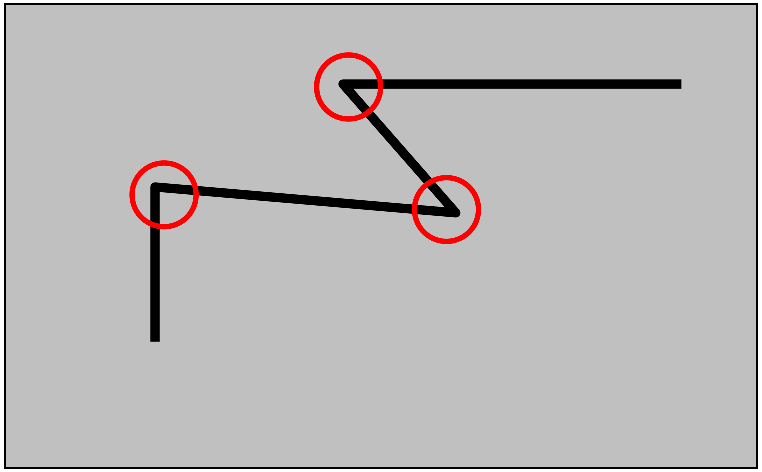
Harris Corner Detector

Slides taken from: "Matching with Invariant Features", Darya Frolova, Denis Simakov, The Weizmann Institute of Science, March 2004

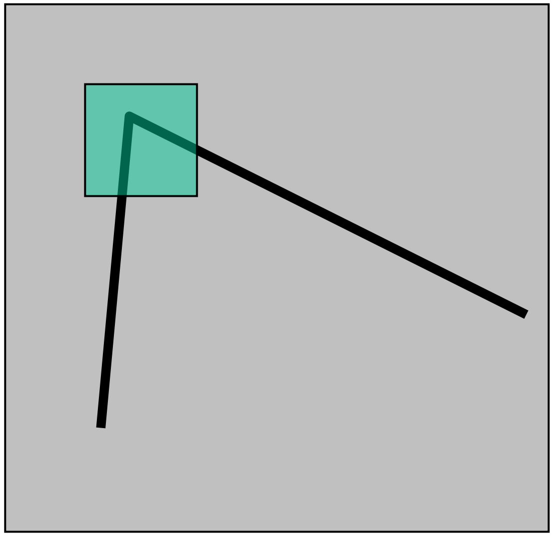
An introductory example:



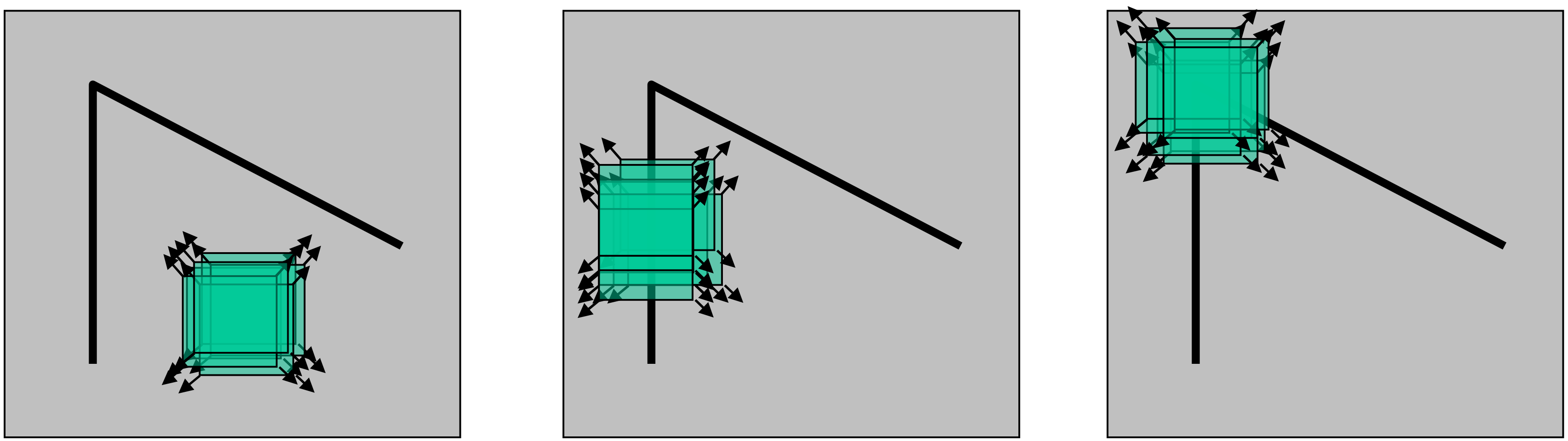
C.Harris, M.Stephens. “A Combined Corner and Edge Detector”. 1988

The Basic Idea

- We should easily recognize the point by looking through a small window
- Shifting a window in *any direction* should give *a large change* in intensity



Basic Idea (2)



"flat" region:
no change in
all directions

"edge":
no change along
the edge direction

"corner":
significant change
in all directions

Mathematics

Change of intensity for the shift $[u,v]$:

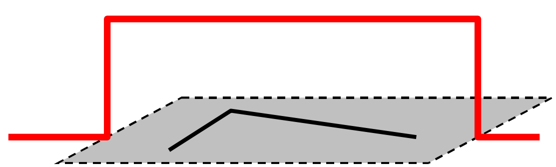
$$E(u,v)=\sum_{x,y} w(x,y)\left[I(x+u,y+v)-I(x,y)\right]^2$$

Window function

Shifted intensity

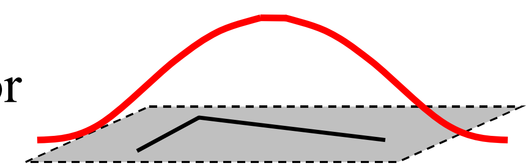
Intensity

Window function $w(x,y)$ =



1 in window, 0 outside

or



Gaussian

Mathematics

For small shifts $[u,v]$ we have a *bilinear* approximation:

$$E(u,v) \simeq \begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix}$$

where M is a 2×2 matrix computed from image derivatives:

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

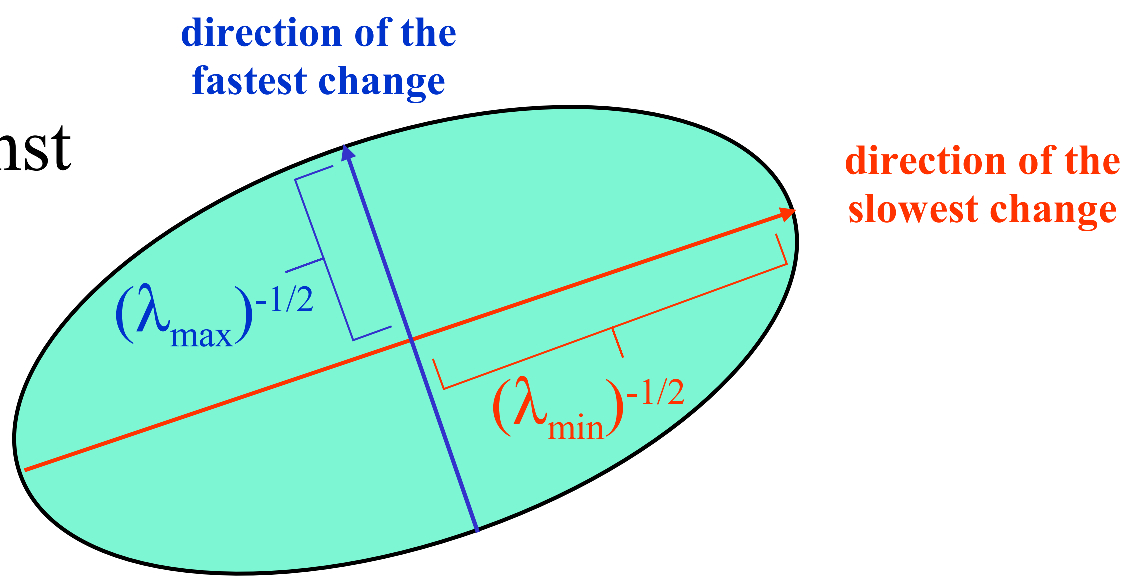
Mathematics

Intensity change in shifting window: eigenvalue analysis

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

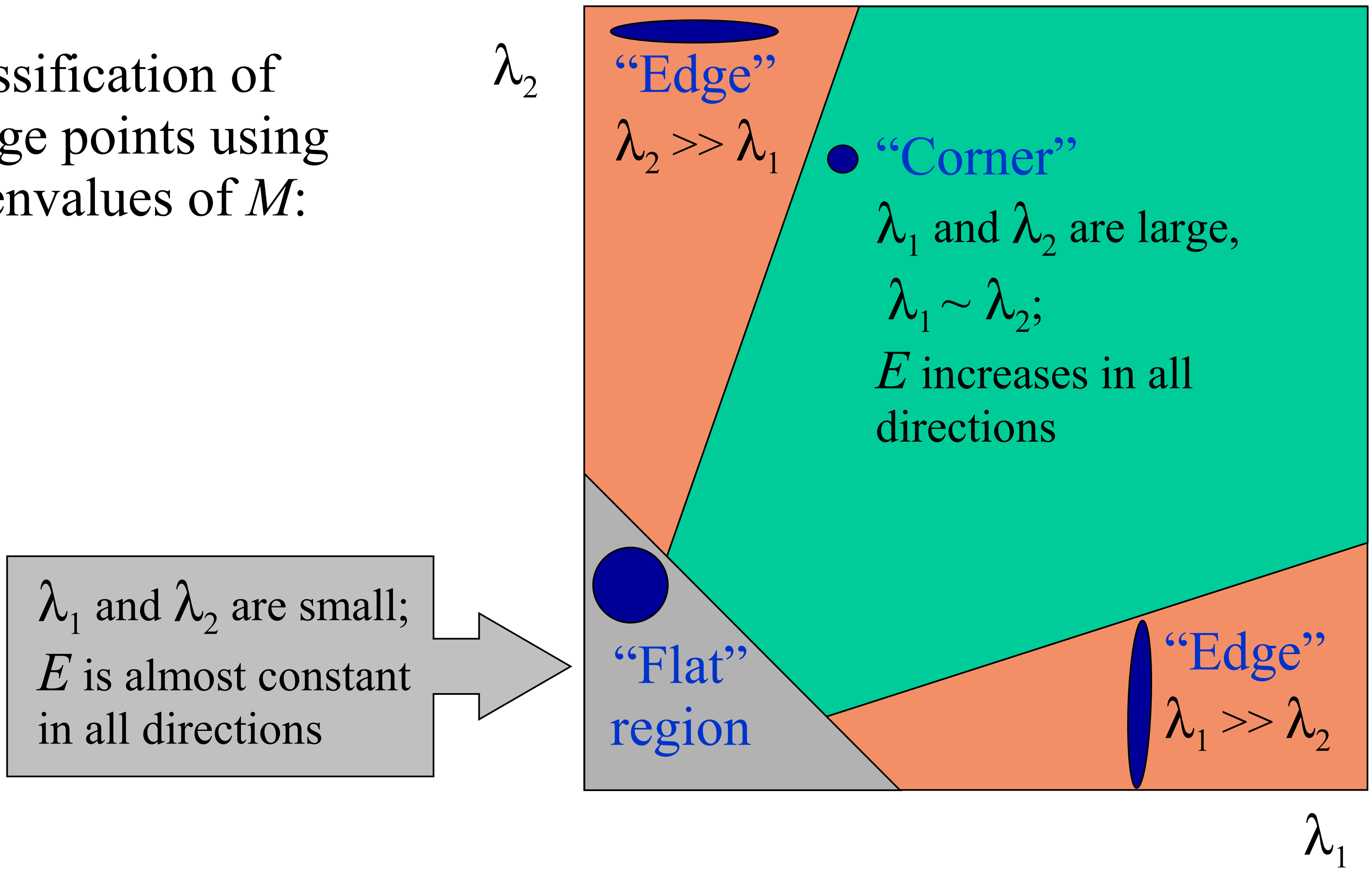
λ_1, λ_2 – eigenvalues of M

Ellipse $E(u,v)$ = const



Mathematics

Classification of image points using eigenvalues of M:



Mathematics

Measure of corner response:

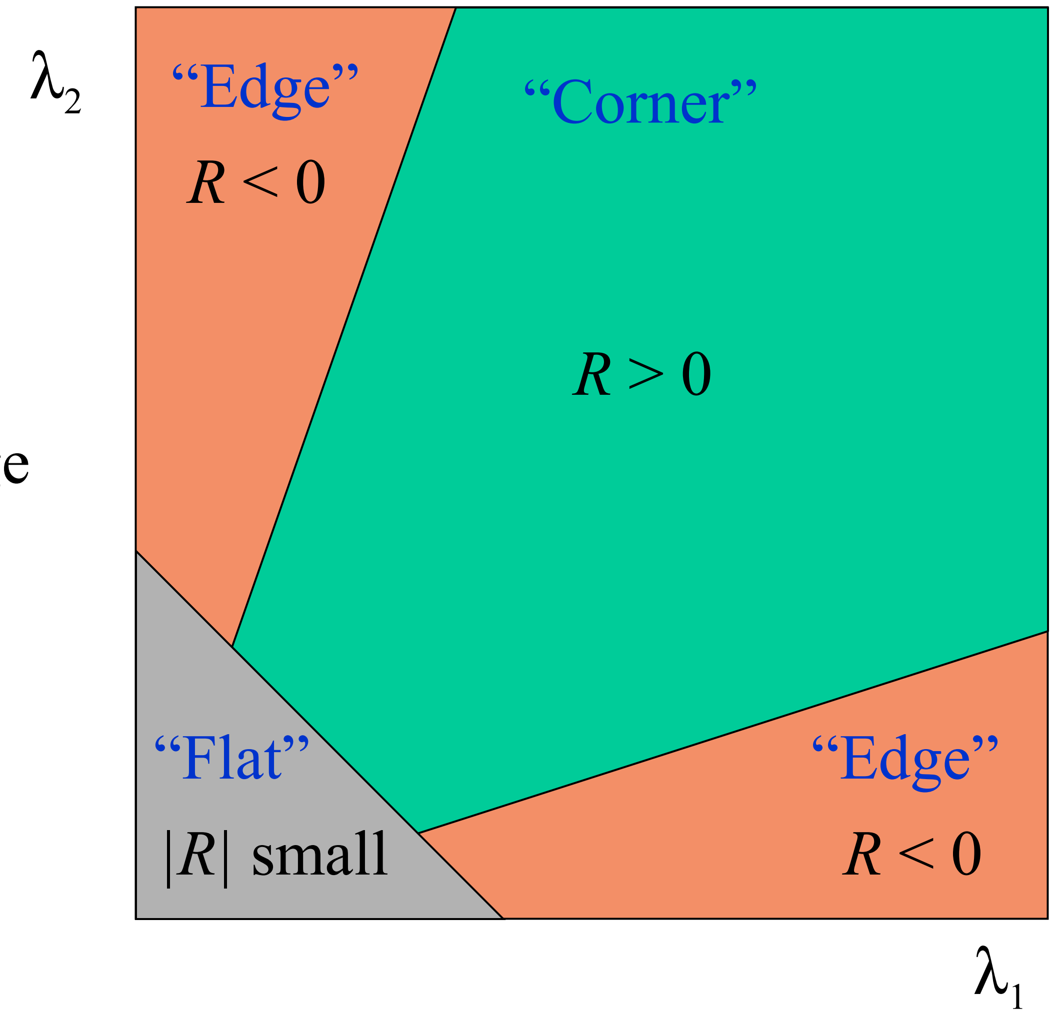
$$R = \det M - k\left(\operatorname{trace} M\right)^2$$

