Facial Expression Recognition

YING SHEN
SSE, TONGJI UNIVERSITY

Outline

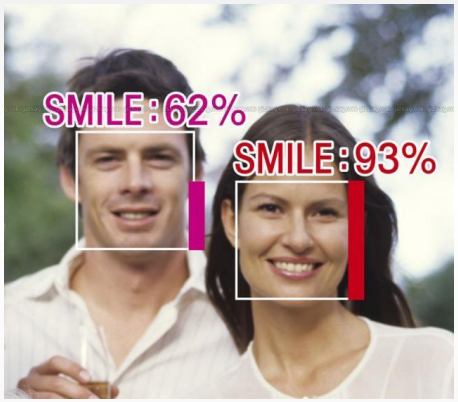
- Introduction
- Facial expression recognition
  - Appearance-based vs. model based
- Active appearance model (AAM)
  - Pre-requisite
  - Principle component analysis (PCA)
  - Procrustes analysis
  - ASM
  - Delaunay triangulation
  - AAM

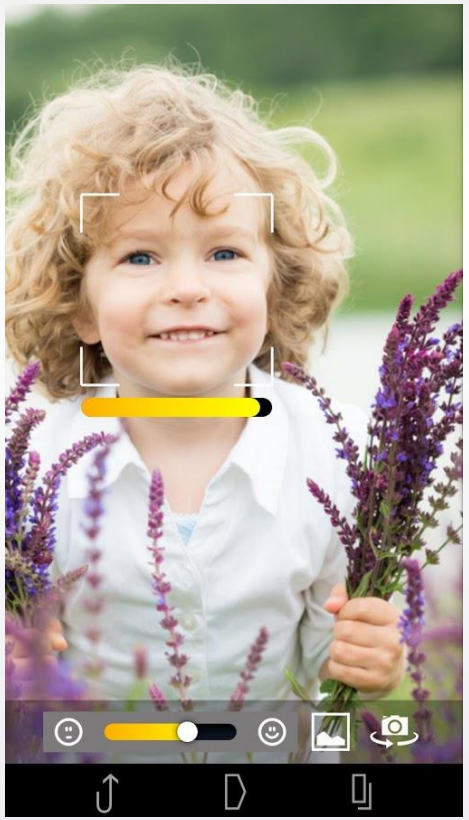
Outline

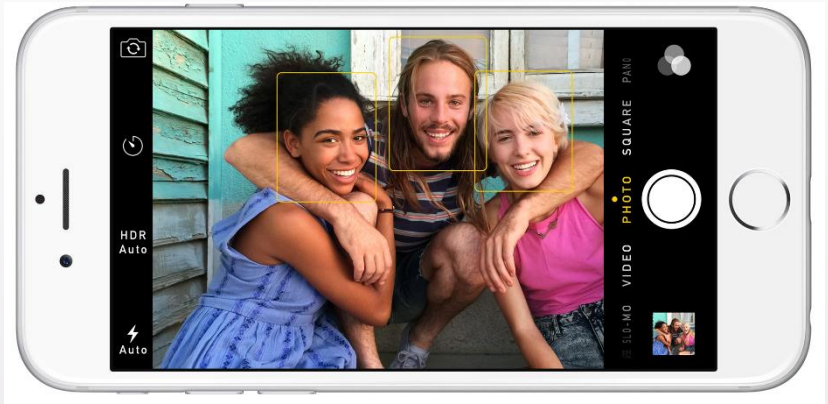
- Introduction
- Facial expression recognition
  - Appearance-based vs. model based
- Active appearance model (AAM)
  - Pre-requisite
  - Principle component analysis (PCA)
  - Procrustes analysis
  - ASM
  - Delaunay triangulation
  - AAM

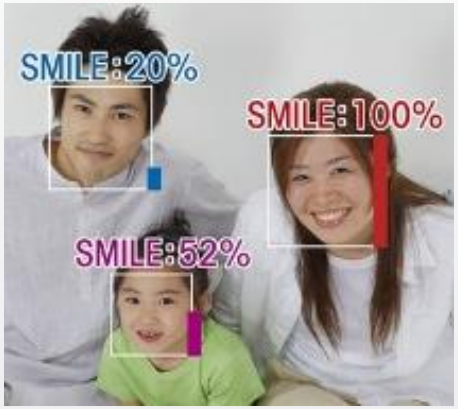
Introduction

- Smile detection









Introduction

- Facial expression recognition

Happy

Surprise

Angry

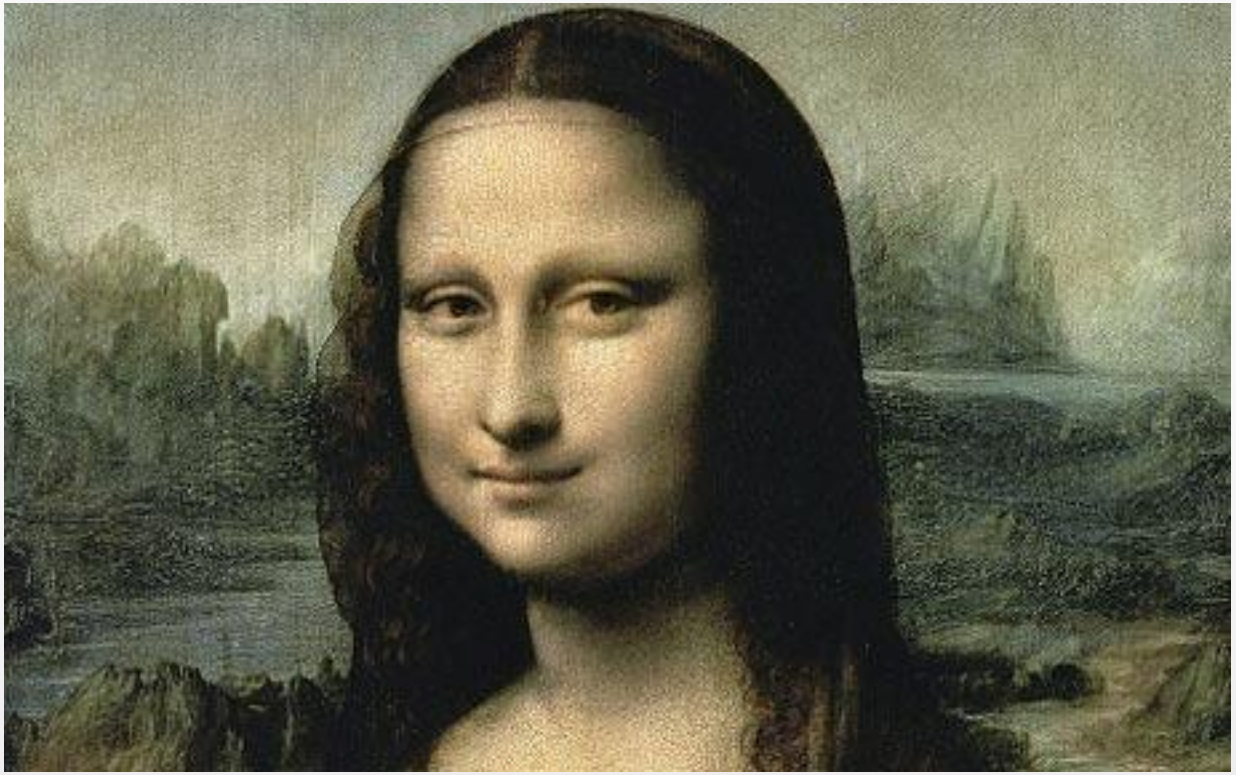
Disgust

Sad

Fear

Introduction

- Facial expression recognition



Happy: 83%

Disgusted: 9%

Fearful: 6%

Angry: 2%

Introduction

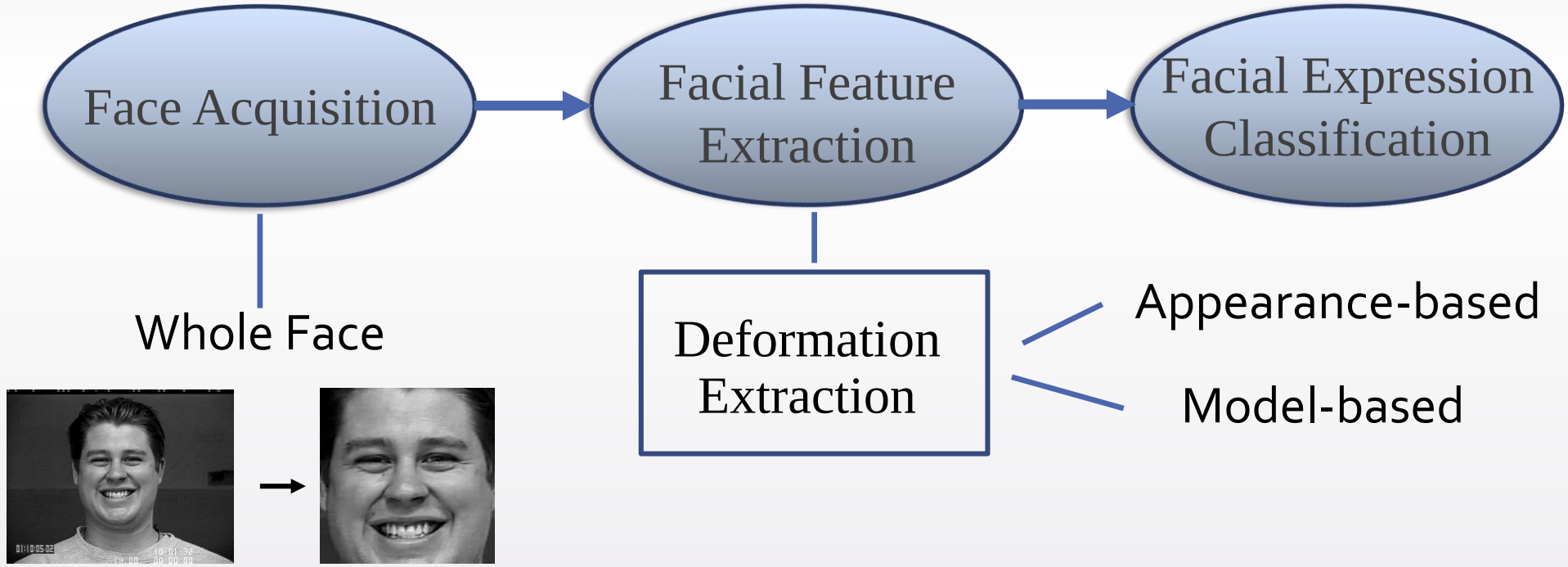
- Factors affect facial expression recognition accuracy
  - Pose
  - Illumination
  - …

Outline

- Introduction
- Facial expression recognition
  - Appearance-based vs. model-based
- Active appearance model (AAM)
  - Pre-requisite
  - Principle component analysis
  - Procrustes analysis
  - ASM
  - Delaunay triangulation
  - AAM

