Carnegie Mellon

Prof. Marios Savvides

# Pattern Recognition Theory

**Lecture 6 : Principal Component Analysis (PCA) - I**

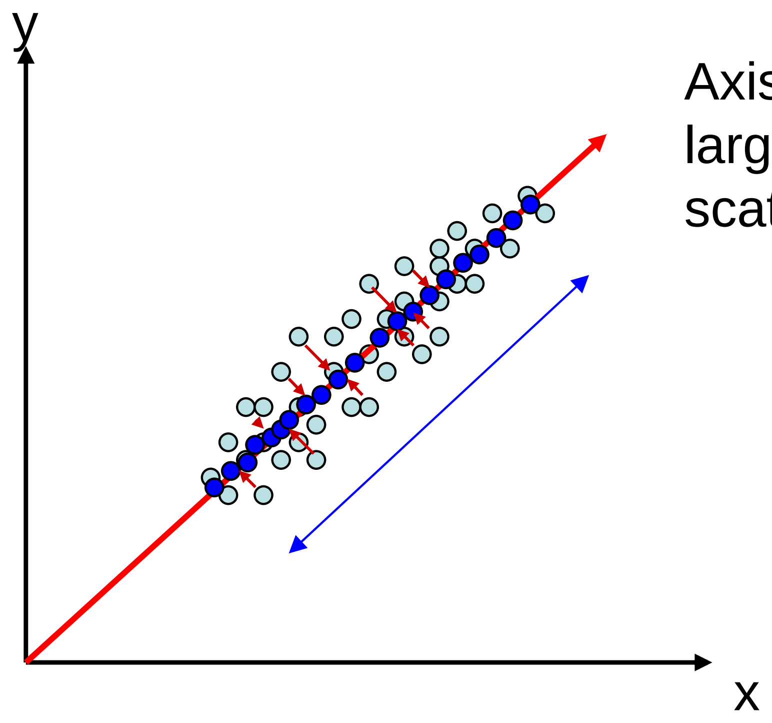
Electrical & Computer ENGINEERING

# PCA

- Also known as Hotelling Transform or Karhunen Loeve Transform
- Very popular pattern recognition tool…also known as “Eigenfaces” by Turk & Pentland
- Commonly used baseline benchmark when testing your new pattern recognition algorithm..i.e. given same training data and testing configuration, can I do better than PCA?
- So….you must DEFINITELY LEARN PCA!

## What is PCA?
## What are we trying to do?

- We want to find projections of data (i.e. direction vectors that we can project the data on to) that describe the maximum variation.



Axis that describes the largest variation (or scatter)

Marios Savvides

## PCA formulation

- We want projection vectors $\omega$ that when we project the data onto these directions we maximize the scatter or variance.

- i.e. we want to maximize

$$\mathrm{Var}\ (\omega^T x)$$

Subject to the constraint that $\omega$ is a direction vector of unit norm i.e. $\|\omega\|=1$

## How is PCA derived?

- Maximize the following objective function:

$$
\begin{aligned}
J(\omega) &= \mathrm{Var}(\omega^T x) \\
&= E(\omega^T x - \omega^T \mu)^2 \\
&= E(\omega^T x - \omega^T \mu)(x^T \omega - \mu^T \omega) \\
&= \omega^T E(x - \mu)(x - \mu)^T \omega \\
&= \omega^T \Sigma \omega
\end{aligned}
$$

Where covariance matrix $\Sigma = E(x - \mu)(x - \mu)^T$

# Derivation

- We want to maximize $J(\omega)$ subject to the projection vectors $\omega$ be unit norm i.e.

- Maximize quadratic term $\omega^T \Sigma \omega$ subject to $\|\omega\|=1$.

- Solution: Form Lagrangian optimization to take care of constraints, take derivative and set to zero to find $\omega$ vectors.

## Derivation

Solve: $$L(\omega,\lambda) = \omega^{\mathbf{T}}\Sigma\omega - \lambda(\omega^{\mathbf{T}}\omega - 1)$$

Take derivative w.r.t $\omega$, set it to zero

$$\frac{\partial L(\omega,\lambda)}{\partial \omega} = 2\Sigma\omega - 2\lambda\omega = 0$$

$$\Sigma\omega = \lambda\omega$$

Standard Eigenvalue/Eigenvector problem (Ax= $\lambda$x). i.e.

the vectors $\omega$ are the eigenvectors of the covariance matrix $\Sigma$.

## PCA

- Ok..do eigenanalysis…find eigenvectors $\omega_i$ and corresponding eigenvalues $\lambda_i$. Which to use though? All of them?
- A *d*x*d* covariance matrix contains max *d* eigenvector/eigenvalue pairs. Do we need to compute all of them?
- Which $\omega_i$ vector and eigenvalue $\lambda_I$ pairs to use?

# PCA

- Lets re-visit our original objective of PCA….
- We want to find projections which describe the biggest variance (or have the maximum scatter of data).
- Remember the variance of projected data is

$$\omega^T \Sigma\, \omega. \quad (1)$$

- And our solution yielded $$\Sigma\omega = \lambda\omega \qquad (2)$$
- Plug (2) in (1) and we get

$$\begin{aligned}\text{projected variance} &= \omega^T \Sigma\, \omega = \omega^T \lambda\omega \\ &= \lambda\, \omega^T \omega \qquad \text{(remember } \|\omega\|=1\text{)} \\ &= \lambda\end{aligned}$$

# PCA

- So this means that the direction vector $\omega_l$ has captures $\lambda_l$ variance.
- So we want to first represent the data using projections with largest variance, so order and keep the $\omega_l$'s with largest $\lambda_{max}$.
- In fact the *d*x*d* covariance matrix will not be full rank. Assume you N training images of size MxM=*d,* then you have AT MOST N-1 non-zero eigenvalues!

## Properties of PCA vectors

- Remember we computed eigenvectors $\omega$ from covariance matrix $\Sigma$.
- IMPORTANT NOTE: $\Sigma$ matrix is symmetric, i.e. $\Sigma = \Sigma^T$ is how you test for symmetry.

Proof:

$$\Sigma = E(x - \mu)(x - \mu)^{T} = [\, E(x - \mu)(x - \mu)^{T}]^{T}$$

$$= E(x - \mu)(x - \mu)^{T} = \Sigma^{T}$$

## So what?

- Symmetric Matrix, has ORTHOGONAL eigenvectors!
- What does this mean?

$\omega_i^T \omega_k = 0$ for all $i \neq k$

$\omega_i^T \omega_k = 1$ for all $i = k$ ($\omega^T \omega = 1$)

(i.e. the inner-product or dot-product between different eigenvectors is 0.) They are un-correlated, i.e. information about how data varies in the direction of one of the eigenvectors tells you nothing about how data varies in the directions of the eigenvectors.

- This also means we have an ORTHOGONAL basis that we can use to represent our data!

# Expanding a signal using a basis

From this slide on: we refer to principal direction vectors $\omega$ as eigenvectors v

- Assume you have a discrete vector signal x.
- You have a set of N basis vectors $\mathbf{v}_i$ which can use to represent a signal as follows:

$$\mathbf{x} = \sum_{i=1}^{N} p_i \mathbf{v}_i = p_1 \begin{bmatrix} | \\ \mathbf{v}_1 \\ | \end{bmatrix} + p_2 \begin{bmatrix} | \\ \mathbf{v}_2 \\ | \end{bmatrix} + .. + p_n \begin{bmatrix} | \\ \mathbf{v}_n \\ | \end{bmatrix} = \mathbf{Vp}$$

i.e. the signal is a linear combination of the basis vectors where the $p_i$ scalars are weight coefficients.

$$\mathbf{V} = \begin{bmatrix} | & | & | & | \\ \mathbf{v_1} & \mathbf{v}_2 & \mathbf{v}_3 \dots & \mathbf{v}_N \\ | & | & | & | \end{bmatrix}$$

$$\mathbf{p} = \begin{bmatrix} p_1 \\ p_2 \\ | \\ p_N \end{bmatrix}$$

# Computing the weight coefficients

$$\mathbf{x} = \sum_{i=1}^{N} p_i \mathbf{v}_i = p_1 \begin{bmatrix} | \\ \mathbf{v}_1 \\ | \end{bmatrix} + p_2 \begin{bmatrix} | \\ \mathbf{v}_2 \\ | \end{bmatrix} + .. + p_n \begin{bmatrix} | \\ \mathbf{v}_n \\ | \end{bmatrix} = \mathbf{V}\mathbf{p}$$

- Since this is an orthogonal basis, we can easily find the coefficients p. These are just the projections of the signal on to each basis.

- i.e. $p_1 = \mathbf{x}^{\mathrm{T}} \mathbf{v}_1$ or $\mathbf{p} = \mathbf{V}^{\mathrm{T}} \mathbf{x}$

$$\mathbf{V}^{\mathrm{T}}\mathbf{x} = \begin{bmatrix} - & \mathbf{v}_1 & - \\ - & \mathbf{v}_2 & - \\ - & \mathbf{v}_3 & - \end{bmatrix} \begin{bmatrix} | \\ \mathbf{x} \\ | \end{bmatrix} = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix}$$

# In PCA we model the variations about the mean.

- Don’t forget that you model the variance about the mean!
- i.e. Don’t forget to subtract the global mean before you analyze your data.

$$\mathbf{p} = \mathbf{V}^{\mathbf{T}}(\mathbf{x} - \mathbf{m})$$

- By Mean we mean the sample mean image/vector of all your training data samples.
- Don’t forget to add the mean back before you reconstruct the data!

$$\mathbf{x} = \sum_{i=1}^{N} p_i \mathbf{v}_i + \mathbf{m} = p_1 \begin{bmatrix} | \\ \mathbf{v}_1 \\ | \end{bmatrix} + p_2 \begin{bmatrix} | \\ \mathbf{v}_2 \\ | \end{bmatrix} + .. + p_n \begin{bmatrix} | \\ \mathbf{v}_n \\ | \end{bmatrix} + \mathbf{m} = \mathbf{V}\mathbf{p} + \mathbf{m}$$

Marios Savvides

## PCA – Properties of

- These projection vectors form a basis for describing the training data in the Minimum Squared Error (MSE) sense.
- I.e. the eigenvectors with most largest eigenvectors capture most of the signal energy and reconstruction ability.
- N samples of data will have a covariance matrix of rank at most N-1. That means you have at MOST N-1 NON-ZERO eigenvalues. And you ONLY care about these corresponding N-1 eigenvectors.