$$\det M = \lambda_1 \lambda_2$$
$$\operatorname{trace} M = \lambda_1 + \lambda_2$$

(k – empirical constant, k = 0.04-0.06)

Mathematics

- R depends only on eigenvalues of M
- R is large for a corner
- R is negative with large magnitude for an edge
- $|R|$ is small for a flat region



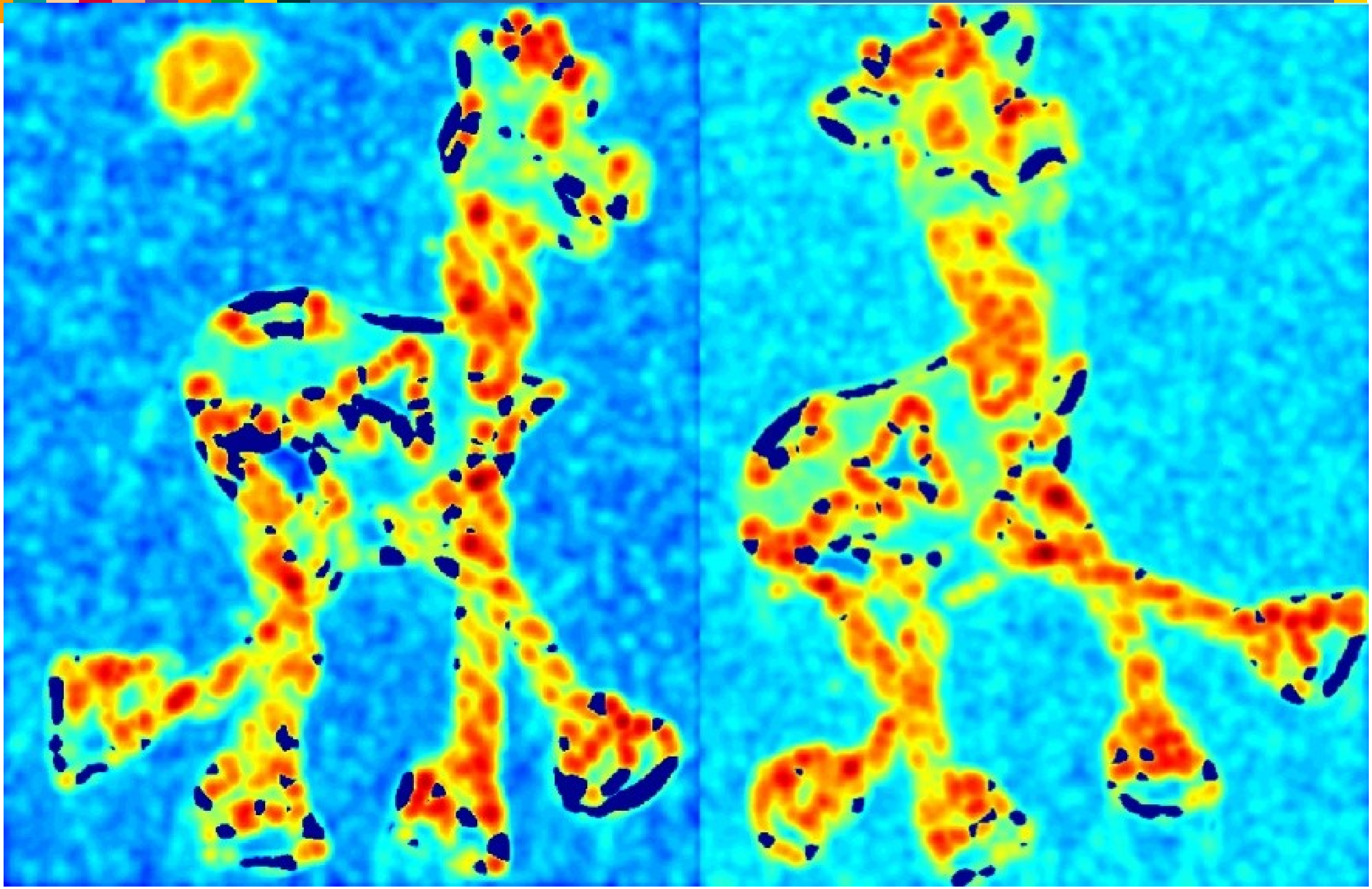
Harris Detector

- The Algorithm:
  - Find points with large corner response function R (R > threshold)
  - Take the points of local maxima of R