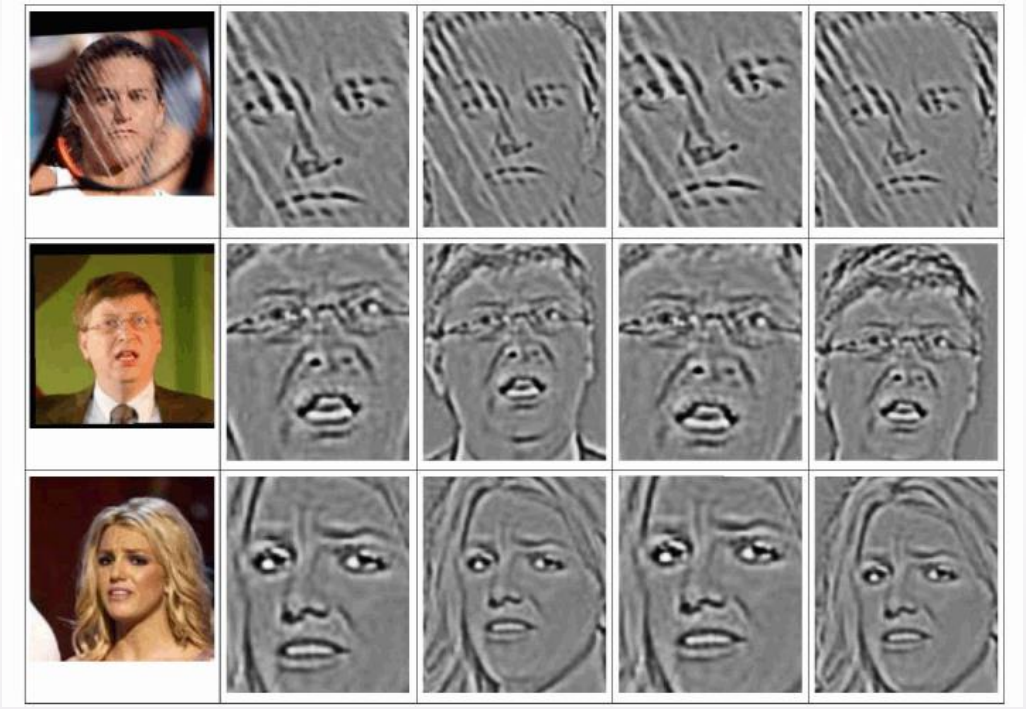
Facial expression recognition

- Framework of automatic facial expression recognition



Facial expression recognition

- Appearance-based methods
  - E.g. Features are extracted using Garbor filter

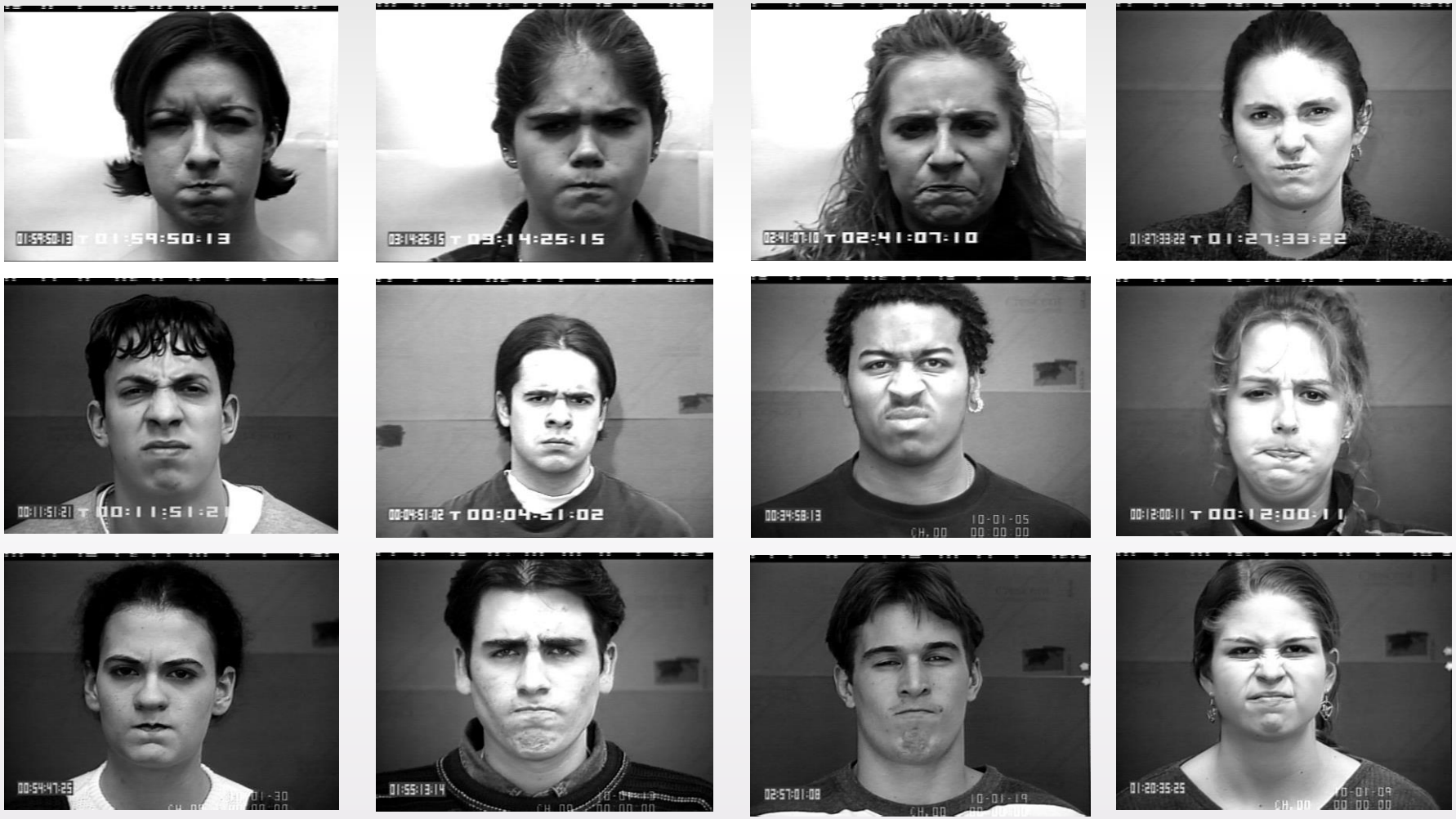


Facial expression recognition

- CMU facial expression database (CK, CK+ database)
  - 8000+ images
  - Six basic expressions
- Training preparation
  - create positive examples
  - prepare negative examples
- Training
  - Train Haar-like features using OpenCV
  - Or train a NN classifier using extracted features

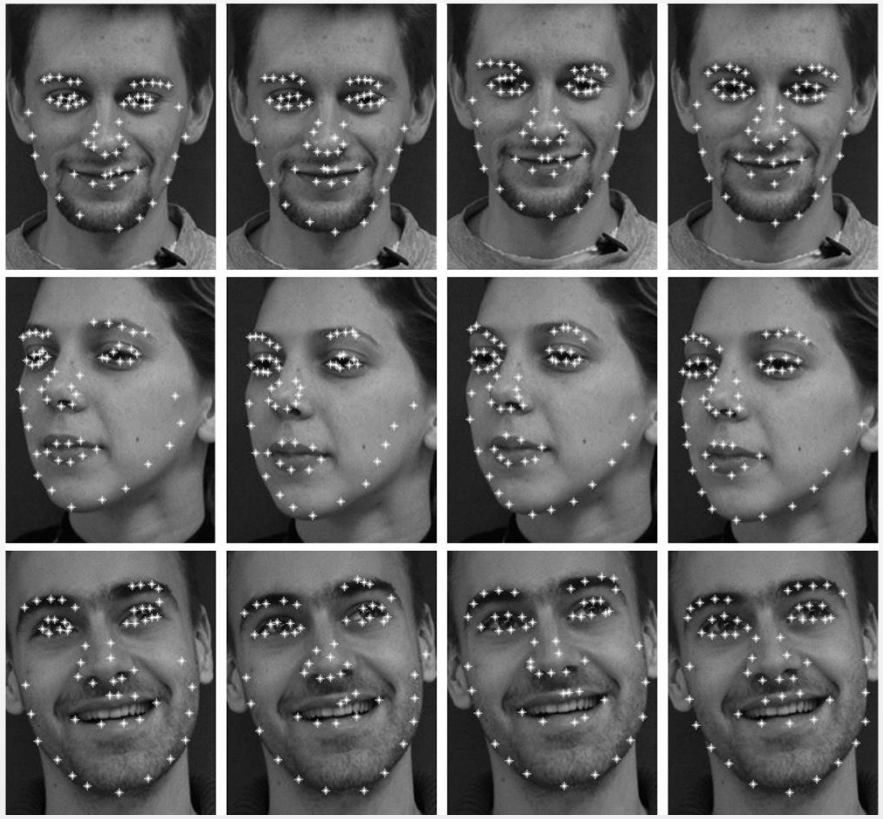
Facial expression recognition

- Angry Positive Examples for Training



Facial expression recognition

- Model-based methods



Outline

- Introduction
- Facial expression recognition
  - Appearance-based vs. model-based
- Active appearance model (AAM)
  - Pre-requisite
  - Principle component analysis (PCA)
  - Procrustes analysis
  - ASM
  - Delaunay triangulation
  - AAM

Pre-requisite: Lagrange multiplier

- Single-variable function

$f(x)$ is differentiable in (a, b). At $x_0 \in (a,b)$, $f(x)$ achieves an extremum

$$\rightarrow \frac{df}{dx}|_{x_0} = 0$$

- Two-variables function

$f(x, y)$ is differentiable in its domain. At (x_0, y_0), $f(x, y)$ achieves an extremum

$$\rightarrow \frac{\partial f}{\partial x}|_{(x_0,y_0)} = 0, \frac{\partial f}{\partial y}|_{(x_0,y_0)} = 0$$

Pre-requisite: Lagrange multiplier

- In general case

If $\mathbf{x}_0$ is a stationary point of $f(\mathbf{x}), \mathbf{x} \in \mathbb{R}^{n\times 1}$

$$\frac{\partial f}{\partial x_1}|_{\mathbf{x}_0} = 0, \frac{\partial f}{\partial x_2}|_{\mathbf{x}_0} = 0, \ldots, \frac{\partial f}{\partial x_n}|_{\mathbf{x}_0} = 0$$

Pre-requisite: Lagrange multiplier

- Lagrange multiplier is a strategy for finding the local extremum of a function subject to equality constraints

Problem: find stationary points for $y = f(\mathbf{x}), \mathbf{x} \in \mathbb{R}^{n\times 1}$

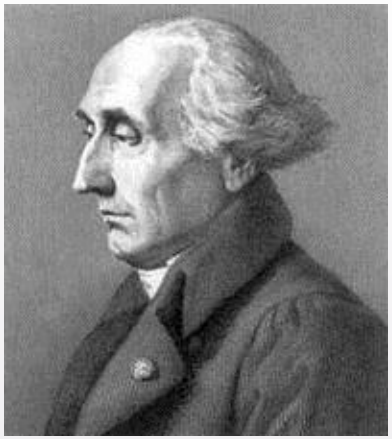
under *m* constraints $g_k(\mathbf{x}) = 0, k = 1, 2, ..., m$

Solution:

$$F(\mathbf{x}; \lambda_1, ..., \lambda_m) = f(\mathbf{x}) + \sum_{k=1}^{m} \lambda_k g_k(\mathbf{x})$$

If $(\mathbf{x}_0, \lambda_{10}, \lambda_{20} ..., \lambda_{m0})$ is a stationary point of F, then,

$\mathbf{x}_0$ is a stationary point of $f(\mathbf{x})$ with constraints



Joseph-Louis Lagrange
Jan. 25, 1736~Apr.10, 1813

Pre-requisite: Lagrange multiplier

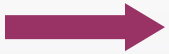
- Lagrange multiplier is a strategy for finding the local extremum of a function subject to equality constraints

Problem: find stationary points for $y = f(\mathbf{x}), \mathbf{x} \in \mathbb{R}^{n\times 1}$

under *m* constraints $g_k(\mathbf{x}) = 0, k = 1, 2, \ldots, m$

Solution:

$$F(\mathbf{x}; \lambda_1, \ldots, \lambda_m) = f(\mathbf{x}) + \sum_{k=1}^{m} \lambda_k g_k(\mathbf{x})$$

$(\mathbf{x}_0, \lambda_{10}, \ldots, \lambda_{m0})$ is a stationary point of F 

$$\frac{\partial F}{\partial x_1} = 0, \frac{\partial F}{\partial x_2} = 0, \ldots, \frac{\partial F}{\partial x_n} = 0, \frac{\partial F}{\partial \lambda_1} = 0, \frac{\partial F}{\partial \lambda_2} = 0, \ldots, \frac{\partial F}{\partial \lambda_m} = 0$$

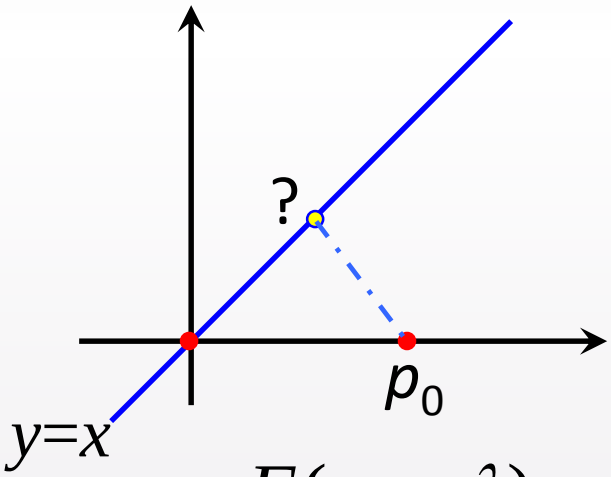
at that point

n + m equations!

