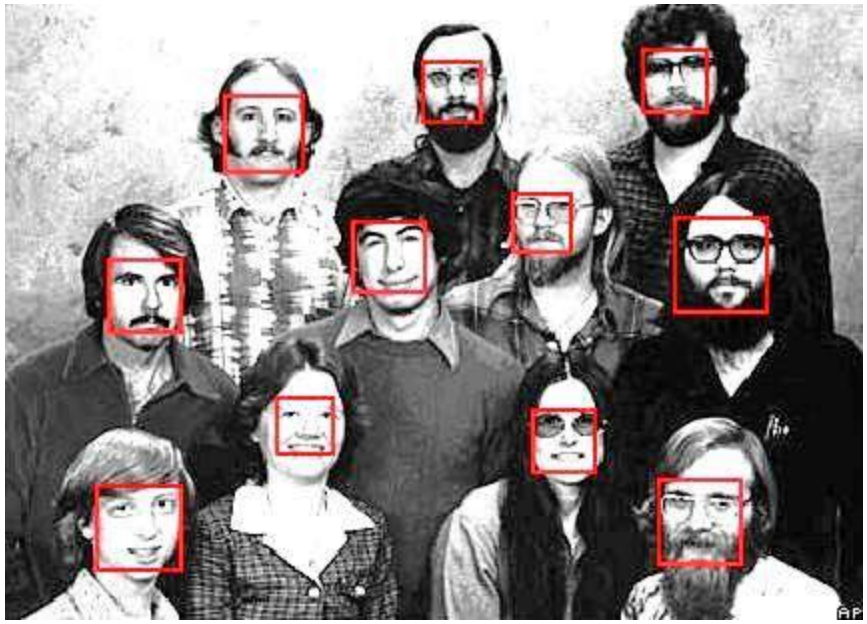
* From Marc Pollefeys COMP 256 2003

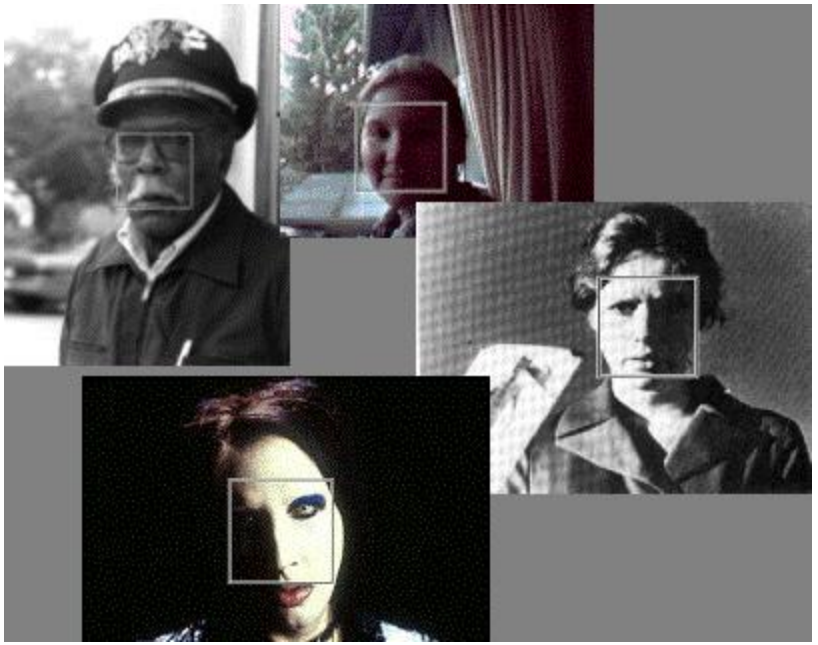
# Recap: Mean-Shift Clustering

- Choose a search window (width and location)
- Compute the mean of the data in the search window
- Center the search window at the new mean location
- Repeat until convergence



# Face detection and Face recognition



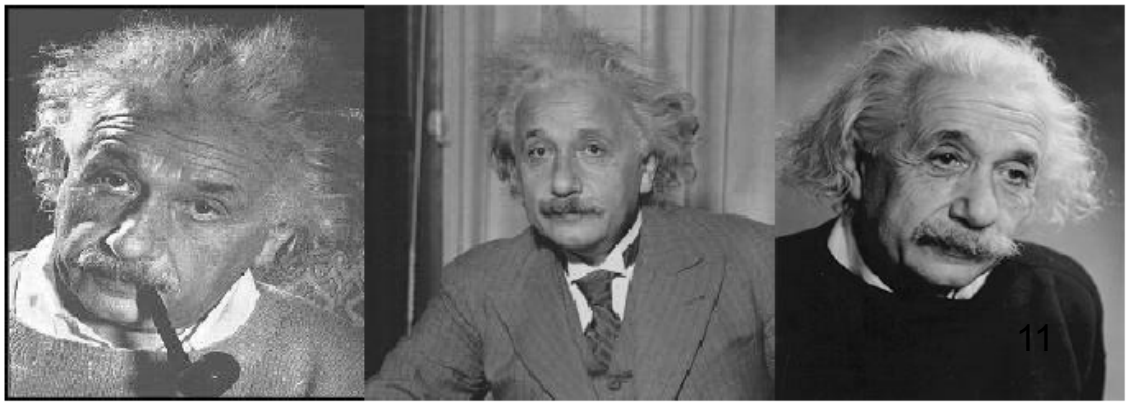


Slide adapted from K. Grauman and D. Lowe

# Natural Face Recognition Ability

- Humans perform face recognition almost instantaneously
- Highly invariant to pose, scale, rotation, lighting changes
- Can handle partial occasions and changes in age
- And we can do all this for faces of several thousand individuals

Who is this?



# Today's lecture

- PCA and EigenFace representation [Turk & Pentland]

- Viola-Jones face detection: [Viola & Jones](next lecture)

# Detection versus Recognition

- Where is it?
  - Object detection
- What is it?
  - Object recognition
- Who is it?
  - Identity recognition/identification/verification.
- What are they doing?
  - Activities recognition.

# Face detection versus recognition

Detection

Recognition

“Sally”

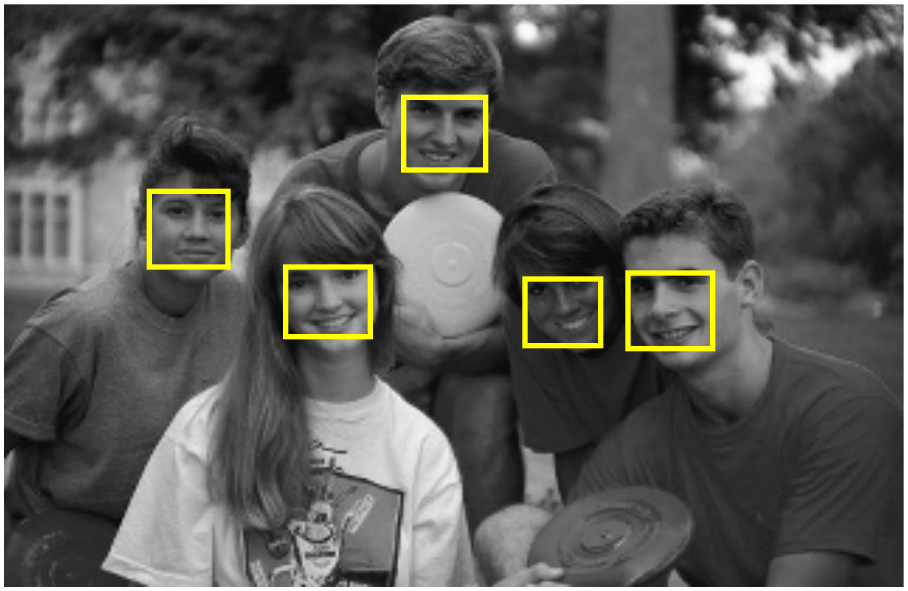
# Consumer application: iPhoto


http://www.apple.com/ilife/iphoto/

# Visual Object Detection

**Task**

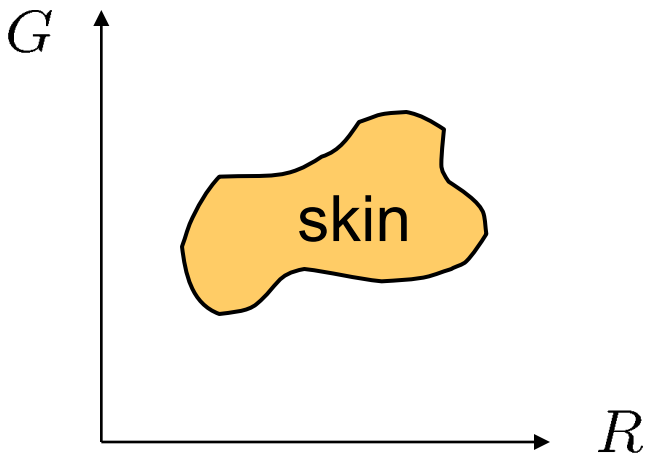
Given an input image, determine if there are objects of a certain category (e.g. faces, people, cars..) in the image and where they are located.