Harris Detector: Workflow

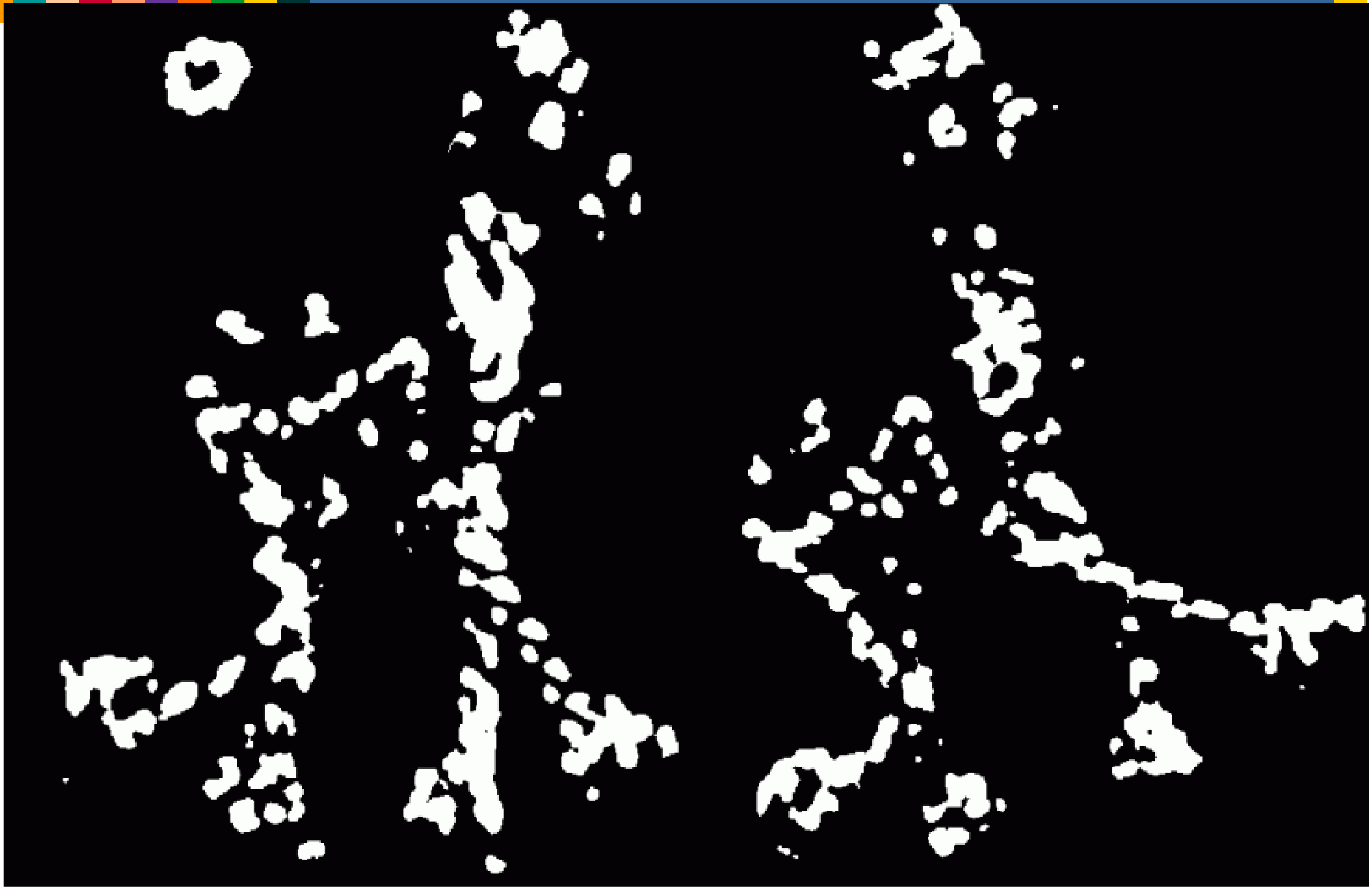
Harris Detector: Workflow

Compute corner response R



Harris Detector: Workflow

Find points with large corner response: R>threshold



Harris Detector: Workflow

Take only the points of local maxima of R

Harris Detector: Workflow

Harris Detector: Summary

- Average intensity change in direction $[u,v]$ can be expressed as a bilinear form:

$$E(u,v) \simeq [u \quad v]\, M \begin{bmatrix} u \\ v \end{bmatrix}$$

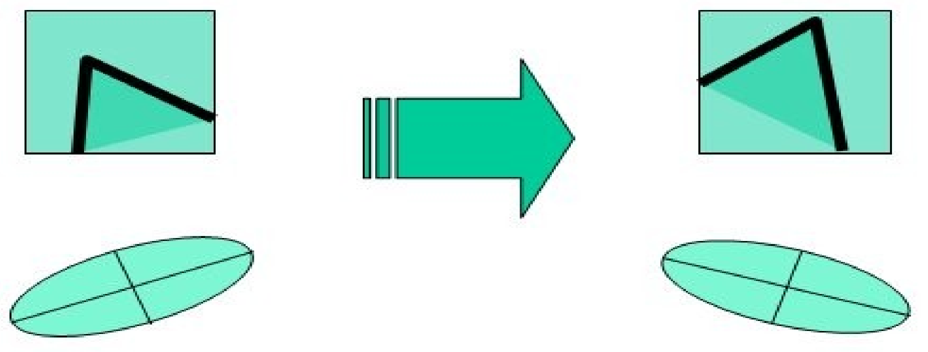
- Describe a point in terms of eigenvalues of M: *measure of corner response*

$$R = det\, M - k\,(trace\, M)^2$$

- A good (corner) point should have a *large intensity change* in *all directions*, i.e. R should be large positive