Pre-requisite: Lagrange multiplier

- Example

  Problem: for a given point p_0 = (1, 0), among all the points lying on the line $y=x$, identify the one having the least distance to p_0.



The distance is

$$f(x,y)=(x-1)^2+(y-0)^2$$

Now we want to find the stationary point of $f(x, y)$ under the constraint

$$g(x,y)=y-x=0$$

According to Lagrange multiplier method, construct another function

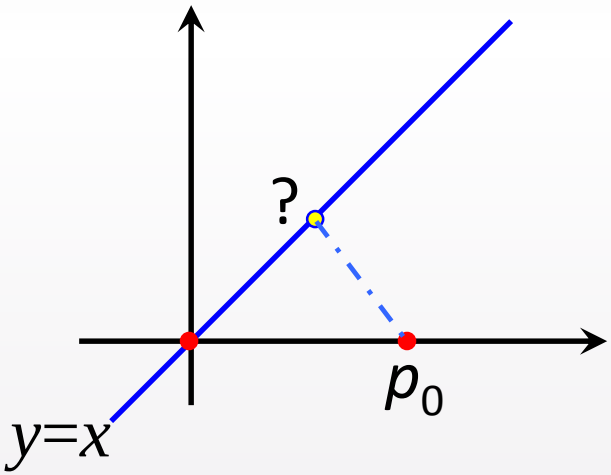
$$F(x,y,\lambda)=f(x)+\lambda g(x)=(x-1)^2+y^2+\lambda(y-x)$$

Find the stationary point for $F(x,y,\lambda)$

Pre-requisite: Lagrange multiplier

- Example

  Problem: for a given point p_0 = (1, 0), among all the points lying on the line $y=x$, identify the one having the least distance to p_0.



$$\begin{cases} \dfrac{\partial F}{\partial x} = 0 \\ \dfrac{\partial F}{\partial y} = 0 \\ \dfrac{\partial F}{\partial \lambda} = 0 \end{cases} \rightarrow \begin{cases} 2(x-1)+\lambda = 0 \\ 2y-\lambda = 0 \\ x-y = 0 \end{cases} \rightarrow \begin{cases} x = 0.5 \\ y = 0.5 \\ \lambda = 1 \end{cases}$$

→ $(0.5, 0.5, 1)$ is a stationary point of $F(x, y, \lambda)$

→ $(0.5, 0.5)$ is a stationary point of $f(x,y)$ under constraints

Pre-requisite: matrix differentiation

- Function is a vector and the variable is a scalar

$$f(t) = \left[f_1(t), f_2(t), \ldots, f_n(t) \right]^T$$

Definition

$$\frac{df}{dt} = \left[\frac{df_1(t)}{dt}, \frac{df_2(t)}{dt}, \ldots, \frac{df_n(t)}{dt} \right]^T$$

Pre-requisite: matrix differentiation

- Function is a matrix and the variable is a scalar

$$f(t)=\begin{bmatrix} f_{11}(t) & f_{12}(t),\ldots, f_{1m}(t) \\ f_{21}(t) & f_{22}(t),\ldots, f_{2m}(t) \\ \vdots & \\ f_{n1}(t) & f_{n2}(t),\ldots, f_{nm}(t) \end{bmatrix}=\left[f_{ij}(t) \right]_{n\times m}$$

Definition

$$\frac{df}{dt}=\begin{bmatrix} \frac{df_{11}(t)}{dt} & \frac{df_{12}(t)}{dt},\ldots, \frac{df_{1m}(t)}{dt} \\ \frac{df_{21}(t)}{dt} & \frac{df_{22}(t)}{dt},\ldots, \frac{df_{2m}(t)}{dt} \\ \vdots & \\ \frac{df_{n1}(t)}{dt} & \frac{df_{n2}(t)}{dt},\ldots, \frac{df_{nm}(t)}{dt} \end{bmatrix}=\left[\frac{df_{ij}(t)}{dt} \right]_{n\times m}$$

Pre-requisite: matrix differentiation

- Function is a scalar and the variable is a vector

$$f(\mathbf{x}), \mathbf{x} = (x_1, x_2, ..., x_n)^T$$

Definition

$$\frac{df}{d\mathbf{x}} = \left[\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, ..., \frac{\partial f}{\partial x_n} \right]^T$$

In a similar way,

$$f(\mathbf{x}), \mathbf{x} = (x_1, x_2, ..., x_n)$$

$$\frac{df}{d\mathbf{x}} = \left[\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, ..., \frac{\partial f}{\partial x_n} \right]$$

Pre-requisite: matrix differentiation

- Function is a vector and the variable is a vector

$$\mathbf{x} = \left[x_1, x_2, ..., x_n \right]^T, \mathbf{y} = \left[y_1(\mathbf{x}), y_2(\mathbf{x}), ..., y_m(\mathbf{x}) \right]^T$$

Definition

$$\frac{d\mathbf{y}}{d\mathbf{x}^T} = \begin{bmatrix} \frac{\partial y_1(\mathbf{x})}{\partial x_1}, \frac{\partial y_1(\mathbf{x})}{\partial x_2}, ..., \frac{\partial y_1(\mathbf{x})}{\partial x_n} \\ \frac{\partial y_2(\mathbf{x})}{\partial x_1}, \frac{\partial y_2(\mathbf{x})}{\partial x_2}, ..., \frac{\partial y_2(\mathbf{x})}{\partial x_n} \\ \vdots \\ \frac{\partial y_m(\mathbf{x})}{\partial x_1}, \frac{\partial y_m(\mathbf{x})}{\partial x_2}, ..., \frac{\partial y_m(\mathbf{x})}{\partial x_n} \end{bmatrix}_{m \times n}$$

Pre-requisite: matrix differentiation

- Function is a vector and the variable is a vector

$$\mathbf{x} = \left[x_1, x_2, \ldots, x_n\right]^T, \mathbf{y} = \left[y_1(\mathbf{x}), y_2(\mathbf{x}), \ldots, y_m(\mathbf{x})\right]^T$$

In a similar way,

$$\frac{d\mathbf{y}^T}{d\mathbf{x}} = \begin{bmatrix} \frac{\partial y_1(\mathbf{x})}{\partial x_1}, \frac{\partial y_2(\mathbf{x})}{\partial x_1}, \ldots, \frac{\partial y_m(\mathbf{x})}{\partial x_1} \\ \frac{\partial y_1(\mathbf{x})}{\partial x_2}, \frac{\partial y_2(\mathbf{x})}{\partial x_2}, \ldots, \frac{\partial y_m(\mathbf{x})}{\partial x_2} \\ \vdots \\ \frac{\partial y_1(\mathbf{x})}{\partial x_n}, \frac{\partial y_2(\mathbf{x})}{\partial x_n}, \ldots, \frac{\partial y_m(\mathbf{x})}{\partial x_n} \end{bmatrix}_{n \times m}$$

Pre-requisite: matrix differentiation

- Function is a vector and the variable is a vector

  Example:

$$\mathbf{y} = \begin{bmatrix} y_1(\mathbf{x}) \\ y_2(\mathbf{x}) \end{bmatrix}, \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}, y_1(\mathbf{x}) = x_1^2 - x_2, y_2(\mathbf{x}) = x_3^2 + 3x_2$$

$$\frac{d\mathbf{y}^T}{d\mathbf{x}} = \begin{bmatrix} \frac{\partial y_1(\mathbf{x})}{\partial x_1} & \frac{\partial y_2(\mathbf{x})}{\partial x_1} \\ \frac{\partial y_1(\mathbf{x})}{\partial x_2} & \frac{\partial y_2(\mathbf{x})}{\partial x_2} \\ \frac{\partial y_1(\mathbf{x})}{\partial x_3} & \frac{\partial y_2(\mathbf{x})}{\partial x_3} \end{bmatrix} = \begin{bmatrix} 2x_1 & 0 \\ -1 & 3 \\ 0 & 2x_3 \end{bmatrix}$$

Pre-requisite: matrix differentiation

- Function is a scalar and the variable is a matrix

$$f(\mathbf{X}), \mathbf{X} \in \mathbb{R}^{m \times n}$$

Definition

$$\frac{df(\mathbf{X})}{d\mathbf{X}} = \begin{bmatrix} \frac{\partial f}{\partial x_{11}} & \frac{\partial f}{\partial x_{12}} & \cdots & \frac{\partial f}{\partial x_{1n}} \\ \cdots & & & \\ \frac{\partial f}{\partial x_{m1}} & \frac{\partial f}{\partial x_{m2}} & \cdots & \frac{\partial f}{\partial x_{mn}} \end{bmatrix}$$

Pre-requisite: matrix differentiation

- Useful results

(1)

$$\mathbf{x}, \mathbf{a} \in \mathbb{R}^{n \times 1}$$

Then,

$$\frac{d\mathbf{a}^T\mathbf{x}}{d\mathbf{x}} = \mathbf{a}, \frac{d\mathbf{x}^T\mathbf{a}}{d\mathbf{x}} = \mathbf{a}$$

How to prove?

Pre-requisite: matrix differentiation

- Useful results

(2) $A \in \mathbb{R}^{m \times n}, \mathbf{x} \in \mathbb{R}^{n \times 1}$ Then, $\frac{d A\mathbf{x}}{d\mathbf{x}^T} = A$

(3) $A \in \mathbb{R}^{m \times n}, \mathbf{x} \in \mathbb{R}^{n \times 1}$ Then, $\frac{d\mathbf{x}^T A^T}{d\mathbf{x}} = A^T$

(4) $A \in \mathbb{R}^{n \times n}, \mathbf{x} \in \mathbb{R}^{n \times 1}$ Then, $\frac{d\mathbf{x}^T A\mathbf{x}}{d\mathbf{x}} = (A + A^T)\mathbf{x}$

(5) $\mathbf{X} \in \mathbb{R}^{m \times n}, \mathbf{a} \in \mathbb{R}^{m \times 1}, \mathbf{b} \in \mathbb{R}^{n \times 1}$ Then, $\frac{d\mathbf{a}^T \mathbf{X}\mathbf{b}}{d\mathbf{X}} = \mathbf{a}\mathbf{b}^T$

(6) $\mathbf{X} \in \mathbb{R}^{n \times m}, \mathbf{a} \in \mathbb{R}^{m \times 1}, \mathbf{b} \in \mathbb{R}^{n \times 1}$ Then, $\frac{d\mathbf{a}^T \mathbf{X}^T \mathbf{b}}{d\mathbf{X}} = \mathbf{b}\mathbf{a}^T$

(7) $\mathbf{x} \in \mathbb{R}^{n \times 1}$ Then, $\frac{d\mathbf{x}^T \mathbf{x}}{d\mathbf{x}} = 2\mathbf{x}$

Outline

- Introduction
- Facial expression recognition
  - Appearance-based vs. model-based
- Active appearance model (AAM)
  - Pre-requisite
  - Principle component analysis
  - Procrustes analysis
  - ASM
  - Delaunay triangulation
  - AAM

Principal Component Analysis (PCA)

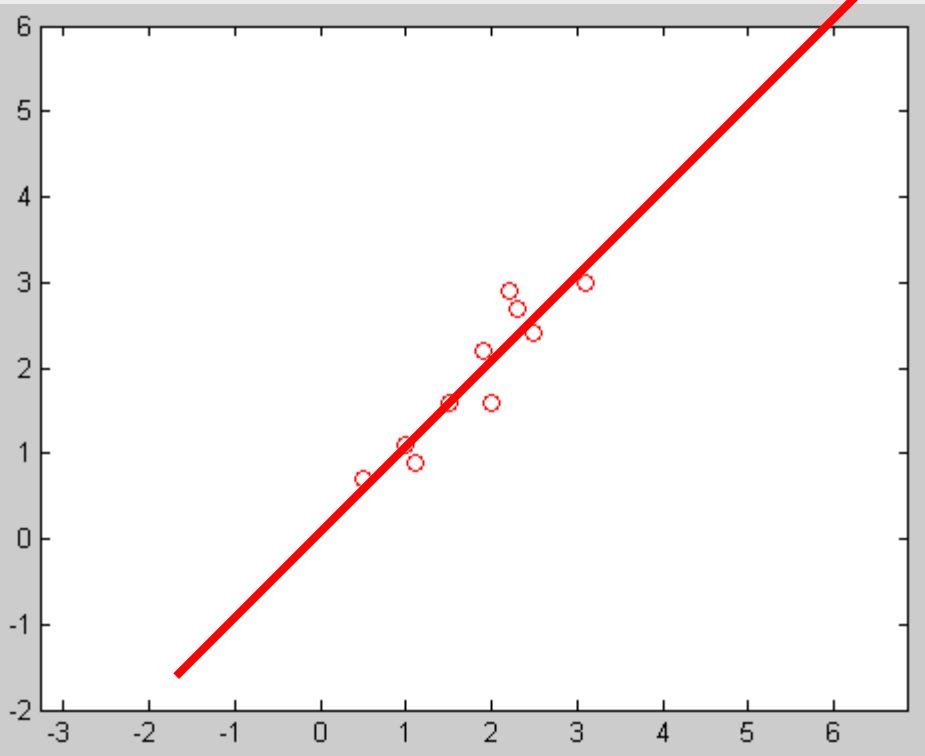
- PCA: converts a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components
- The number of principal components is less than or equal to the number of original variables
- This transformation is defined in such a way that the first principal component has the largest possible variance, and each succeeding component in turn has the highest variance possible under the constraint that it be orthogonal to (i.e., uncorrelated with) the preceding components

Principal Component Analysis (PCA)

- Illustration

How to find?

x , y
(2.5, 2.4)
(0.5, 0.7)
(2.2, 2.9)
(1.9, 2.2)
(3.1, 3.0)
(2.3, 2.7)
(2.0, 1.6)
(1.0, 1.1)
(1.5, 1.6)
(1.1, 0.9)



De-correlation!