# Common face detection methods

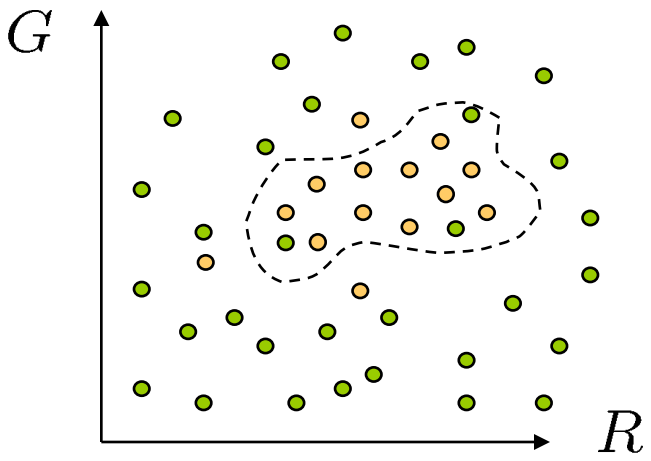
- Colour segmentation
- Template matching
- Eigen Face: Face space representation
- Viola-Jones Adaboost face detection
- SIFT BoW detection
- Deep-Net  detection
- …

# Skin color detection



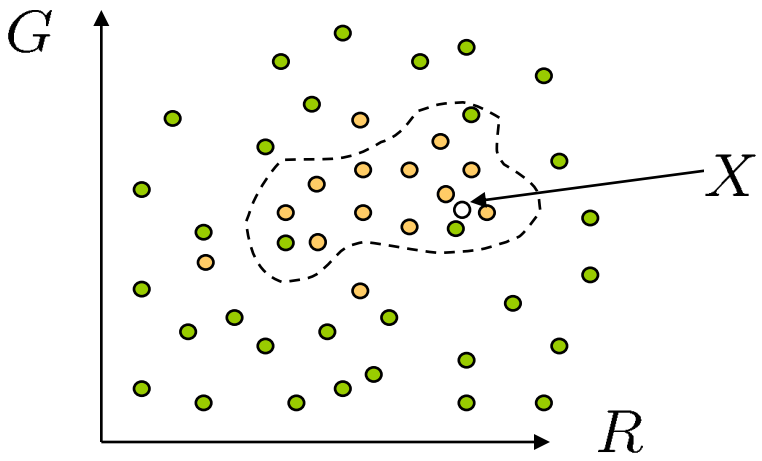

- Skin pixels have a distinctive range of colors
  - Corresponds to region(s) in RGB color space
- Skin classifier
  - A pixel X = (R,G,B) is skin if it is in the skin (color) region
  - How to find this region?

# Skin colour detection




- **Learn** the skin region from examples
  - Manually label skin/non pixels in one or more “training images”
  - Plot the training data in RGB space
    - skin pixels shown in orange, non-skin pixels shown in gray
    - some skin pixels may be outside the region, non-skin pixels inside.

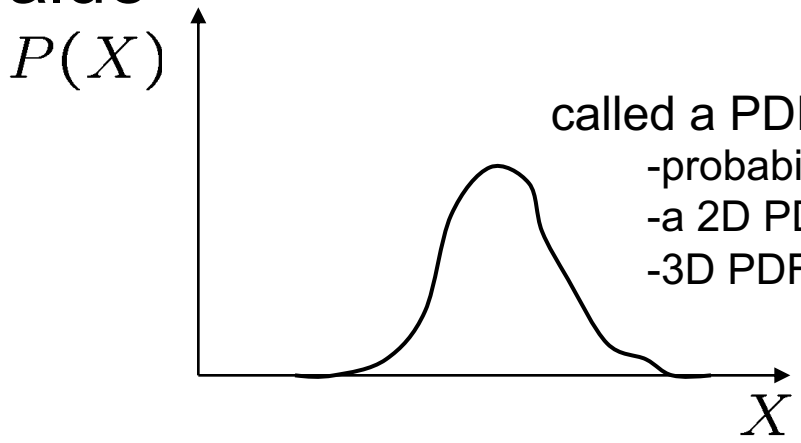
# Skin classifier




- Given X = (R,G,B): how to determine if it is skin or not?
  - Nearest neighbor
    - find labeled pixel closest to X
  - Find plane/curve that separates the two classes
    - popular approach: Support Vector Machines (SVM)
  - Data modeling
    - fit a probability density/distribution model to each class

# Probability

- X is a random variable
- P(X) is the probability that X achieves a certain value




called a PDF

-probability distribution/density function
-a 2D PDF is a surface
-3D PDF is a volume

$$0 \leq P(X) \leq 1$$

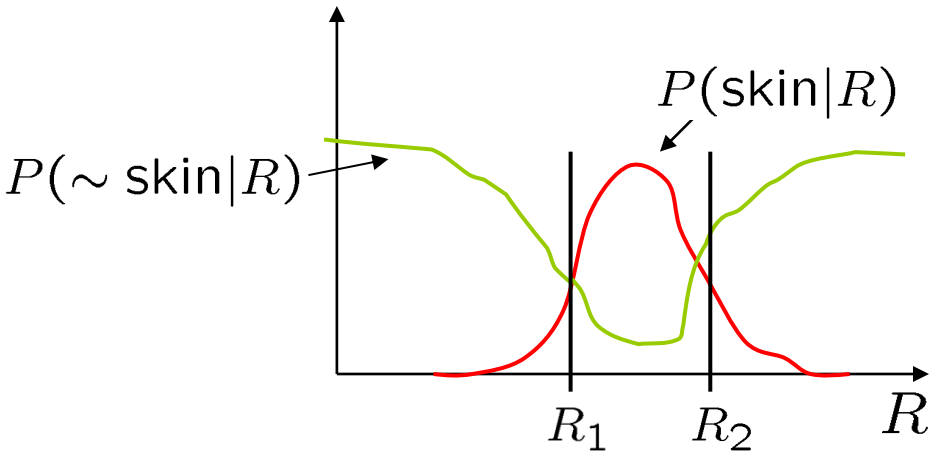
$$\int_{-\infty}^{\infty} P(X)dX = 1$$

continuous X

$$\sum P(X) = 1$$

discrete X

# Probabilistic skin classification



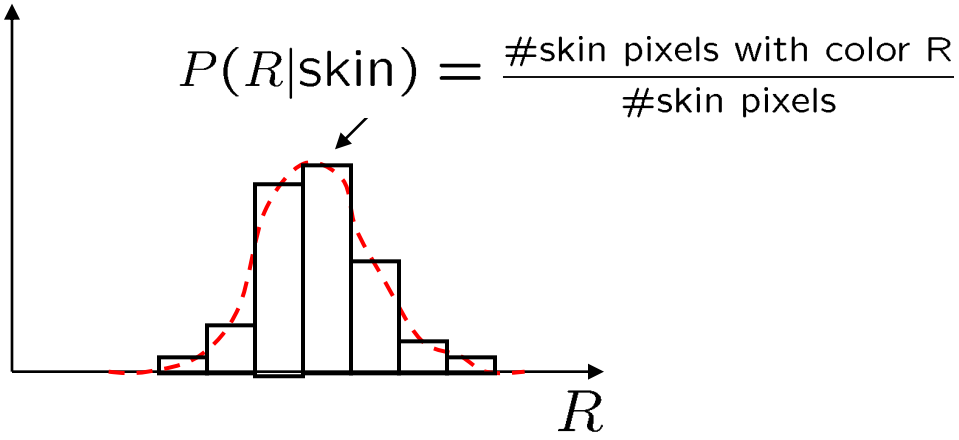

- Model PDF / uncertainty
  - Each pixel has a probability of being skin or not skin

$$P(\sim \text{skin}|R) = 1 - P(\text{skin}|R)$$

- Skin classifier
  - Given X = (R,G,B): how to determine if it is skin or not?
  - Choose interpretation of highest probability

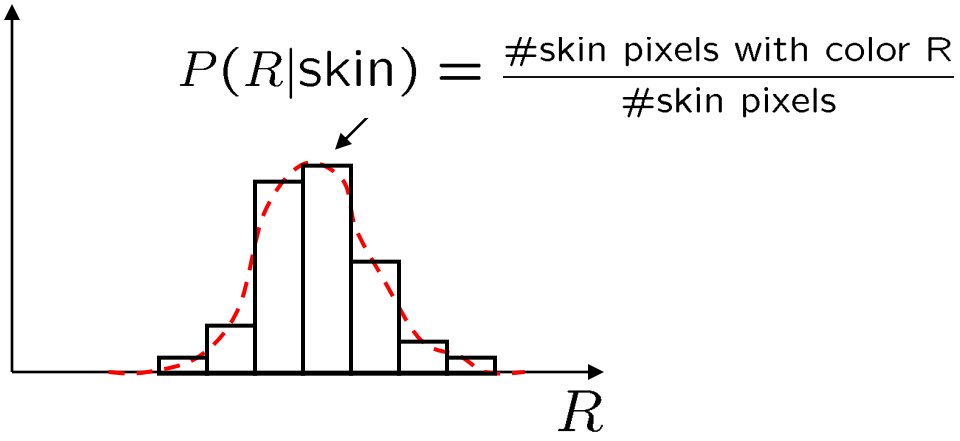
- Where do we get $P(\text{skin}|R)$ and $P(\sim \text{skin}|R)$ ?

# Learning conditional PDF's




- We can calculate P(R | skin) from a set of training images
- It is simply a histogram over the pixels in the training images
  - each bin $R_i$ contains the proportion of skin pixels with color $R_i$

# Learning Conditional-PDF




- We can calculate P(R | skin) from a set of training images
- **But this conditional PDF (aka. Likelihood) is not quite what we want !**
  - Why not? How to determine if a pixel is skin?
  - We want P(skin | R), not P(R | skin)
  - How can we get it?