## How do I compute these in Matlab?

- Use MATLAB command “eigs” and “eig”
- “Eigs” is nice because it can find only a few eigenvectors,
- [v,d]=eigs(Σ,N) which finds the N eigenvectors with the largest N eigenvalues.

## Practical Issues in computing PCA

- Assume we have image data of size 200x200 pixels.
- This means that our data is 40,000 dimensions!
- What is the size of Covariance Matrix?

$$\Sigma = E(x - \mu)(x - \mu)^T \Rightarrow 40{,}000 \times 40{,}000$$

- This is a BIG MATRIX!..infact you CAN NOT create this matrix in MATLAB, no matter how much memory you have!

## Also…

- Well we only have N training samples, where N<<d (d=dimensionality of our problem=40,000).
- Since N is much smaller than d and we will have AT MOST N-1 eigenvectors (assuming N-1 linearly independent data samples).
- It’s not computationally feasible or possible to work with large covariance matrices.

## Instead…I show you the Gram Matrix Trick!

- We know that $\Sigma = E(x - \mu)(x - \mu)^T = XX^T$

  Must solve $\Sigma v = \lambda v$

  $XX^T v = \lambda v$ (pre-mult by $X^T$) (1)

  $X^T XX^T v = \lambda X^T v$ ($v' = X^T v$) (2)

  Solve eigenvalue problem $X^T X v' = \lambda v'$

$X^T X$ is a gram or inner-product matrix, its size is not dependent on dimensionality of data but rather on the number of data samples N. It is of size NxN and hence easier to compute if N<<d.

## Ok..but how do I compute v from v' ?

- Remember Eq(1) and Eq (2):

$$X X^T v = \lambda v \quad (1)$$

$$v' = X^T v \quad (2)$$

Recognizing and subst (2) in (1) we get

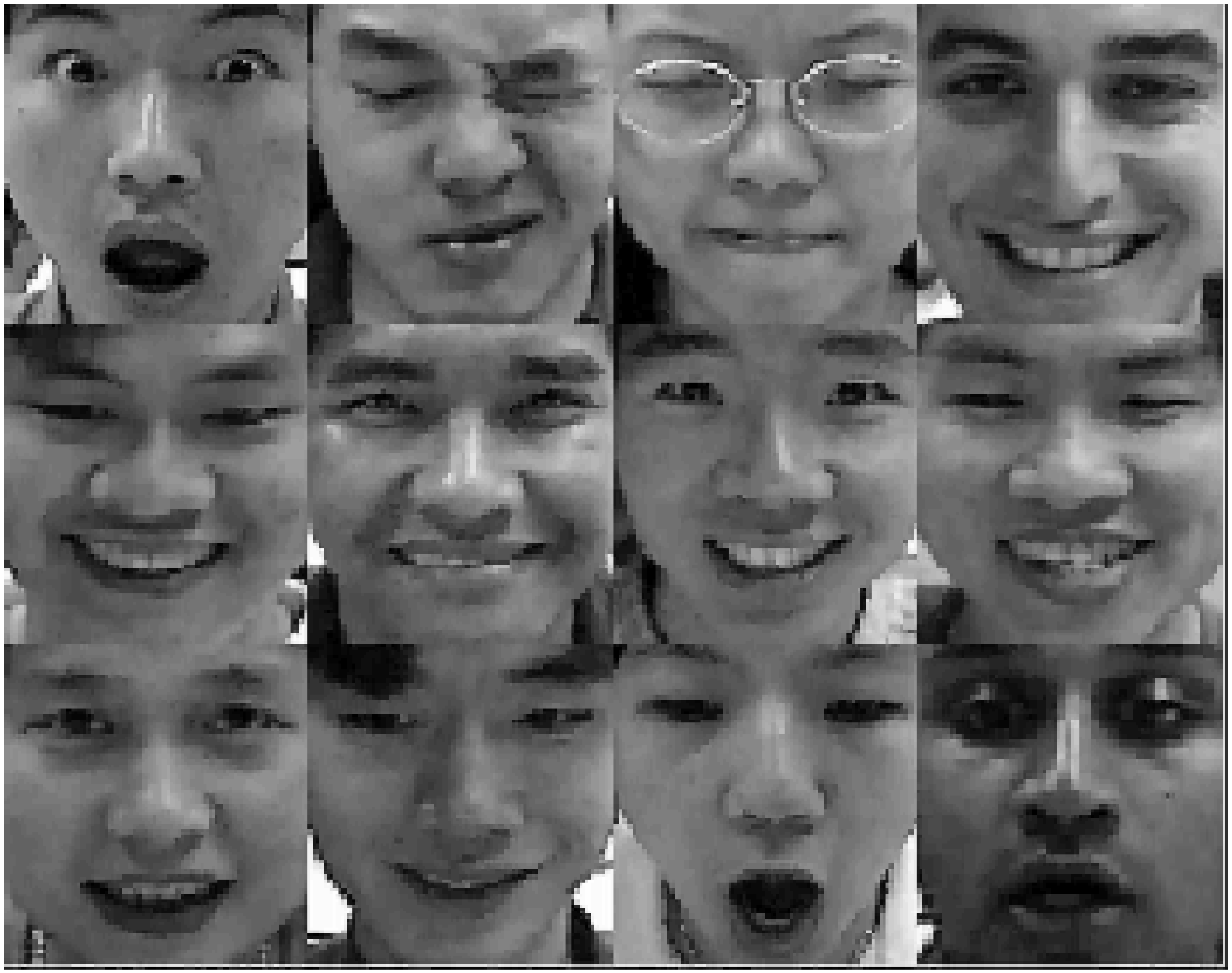
$$X v' = \lambda v$$

Thus v=Xv'. We don't care about scaling term because we will anyway make eigenvector orthonormal i.e. $||v||=1$.

## Lets see an example on real face data.

- We use CMU’s AMP Lab facial Expression database.
- 13 people.
- Images are 64x64 cropped and centered facial images.
- Variations are due to varying expressions in the video sequence.
- 75 images in each person’s video sequence

# AMP Lab Facial Expression DB

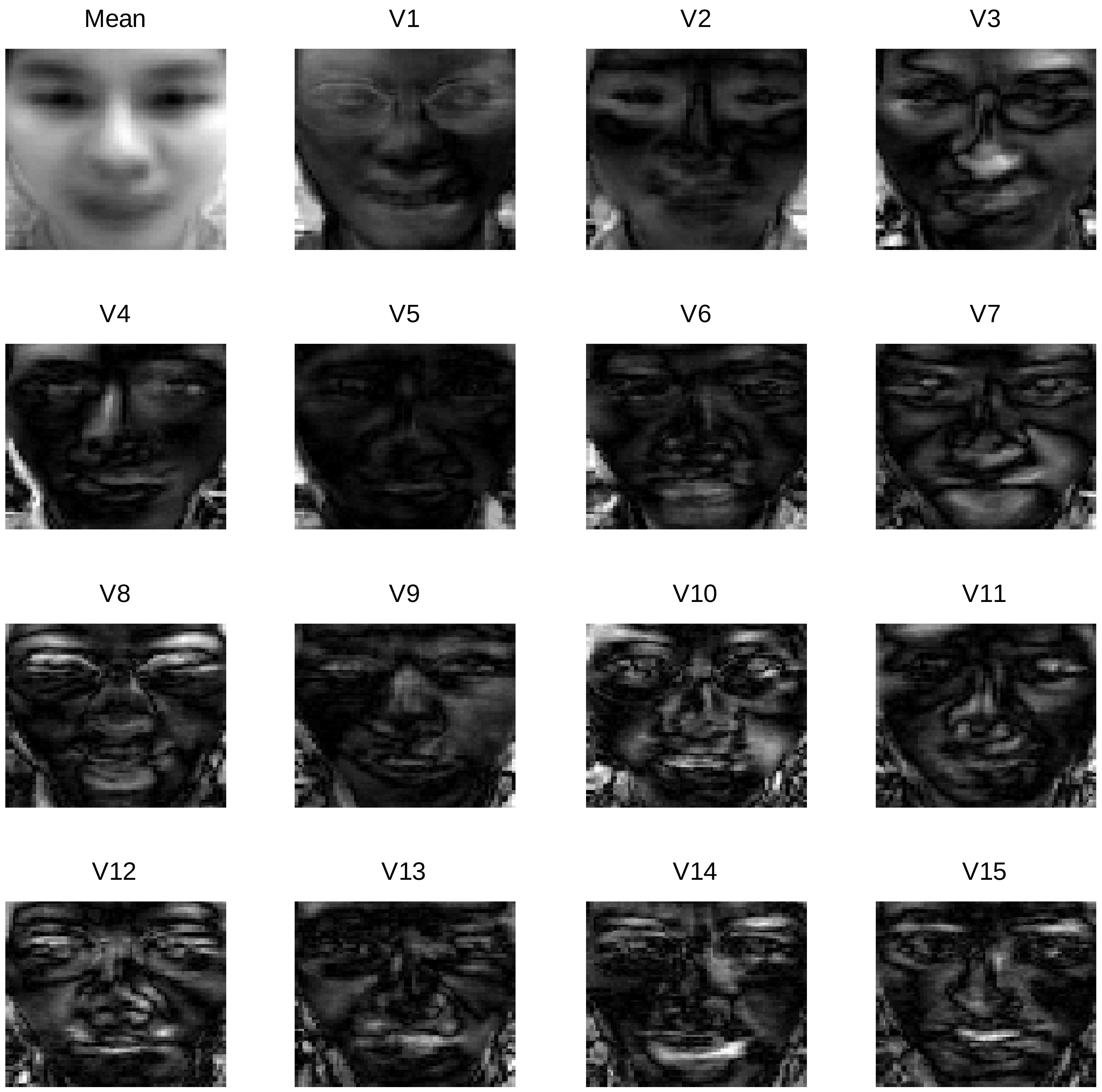


Mario Savvides

## Experiment:

- Take the first 5 images per person and use them as training data
- Total of 5x13=65 training images
- Do PCA and compute basis eigenvectors.
- Measure Reconstruction ability
- Perform Dimensionality Reduction (to 3d, i.e. only use a few eigenvectors and look at how data clusters in this 3D linear subspace.