Along which orientation the data points scatter most?

Principal Component Analysis (PCA)

- Identify the orientation with largest variance

Suppose $\mathbf{X}$ contains n data points, and each data point is p-dimensional, that is

$$\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}, \mathbf{x}_i \in \mathbb{R}^{p\times 1}, \mathbf{X} \in \mathbb{R}^{p\times n}$$

Now, we want to find such a unit vector α_1 ,

$$\alpha_1 = \arg\max_{\alpha} \left(\mathrm{var}\left(\alpha^T \mathbf{X} \right) \right), \alpha \in \mathbb{R}^{p\times 1}$$

Principal Component Analysis (PCA)

- Identify the orientation with largest variance

$$\text{var}\left(\alpha^T \mathbf{X}\right) = \frac{1}{n-1}\sum_{i=1}^{n}(\alpha^T \mathbf{x}_i - \alpha^T \mu)^2 = \frac{1}{n-1}\sum_{i=1}^{n}\alpha^T(\mathbf{x}_i - \mu)(\mathbf{x}_i - \mu)^T\alpha$$

$$= \alpha^T C \alpha$$

(Note that: $\alpha^T(\mathbf{x}_i - \mu) = (\mathbf{x}_i - \mu)^T\alpha$)

where $\mu = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i$

and $C = \frac{1}{n-1}\sum_{i=1}^{n}(\mathbf{x}_i - \mu)(\mathbf{x}_i - \mu)^T$ is the **covariance matrix**

Principal Component Analysis (PCA)

- Identify the orientation with largest variance

Since α is unit, $\alpha^T \alpha = 1$

Based on Lagrange multiplier method, we need to,

$$\arg\max_{\alpha} \left(\alpha^T C \alpha - \lambda \left(\alpha^T \alpha - 1 \right) \right)$$

$$0 = \frac{d\left(\alpha^T C \alpha - \lambda \left(\alpha^T \alpha - 1 \right) \right)}{d\alpha} = 2C\alpha - 2\lambda\alpha \rightarrow C\alpha = \lambda\alpha$$

α is C's eigen-vector

Since,

$$\max\left(\mathrm{var}\left(\alpha^T \mathbf{X} \right) \right) = \max\left(\alpha^T C \alpha \right) = \max\left(\alpha^T \lambda \alpha \right) = \max\left(\lambda \right)$$

Thus,

Principal Component Analysis (PCA)

- Identify the orientation with largest variance

Thus, α_1 should be the eigen-vector of C corresponding to the largest eigen-value of C

What is another orientation α_2 , orthogonal to α_1 , and along which the data can have the second largest variation?

Answer: it is the eigen-vector associated to the second largest eigen-value λ_2 of C and such a variance is λ_2

Principal Component Analysis (PCA)

- Identify the orientation with largest variance

Results: the eigen-vectors of C forms a set of orthogonal basis and they are referred as **Principal Components** of the original data $\mathbf{X}$

You can consider PCs as a set of orthogonal coordinates. Under such a coordinate system, variables are not correlated.

Principal Component Analysis (PCA)

- Express data in PCs

Suppose $\{\alpha_1, \alpha_2, ..., \alpha_p\}$ are PCs derived from $\mathbf{X}$, $\mathbf{X} \in \mathbb{R}^{p \times n}$

Then, a data point $\mathbf{x}_i \in \mathbb{R}^{p \times 1}$ can be linearly represented by $\{\alpha_1, \alpha_2, ..., \alpha_p\}$, and the representation coefficients are

$$\mathbf{c}_i = \begin{pmatrix} \alpha_1^T \\ \alpha_2^T \\ \vdots \\ \alpha_p^T \end{pmatrix} \mathbf{x}_i$$

Actually, $\mathbf{c}_i$ is the coordinates of $\mathbf{x}_i$ in the new coordinate system spanned by $\{\alpha_1, \alpha_2, ..., \alpha_p\}$

Principal Component Analysis (PCA)

- Illustration

x , y
(2.5, 2.4)
(0.5, 0.7)
(2.2, 2.9)
(1.9, 2.2)
(3.1, 3.0)
(2.3, 2.7)
(2.0, 1.6)
(1.0, 1.1)
(1.5, 1.6)
(1.1, 0.9)

$$\mathbf{X} = \begin{pmatrix} 2.5 & 0.5 & 2.2 & 1.9 & 3.1 & 2.3 & 2.0 & 1.0 & 1.5 & 1.1 \\ 2.4 & 0.7 & 2.9 & 2.2 & 3.0 & 2.7 & 1.6 & 1.1 & 1.6 & 0.9 \end{pmatrix}$$

$$\text{cov}(\mathbf{X}) = \begin{pmatrix} 5.549 & 5.539 \\ 5.539 & 6.449 \end{pmatrix}$$

Eigen-values = $11.5562, 0.4418$

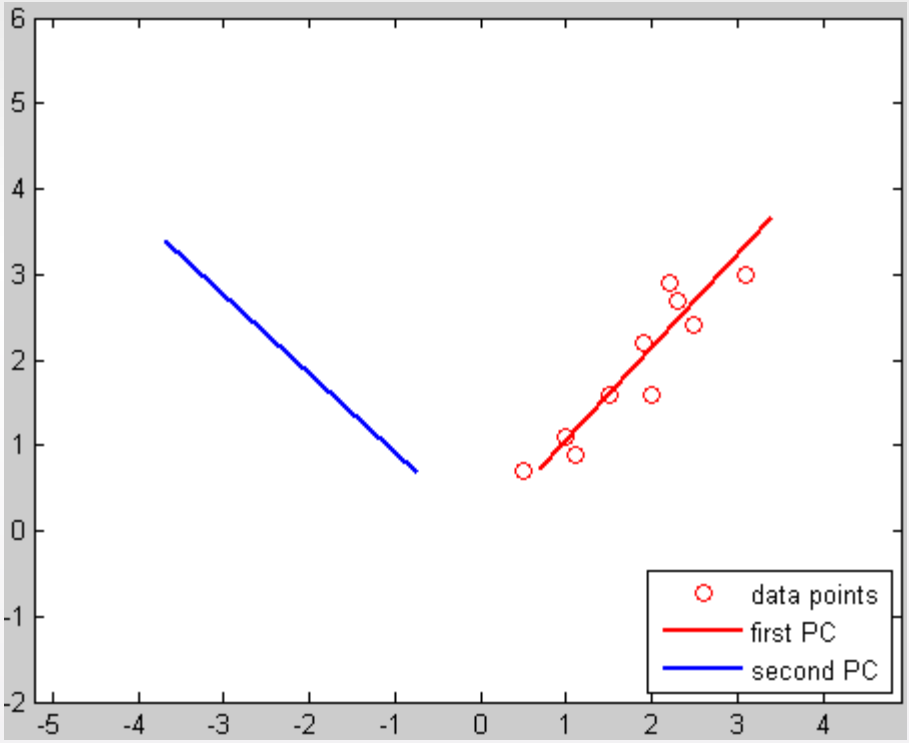
Corresponding eigen-vectors:

$$\alpha_1 = \begin{pmatrix} 0.6779 \\ 0.7352 \end{pmatrix}$$

$$\alpha_2 = \begin{pmatrix} -0.7352 \\ 0.6779 \end{pmatrix}$$

Principal Component Analysis (PCA)

- Illustration



Principal Component Analysis (PCA)

- Illustration

Coordinates of the data points in the new coordinate system

$$newC = \begin{pmatrix} \alpha_1^T \\ \alpha_2^T \end{pmatrix} \mathbf{X}$$

$$\begin{pmatrix} 0.6779 & 0.7352 \\ -0.7352 & 0.6779 \end{pmatrix} \mathbf{X}$$

$$= \begin{pmatrix} 3.459 & 0.854 & 3.623 & 2.905 & 4.307 & 3.544 & 2.532 & 1.487 & 2.193 & 1.407 \\ -0.211 & 0.107 & 0.348 & 0.094 & -0.245 & 0.139 & -0.386 & 0.011 & -0.018 & -0.199 \end{pmatrix}$$

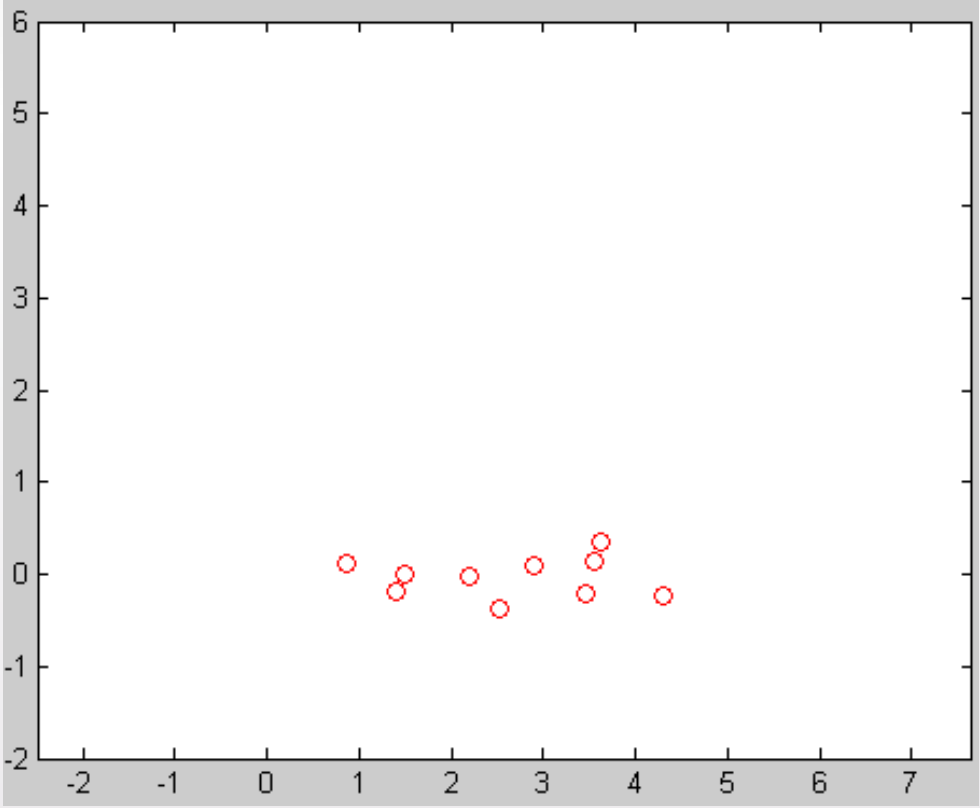
Principal Component Analysis (PCA)

- Illustration

Coordinates of the data points in the new coordinate system

Draw $newC$ on the plot

In such a new system, two variables are linearly independent!



