Biomedical Imaging

& Analysis

Lecture 6, Part 7. Fall 2014

Basic Image Processing / Filtering (VII)

*[**Text**: Ch: 10, Gonzalez and Woods, Digital Image Processing (3rd Edition) + Papers / Reading Assignments on Blackboard.]*

Prahlad G Menon, PhD

Assistant Professor

Sun Yat-sen University – Carnegie Mellon University (SYSU-CMU)

Joint Institute of Engineering

Principal Component Analysis (PCA)

- **Pattern recognition in high-dimensional spaces**
  - Problems arise when performing recognition in a high-dimensional space (curse of dimensionality).
  - Significant improvements can be achieved by first mapping the data into a *lower-dimensional sub*-space.

$$x = \begin{bmatrix} a_1 \\ a_2 \\ \dots \\ a_N \end{bmatrix} --> \textit{reduce dimensionality} --> y = \begin{bmatrix} b_1 \\ b_2 \\ \dots \\ b_K \end{bmatrix} \quad (K << N)$$

  - The goal of PCA is to reduce the dimensionality of the data **while retaining as much information as possible** in the original dataset.

Principal Component Analysis (PCA)

- **Dimensionality reduction**
  - PCA allows us to compute a linear transformation that maps data from a high dimensional space to a lower dimensional sub-space.

K x N

$$y = Tx \text{ where } T = \begin{bmatrix} t_{11} & t_{12} & \cdots & t_{1N} \\ t_{21} & t_{22} & \cdots & t_{2N} \\ \cdots & \cdots & \cdots & \cdots \\ t_{K1} & t_{K2} & \cdots & t_{KN} \end{bmatrix}$$

$$\begin{aligned} b_1 &= t_{11}a_1 + t_{12}a_2 + \ldots + t_{1n}a_N \\ b_2 &= t_{21}a_1 + t_{22}a_2 + \ldots + t_{2n}a_N \\ &\cdots \\ b_K &= t_{K1}a_1 + t_{K2}a_2 + \ldots + t_{KN}a_N \end{aligned}$$

Principal Component Analysis (PCA)

- **Methodology**
  - Suppose $x_1, x_2, \ldots, x_M$ 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 $A = [\Phi_1 \ \Phi_2 \cdots \Phi_M]$ (NxM matrix), then compute:

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

(sample **covariance** matrix, NxN, 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: $u_1, u_2, \ldots, u_N$

Principal Component Analysis (PCA)

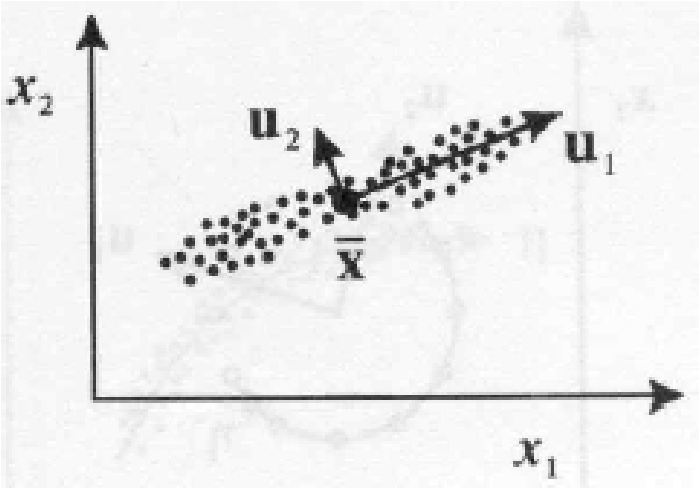
- **Linear transformation implied by PCA**
  - The effective linear transformation $R^N \rightarrow R^K$ that performs the dimensionality reduction is:

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

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

Principal Component Analysis (PCA)

- **Geometric interpretation**
  - PCA projects the data along the directions where the data varies the 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.