# Bayes' Law

$$P(X|Y) = \frac{P(Y|X)P(X)}{P(Y)}$$

- In the context of our problem:

what we can measure (**likelihood**)

domain knowledge (**prior**)

$$P(\text{skin}|R) = \frac{P(R|\text{skin})\ P(\text{skin})}{P(R)}$$

what we want (**posterior**)

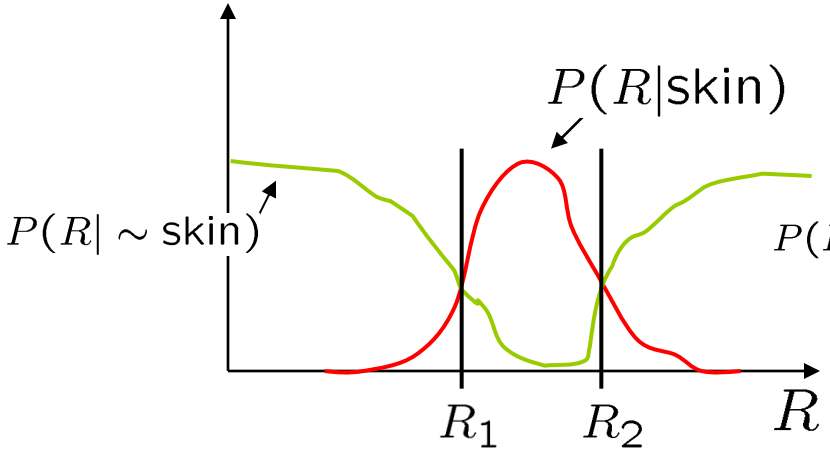
**normalization** term

$$P(R) = P(R|\text{skin})P(\text{skin}) + P(R|\sim\text{skin})P(\sim\text{skin})$$

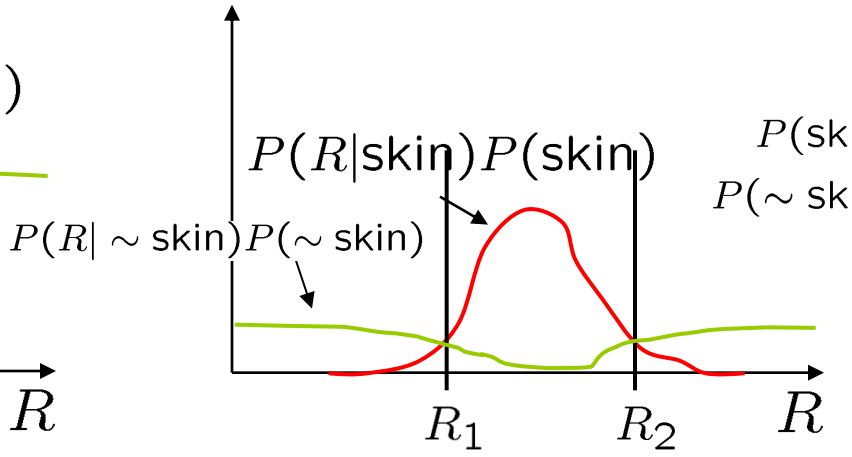
What can we use for the prior P(skin)?

- Domain knowledge:
  - P(skin) may be larger if we know the image contains a person
  - For a portrait, P(skin) may be higher for pixels in the center
- Learn the prior from the training set. How?
  - P(skin) is proportion of skin pixels in training set

# Bayesian estimation




likelihood




posterior (unnormalized)

- Bayesian estimation
  - Goal is to choose the label (skin or ~skin) that maximizes the posterior ↔ minimizes probability of misclassification
    - this is called **Maximum A Posteriori (MAP) estimation**

# Skin detection results

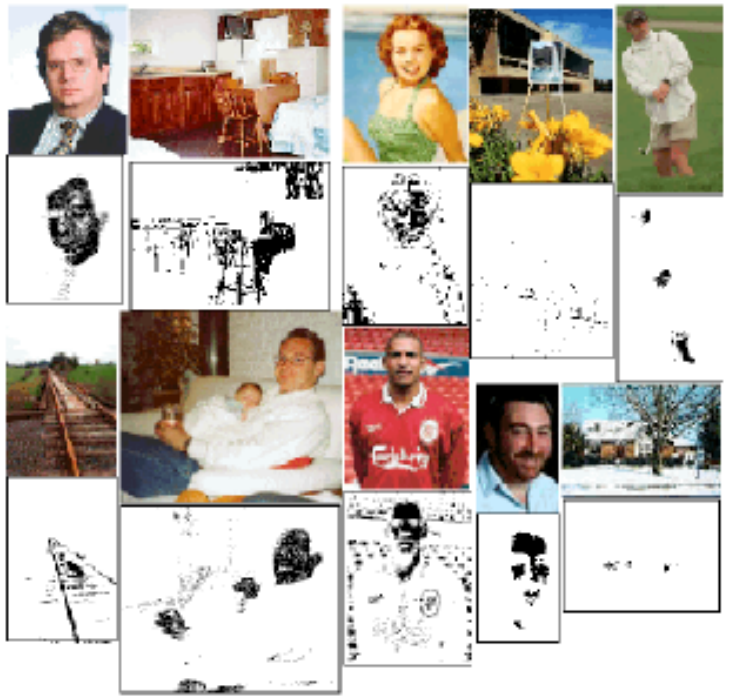


**Figure 25.3.** The figure shows a variety of images together with the output of the skin detector of Jones and Rehg applied to the image. Pixels marked black are skin pixels, and white are background. Notice that this process is relatively effective, and could certainly be used to focus attention on, say, faces and hands. *Figure from "Statistical color models with application to skin detection," M.J. Jones and J. Rehg, Proc. Computer Vision and Pattern Recognition, 1999 © 1999, IEEE*

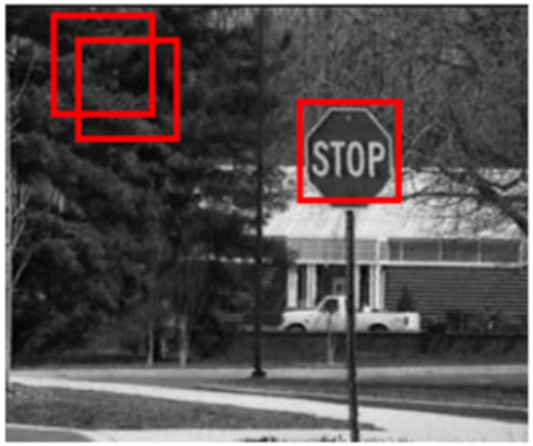
# Template Matching

Objects can be represented by storing sample images or "templates"


Stop sign template

# Hypotheses fromTemplate Matching

- Place the template at every location on the given image.
  - Compare the pixel values in the template with the pixel values in the underlying region of the image.
  - If a "good" match is found, announce that the object is present in the image.


- Possible measures are: SSD, SAD, Cross-correlation, Normalized Cross-correlation, max difference, etc.

# Limitations of Template Matching

- If the object appears scaled, rotated, or skewed on the image, the match will not be good.

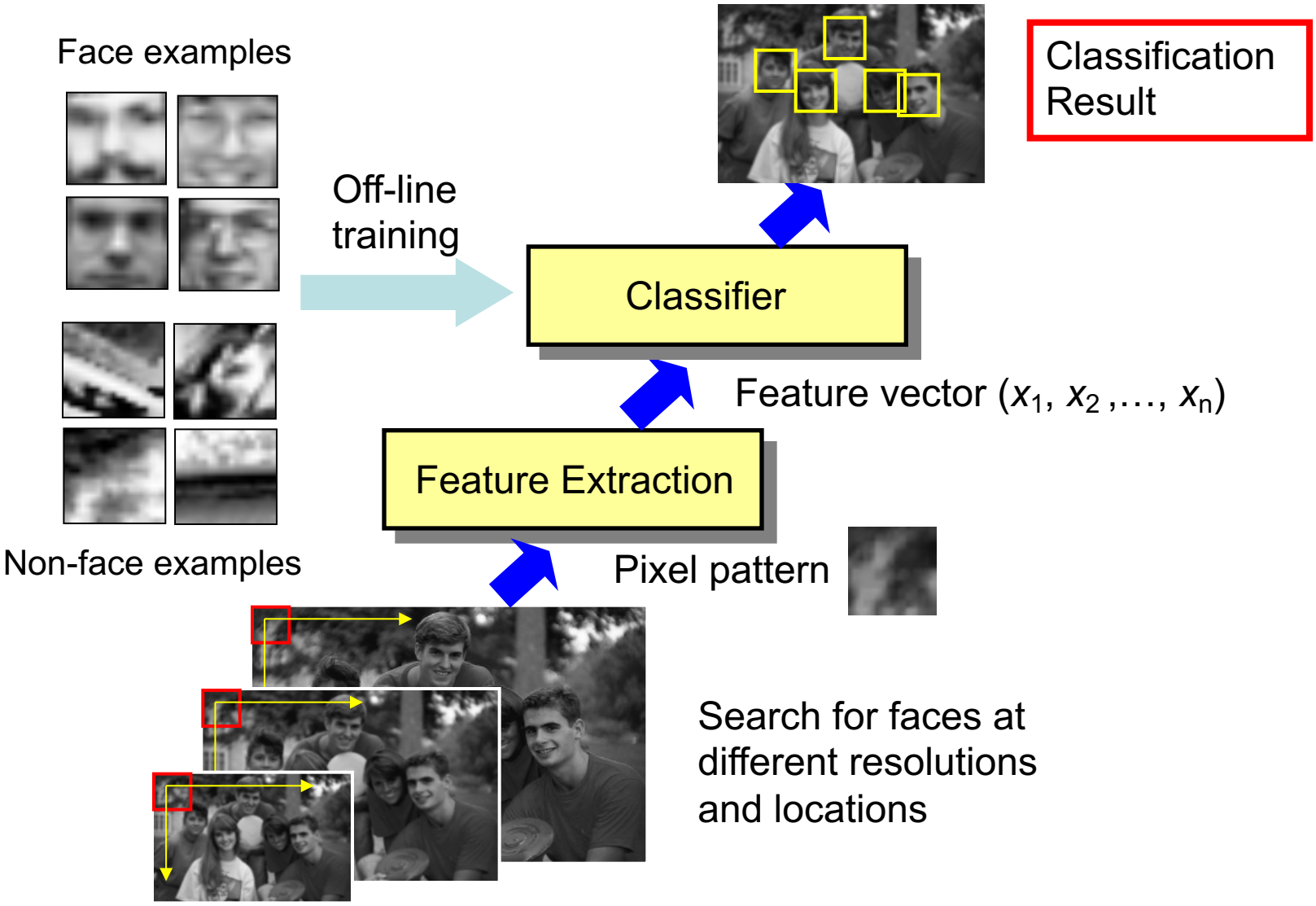

# Solution:

- Search for the template and possible transformations of the template:

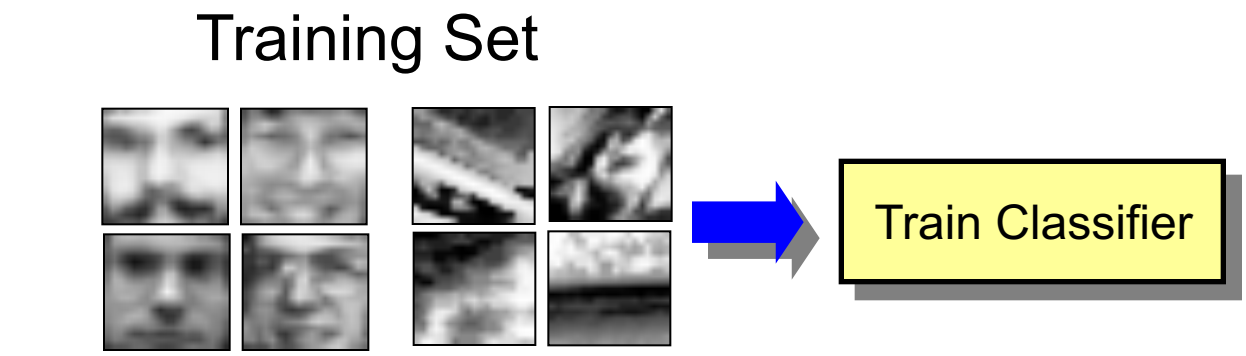


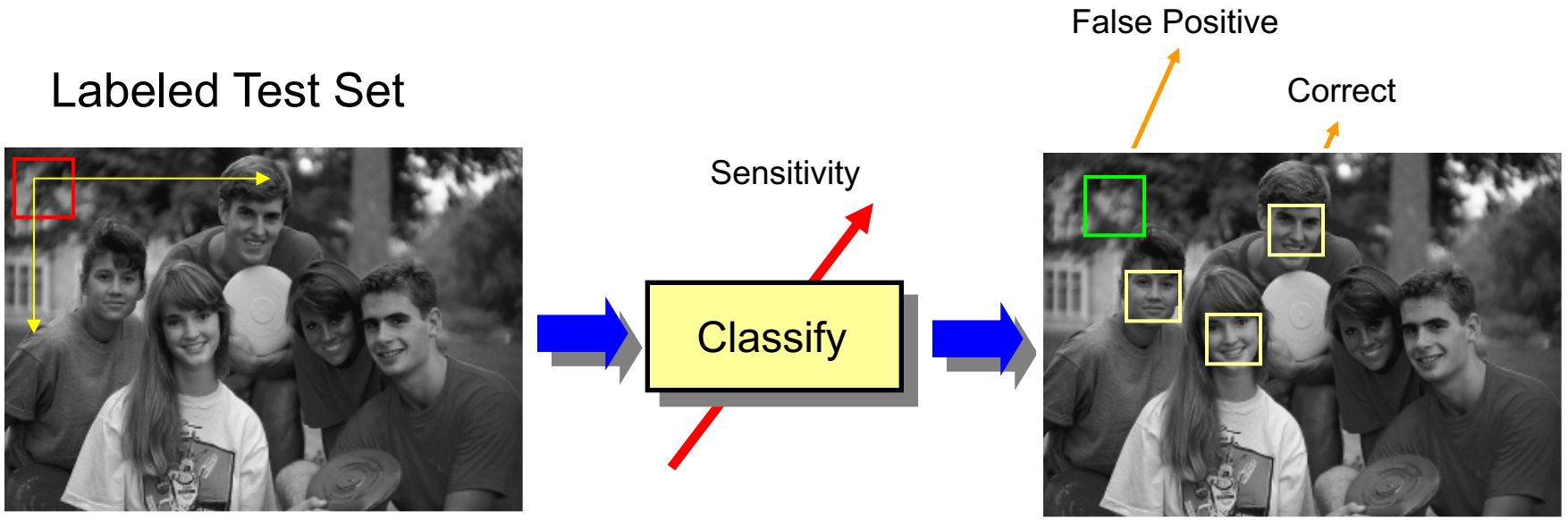
Not very efficient! (but doable …)

# Face Detection – basic scheme



# Training and Testing



# Face detection

- Basic idea: slide a window across image and classify (evaluate) a facial feature representation at every location

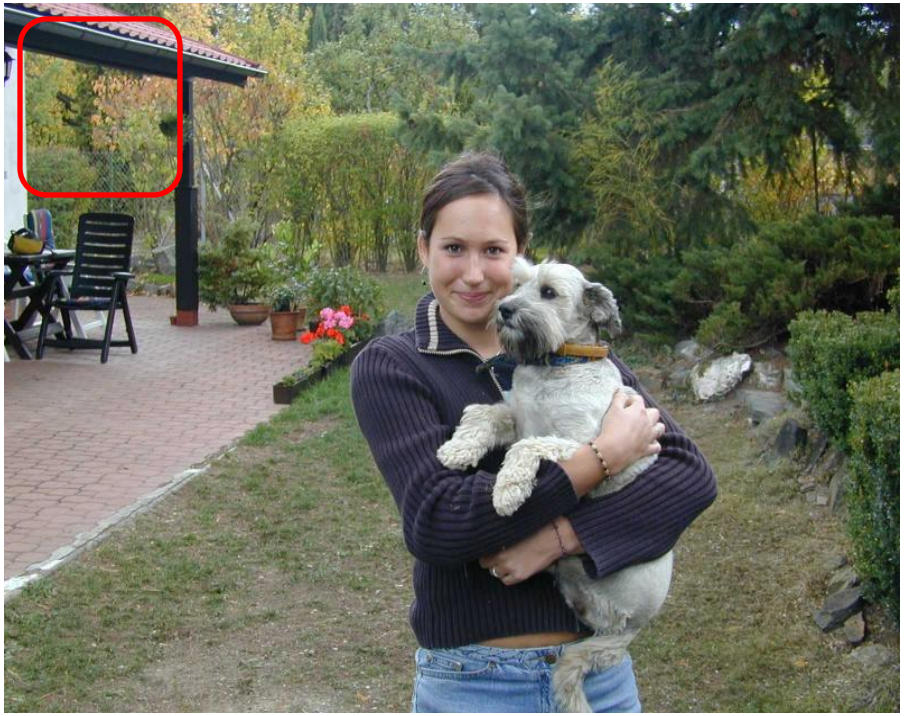
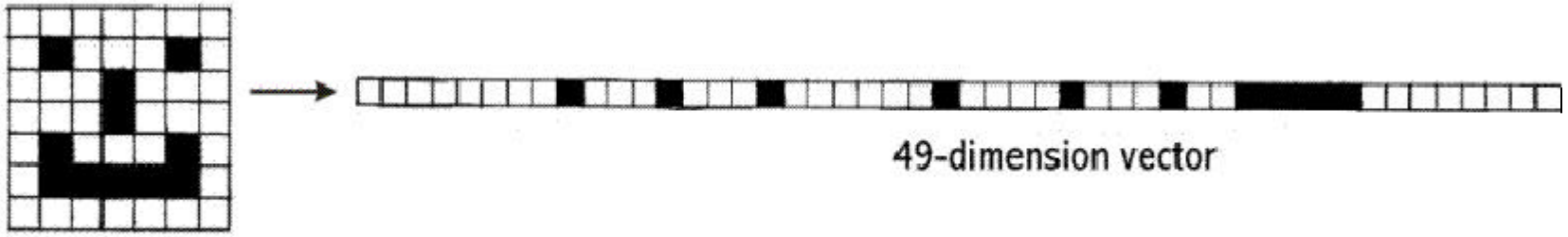


Image feature representation:

Dimensionality reduction

# High Dimensional Correlated Data

- Images as a high dimensional vector




- A typical image used for image processing will be 512x512= 262144 dimension vector!
- (Registered) face images are highly correlated

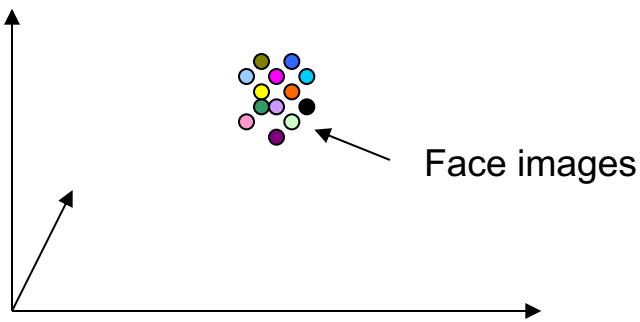



Image space (high dimensional space of all possible images)