Principal Component Analysis (PCA)

- Data dimension reduction with PCA

Suppose $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^{n}, \mathbf{x}_i \in \mathbb{R}^{p\times 1}$, $\{\alpha_i\}_{i=1}^{p}, \alpha_i \in \mathbb{R}^{p\times 1}$ are the PCs

If all of $\{\alpha_i\}_{i=1}^{p}$ are used, $\mathbf{c}_i = \begin{pmatrix} \alpha_1^T \\ \alpha_2^T \\ \vdots \\ \alpha_p^T \end{pmatrix} \mathbf{x}_i$ is still p-dimensional

If only $\{\alpha_i\}_{i=1}^{m}, m < p$ are used, $\mathbf{c}_i$ will be m-dimensional

That is, the dimension of the data is reduced!

Principal Component Analysis (PCA)

- Illustration

Coordinates of the data points in the new coordinate system

$$newC = \begin{pmatrix} 0.6779 & 0.7352 \\ -0.7352 & 0.6779 \end{pmatrix} \mathbf{X}$$

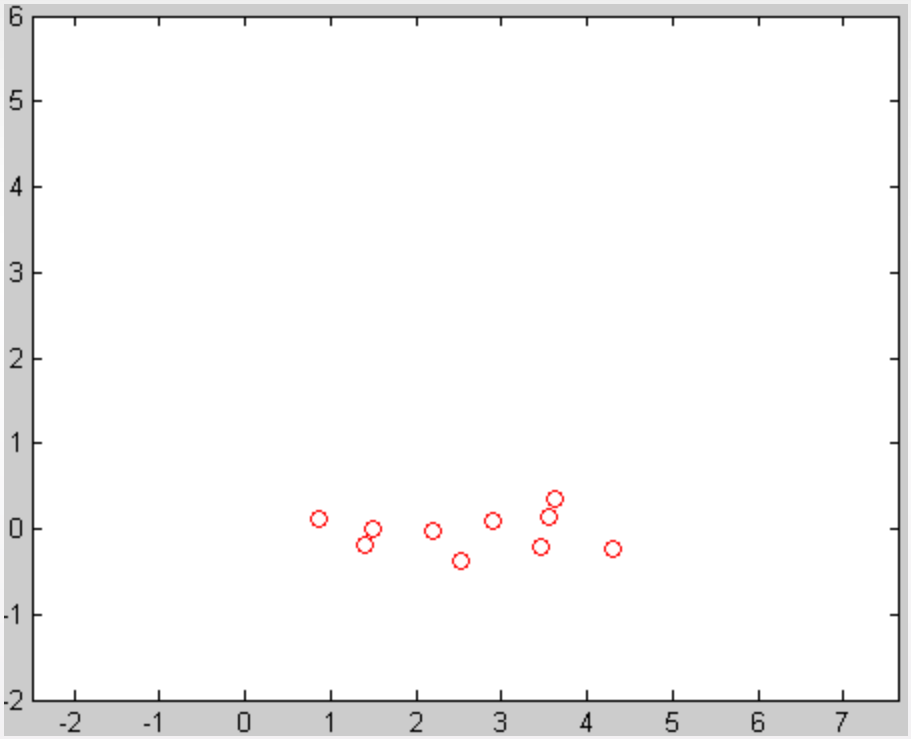
If only the first PC (corresponds to the largest eigen-value) is remained

$$newC = \begin{pmatrix} 0.6779 & 0.7352 \end{pmatrix} \mathbf{X}$$

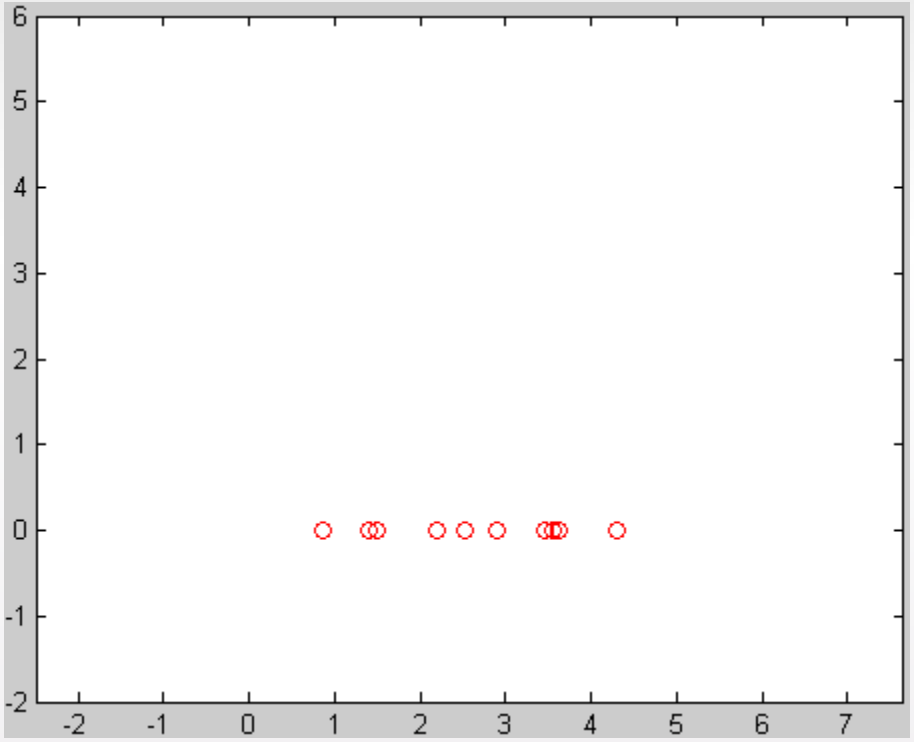
$$= \begin{pmatrix} 3.459 & 0.854 & 3.623 & 2.905 & 4.307 & 3.544 & 2.532 & 1.487 & 2.193 & 1.407 \end{pmatrix}$$

Principal Component Analysis (PCA)

- Illustration



All PCs are used



Only 1 PC is used

Dimension reduction!

Principal Component Analysis (PCA)

- Illustration

If only the first PC (corresponds to the largest eigen-value) is remained

$$newC = \begin{pmatrix} 0.6779 & 0.7352 \end{pmatrix} \mathbf{X}$$

$$= \begin{pmatrix} 3.459 & 0.854 & 3.623 & 2.905 & 4.307 & 3.544 & 2.532 & 1.487 & 2.193 & 1.407 \end{pmatrix}$$

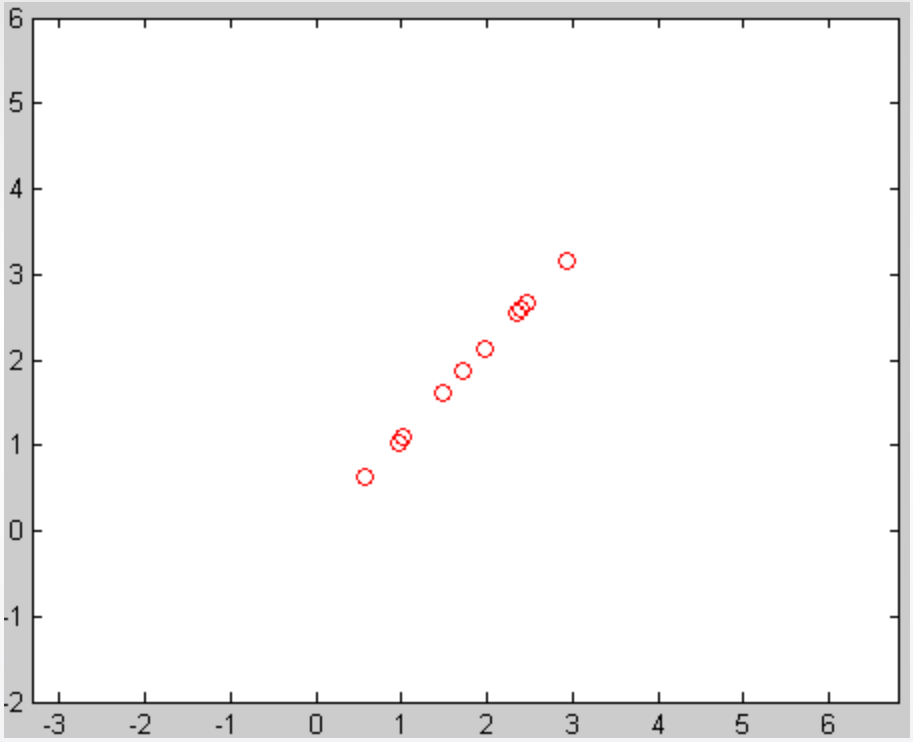
How to recover $newC$ to the original space? Easy

$$\begin{pmatrix} 0.6779 & 0.7352 \end{pmatrix}^T newC$$

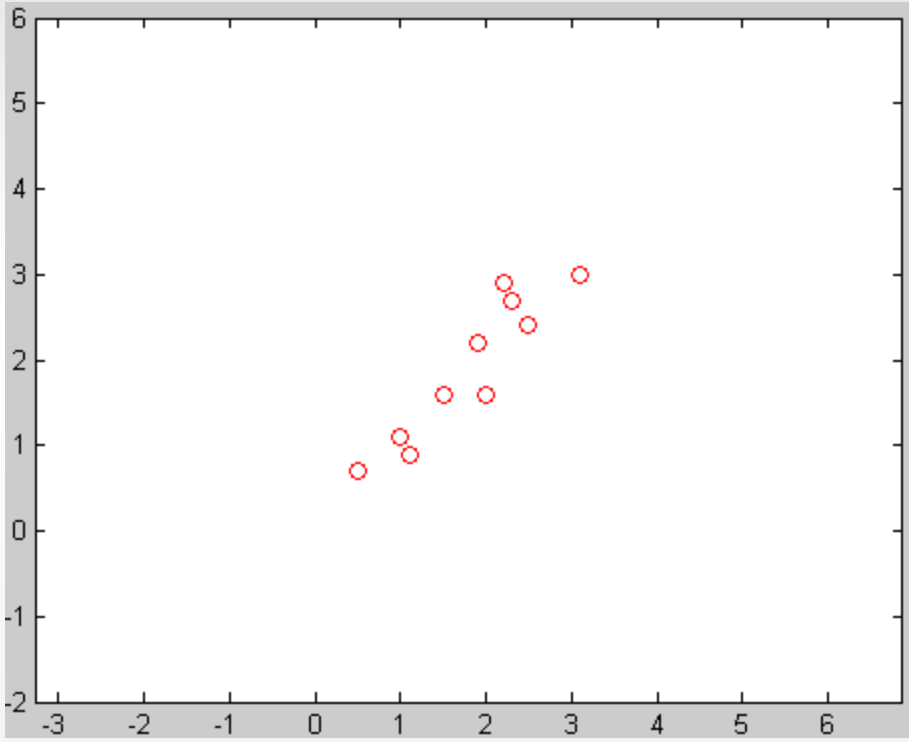
$$= \begin{pmatrix} 0.6779 \\ 0.7352 \end{pmatrix} \begin{pmatrix} 3.459 & 0.854 & 3.623 & 2.905 & 4.307 & 3.544 & 2.532 & 1.487 & 2.193 & 1.407 \end{pmatrix}$$

Principal Component Analysis (PCA)

- Illustration



Data recovered if only 1 PC used



Original data

Outline

- Introduction
- Facial expression recognition
  - Appearance-based vs. model-based
- Active appearance model (AAM)
  - Pre-requisite
  - Principle component analysis
  - Procrustes analysis
  - ASM
  - Delaunay triangulation
  - AAM

Procrustes analysis

- Who is Procrustes?