Principal Component Analysis (PCA)

- **How to choose K (i.e., number of principal components) ?**
  - To choose K, use 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 in our data. If K=N, then we “preserve” 100% of the information in our data.
  - PCA minimizes the reconstruction error, e, and it can be shown that the error is equal to:

$$e = \|x - \hat{x}\| \quad \rightarrow \quad e = 1/2 \sum_{i=K+1}^{N} \lambda_i$$

PCA **Standardization**

- **Standardization**
  - The principal components are dependent on the *units* used to measure the original variables as well as on the *range* of values they assume.
  - Always standardize the data prior to using PCA.
  - A common standardization method is to transform all the data to have zero mean and unit standard deviation:

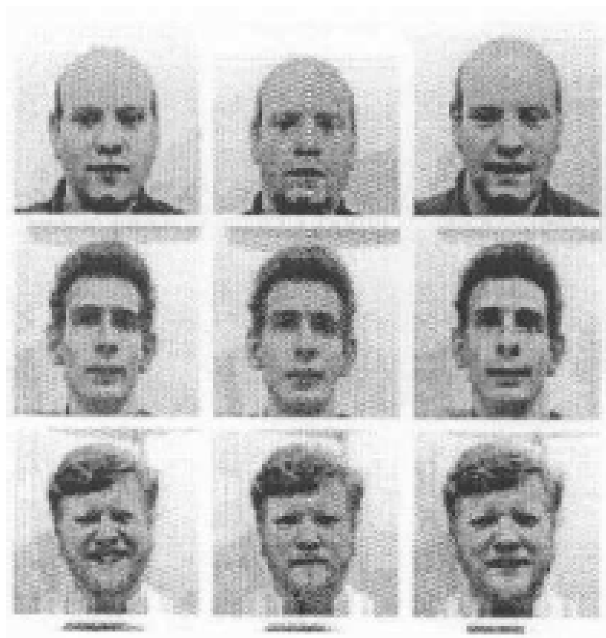
$$\frac{x_i - \boldsymbol{\mu}}{\boldsymbol{\sigma}} \quad (\boldsymbol{\mu} \text{ and } \boldsymbol{\sigma} \text{ are the mean and standard deviation of } x_i\text{'s})$$

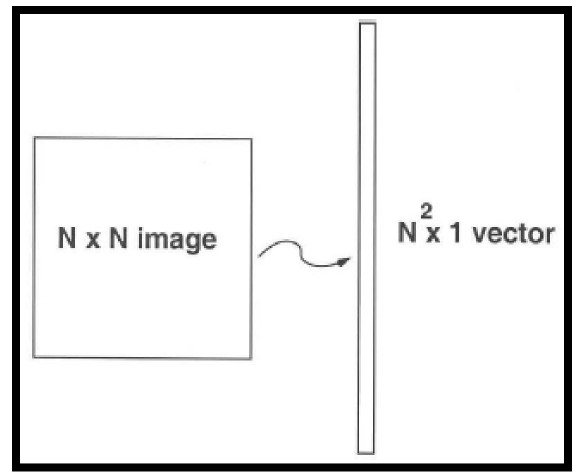
Application to **Faces**

- **Computation of low-dimensional basis (i.e.,eigenfaces):**

Step 1: obtain face images I_1, I_2, ..., I_M (training faces)

(**very important:** the face images must be *centered* and of the same *size*)





Step 2: represent every image I_i as a vector Γ_i

Application to Faces

- Computation of the **eigenfaces** – contd…

Step 3: compute the average face vector Ψ:

$$\Psi = \frac{1}{M} \sum_{i=1}^{M} \Gamma_i$$

Step 4: subtract the mean face:

$$\Phi_i - \Gamma_i - \Psi$$

Step 5: compute the covariance matrix C:

$$C = \frac{1}{M} \sum_{n=1}^{M} \Phi_n \Phi_n^T = \frac{1}{M} A A^T \quad (N^2 \text{x} N^2 \text{ matrix})$$

$$\text{where } A = [\Phi_1 \; \Phi_2 \; \cdots \; \Phi_M] \quad (N^2 \text{x} M \text{ matrix})$$

Application to **Faces**

- Computation of the **eigenfaces** – contd...