# Starting idea of “eigenfaces”

1. Treat pixels as a vector

$$\mathbf{x}$$

2. Recognize face by nearest neighbor

$$\mathbf{y}_1 \cdots \mathbf{y}_n$$

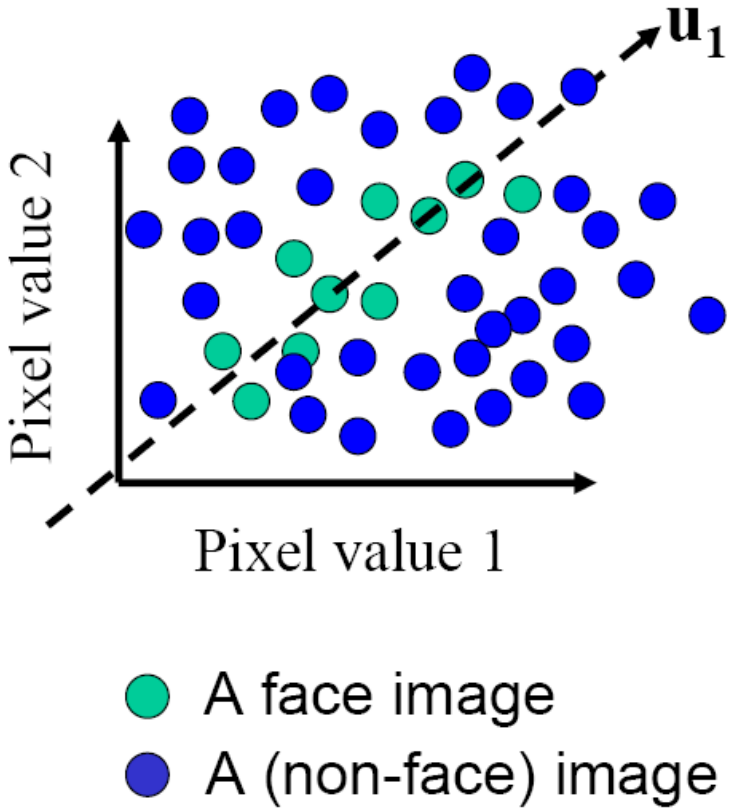
$$k = \underset{k}{\operatorname{argmin}} \left\| \mathbf{y}_k{}^T - \mathbf{x} \right\|$$

# The space of all face images

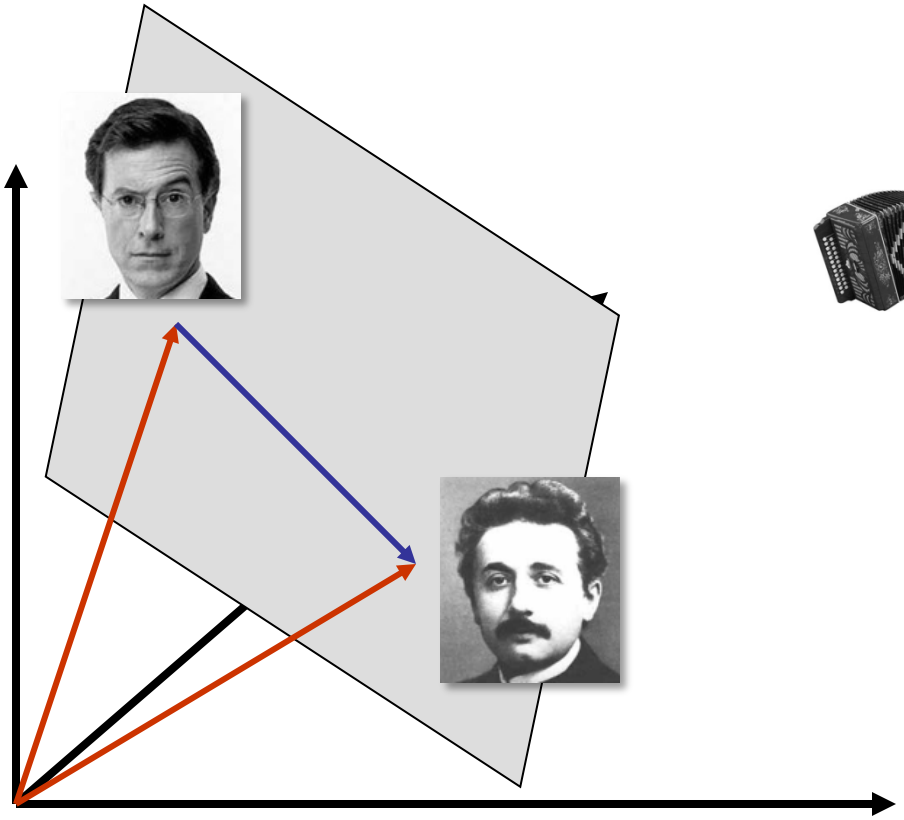
- When viewed as vectors of pixel values, face images are extremely high-dimensional
  - 512x512 image = 262,144 dimensions
  - Slow and lots of storage
- But very few 262,144-dimension long vectors are valid face images;  real face vectors are sparse  (i.e. sparse distribution)  in the data space.
- We want to effectively model the subspace of face images

# The space of all face images

- Eigenface idea:
  - Construct a low-dimensional linear subspace that best explains the variation in the set of face images



# Dimensionality reduction





- The set of faces is a “subspace” of the set of all images
  - Suppose it is K dimensional
  - We can find the best subspace using PCA
  - This is like fitting a “hyper-plane” to the set of faces
    - spanned by vectors $\mathbf{v_1}$, $\mathbf{v_2}$, ..., $\mathbf{v_K}$
    - any face $\mathbf{x} \approx \overline{\mathbf{x}} + a_1\mathbf{v_1} + a_2\mathbf{v_2} + \ldots + a_k\mathbf{v_k}$

# PCA

- General dimensionality reduction technique
- Preserves most of variance with a much more compact representation
  - Lower storage requirements (eigenvectors + a few numbers per face)
  - Faster matching

# Principal Component Analysis (PCA)

- Given: N data points $\mathbf{x}_1, \ldots, \mathbf{x}_N$ in $R^d$
- We want to find a new set of features that are linear combinations of original ones:

$$w = \mathbf{u}^T(\mathbf{x}_i - \boldsymbol{\mu})$$

(**μ**: mean of data points)

- Choose unit vector **u** in $R^d$ that captures the most data variance

# Principal Component Analysis

- Direction that maximizes the variance of the projected data:

$$\underset{\mathbf{u}}{\text{Maximize}} \quad \frac{1}{N}\sum_{i=1}^{N} \underbrace{\mathbf{u}^{\mathrm{T}}(\mathbf{x}_i - \mu)}_{\text{Projection of data point}}(\mathbf{u}^{\mathrm{T}}(\mathbf{x}_i - \mu))^{\mathrm{T}}$$

subject to $||\mathbf{u}||=1$

$$= \mathbf{u}^{\mathrm{T}}\underbrace{\left[1/N\sum_{i=1}^{N}(\mathbf{x}_i - \mu)(\mathbf{x}_i - \mu)^{\mathrm{T}}\right]}_{\text{Covariance matrix of data}}\mathbf{u}$$

$$= \mathbf{u}^{\mathrm{T}}\Sigma\mathbf{u}$$

→ **The direction that maximizes the variance is the eigenvector associated with the largest eigenvalue of Σ**

**PCA algorithm.**

(i) Find the centroid of the points $\mathbf{y}_i$, given by

$$\bar{\mathbf{y}} = \frac{1}{k}\sum_{i=1}^{k} \mathbf{y}_i \ .$$

(ii) Form the scatter matrix

$$\mathtt{Y} = \sum_{i=1}^{k} (\mathbf{y}_i - \bar{\mathbf{y}})(\mathbf{y}_i - \bar{\mathbf{y}})^{\mathsf{T}} \ .$$

(iii) Let $\mathtt{A}$ be the matrix whose columns are the eigenvectors corresponding to the $m$ **largest** eigenvalues of $\mathtt{Y}$.

(iv) The best hyperplane fit to the points $\{\mathbf{y}_i\}$ is the affine subspace of points of the form $\mathbf{x}' = \mathtt{A}\mathbf{x} + \bar{\mathbf{y}}$, for $\mathbf{x} \in \mathbb{R}^m$.

(v) This can also be written as the points $\mathbf{x}'$ satisfying the equation

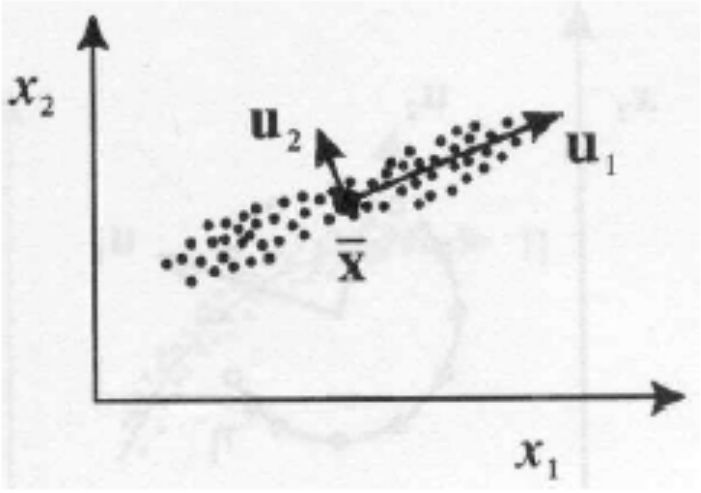
$$(\mathtt{A}^{\perp})^{\mathsf{T}}(\mathbf{x}' - \bar{\mathbf{y}}) = \mathbf{0}$$

where $\mathtt{A}^{\perp}$ is the matrix consisting of the columns of $\mathtt{Y}$ corresponding to the $n - m$ **smallest** eigenvalues.

Table 1.1. PCA algorithm

# Geometric interpretation

- PCA projects the data along the directions where the data varies most.
- These directions are determined by the eigenvectors of the covariance matrix corresponding to the largest eigenvalues.
- The magnitude of the eigenvalues corresponds to the variance of the data along the eigenvector directions.



# PCA for dimension reduction

- Lower dimensionality basis
  - Approximate vectors by finding a basis in an appropriate lower dimensional space.