# What do the eigenvectors look like?



Marios Savvides

# How much reconstruction?

- If you want to say reconstruct an image with only 20% MSE (or 80% reconstruction)
- Then use the eigenvectors with largest M eigenvalues from the total of N non-zero eigenvalues satisfying this ratio:

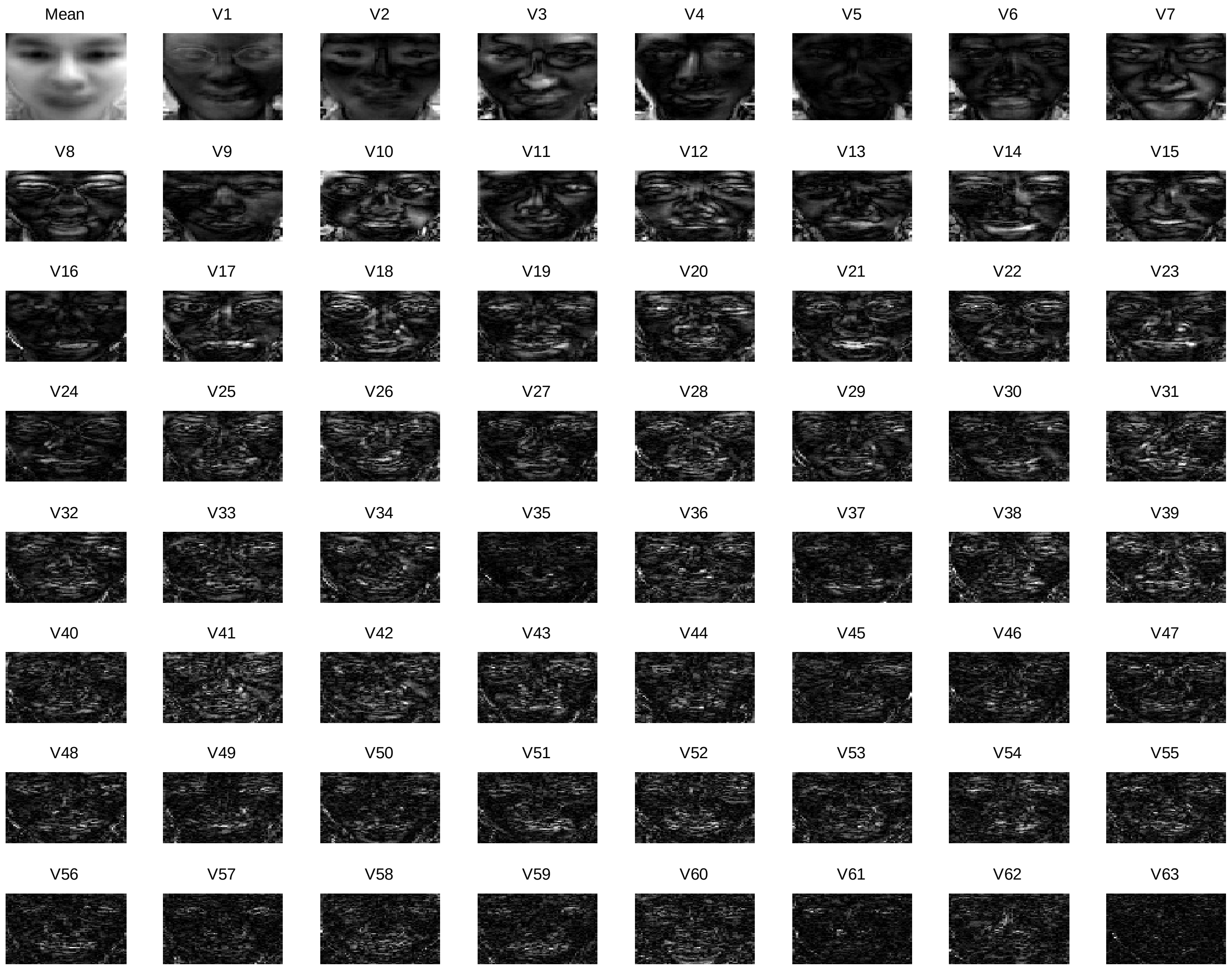
$$reconstruction\% = \frac{\displaystyle\sum_{i=1}^{M} \lambda_i}{\displaystyle\sum_{i=1}^{N} \lambda_i} * 100$$

Sum of the eigenvalues of the kept eigenvectors

Sum of all eigenvalues

Marios Savvides

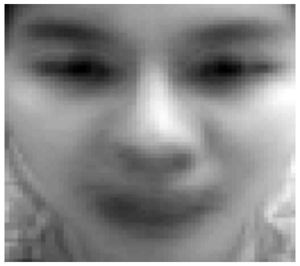
# All 64 eigenvectors..do we need all?



Noisy..not much info..will not contribute to image reconstruction

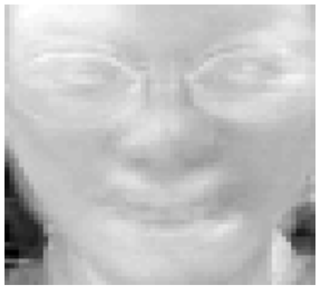
## See how it works…(use only the 3 most dominant eigenvectors)

Mean



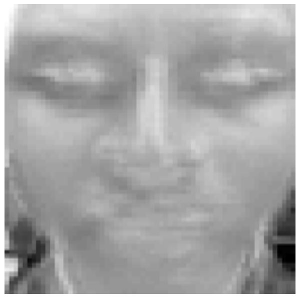
+ 230

v1



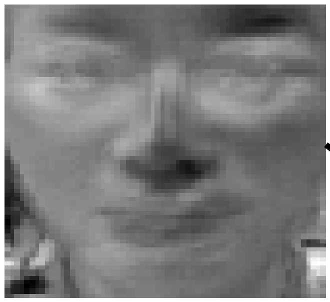
- 917

v2



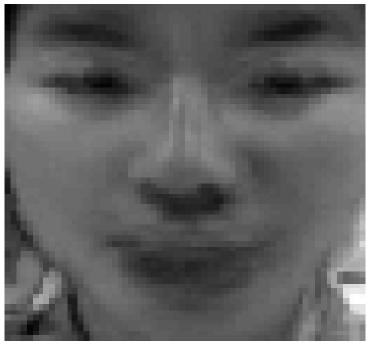
+ 1050

v3

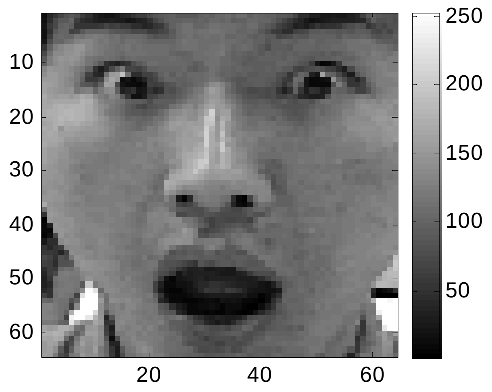


=

MSE=758.13



reconstructed $\tilde{\mathbf{x}}$

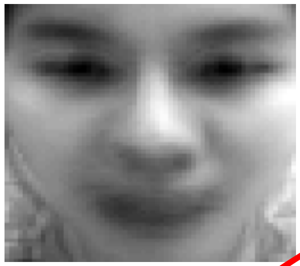


Original $\mathbf{x}$

Marios Savvides

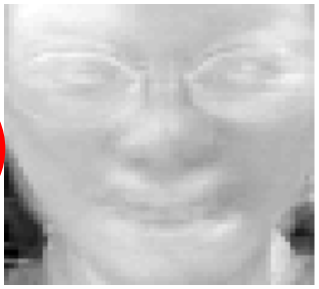
# How do we compute projection coefficients p?