Step 6: compute the eigenvectors u_i of AA^T $\Rightarrow$ $AA^T u_i = \lambda_i u_i$

The matrix AA^T is very large --> not practical !!

Step 6.1: consider the matrix $A^T A$ ($M \times M$ matrix)

Step 6.2: compute the eigenvectors v_i of $A^T A$

$$A^T A v_i = \mu_i v_i$$

What is the relationship between u_i and v_i?

$A^T A v_i = \mu_i v_i \Rightarrow AA^T A v_i = \mu_i A v_i \Rightarrow$

$$u_i = A v_i \quad and \quad \lambda_i = \mu_i$$

Thus, AA^T and $A^T A$ have the same eigenvalues and their eigenvectors are related as follows: $u_i = A v_i$!!

Application to **Faces**

- Computation of the **eigenfaces** – contd..

Note 1: AA^T can have up to N^2 eigenvalues and eigenvectors.

Note 2: $A^T A$ can have up to M eigenvalues and eigenvectors.

Note 3: The M eigenvalues of $A^T A$ (along with their corresponding eigenvectors) correspond to the M *largest* eigenvalues of AA^T (along with their corresponding eigenvectors).

Step 6.3: compute the M best eigenvectors of AA^T: $u_i = Av_i$

(**important:** normalize u_i such that $\|u_i\| = 1$)

Step 7: keep only K eigenvectors (corresponding to the K largest eigenvalues)