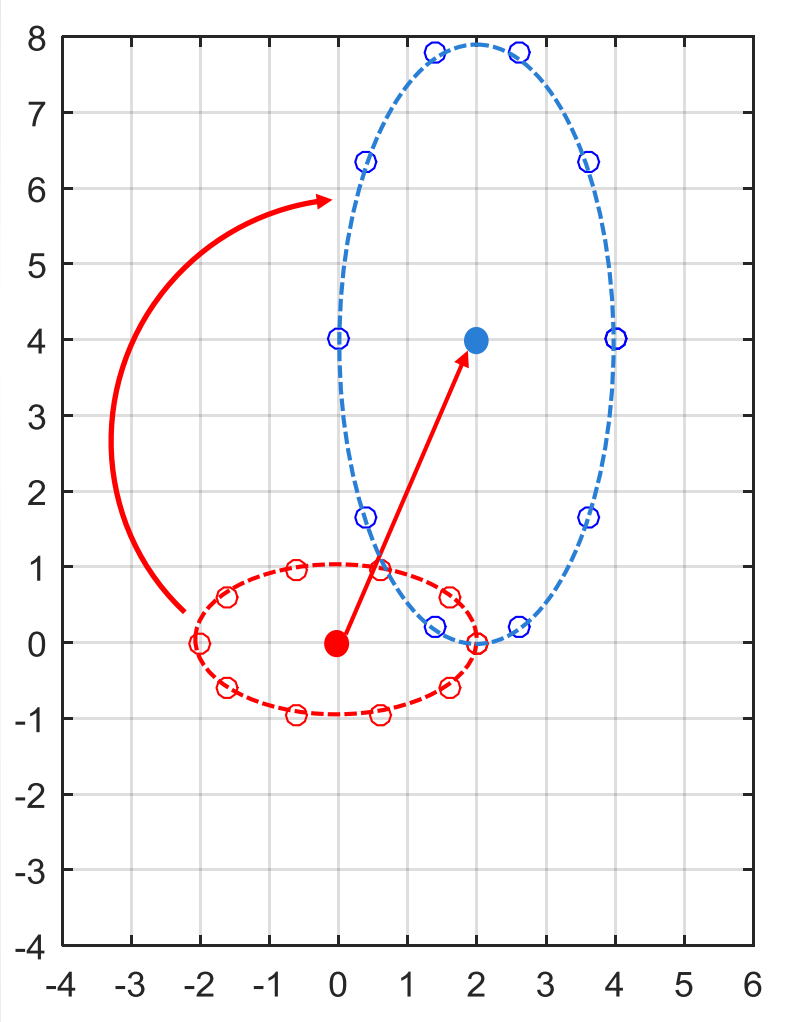
Procrustes analysis

- Problem:
  - Suppose we have two sets of point correspondence pairs (m_1, m_2,…, m_N) and (n_1, n_2,…, n_N).
  - We want to find a scale factor s, an orthogonal matrix R and a vector t so that:

$$\Sigma^2 = \sum_{i=1}^{N} \left\| m_i - \left(sR(n_i) + T \right) \right\|^2 \qquad (1)$$

Procrustes analysis

- In 2D space:



Procrustes analysis

- How to compute R and T?
  - We assume that there is a similarity transform between point sets $\{m_i\}_{i=1}^{N}$ and $\{n_i\}_{i=1}^{N}$
  - *Find s, R and T to minimize*

$$\Sigma^2 = \sum_{i=1}^{N} e_i^2 = \sum_{i=1}^{N} \left\| m_i - \left(sR(n_i) + T \right) \right\|^2 \qquad (1)$$

  - Let

$$\bar{m} = \frac{1}{N}\sum_{i=1}^{N} m_i, \bar{n} = \frac{1}{N}\sum_{i=1}^{N} n_i, m_i' = m_i - \bar{m}, n_i' = n_i - \bar{n}$$

  - Note that $\sum_{i=1}^{N} m_i' = \mathbf{0}, \sum_{i=1}^{N} n_i' = \mathbf{0}$

Procrustes analysis

Then:

$$e_i = m_i - sR(n_i) - T = m_i^{'} + \overline{m} - sR(n_i^{'} + \overline{n}) - T = m_i^{'} + \overline{m} - sR(n_i^{'}) - sR(\overline{n}) - T$$

$$= m_i^{'} - sR(n_i^{'}) - \left(T - \overline{m} + sR(\overline{n})\right) = m_i^{'} - sR(n_i^{'}) - e_0$$

$e_0 = T - \overline{m} + sR(\overline{n})$ is independent from $\{m_i^{'}, n_i^{'}\}$

(1) can be rewritten as:

$$\Sigma^2 = \sum_{i=1}^{N} e_i^2 = \sum_{i=1}^{N} \left\| m_i^{'} - sR(n_i^{'}) - e_0 \right\|^2 = \sum_{i=1}^{N} \left\| m_i^{'} - sR(n_i^{'}) \right\|^2 - 2e_0 \cdot \sum_{i=1}^{N} \left(m_i^{'} - sR(n_i^{'}) \right) + Ne_0^2$$

$$= \sum_{i=1}^{N} \left\| m_i^{'} - sR(n_i^{'}) \right\|^2 - 2e_0 \cdot \sum_{i=1}^{N} \left(m_i^{'} \right) + 2e_0 \cdot \sum_{i=1}^{N} \left(sR(n_i^{'}) \right) + Ne_0^2$$

$$= \sum_{i=1}^{N} \left\| m_i^{'} - sR(n_i^{'}) \right\|^2 + Ne_0^2$$

Variables are separated and can be minimized separately.

$e_0^2 = 0 \Leftrightarrow T = \overline{m} - sR(\overline{n})$ If we have *s* and *R*, *T* can be determined.

Procrustes analysis

Then the problem simplifies to: how to minimize

$$\Sigma^2 = \sum_{i=1}^{N} \left\| m_i^{'} - sR(n_i^{'}) \right\|^2$$

Consider its geometric meaning here.

We revise the error item as a symmetrical one:

$$\frac{1}{s}\Sigma^2 = \sum_{i=1}^{N} \left\| \frac{1}{\sqrt{s}} m_i^{'} - \sqrt{s} R(n_i^{'}) \right\|^2 = \frac{1}{s}\sum_{i=1}^{N} \left\| m_i^{'} \right\|^2 - 2\sum_{i=1}^{N} m_i^{'} \cdot R(n_i^{'}) + s\sum_{i=1}^{N} \left\| R(n_i^{'}) \right\|^2$$

$$= \frac{1}{s}\sum_{i=1}^{N} \left\| m_i^{'} \right\|^2 - 2\sum_{i=1}^{N} m_i^{'} \cdot R(n_i^{'}) + s\sum_{i=1}^{N} \left\| n_i^{'} \right\|^2$$

P D Q

Variables are separated.

$$\frac{1}{s}\Sigma^2 = \frac{1}{s}P - 2D + sQ = \left(\sqrt{s}\sqrt{Q} - \frac{1}{\sqrt{s}}\sqrt{P} \right)^2 + 2(\sqrt{PQ} - D)$$

Thus,

Procrustes analysis

$$\left(\sqrt{s}\sqrt{Q}-\frac{1}{\sqrt{s}}\sqrt{P}\right)^2=0\Leftrightarrow s=\sqrt{\frac{P}{Q}}=\sqrt{\frac{\sum_{i=1}^{N}\left\|m_i^{'}\right\|^2}{\sum_{i=1}^{N}\left\|n_i^{'}\right\|^2}}$$

Determined!.

Then the problem simplifies to: how to maximize

$$D=\sum_{i=1}^{N}m_i^{'}\cdot R(n_i^{'})$$

Note that: D is a real number.

$$D=\sum_{i=1}^{N}m_i^{'}\cdot Rn_i^{'}=\sum_{i=1}^{N}\left(m_i^{'}\right)^T Rn_i^{'}=trace\left(\sum_{i=1}^{N}Rn_i^{'}\left(m_i^{'}\right)^T\right)=trace\left(RH\right)$$

$$H\equiv\sum_{i=1}^{N}n_i^{'}\left(m_i^{'}\right)^T$$

Now we are looking for an orthogonal matrix *R* to maximize the trace of *RH.*

Procrustes analysis

Lemma

For any positive semi-definite matrix C and any orthogonal matrix B:

$$trace(C) \geq trace(BC)$$

Proof:

From the positive definite property of *C*, $\exists A, C = AA^T$

where A is a non-singular matrix.

Let a_i be the *i*th column of A. Then

$$trace(BAA^T) = trace(A^T BA) = \sum_i a_i^T (Ba_i)$$

According to Schwarz inequality: $|< x, y >| \leq \|x\| \|y\|$

$$a_i^T (Ba_i) \leq \|a_i^T\| \|Ba_i\| = \sqrt{(a_i^T a_i)(a_i^T B^T Ba_i)} = a_i^T a_i$$

Hence,

$trace(BAA^T) \leq \sum_i a_i^T a_i = trace(AA^T)$ that is, $trace(BC) \leq trace(C)$

Procrustes analysis

Consider the SVD of $H \equiv \sum_{i=1}^{N} n_i' \left(m_i'\right)^T \quad H = U \Lambda V^T$

According to the property of SVD, U and V are orthogonal matrices, and Λ is a diagonal matrix with nonnegative elements.

Now let $X = VU^T$

Note that: X is orthogonal.

We have $XH = VU^T U \Lambda V^T = V \Lambda V^T$ which is positive semi-definite.

Thus, from the lemma, we know: for any orthogonal matrix *B*

$$trace(XH) \geq trace(BXH)$$

for any orthogonal matrix Ψ

$$trace(XH) \geq trace(\Psi H)$$

It's time to go back to our objective now… *R should be X*

Procrustes analysis

Now, *s*, *R* and *T* are all determined.

$$H \equiv \sum_{i=1}^{N} n_i^{'} \left(m_i^{'} \right)^T = U \Lambda V^T$$

$$R = VU^T \qquad s = \sqrt{\frac{\sum_{i=1}^{N} \left\| m_i^{'} \right\|^2}{\sum_{i=1}^{N} \left\| n_i^{'} \right\|^2}} \qquad T = \overline{m} - sR(\overline{n})$$

Procrustes analysis

- Problem: Given two configuration matrices **X** and **Y** with the same dimensions, find rotation and translation that would bring **Y** as close as possible to **X**.

1) Find the centroids (mean values of columns) of **X** and **Y**. Call them $\bar{\mathbf{x}}$ and $\bar{\mathbf{y}}$.
2) Remove from each column corresponding mean. Call the new matrices $\mathbf{X}_{new}$ and $\mathbf{Y}_{new}$
3) Find $\mathbf{Y}_{new}{}^{T}\mathbf{X}_{new}$. Find the SVD of this matrix $\mathbf{Y}_{new}{}^{T}\mathbf{X}_{new} = \mathbf{UDV}^{T}$
4) Find the rotation matrix $R = \mathbf{UV}^{T}$. Find scale factor $s = \sqrt{\frac{\sum_{i=1}^{N} \|\mathbf{X}_{new}\|^2}{\sum_{i=1}^{N} \|\mathbf{Y}_{new}\|^2}}$
5) Find the translation vector $T = \bar{\mathbf{x}} - sR\bar{\mathbf{y}}$.

Some modifications

- Sometimes it is necessary to weight some variables down and others up. In these cases Procrustes analysis can be performed using weights. We want to minimize the function:

$$M^2 = tr(\mathbf{W}(\mathbf{X} - \mathbf{AY})(\mathbf{X} - \mathbf{AY})^T)$$

- This modification can be taken into account if we find SVD of $\mathbf{Y}^T\mathbf{WX}$ instead of $\mathbf{Y}^T\mathbf{X}$

Outline

- Introduction
- Facial expression recognition
  - Appearance-based vs. model-based
- Active appearance model (AAM)
  - Pre-requisite
  - Principle component analysis
  - Procrustes analysis
  - ASM
  - Delaunay triangulation
  - AAM

Active shape model (ASM)

- Collect training samples
  - 30 neutral faces
  - 30 smile faces

ASM

- Add landmarks
  - 26 points each face
  - $\mathbf{X} = [\mathbf{x}_1\ \mathbf{x}_2\ \ldots\ \mathbf{x}_N]$; $N = 60$.