Mean (m)



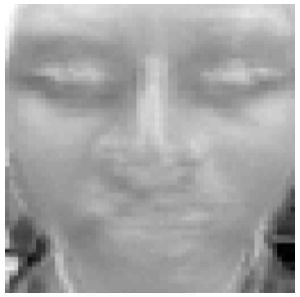
+ 230

v1



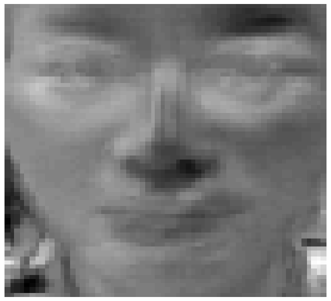
- 917

v2



+ 1050

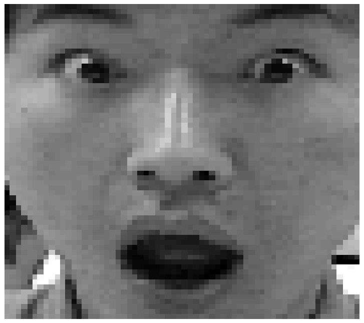
v3



## Step 1:

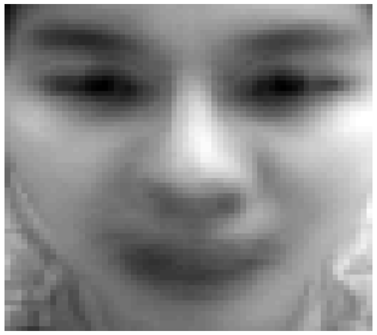
**Subtract mean image**

**x**



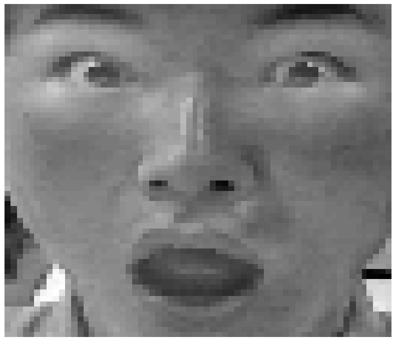
-

**m**



=

**c**



## Step 2:

Project onto eigenvector

$$\sum_{x=1}^{X}\sum_{y=1}^{Y}$$

.*

(element multip

= 230

# Projections in vector form

Step 2:

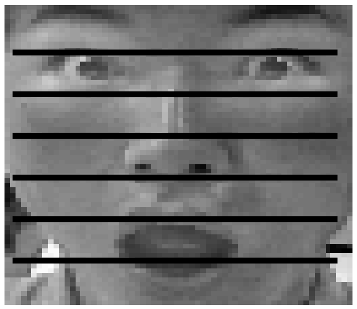
Project onto eigenvector

$$\sum_{x=1}^{X}\sum_{y=1}^{Y}$$

$.*$

(element multip

$= \boxed{230}$

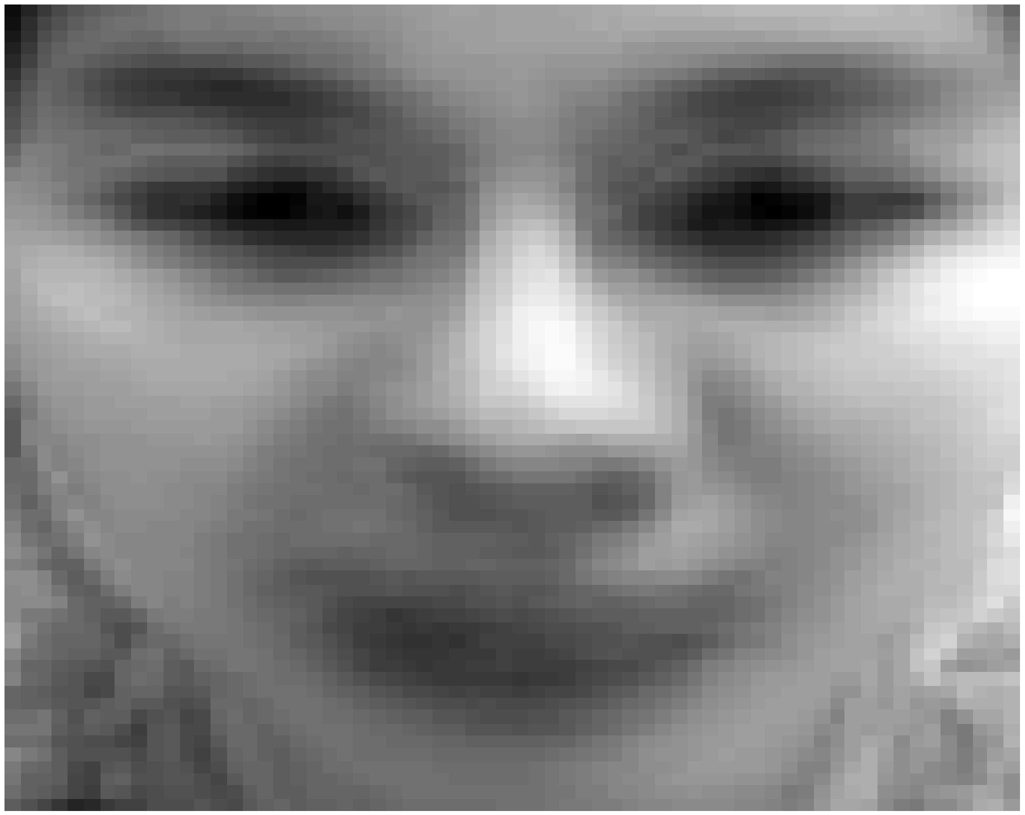


$\Rightarrow \begin{bmatrix} | \\ | \\ | \\ | \\ | \end{bmatrix} = \mathbf{x}$

$\Rightarrow \begin{bmatrix} | \\ | \\ | \\ | \\ | \end{bmatrix} = \mathbf{v}_1$

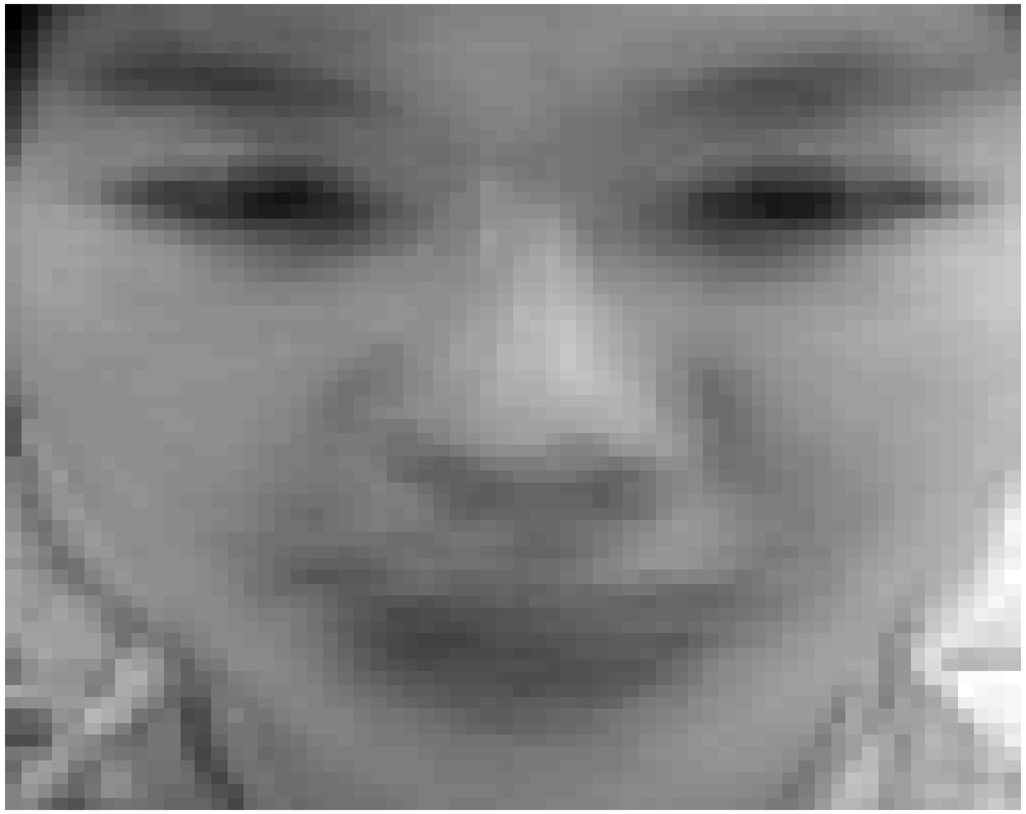
$$\mathbf{p}_1 = \mathbf{x}^{\mathbf{T}}\mathbf{v}_1 = 230$$

MSE=1233.16



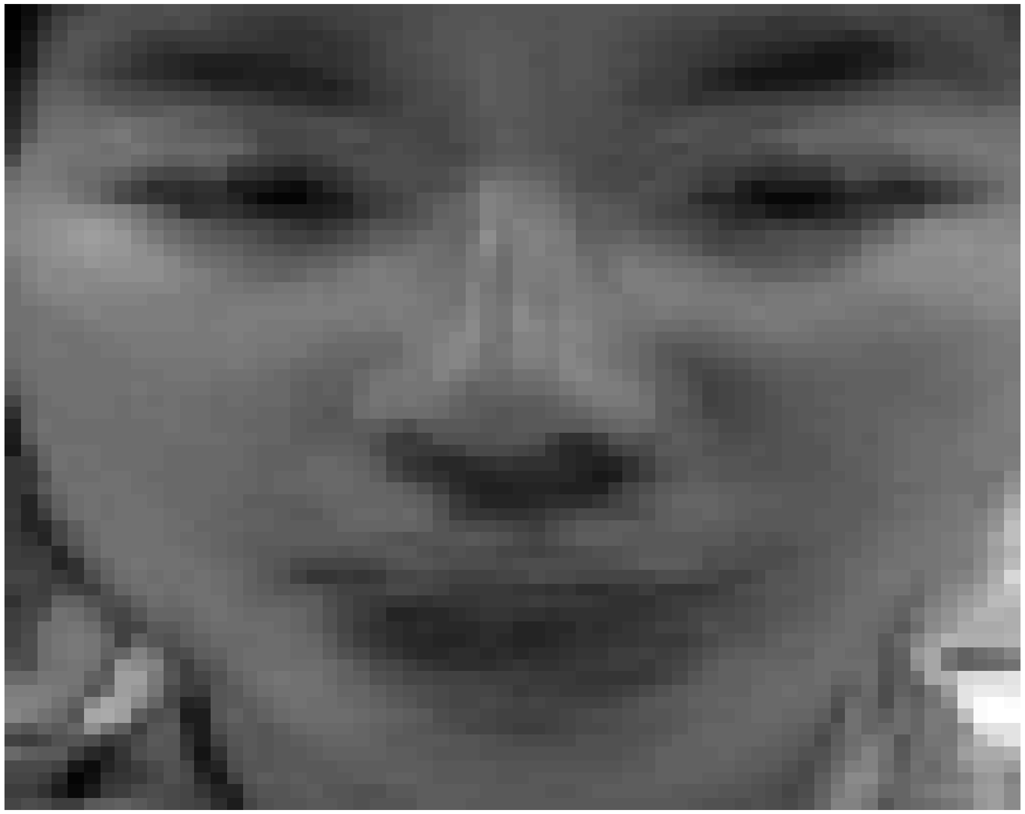
- If we use only 1 eigenvector, MSE=1233 (max Eigenvalue)