Harris Detector: Properties

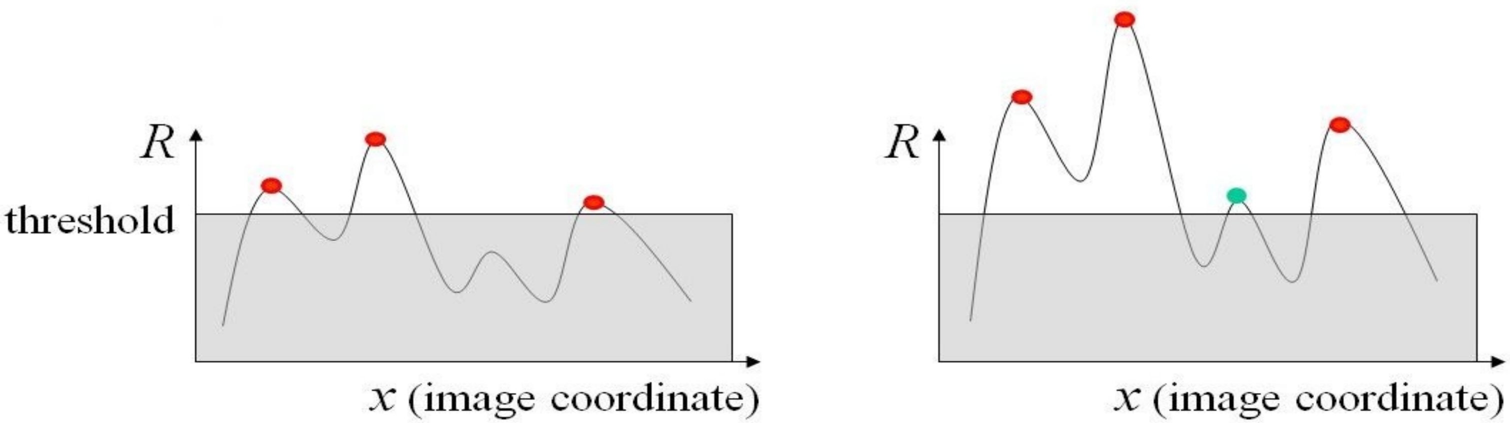
- Rotation invariance



- Ellipse rotates but its shape (i.e. eigenvalues) remains the same

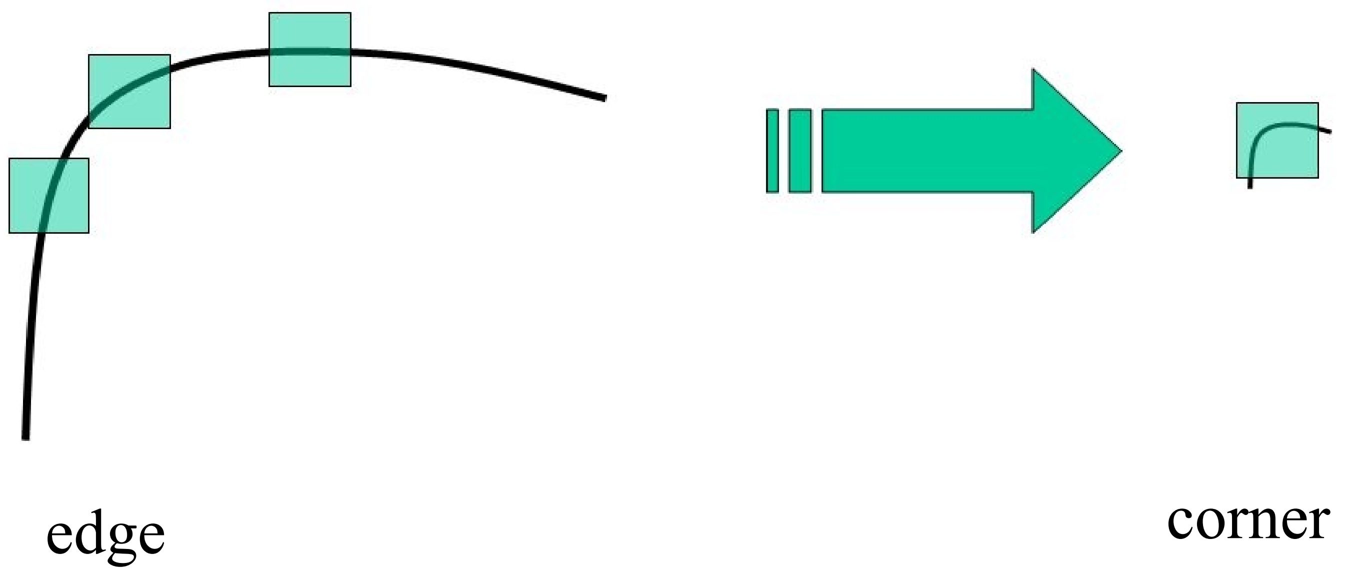
Harris Detector: Properties

- Partial invariance to affine intensity change
- Only derivatives are used => invariance to intensity shift $I \rightarrow I + b$
- Intensity scale: $I \rightarrow a\,I$



Harris Detector: Properties

- But: non-invariant to image scale!



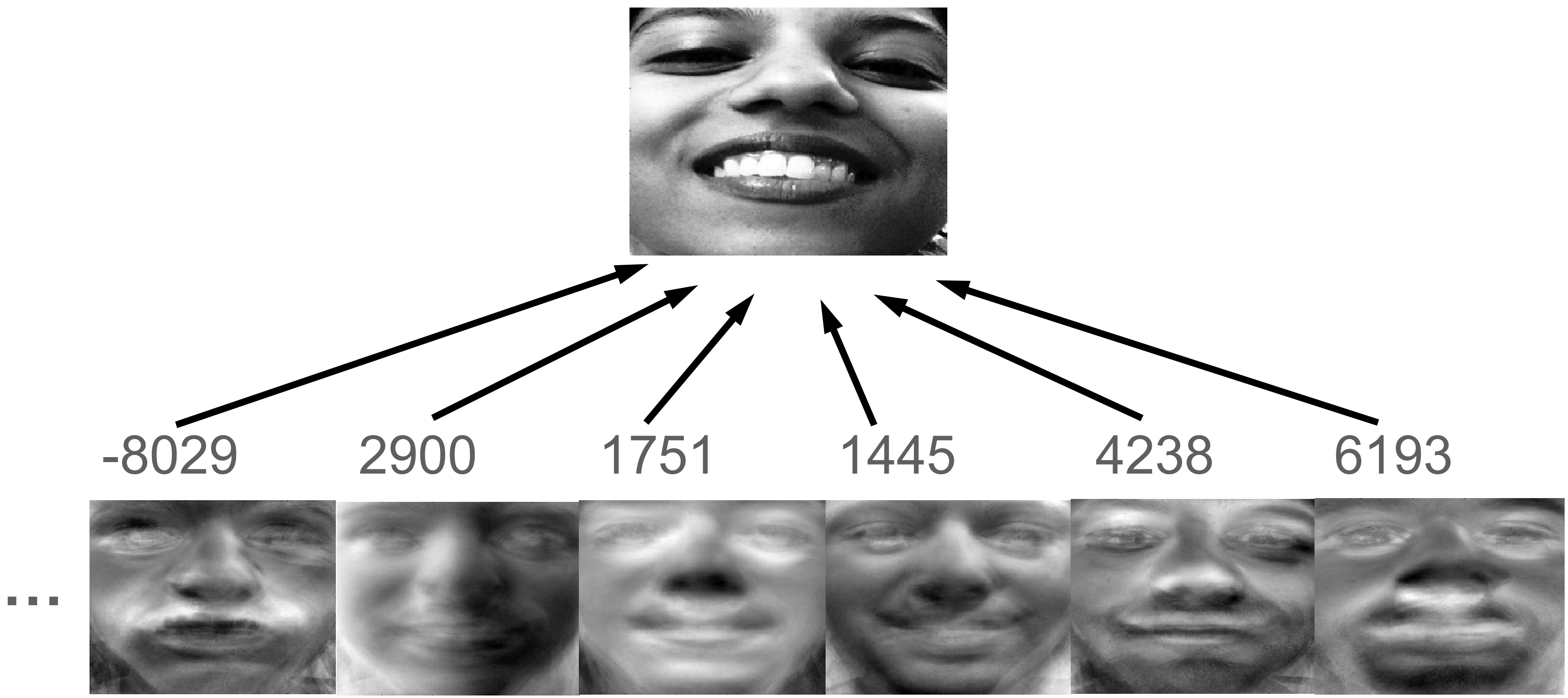
Face Recognition

Slides taken from Jeremy Wyatt

("www.cs.bham.**ac**.uk/~jlw/vision/**face_recognition**.ppt")