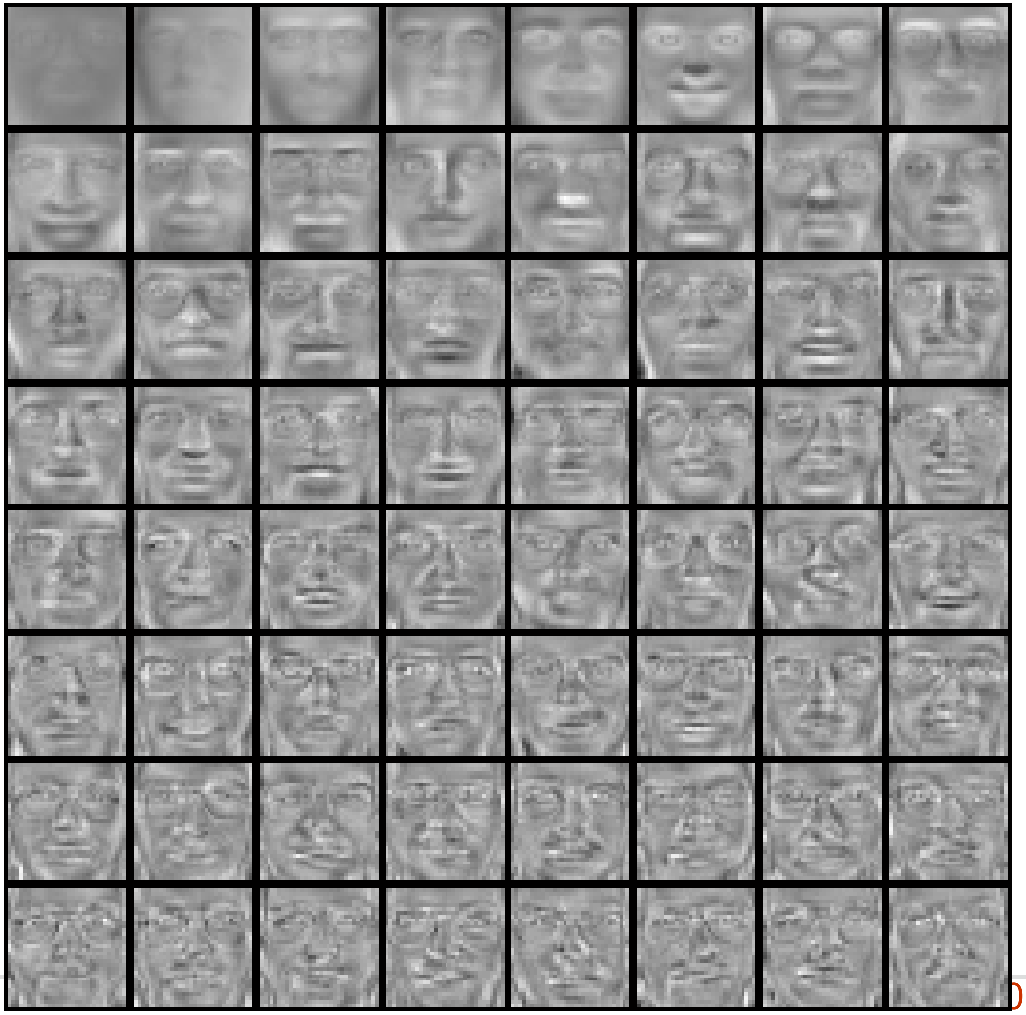
Eigenfaces example

Training images

Eigenfaces example

Mean: μ

Top eigenvectors: $u_1, \ldots u_k$



Application to **Faces**

- Projecting faces onto the computed basis of Principal Components:

  - Each face (minus the mean) Φ_i in the training set can be represented as a linear combination of the best K eigenvectors:

$$\hat{\Phi}_i - mean = \sum_{j=1}^{K} w_j u_j, \quad (w_j = u_j^T \Phi_i)$$

(we call the u_j's *eigenfaces*)

Face reconstruction:

Eigenfaces for **Face Detection**

- Given an unknown image Γ

Step 1: compute $\Phi = \Gamma - \Psi$

Step 2: compute $\hat{\Phi} = \sum_{i=1}^{K} w_i u_i \quad (w_i = u_i^T \Phi)$

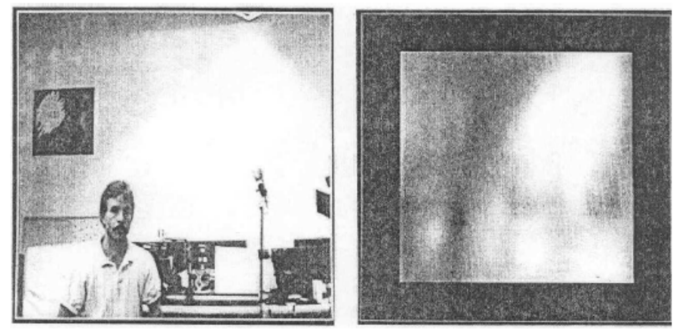
$(where \|u_i\| = 1)$

Step 3: compute $e_d = \|\Phi - \hat{\Phi}\|$

Step 4: if $e_d < T_d$, then Γ is a face.

Visualization of DFFS:

$e_d = \|\Phi - \hat{\Phi}\|$



– The distance e_d is called *distance from face space (DFFS) …*

Eigenfaces for **Face Recognition**

- Given an unknown face image Γ (centered and of the same size like the training faces) follow these steps:

Step 1: normalize Γ: $\Phi = \Gamma - \Psi$

Step 2: project on the eigenspace

$$\hat{\Phi} = \sum_{i=1}^{K} w_i u_i \quad (w_i = u_i^T \Phi) \qquad (where \| u_i \| = 1)$$

Step 3: represent Φ as: $\Omega = \begin{bmatrix} w_1 \\ w_2 \\ \dots \\ w_K \end{bmatrix}$

Step 4: find $e_r = \min_l \|\Omega - \Omega^l\|$ *where* $\| \Omega - \Omega^l \| = \sum_{i=1}^{K} (w_i - w_i^l)^2$

Step 5: if $e_r < T_r$, then Γ is recognized as face l from the training set.