MSE=1027.63



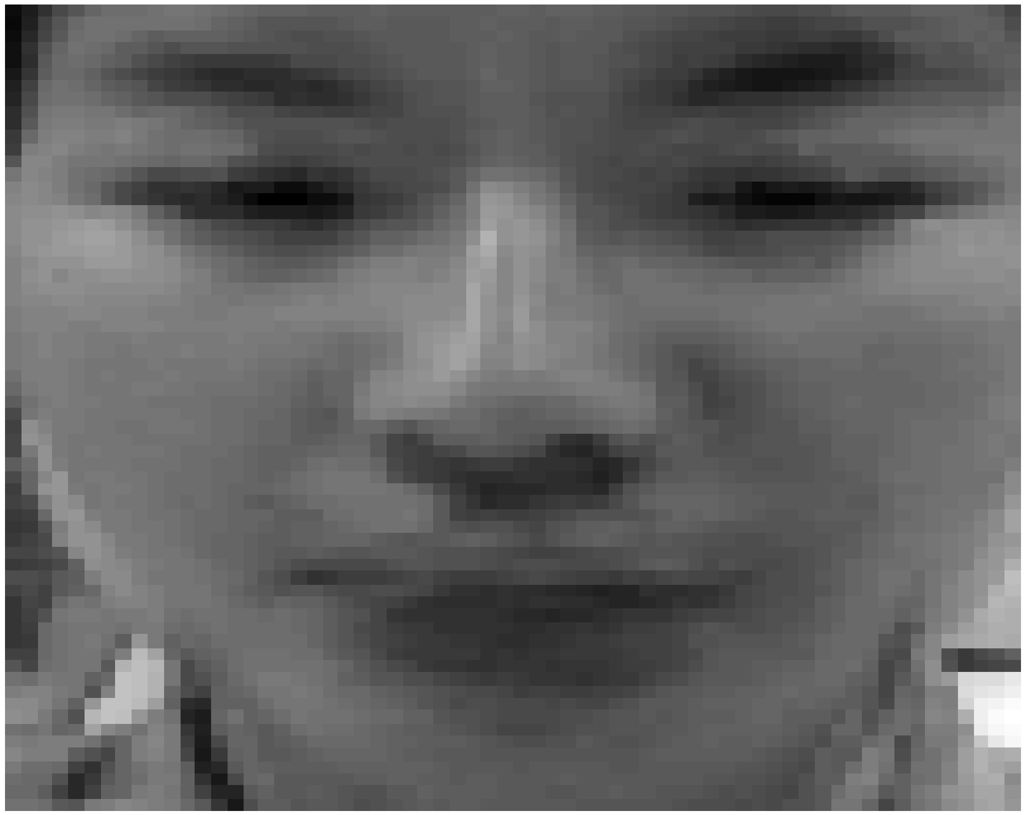
- If we use 2 eigenvectors, MSE=1027

MSE=758.13



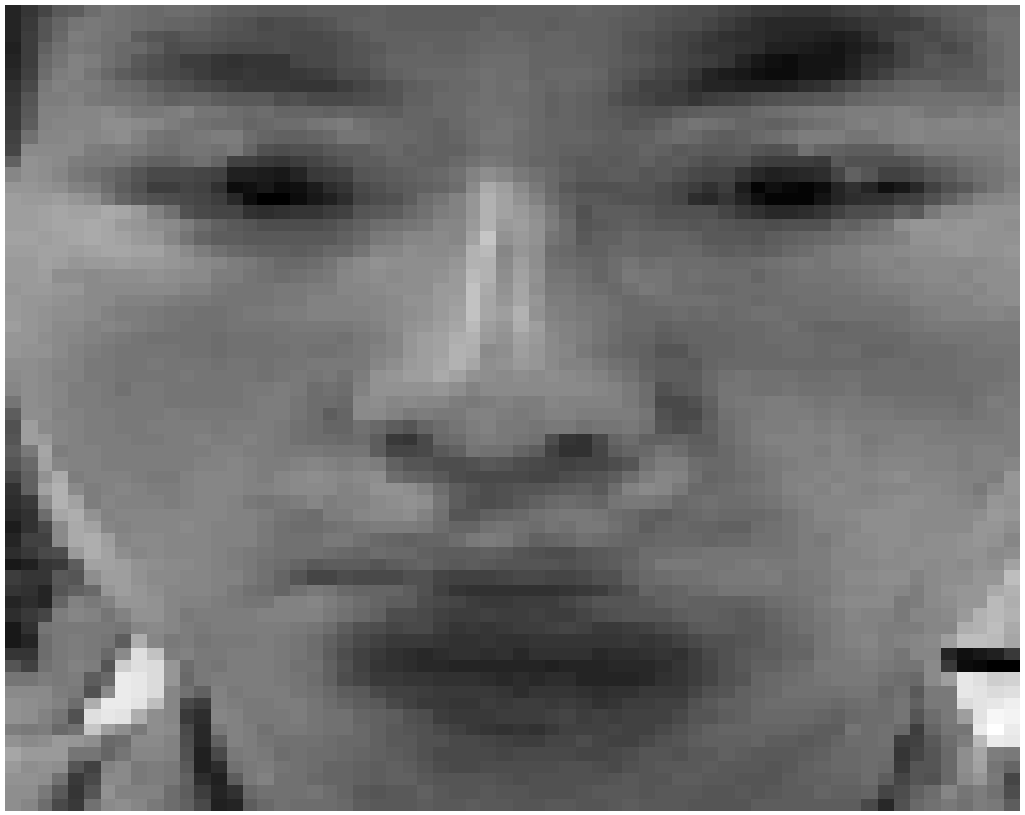
- If we use 3 eigenvectors, MSE=758

MSE=634.54



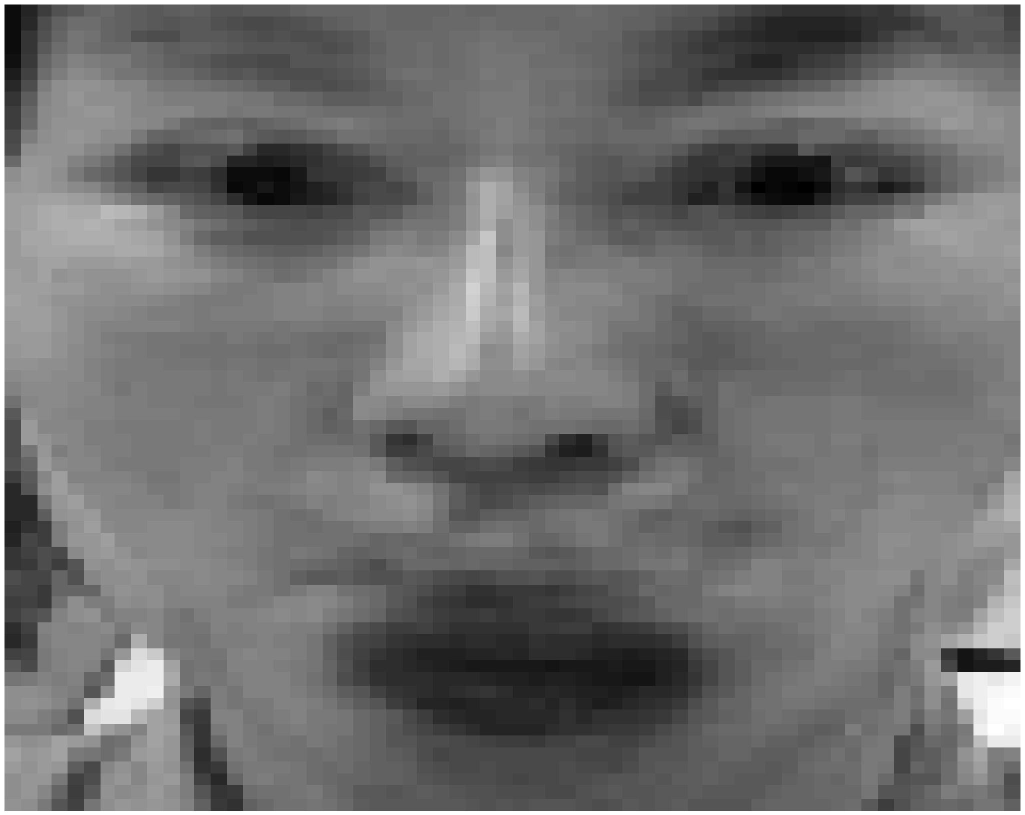
- If we use 4 eigenvectors, MSE=634

MSE=399.08



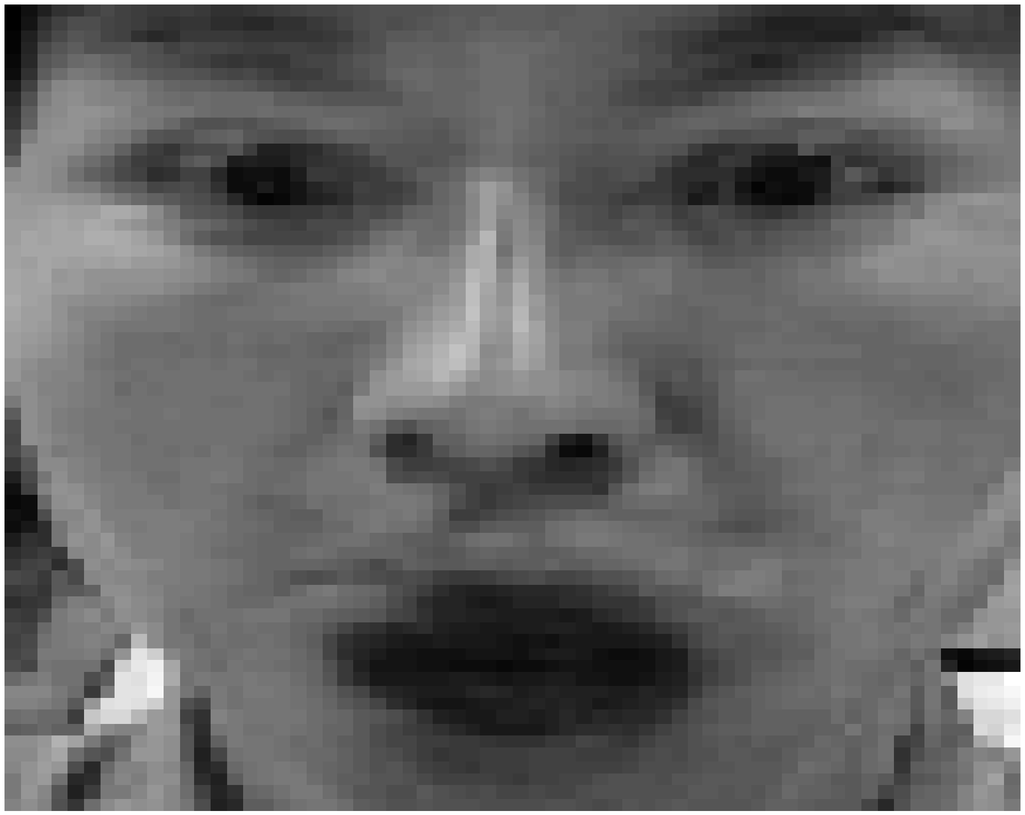
- If we use 7 eigenvectors, MSE=399