(1) Higher-dimensional space representation:

$$x = a_1 v_1 + a_2 v_2 + \cdots + a_N v_N$$

$v_1, v_2, ..., v_N$ is a basis of the $N$-dimensional space

(2) Lower-dimensional space representation:

$$\hat{x} = b_1 u_1 + b_2 u_2 + \cdots + b_K u_K$$

$u_1, u_2, ..., u_K$ is a basis of the $K$-dimensional space

- *Note:* if both bases have the same size ($N = K$), then $x = \hat{x}$)

# PCA Algorithm

- Suppose $x_1$, $x_2$, ..., $x_M$ are $N$ x 1 vectors

Step 1: $\bar{x} = \frac{1}{M}\sum_{i=1}^{M} x_i$

Step 2: subtract the mean: $\Phi_i = x_i - \bar{x}$ (i.e., center at zero)

Step 3: form the matrix $A = [\Phi_1\ \Phi_2 \cdots \Phi_M]$ ($N$ x $M$ matrix), then compute:

$$C = \frac{1}{M}\sum_{n=1}^{M} \Phi_n \Phi_n^T = \frac{1}{M} AA^T$$

(sample **covariance** matrix, $N$ x $N$, characterizes the *scatter* of the data)

Step 4: compute the eigenvalues of $C$: $\lambda_1 > \lambda_2 > \cdots > \lambda_N$

Step 5: compute the eigenvectors of $C$: $u_1, u_2, \ldots, u_N$

# PCA Algorithm

- Since $C$ is symmetric, $u_1, u_2, \ldots, u_N$ form a basis, (i.e., any vector $x$ or actually $(x - \bar{x})$, can be written as a linear combination of the eigenvectors):

$$x - \bar{x} = b_1 u_1 + b_2 u_2 + \cdots + b_N u_N = \sum_{i=1}^{N} b_i u_i \qquad b_i = \frac{(x - \bar{x}).u_i}{(u_i.u_i)}$$

Step 6: (**dimensionality reduction step**) keep only the terms corresponding to the $K$ largest eigenvalues:

$$\hat{x} - \bar{x} = \sum_{i=1}^{K} b_i u_i \text{ where } K << N$$

- The representation of $\hat{x} - \bar{x}$ into the basis $u_1, u_2, \ldots, u_K$ is thus

$$\begin{bmatrix} b_1 \\ b_2 \\ \ldots \\ b_K \end{bmatrix}$$

# PCA algorithm

- The linear transformation $R^N \rightarrow R^K$ that performs the dimensionality reduction is:

$$\begin{bmatrix} b_1 \\ b_2 \\ \dots \\ b_K \end{bmatrix} = \begin{bmatrix} u_1^T \\ u_2^T \\ \dots \\ u_K^T \end{bmatrix} (x - \bar{x}) = U^T (x - \bar{x})$$

(i.e., simply computing coefficients of linear expansion)

**PCA algorithm.**

(i) Find the centroid of the points $\mathbf{y}_i$, given by

$$\bar{\mathbf{y}} = \frac{1}{k}\sum_{i=1}^{k} \mathbf{y}_i \ .$$

(ii) Form the scatter matrix

$$\mathtt{Y} = \sum_{i=1}^{k} (\mathbf{y}_i - \bar{\mathbf{y}})(\mathbf{y}_i - \bar{\mathbf{y}})^{\mathsf{T}} \ .$$

(iii) Let $\mathtt{A}$ be the matrix whose columns are the eigenvectors corresponding to the $m$ **largest** eigenvalues of $\mathtt{Y}$.

(iv) The best hyperplane fit to the points $\{\mathbf{y}_i\}$ is the affine subspace of points of the form $\mathbf{x}' = \mathtt{A}\mathbf{x} + \bar{\mathbf{y}}$, for $\mathbf{x} \in \mathbb{R}^m$.

(v) This can also be written as the points $\mathbf{x}'$ satisfying the equation

$$(\mathtt{A}^{\perp})^{\mathsf{T}}(\mathbf{x}' - \bar{\mathbf{y}}) = \mathbf{0}$$

where $\mathtt{A}^{\perp}$ is the matrix consisting of the columns of $\mathtt{Y}$ corresponding to the $n - m$ **smallest** eigenvalues.

Table 1.1. PCA algorithm

# Eigenfaces (PCA on face images)

1. Compute covariance matrix of face images

2. Compute the principal components ("eigenfaces")
   - K eigenvectors with largest eigenvalues

3. Represent all face images in the dataset as linear combinations of eigenfaces
   - Perform nearest neighbor on these coefficients

M. Turk and A. Pentland, Face Recognition using Eigenfaces, CVPR 1991

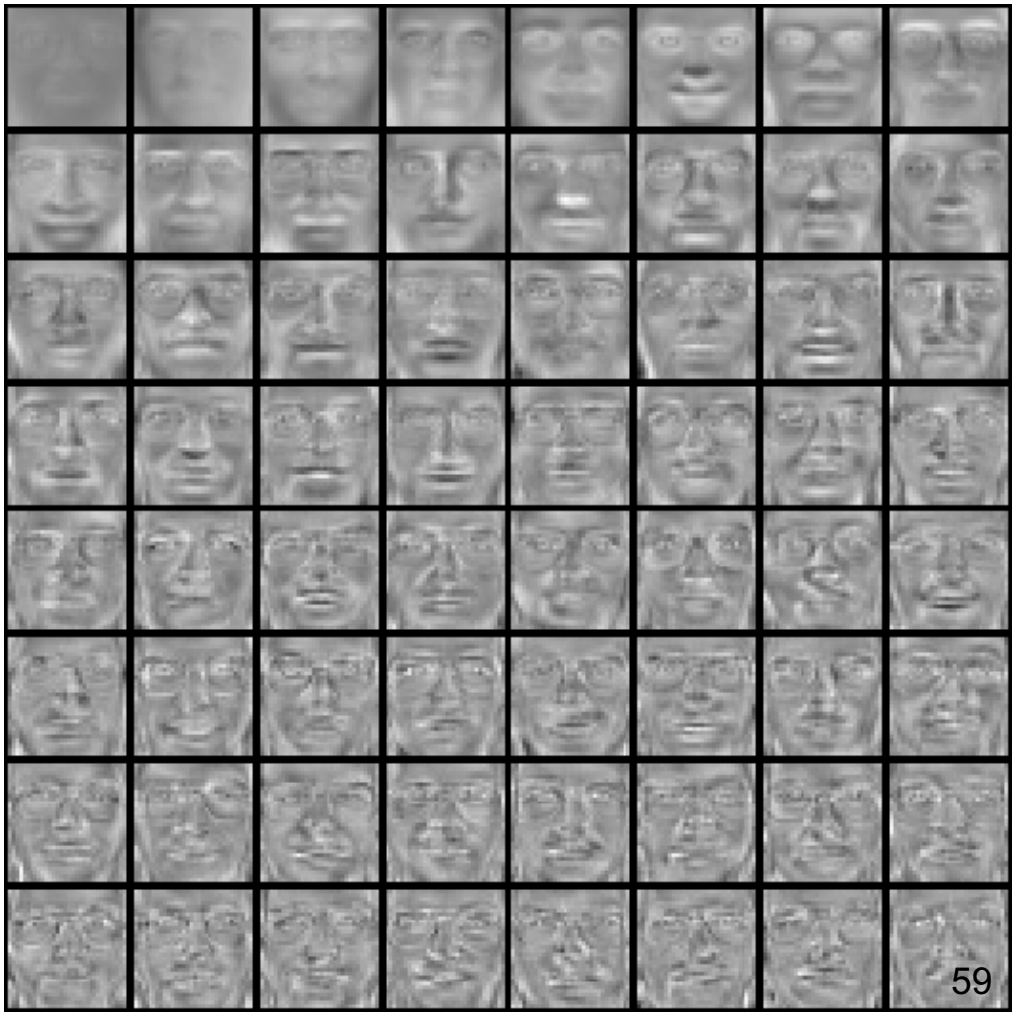
# Eigenfaces example

- Training images
- $\mathbf{x}_1, \ldots, \mathbf{x}_N$

# Eigenfaces example

Mean: μ

Top eigenvectors: $u_1,\ldots u_k$



# Representation and reconstruction

- Face **x** in "face space" coordinates:

$$\mathbf{x} \rightarrow [\mathbf{u}_1^{\mathrm{T}}(\mathbf{x} - \mu), \ldots, \mathbf{u}_k^{\mathrm{T}}(\mathbf{x} - \mu)]$$

$$= \quad w_1, \ldots, w_k$$

# Representation and reconstruction

- Face **x** in "face space" coordinates:

$$\mathbf{x} \rightarrow [\mathbf{u}_1^{\mathrm{T}}(\mathbf{x}-\mu), \ldots, \mathbf{u}_k^{\mathrm{T}}(\mathbf{x}-\mu)]$$

$$= \quad w_1, \ldots, w_k$$

- Reconstruction:

$$\hat{x} = \mu + w_1u_1 + w_2u_2 + w_3u_3 + w_4u_4 + \ldots$$

# Reconstruction example

- The visualization of eigenvectors:

These are the first 4 eigenvectors from a training set of 400 images (ORL Face Database). They look like faces, hence called Eigenface.

# Reconstruction

P = 4

P = 200

P = 400

After computing eigenfaces using 400 face images from ORL face database

# How to choose K?

- Choose *K* using the following criterion:

$$\frac{\sum_{i=1}^{K} \lambda_i}{\sum_{i=1}^{N} \lambda_i} > Threshold \quad \text{(e.g., 0.9 or 0.95)}$$

- In this case, we say that we “preserve” 90% or 95% of the information (variance) in the data.

- If K=N, then we “preserve” 100% of the information in the data.