Reconstruction using partial information

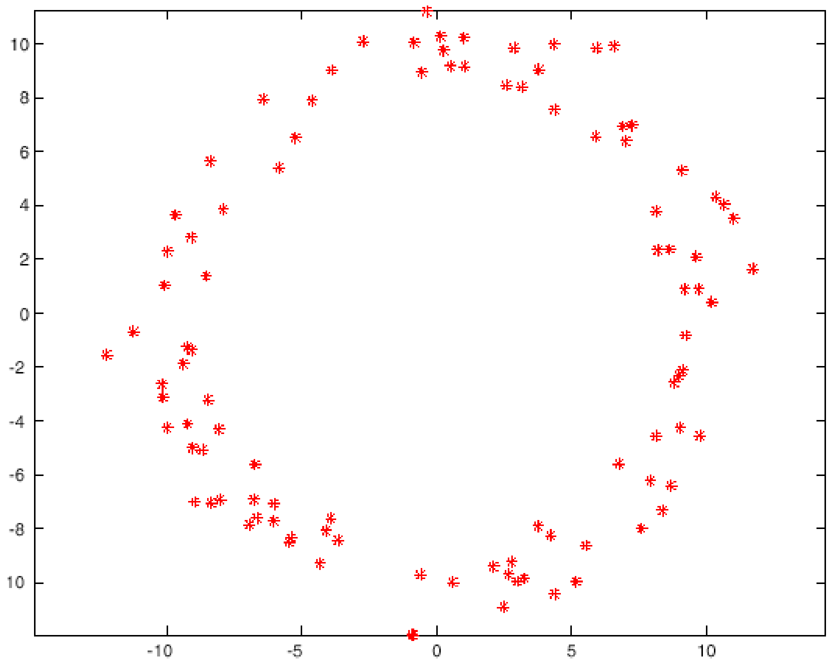
- Robust to **partial face occlusion**:

Input

Reconstructed

Eigen-Faces: **Limitations**

- PCA assumes that the data follows a Gaussian distribution (mean μ, covariance matrix Σ)…
- Eg: The shape of this dataset is not well described by its principal components –