MSE=285.08

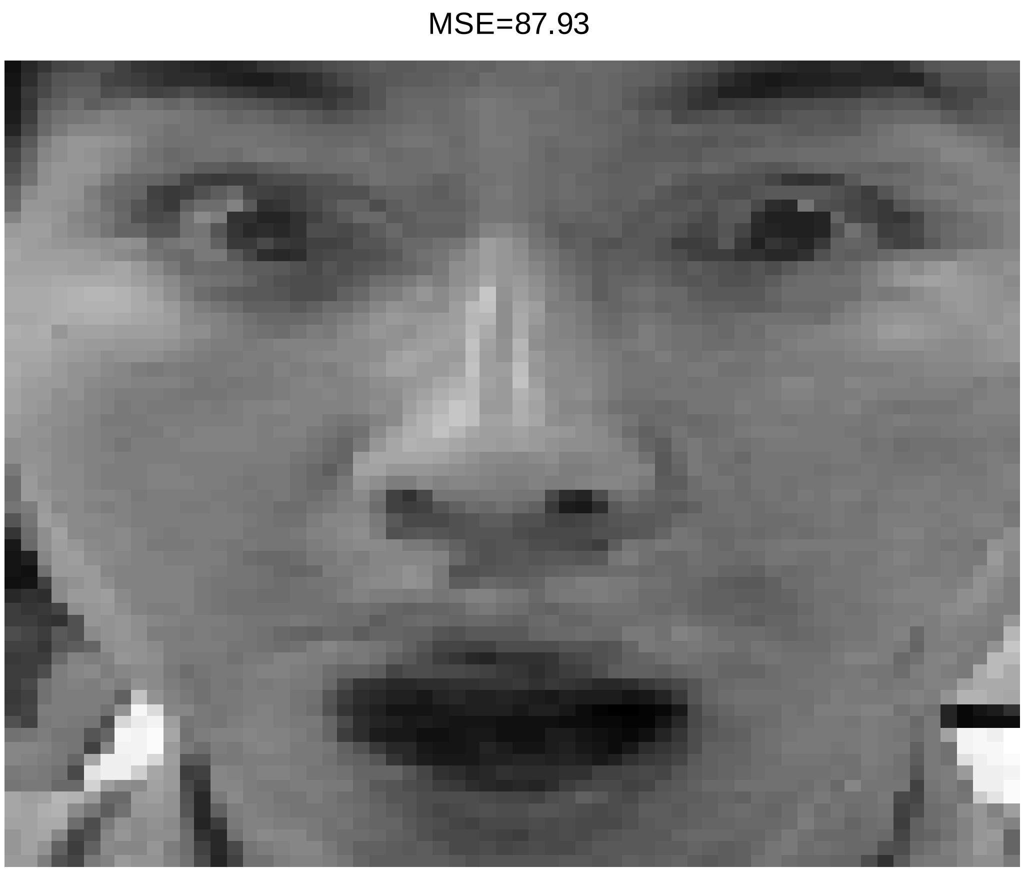


- If we use 8 eigenvectors, MSE=285

MSE=216.88

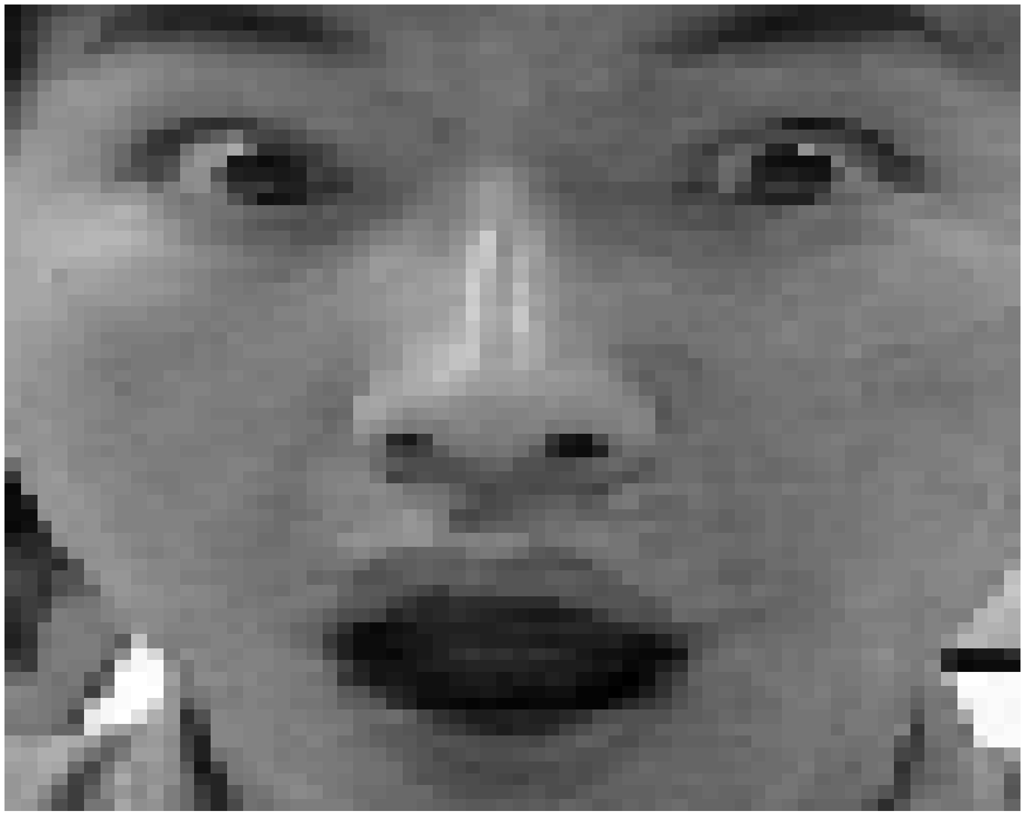


- If we use 13 eigenvectors, MSE=216



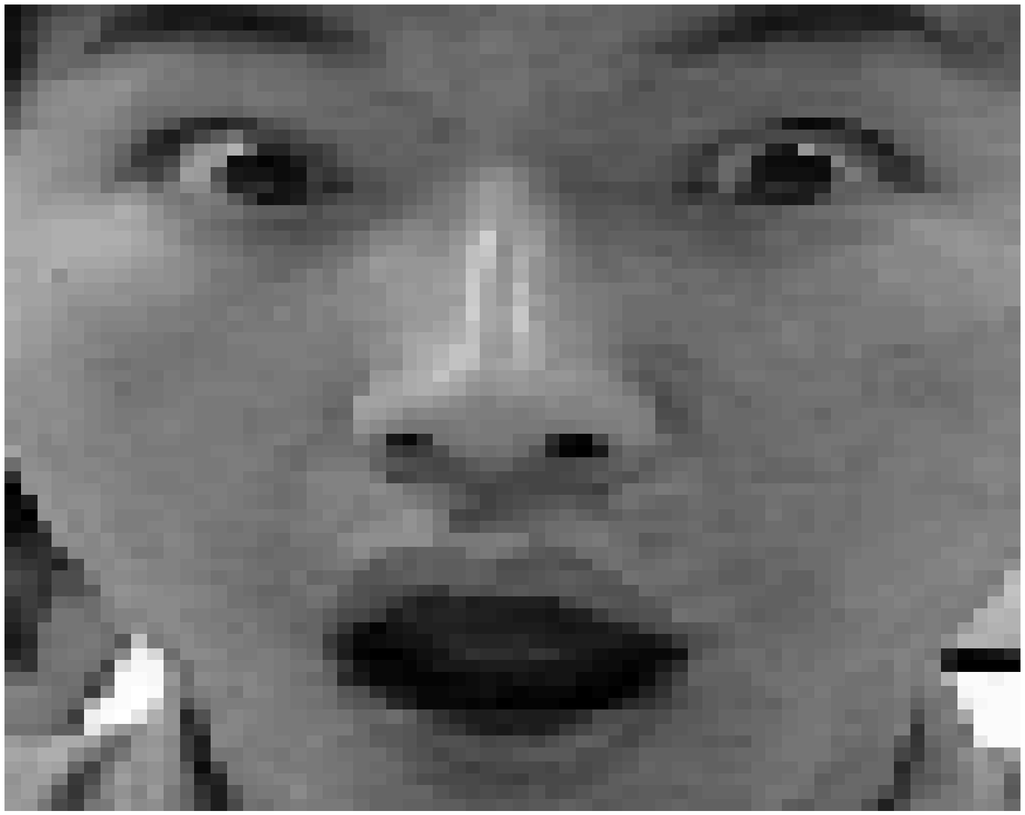
- If we use 20 eigenvectors, MSE=87

MSE=20.55



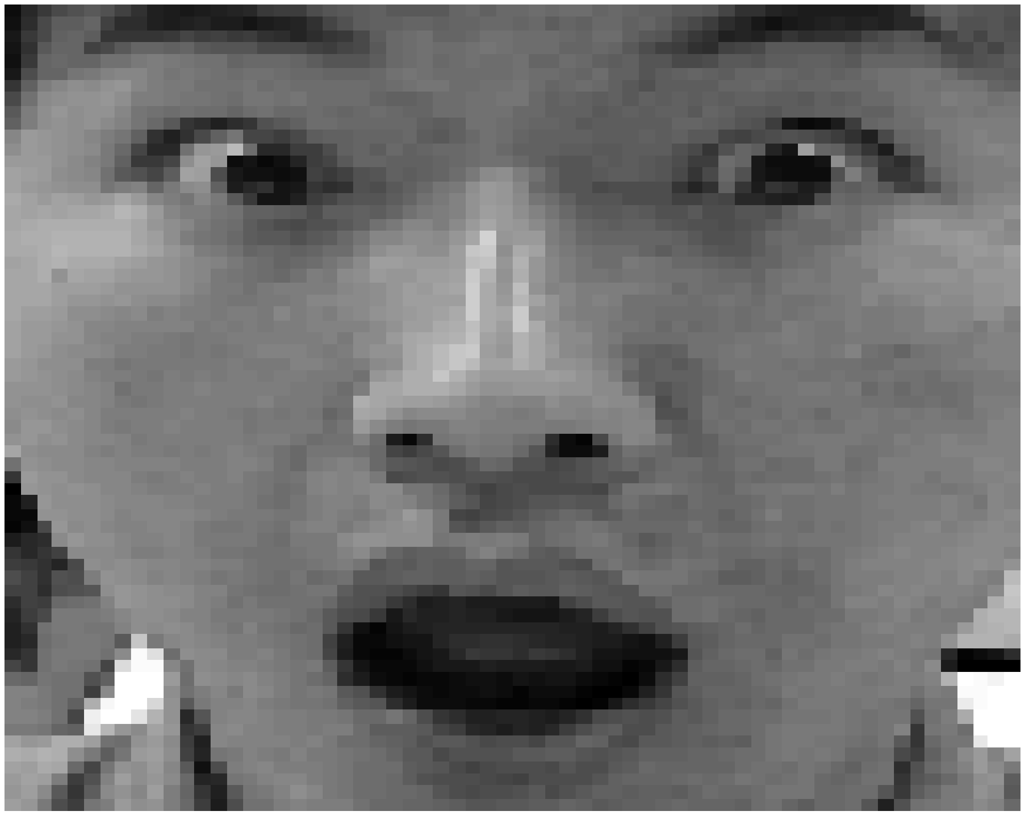