ASM

- Align training faces together using Procrustes analysis

  Transform $\mathbf{x}_i$ to $\mathbf{x}_j$: Rotation (R_i), Scale (s_i), Transformation (T_i)

$$\min Z_i * Z_i^T$$

$$Z_i = \mathbf{x}_j - (s * R * (\mathbf{x}_i) - T)$$

  Consider a weight matrix W:

  D_{kl} represents the distance between the point k and l in one image and V_{Dkl} represents the variance of D_{kl} in different images

$$w_k = (\sum_{l=1}^{N} V_{D_{kl}})^{-1} \qquad W = \begin{pmatrix} w_1 & & \\ & \ddots & \\ & & w_m \end{pmatrix}$$

  m is the dimension of $\mathbf{x}_i$

ASM

- We want to minimize

$$E_i = \min Z_i * W * Z_i^T$$

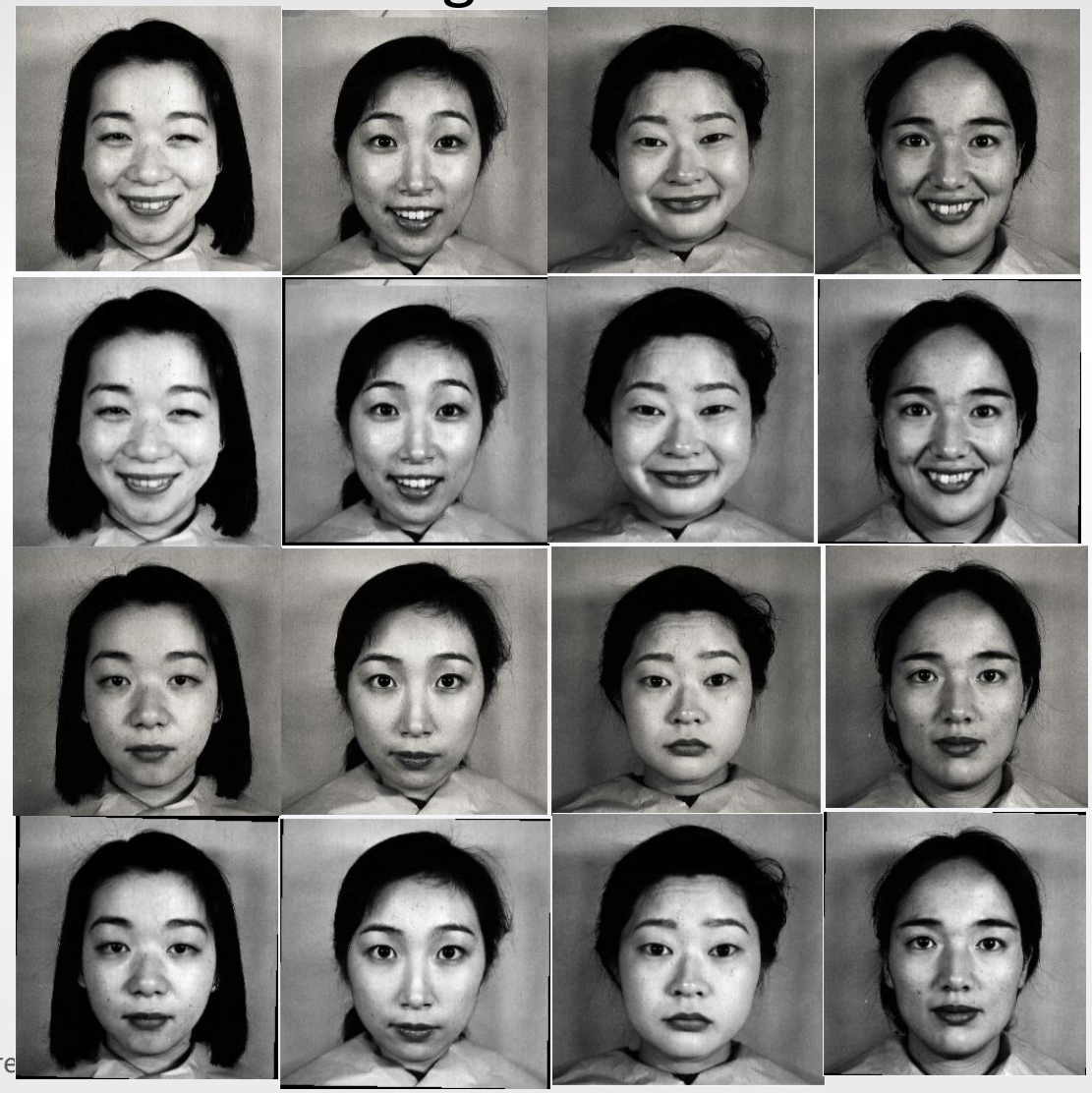
1) Find the centroids (mean values of columns) of $\mathbf{x}_i$ and $\mathbf{x}_j$.
2) Remove from each column corresponding mean.
3) Find the SVD of this matrix $\mathbf{x}_{i\_new} * W * \mathbf{x}_{j\_new}^T = \mathbf{UDV}^T$
4) Find the rotation matrix $\mathbf{A} = \mathbf{UV}^T$. Find the scale factor and translation vector as shown in page 59.

ASM

- Align training faces using Procrustes analysis
- Steps:
  1. Align the other faces with the first face
  2. Compute the mean face $\bar{\mathbf{x}} = \sum_{i=1}^{N} \mathbf{x}_i$
  3. Align training faces with the mean face
  4. Repeat step 2 and 3 until the discrepancies between training faces and the mean face won't change

ASM

- Faces after alignment



ASM

- Construct models of faces
  - Compute mean face $\bar{\mathbf{x}} = \sum_{i=1}^{N} \mathbf{x}_i$
  - Calculate covariance matrix

$$S = \frac{1}{N} \sum_{i=1}^{N} (\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^T$$

  - Find its eigenvalues ($\lambda_1, \lambda_2, \ldots, \lambda_m$) and eigenvectors $\mathbf{P} = (p_1, p_2, \ldots, p_m)$
  - Choose the first t largest eigenvalues

$$\sum_{i=1}^{t} \lambda_i >= f_v V_T, V_T = \sum_t \lambda_i$$

Usually $f_v = 0.98$

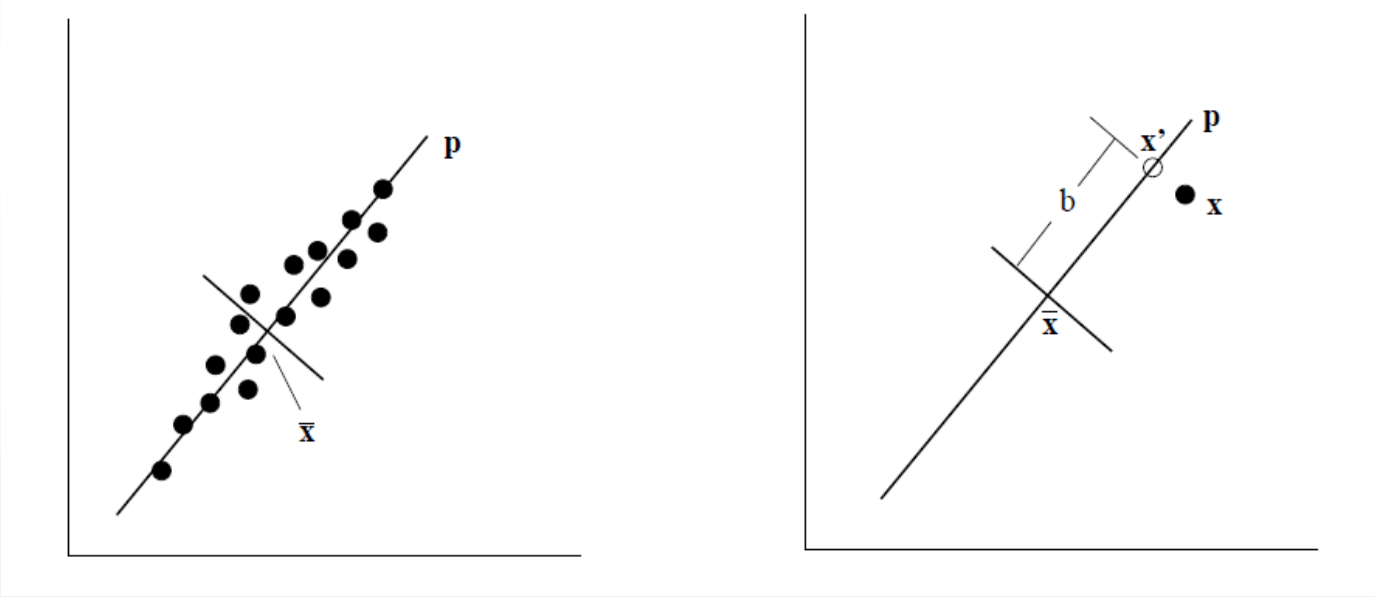
Dimension reduction: 26*2→27

ASM

- We can approximate a training face **x**

$$\mathbf{x} \approx \bar{\mathbf{x}} + \mathbf{P}\mathbf{b}$$

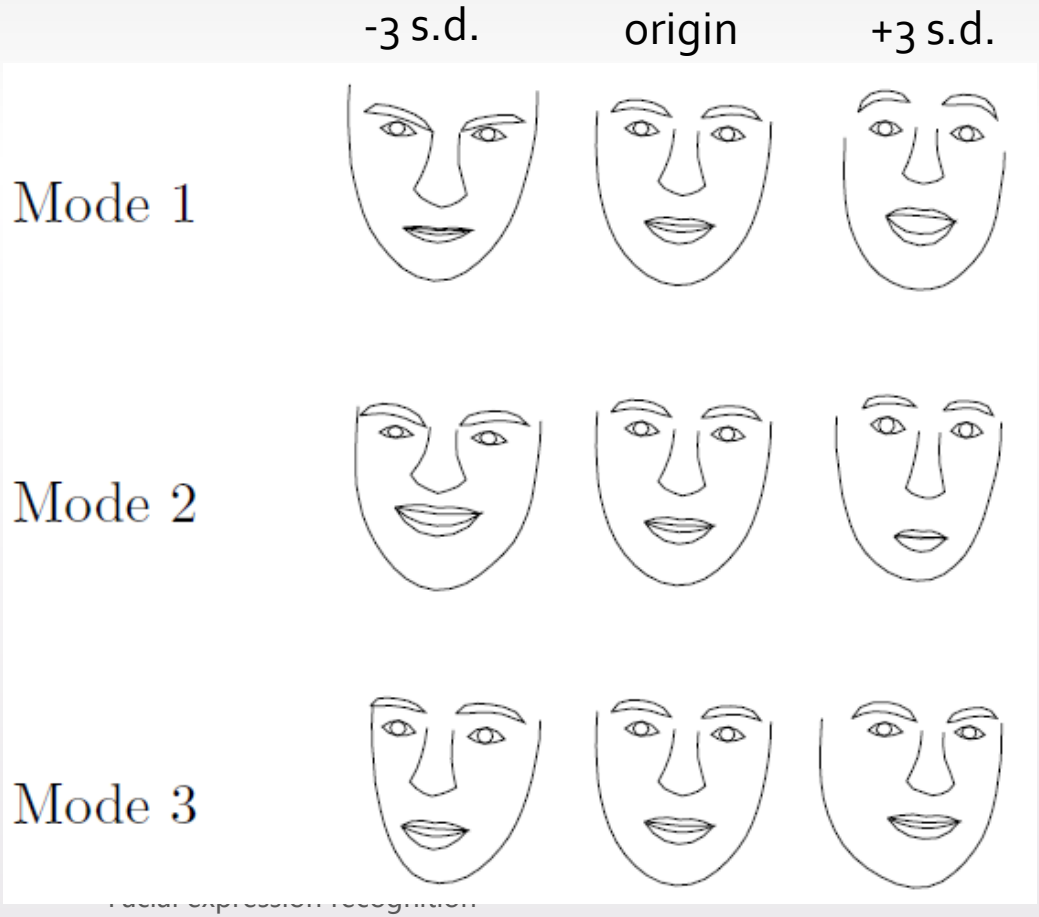
$$\mathbf{b} = \mathbf{P}^T(\mathbf{x} - \bar{\mathbf{x}})$$

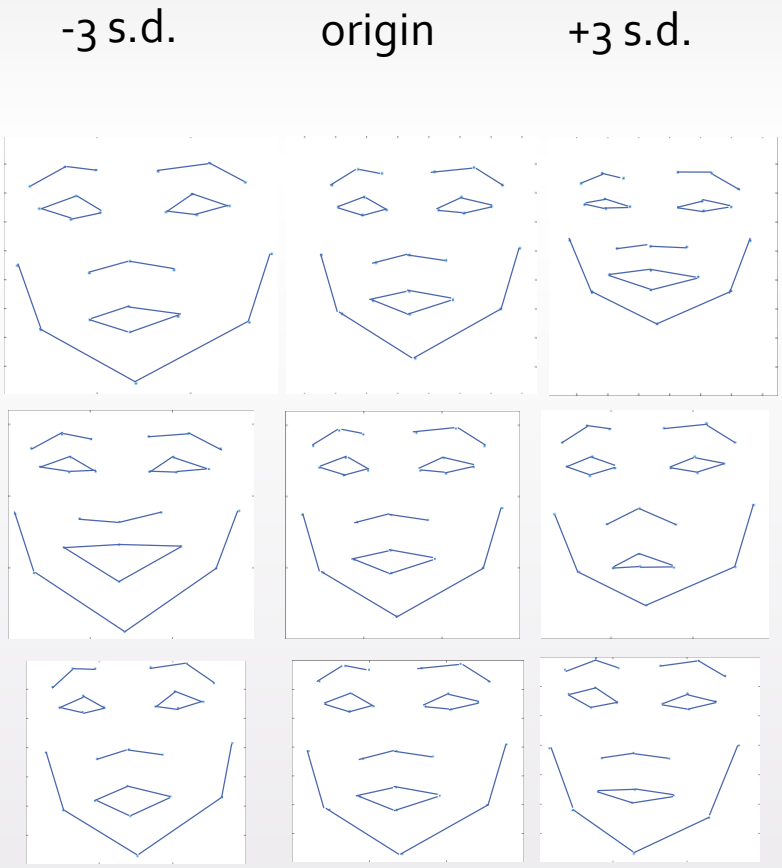


- b_i is called the i^{th} mode of the model
- constraint: $|b_i| <= 3\sqrt{\lambda_i}$