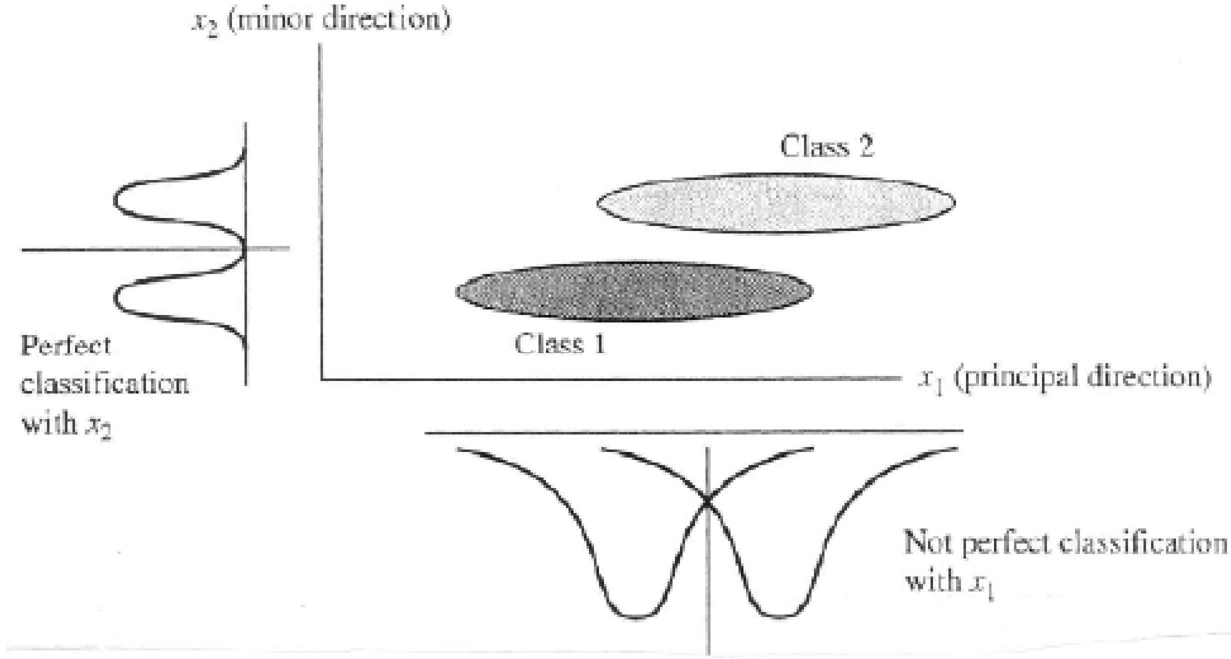


Eigen-Faces: **Limitations**

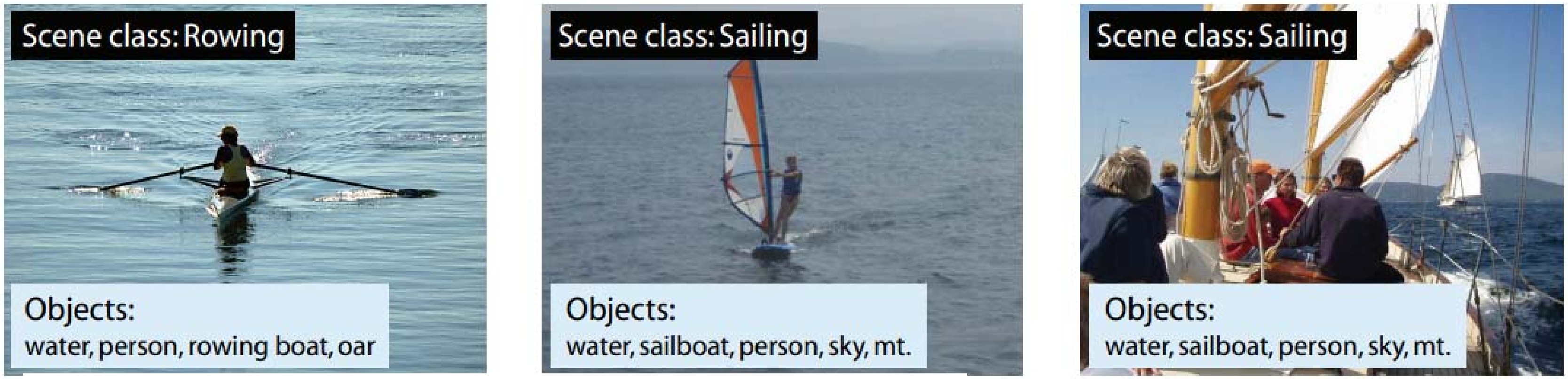
- **Background** changes cause problems
  - De-emphasize the outside of the face (e.g., by multiplying the input image by a 2D Gaussian window centered on the face).
- **Light changes** degrade performance
  - Light normalization helps.
- Performance decreases quickly with changes to **face size**
  - Multi-scale eigenspaces.
  - Scale input image to multiple sizes.
- Performance decreases with changes to **face orientation** (but not as fast as with scale changes)
  - Plane rotations are easier to handle.
  - Out-of-plane rotations are more difficult to handle.