- If we use 30 eigenvectors, MSE=20

MSE=6.84

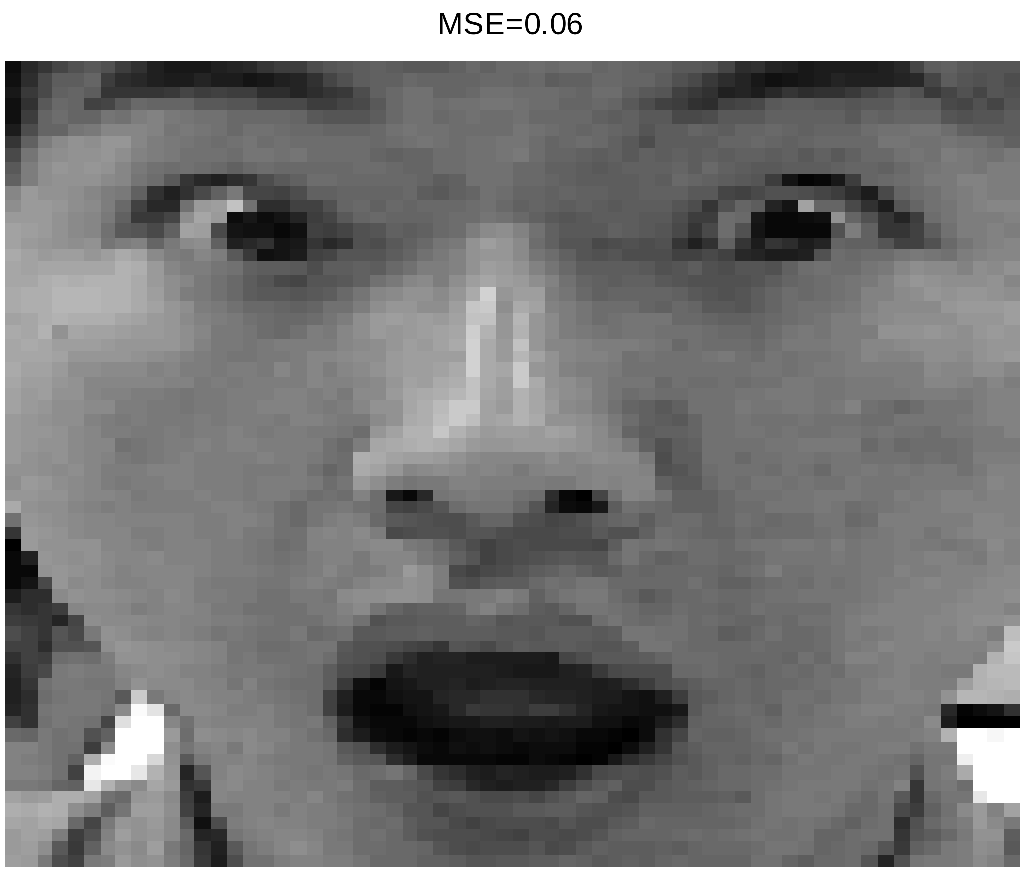


- If we use 45 eigenvectors, MSE=6.8

MSE=2.14

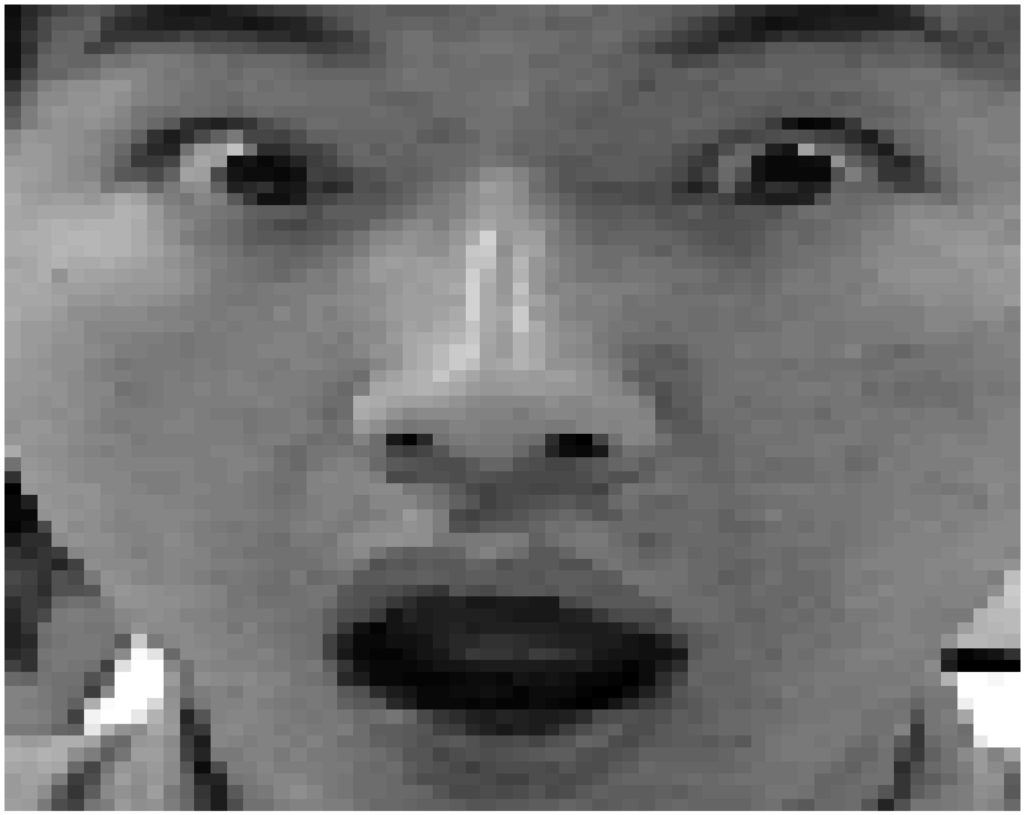


- If we use 50 eigenvectors, MSE=2.1



- If we use 60 eigenvectors, MSE=0.06

MSE=0.00



- If we use 64 eigenvectors, MSE=0

(NOTE: We are using ALL NON-ZERO EigenVALUE Eigenvectors!)