# Eigenfaces

Computed using 400 face images from ORL face database

# **Recognition** with eigenfaces

Process labeled training images

- Find mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$
- Find k principal components (eigenvectors of $\boldsymbol{\Sigma}$) $\mathbf{u}_1,\ldots\mathbf{u}_k$
- Project each training image $\mathbf{x}_i$ onto subspace spanned by principal components:
  $(w_{i1},\ldots,w_{ik}) = (\mathbf{u}_1^T(\mathbf{x}_i - \boldsymbol{\mu}), \ldots, \mathbf{u}_k^T(\mathbf{x}_i - \boldsymbol{\mu}))$

Given novel image $\mathbf{x}$

- Project onto subspace:
  $(w_1,\ldots,w_k) = (\mathbf{u}_1^T(\mathbf{x} - \boldsymbol{\mu}), \ldots, \mathbf{u}_k^T(\mathbf{x} - \boldsymbol{\mu}))$
- Optional: check reconstruction error $\hat{\mathbf{x}} - \mathbf{x}$ to determine whether image is really a face
- Classify as closest training face in k-dimensional subspace



M. Turk and A. Pentland, Face Recognition using Eigenfaces, CVPR 1991

# Reconstruction from partial information

- Robust to partial face occlusion.

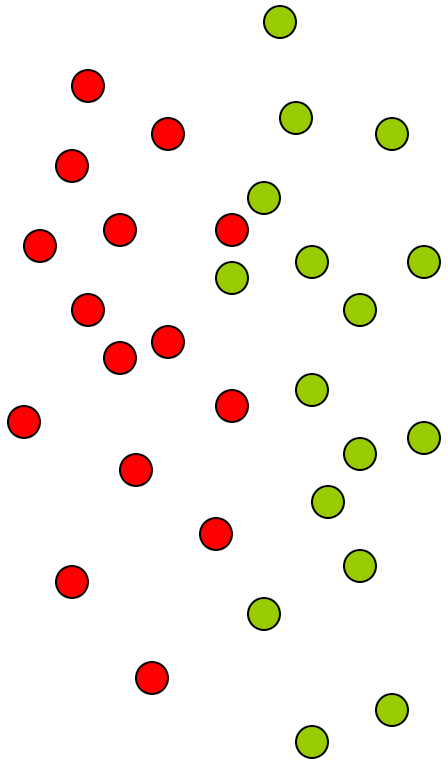
Input Reconstructed

# Limitations

Global appearance method: not robust to misalignment, background variation

# Limitations

- The direction of maximum variance is not always good for classification



# A more discriminative subspace: FLD

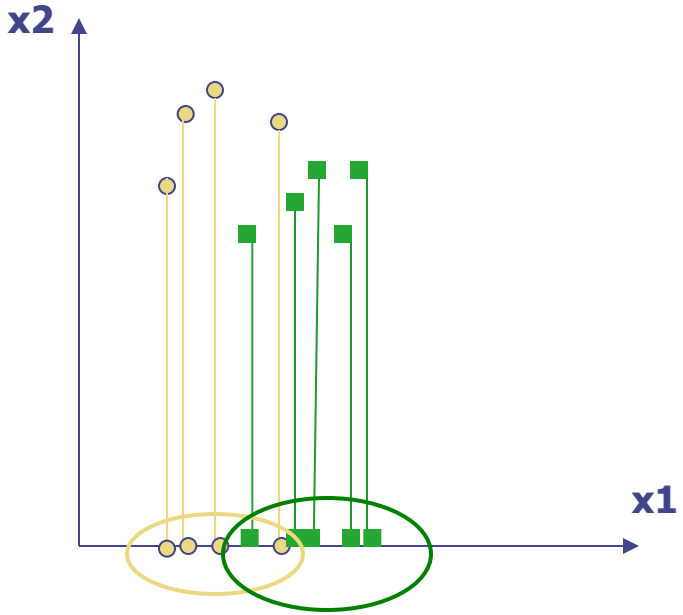
- Fisher Linear Discriminants → "Fisher Faces"

- PCA preserves maximum variance

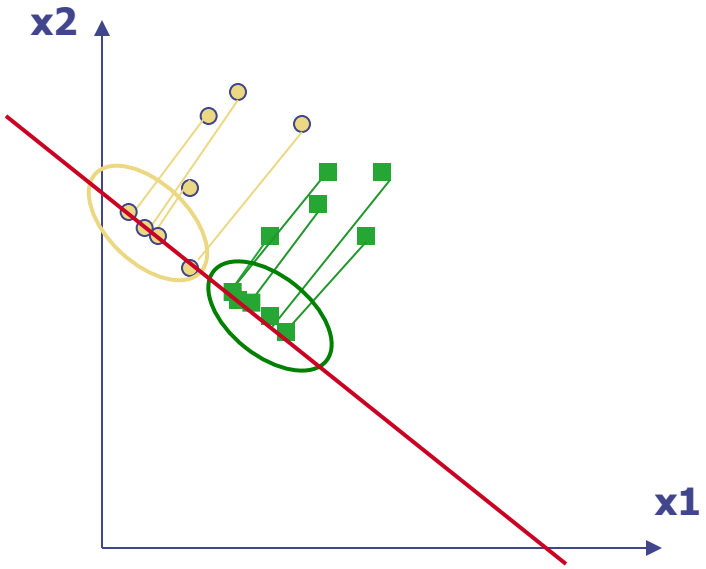
- FLD preserves discrimination
  - Find projection that maximizes scatter between classes and minimizes scatter within classes

# Illustration of the Projection

- Using two classes as an example:

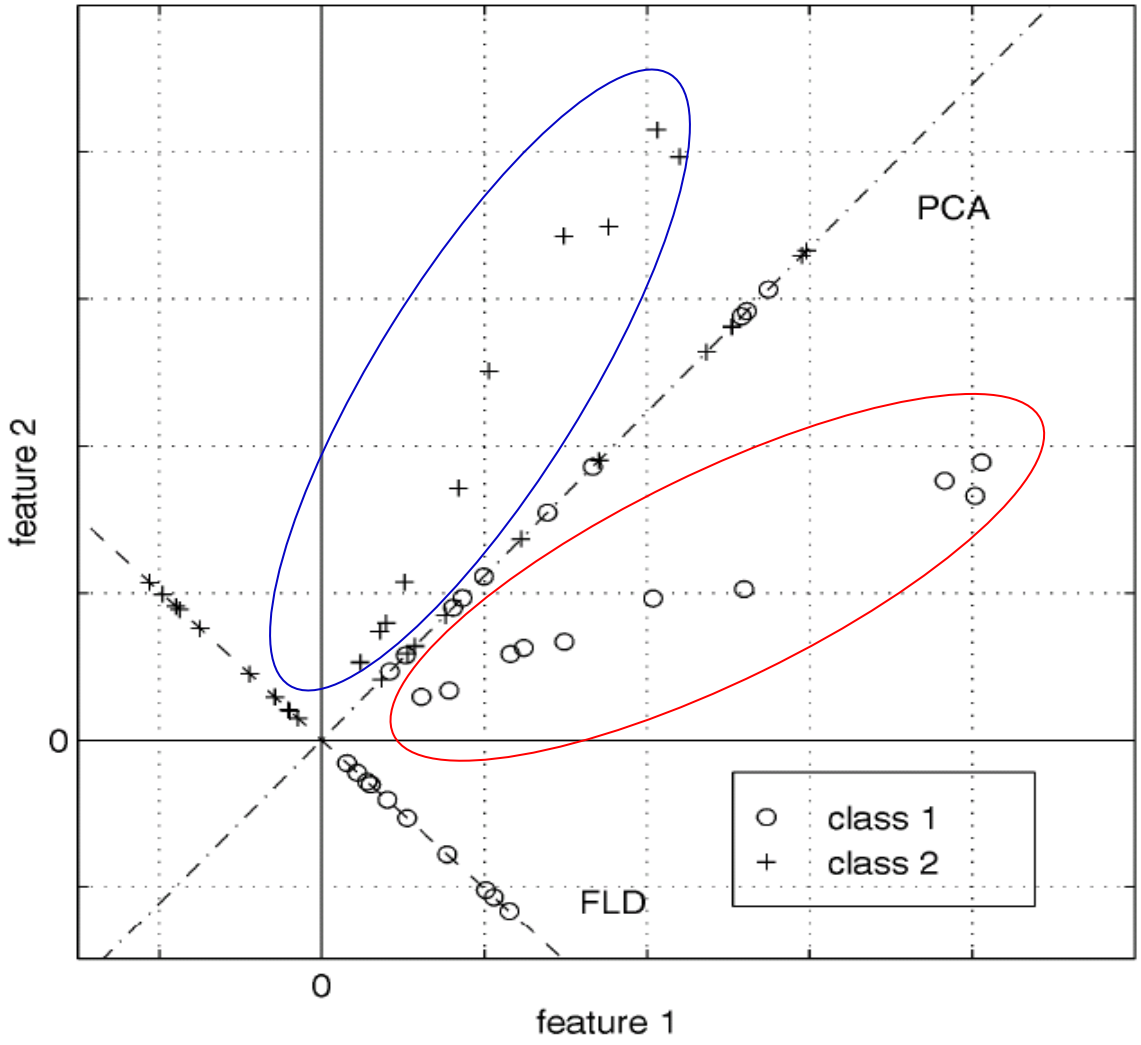



Poor Projection




Good

# Comparing with PCA



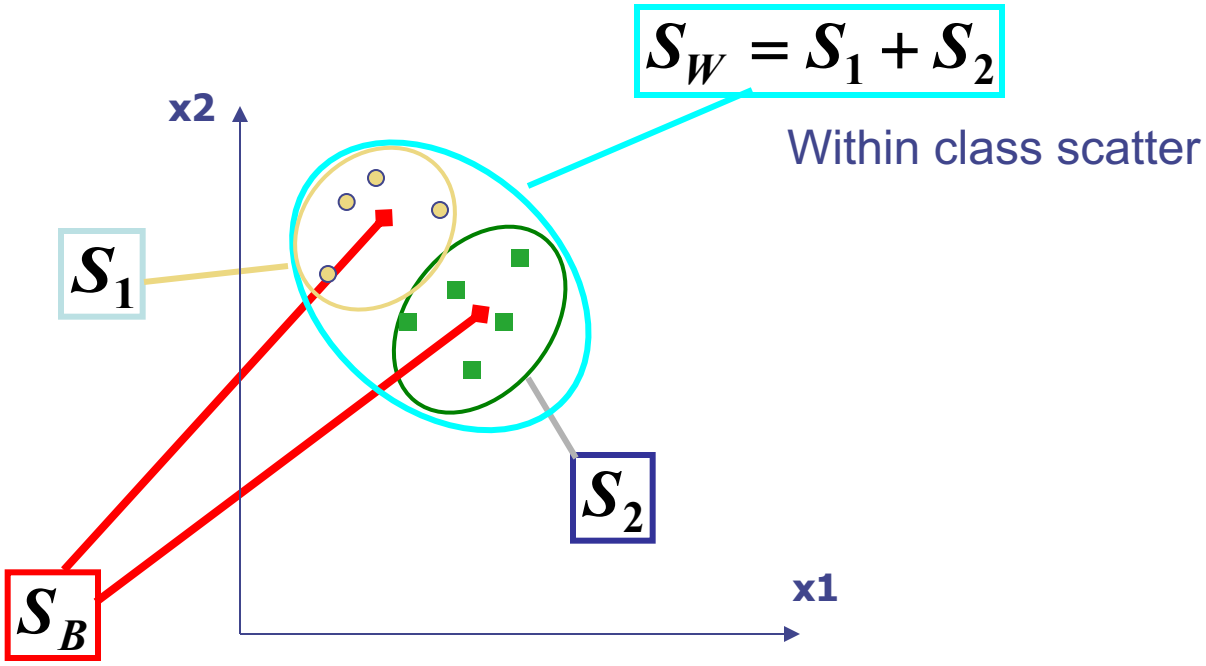
# Variables

- N Sample images: $\{x_1, \cdots, x_N\}$
- c classes: $\{\chi_1, \cdots, \chi_c\}$
- Average of each class: $\mu_i = \frac{1}{N_i} \sum_{x_k \in \chi_i} x_k$
- Average of all data: $\mu = \frac{1}{N} \sum_{k=1}^{N} x_k$

# Scatter Matrices

- Scatter of class i: $$S_i = \sum_{x_k \in \chi_i} (x_k - \mu_i)(x_k - \mu_i)^T$$
- Within class scatter: $$S_W = \sum_{i=1}^{c} S_i$$
- Between class scatter: $$S_B = \sum_{i=1}^{c} N_i (\mu_i - \mu)(\mu_i - \mu)^T$$

# Illustration



# Mathematical Formulation

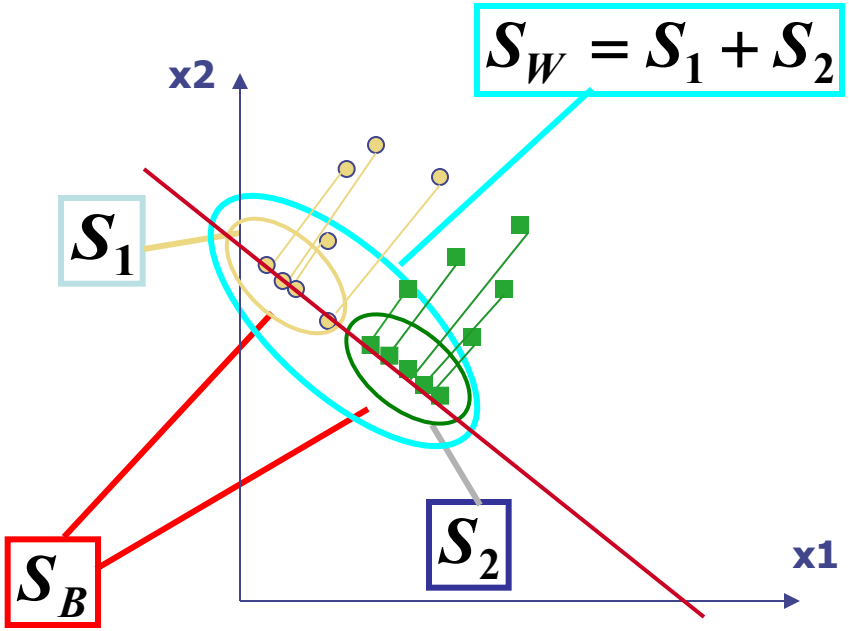
- After projection $y_k = W^T x_k$
  - Between class scatter $\tilde{S}_B = W^T S_B W$
  - Within class scatter $\tilde{S}_W = W^T S_W W$
- Objective

$$W_{opt} = \arg\max_{\mathrm{W}} \frac{\left|\tilde{S}_B\right|}{\left|\tilde{S}_W\right|} = \arg\max_{\mathrm{W}} \frac{\left|W^T S_B W\right|}{\left|W^T S_W W\right|}$$

- Solution: Generalized Eigenvectors

$$S_B w_i = \lambda_i S_W w_i \qquad i = 1, \ldots, m$$

# Illustration



# Recognition with FLD

- Similar to “eigenfaces”
- Compute within-class and between-class scatter matrices

$$S_i = \sum_{x_k \in \chi_i} (x_k - \mu_i)(x_k - \mu_i)^T \qquad S_W = \sum_{i=1}^{c} S_i \qquad S_B = \sum_{i=1}^{c} N_i (\mu_i - \mu)(\mu_i - \mu)^T$$

- Solve generalized eigenvector problem

$$W_{opt} = \arg\max_{\mathrm{w}} \frac{\left| W^T S_B W \right|}{\left| W^T S_W W \right|} \qquad S_B w_i = \lambda_i S_W w_i \qquad i = 1, \ldots, m$$

- Project to FLD subspace and classify by nearest neighbor

$$\hat{x} = {W_{opt}}^T x$$

# Results: Eigenface vs. Fisherface

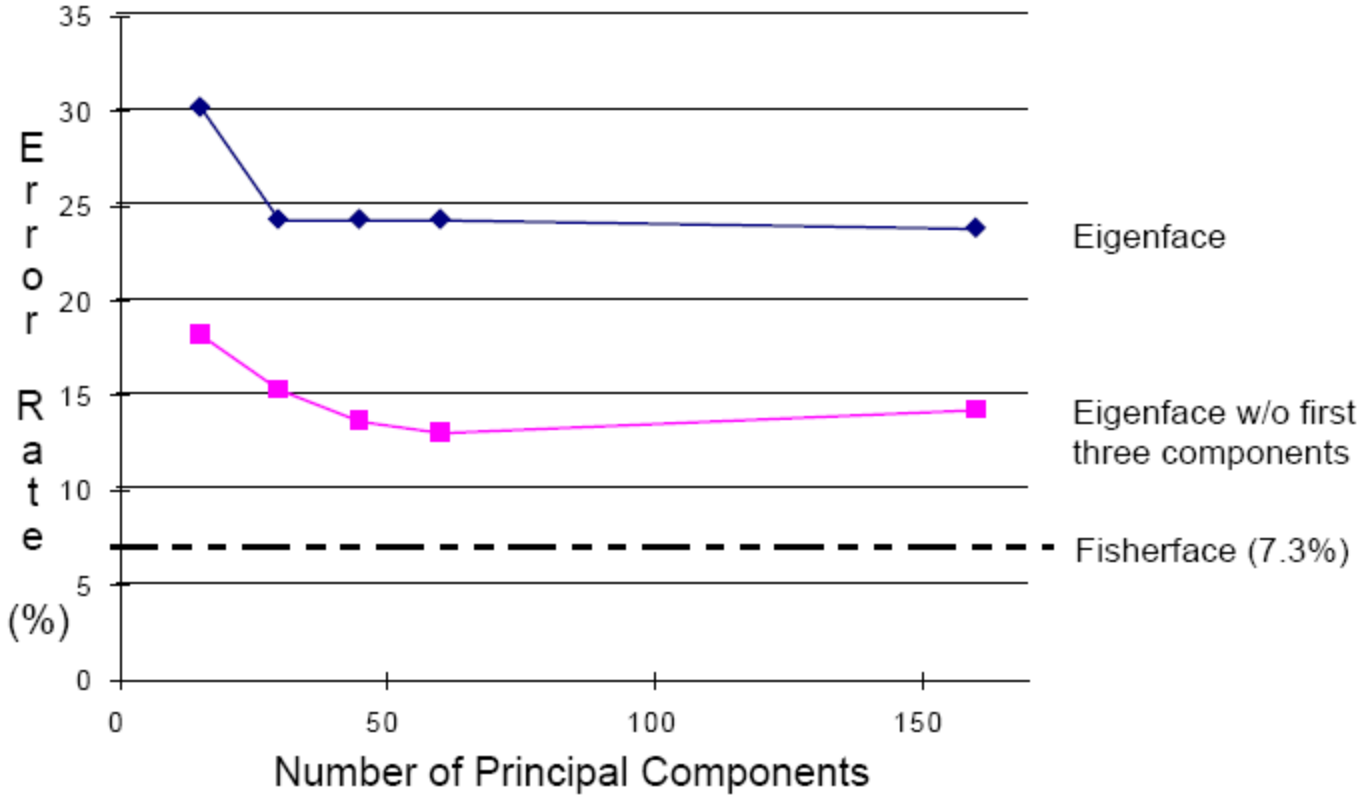
- Input: 160 images of 16 people
- Train: 159 images
- Test: 1 image

- Variation in Facial Expression, Eyewear, and Lighting




Reference: Eigenfaces vs. Fisherfaces, Belheumer et al., PAMI 1997

# Eigenfaces vs. Fisherfaces






Reference: Eigenfaces vs. Fisherfaces, Belheumer et al., PAMI 1997

# Readings

- Chapter 5.2.3 in the book:

Computer Vision: Algorithms and Applications

# Next lecture:

- Viola-Jones face detection
  - Boosting/Adaboost
- Other object detection techniques
  - Bag of words
  - DPM
  - Deep learning