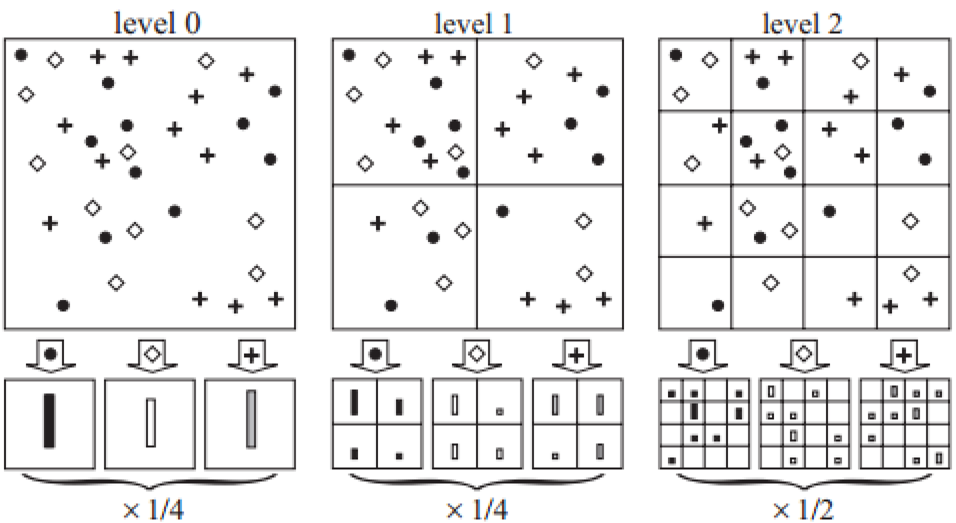
Eigen-Faces: **Limitations**

– PCA is not always an optimal dimensionality-reduction procedure for classification purposes:



Action Recognition – Making sense of Features in a Scene





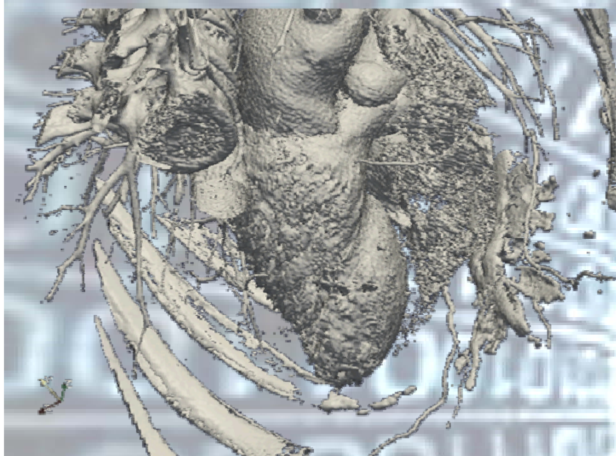
Spatial pyramid representation

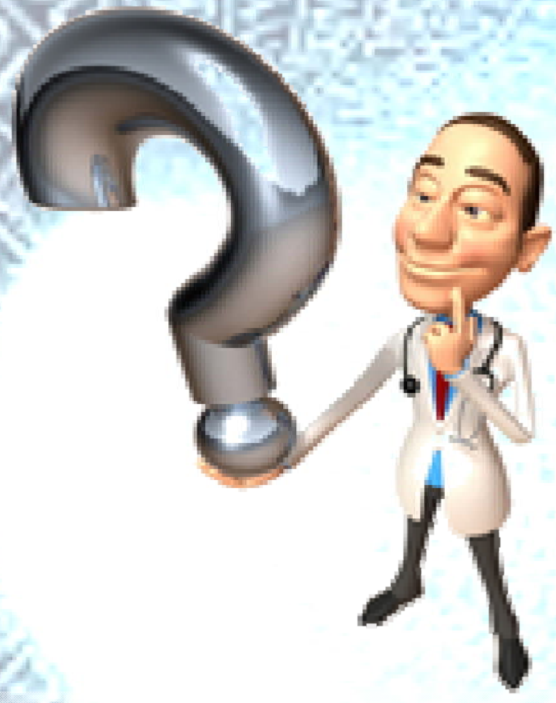
Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories.

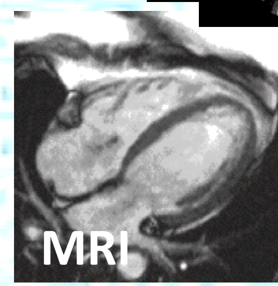
S. Lazebnik, C. Schmid and J. Ponce

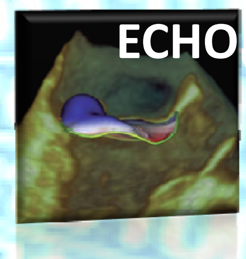
B I A 2 0 1 4

Carnegie Mellon University









55

Prahlad G Menon, PhD

www.justcallharry.com

+1 412-259-3031

pgmenon@andrew.cmu.edu