Marios Savvides

# Feature Extraction via Dimensionality Reduction

- Lets throw away many eigenvectors and only use a couple most dominant ones (this will perform dimensionality reduction)
- We project our data samples onto these vectors and store the projection coefficient for each projected data sample.

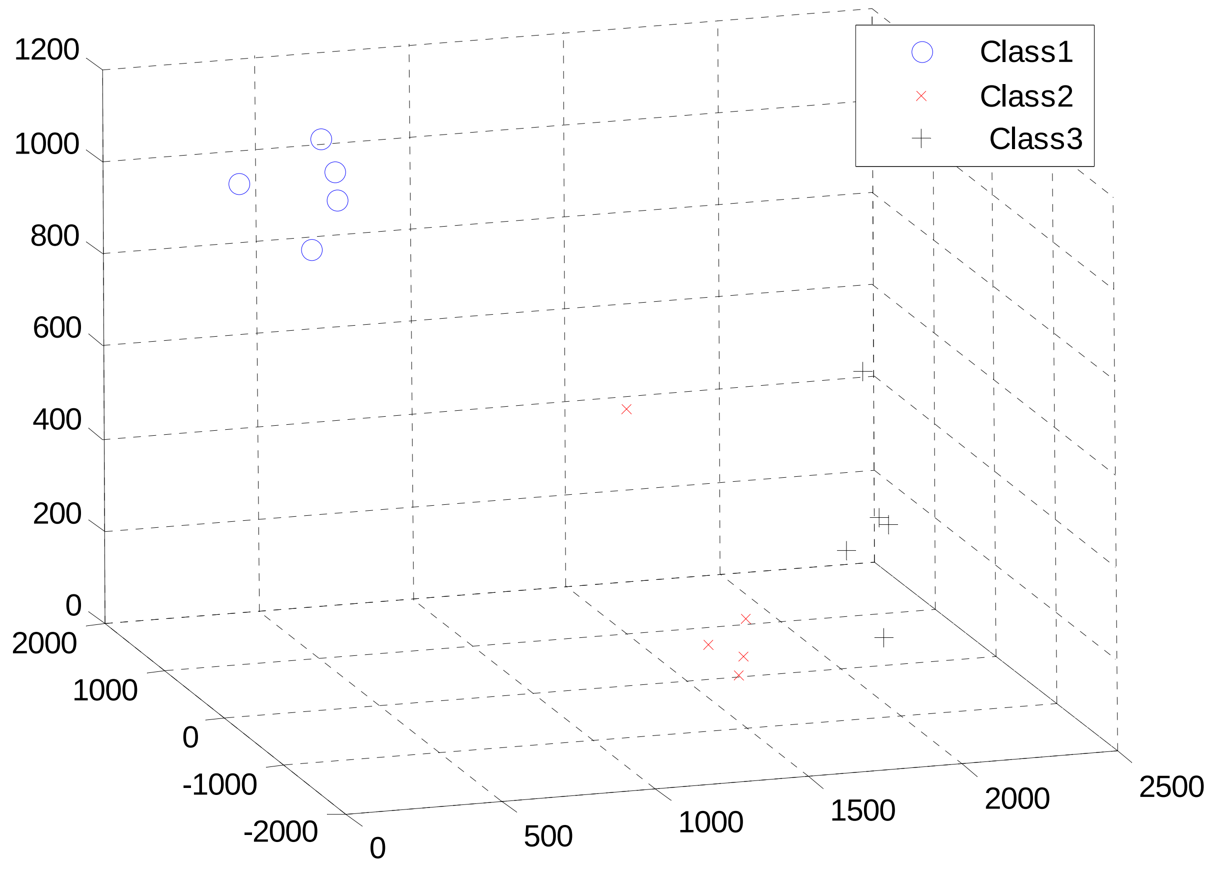
$$\mathbf{p} = \mathbf{V}^{\mathbf{T}}(\mathbf{x} - \mathbf{m})$$

- So if we have N classes with k images per class then we will have kxN projection vectors **p.** We only need to store these vectors and not the original training data to do classification.

## E.g.

- In our example we have 64x64 images, however in this toy example we performed PCA (global PCA since it uses all classes) and then only kept the 3 most dominant eigenvectors (i.e. with the 3 largest eigenvalues).
- Project each 64x64=4096 dimensional training image onto this basis to get a 3-dimensional vector. That 3d vector represents each training sample in a 3d subspace.

## The 3D linear subspace containing the projected training samples from class 1, 2 and 3

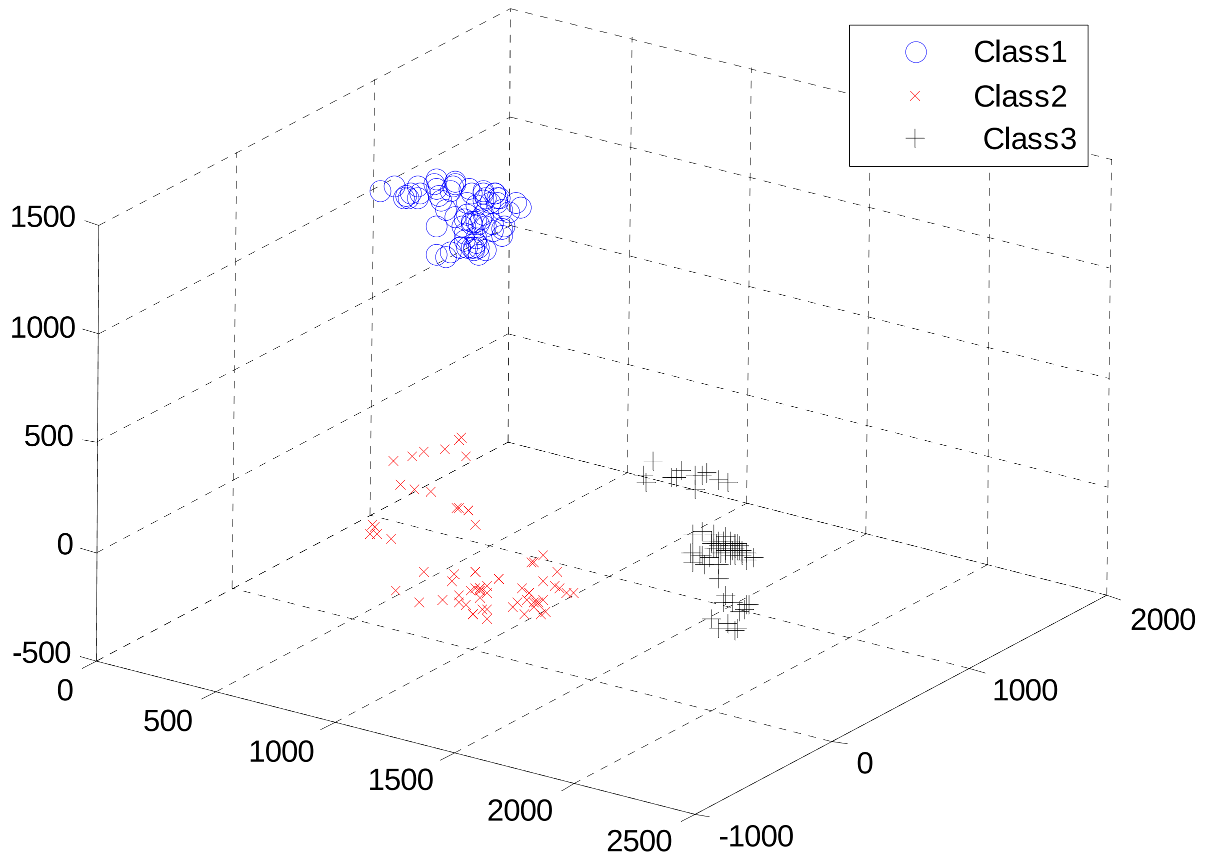


Marios Savvides

## How do we classify?

- Given a test face, we project onto this 3D linear subspace (the 3 eigenvectors) to get a 3d-vector.
- Then we can do a simple Nearest Neighbor search (euclidean or L2) distance to all the stored training vectors to find which is closest to and label the data.

# Test: We projected all 75 images per person in the 3D PCA subspace



- We can see the 3 data classes clusters.

Marios Savvides

## PCA Variants

- What I described here was “Global PCA”

i.e. PCA was performed on all classes to find a universal linear subspace.

- However we can also do Individual PCA (IPCA), i.e. build a subspace for each class using the images from that class.
  - How do we test?
  - We measure the reconstruction error between the test image and the reconstructed image from each subspace.
  - We label the test image with the subspace/basis that yielded the smallest reconstruction error.

## Pros & Cons?

- IPCA you need to have more data for each person (won’t work with just 1 image/person).
- However, you don’t need to re-train the system everytime you add a person in the database (as with Global PCA).