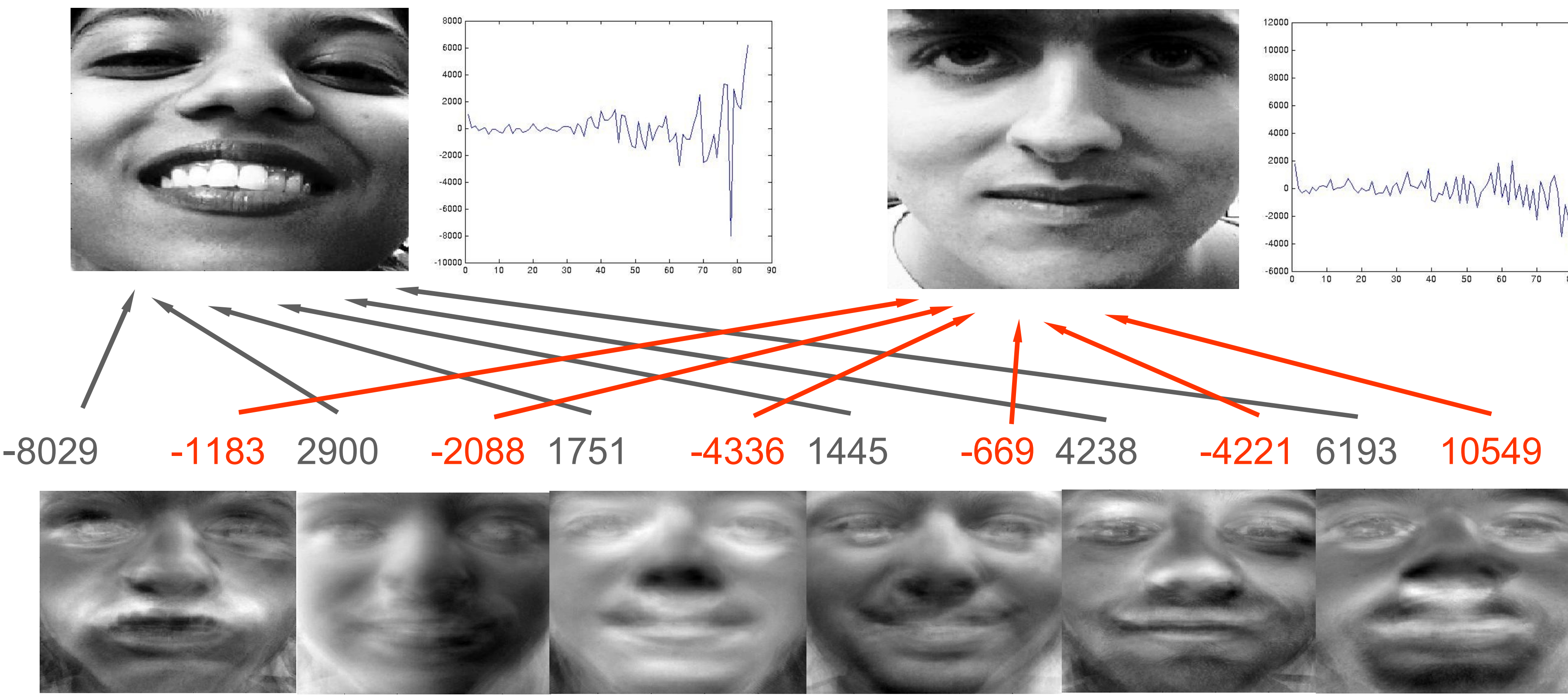
Eigenfaces: The Idea

- Think of a face as being a weighted combination of some "component" or "basis" faces
- These basis faces are called eigenfaces



Face Representation

- These basis faces can be differently weighted to represent any face
- So we can use different vectors of weights to represent different faces



Learning Eigenfaces

- How do we pick the set of basis faces?
- We take a set of real training faces

…

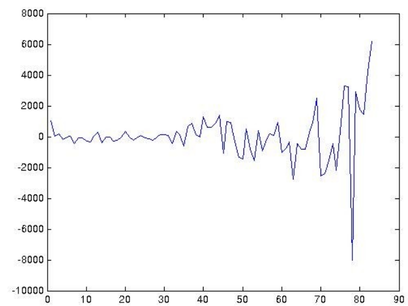
Then we find (learn) a set of basis faces which best represent the differences between them

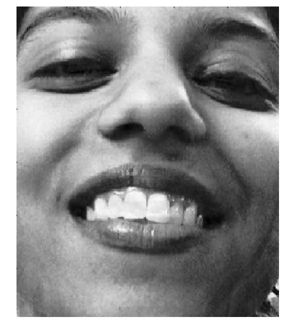
We'll use a statistical criterion for measuring this notion of "best representation of the differences between the training faces"

We can then store each face as a set of weights for those basis faces

Using Eigenfaces: Recognition & Reconstruction

- We can use the eigenfaces in two ways
- 1: we can store and then reconstruct a face from a set of weights





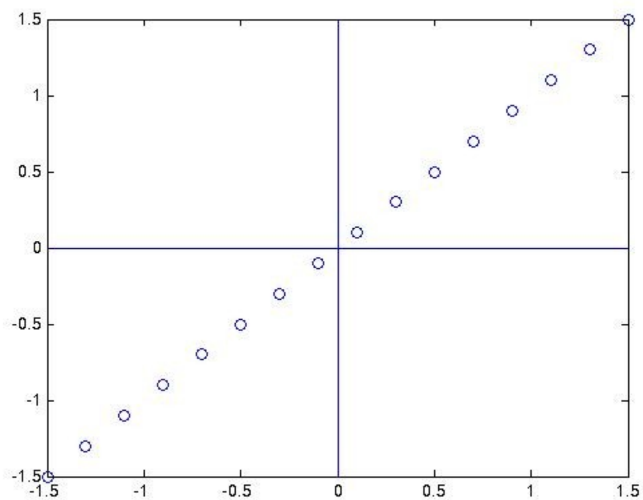
- 2: we can recognize a new picture of a familiar face

Learning Eigenfaces

- How do we learn them?
- We use a method called Principal Component Analysis (PCA)
- To understand it we will need to understand
  - What an eigenvector is
  - What covariance is
- But first we will look at what is happening in PCA qualitatively

Subspaces

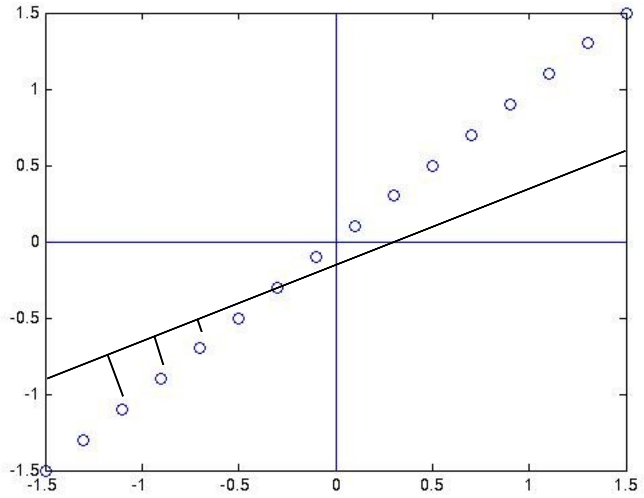
- Imagine that our face is simply a (high dimensional) vector of pixels
- We can think more easily about 2D vectors



- Here we have data in two dimensions
- But we only really need one dimension to represent it

Finding Subspaces

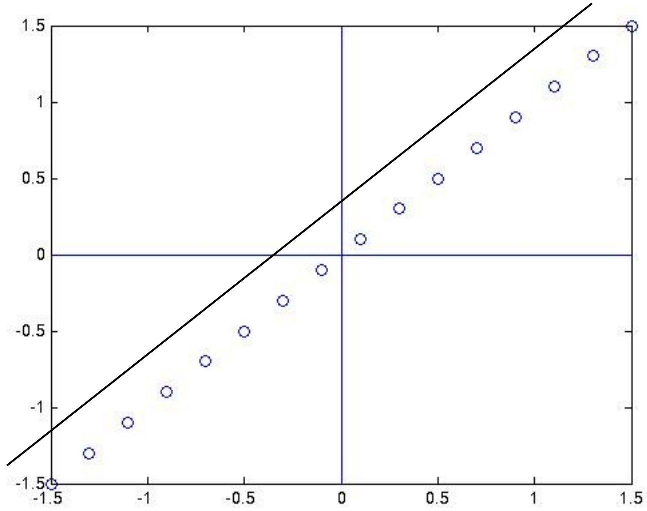
- Suppose we take a line through the space

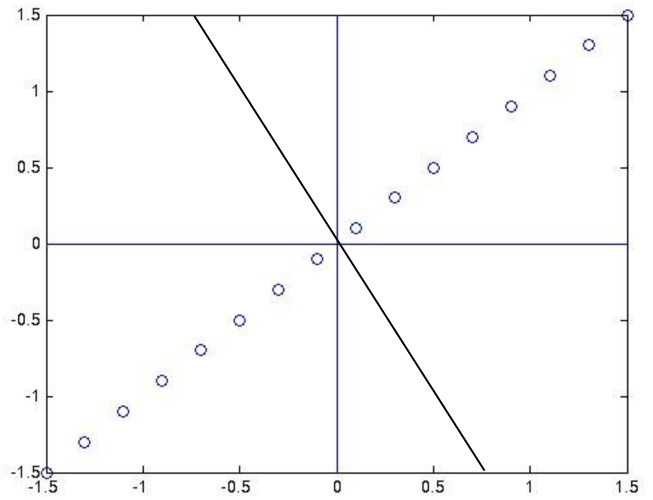


- And then take the projection of each point onto that line
- This could represent our data in “one” dimension

Finding Subspaces

- Some lines will represent the data in this way well, some badly



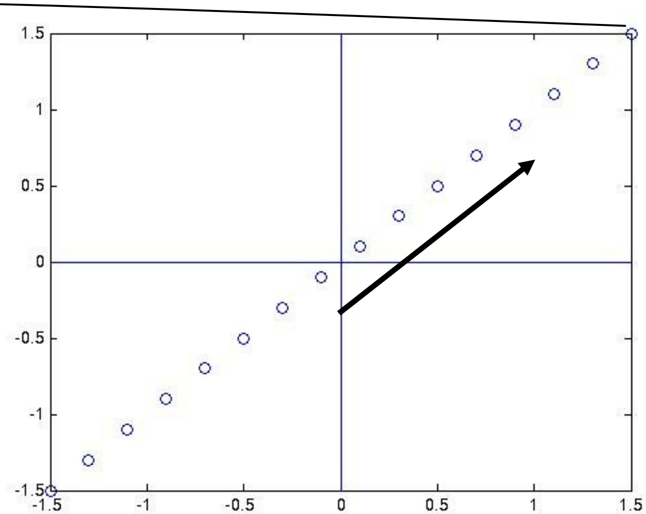


- This is because the projection onto some lines separates the data well, and the projection onto some lines separates it badly

Finding Subspaces

- Rather than a line we can perform roughly the same trick with a vector v

$$\Phi_{16} = \frac{3}{2} v$$



$$v = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

- Now we have to scale the vector to obtain any point on the line

$$\Phi_i = \mu v$$

Eigenvectors

An eigenvector is a vector v that obeys the following rule:

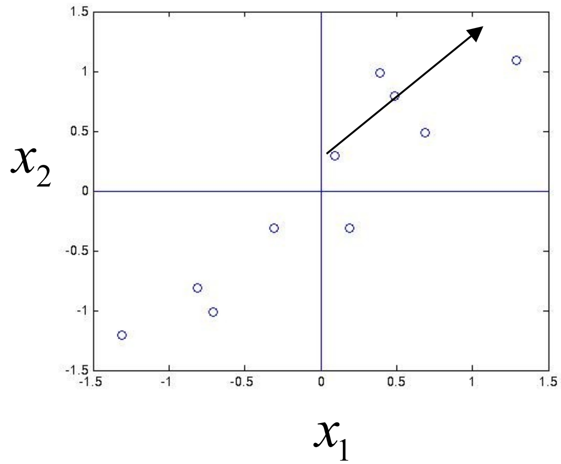
$$\mathbf{A}v = \mu v$$

Where $\mathbf{A}$ is a matrix, and μ is a scalar (called the eigenvalue)

- Only square matrices have eigenvectors
- Not all square matrices have eigenvectors
- An n *by* n matrix has at most n distinct eigenvectors
- All the distinct eigenvectors of a matrix are orthogonal

Covariance

- Which single vector can be used to separate these points as much as possible?



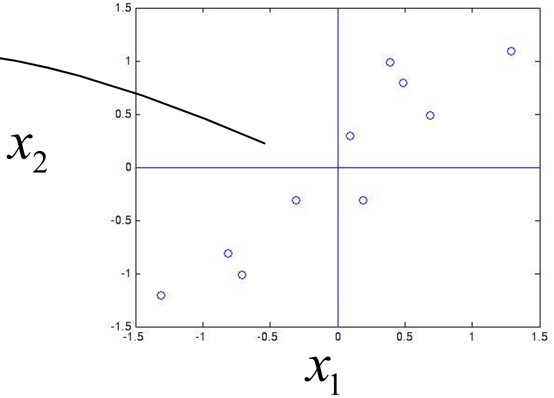
- This vector turns out to be a vector expressing the direction of the correlation
- Here are two variables x_1 and x_2
- They co-vary (y tends to change in roughly the same direction as x)

Covariance

- The covariances can be expressed as a matrix

$$C = \begin{array}{cc} x_1 & x_2 \\ \end{array}$$

$$\begin{array}{c|c} & \begin{array}{cc} x_1 & x_2 \end{array} \\ C = & \begin{bmatrix} .617 & .615 \\ .615 & .717 \end{bmatrix} \begin{array}{c} x_1 \\ x_2 \end{array} \end{array}$$

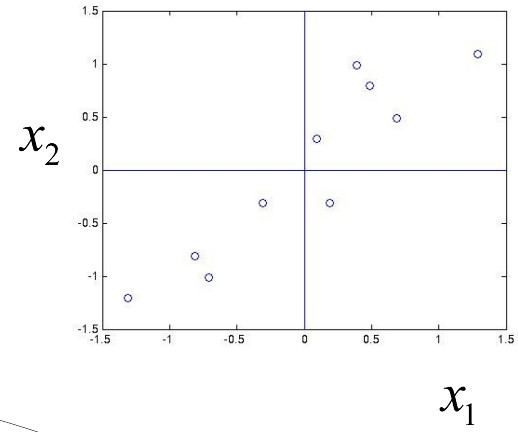


- The diagonal elements are the variances e.g. Var(x_1)
- The covariance of two variables is:

$$\mathrm{cov}(x_1, x_2) = \frac{\sum_{i=1}^{n} (x_1^i - \bar{x}_1)(x_2^i - \bar{x}_2)}{n-1}$$

Eigenvectors of the covariance matrix

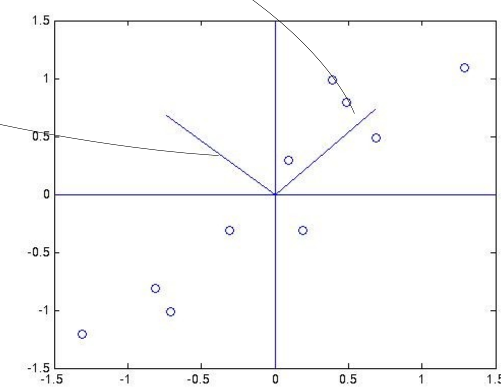
- The covariance matrix has eigenvectors



covariance matrix $C = \begin{bmatrix} .617 & .615 \\ .615 & .717 \end{bmatrix}$

eigenvectors $v_1 = \begin{bmatrix} -.735 \\ .678 \end{bmatrix}$ $v_2 = \begin{bmatrix} .678 \\ .735 \end{bmatrix}$

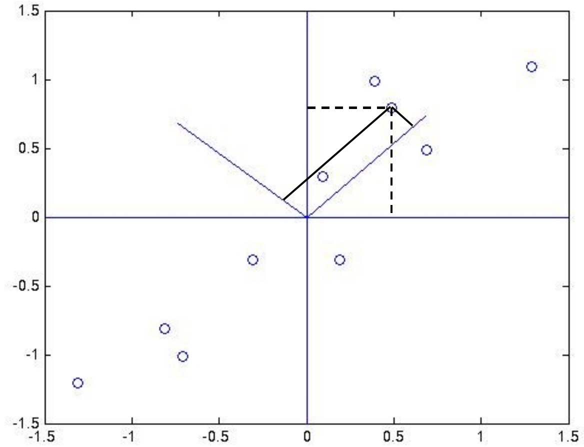
eigenvalues $\mu_1 = 0.049$ $\mu_2 = 1.284$



- Eigenvectors with larger eigenvalues correspond to directions in which the data varies more
- Finding the eigenvectors and eigenvalues of the covariance matrix for a set of data is called principal component analysis

Expressing points

- Suppose you think of your eigenvectors as specifying a new vector space



- i.e. one can reference any point in terms of those eigenvectors
- A point’s position in this new coordinate system is what we earlier referred to as its “weight vector”
- For many data sets you can cope with fewer dimensions in the new space than in the old space

Eigenfaces

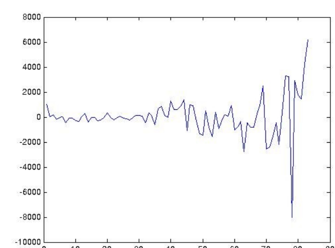
- In the face case:
  - the face is a point in a high-dimensional space
  - the training set of face pictures is our set of points
- Training:
  - calculate the covariance matrix of the faces
  - find the eigenvectors of the covariance matrix
- These eigenvectors are the eigenfaces or basis faces
- Eigenfaces with bigger eigenvalues will explain more of the variation in the set of faces, i.e. will be more distinguishing

Eigenfaces: image space to face space

- When we see an image of a face we can transform it to face space

$$\mathbf{w}_k = \mathbf{x}^{\mathbf{i}} . v_k$$

- There are k=1…n eigenfaces v_k
- The i^{th} face in image space is a vector $\mathbf{x}^{\mathbf{i}}$
- The corresponding weight is $\mathbf{w}_k$
- We calculate the corresponding weight for every eigenface

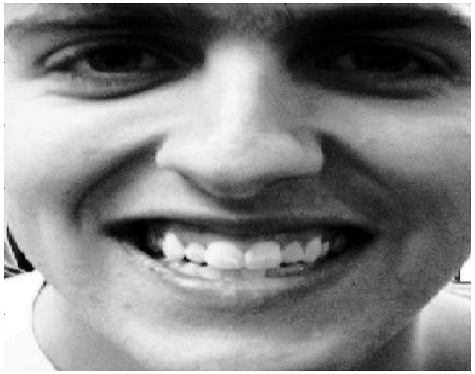


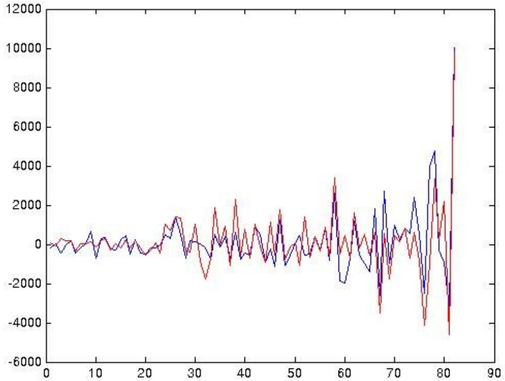
Recognition in face space

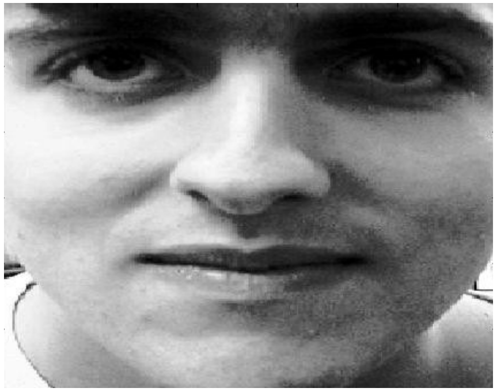
- Recognition is now simple. We find the euclidean distance ***d*** between our face and all the other stored faces in face space:

$$d(w^1, w^2) = \sqrt{\sum_{i=1}^{n} \left(w_i^1 - w_i^2 \right)^2}$$

- The closest face in face space is the chosen match

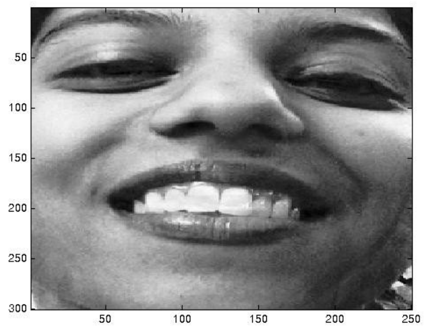




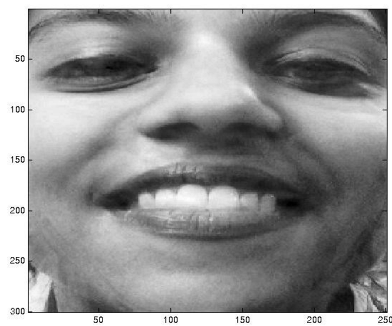


Reconstruction

- The more eigenfaces you have the better the reconstruction, but you can have high quality reconstruction even with a small number of eigenfaces

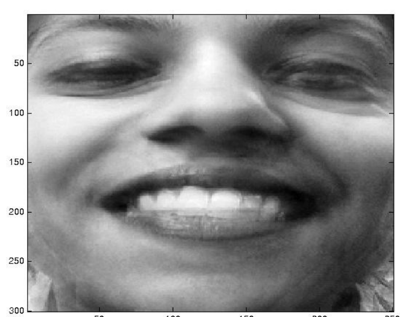


82

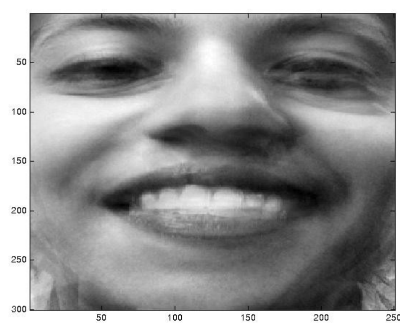


70

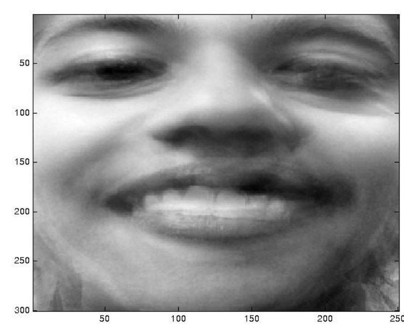
50



30



20



10

Summary

- Statistical approach to visual recognition
- Also used for object recognition
- Problems
- Reference: M. Turk and A. Pentland (1991). Eigenfaces for recognition, *Journal of Cognitive Neuroscience*, 3(1): 71–86.