ASM

- Effect of varying the first three shape parameters in turn between ±3 s.d. from the mean value





ASM

- Active Shape Model Algorithm

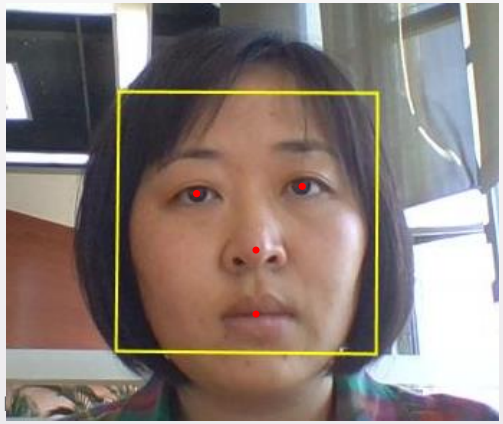
1. Examine a region of the image around each point x_i to find the best nearby match for the point x_i'
2. Update the parameters (T, s, R, $\mathbf{b}$) to best fit the new found points $\mathbf{x}'$
3. Apply constraints to the parameters, $\mathbf{b}$, to ensure plausible shapes (e.g. limit so $|b_i| < 3\sqrt{\lambda_i}$).
4. Repeat until convergence.

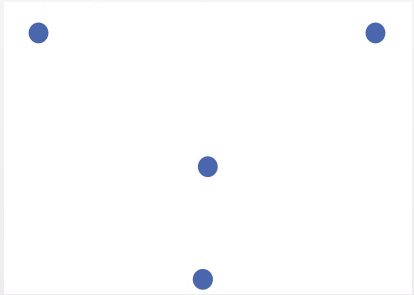
Q1: How to find corresponding points in a new image?

ASM

- Find the initial corresponding points
  - Detect face using Viola-Jones face detector
  - Estimate positions of eyes centers, nose center, and mouse center
  - Align the corresponding positions on the mean face to the estimated positions of eyes centers, nose center, and mouse center (s_{init}, R_{init}, T_{init}) on the new face

$$\mathbf{x}_{init} = s_{init} * R_{init} * \bar{\mathbf{x}} + T_{init}$$

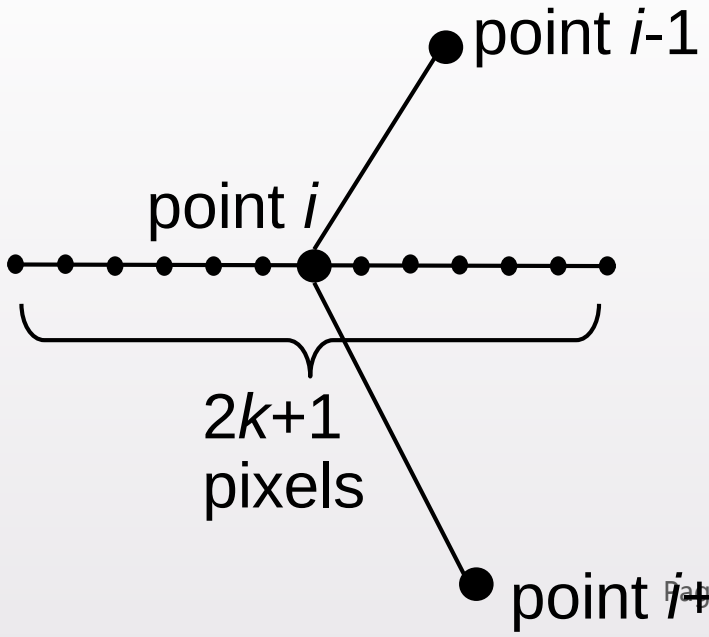




ASM

- Construct local features for each point in the training samples
  - For a given point, we sample along a profile k pixels either side of the model point in the i^{th} training image.
  - Instead of sampling absolute grey-level values, we sample derivatives and put them in a vector $\boldsymbol{g}_i$
  - Normalize the sample:

$$\mathbf{g}_i = \frac{\mathbf{g}_i}{\sum_j \mathbf{g}_{ij}}$$



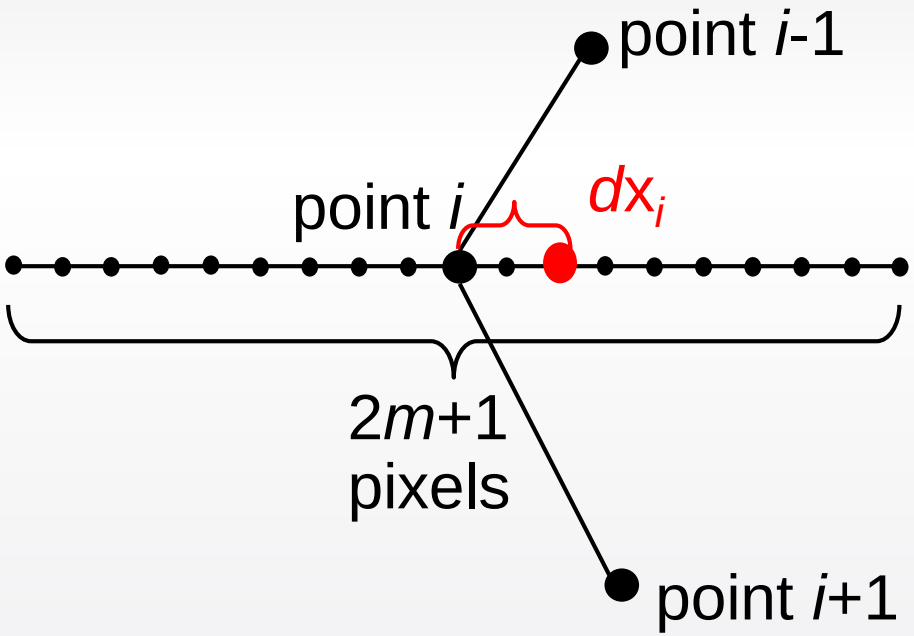
ASM

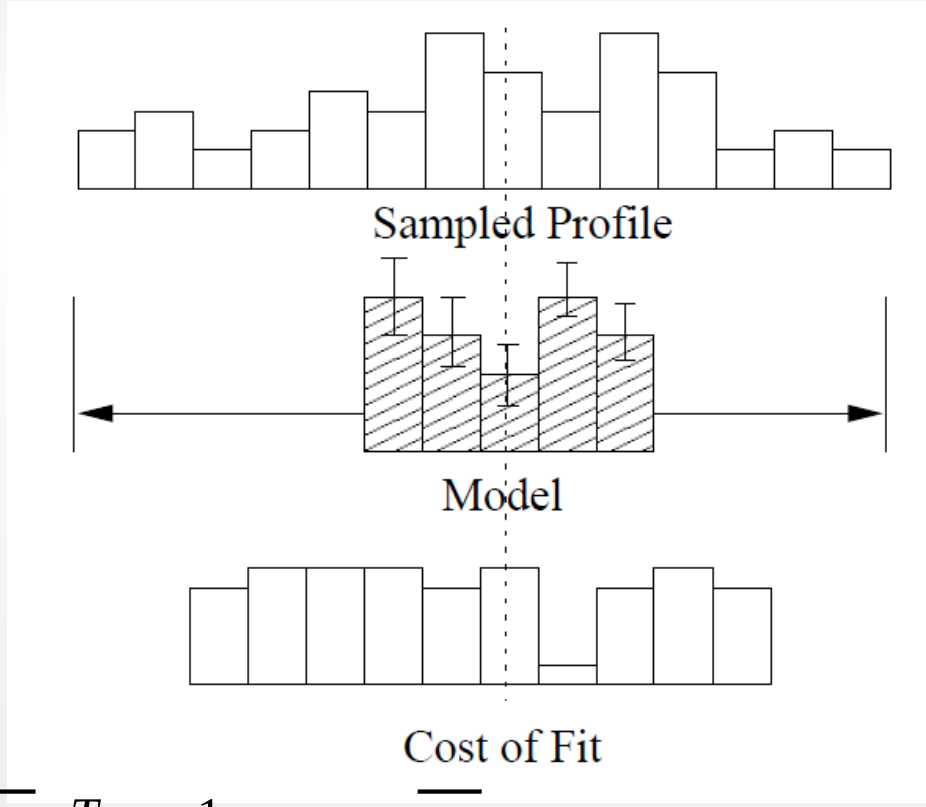
- For each training image, we can get a set of normalized samples $\{\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_N\}$ for point *i*
- We assume that these $\mathbf{g}_i$ are distributed as a multivariate Gaussian, and estimate their mean $\overline{\boldsymbol{g_i}}$ and covariance S_{g_i}.
- This gives a statistical model for the grey-level profile about the point *i*
- Given a new sample $\mathbf{g}_s$, the distance of $\mathbf{g}_s$ to $\overline{\boldsymbol{g_i}}$ can be computed using the Mahalanobis distance

$$d(\mathbf{g}_s, \overline{\mathbf{g}_i}) = (\mathbf{g}_s - \overline{\mathbf{g}_i})^T S_{g_i}^{-1} (\mathbf{g}_s - \overline{\mathbf{g}_i})$$

ASM

- During search we sample a profile *m* pixels from either side of each initial point (*m* > *k*) on the new face.





$$\min_{\mathbf{g}_s} d(\mathbf{g}_s, \overline{\mathbf{g}_i}) = (\mathbf{g}_s - \overline{\mathbf{g}_i})^T S_{g_i}^{-1} (\mathbf{g}_s - \overline{\mathbf{g}_i})$$

ASM

- Some constraints on dx_i
  - $|dx_i| = 0$ if $|d_{best}| <= \delta$
  - $|dx_i| = 0.5 d_{best}$ if $\delta <= |d_{best}| <= d_{max}$
  - $|dx_i| = 0.5 d_{max}$ if $|d_{best}| > d_{max}$

ASM

- Apply one iteration of ASM algorithm

1. Examine a region of the image around each point x_i to find the best nearby match for the point x_i':

$$x_i' = x_i + dx_i$$

2. Update the parameters (T, s, R, $\mathbf{b}$) to best fit the new found points $\mathbf{x}'$
3. Apply constraints to the parameters, $\mathbf{b}$, to ensure plausible shapes (e.g. limit so $|b_i| < 3\sqrt{\lambda_i}$).
4. Repeat until convergence.

Q2: How to find T, s, R, and $\mathbf{b}$ for $\mathbf{x}'$?

ASM

- Suppose we want to match a model **x** to a new set of image points **y**
- We wish to find (*T*, *s*, *R*, **b**) that can minimize

$$\min |\mathbf{y} - s * R(\mathbf{x}) - T|^2$$

$$= \min |\mathbf{y} - s * R(\bar{\mathbf{x}} + \mathbf{Pb}) - T|^2$$

ASM

- A simple iterative approach to achieving the minimum
  1. Initialize the shape parameters, **b**, to zero (the mean shape).
  2. Generate the model point positions using $\mathbf{x} = \bar{\mathbf{x}} + \mathbf{Pb}$
  3. Find the pose parameters (s, R, T) which best align the model points **x** to the current found points **y**

$$\min \left| \mathbf{y} - s * R(\mathbf{x}) - T \right|^2$$

  4. Project **y** into the model co-ordinate frame by inverting the transformation :

$$\mathbf{y}' = R^{-1}(\mathbf{y} + T) / s$$

  5. Project **y** into the tangent plane to $\bar{\mathbf{x}}$ by scaling: $\mathbf{y}'' = \mathbf{y}'/(\mathbf{y}' \cdot \bar{\mathbf{x}})$.
  6. Update the model parameters to match to **y**''

$$\mathbf{b} = \mathbf{P}^T (\mathbf{y}'' - \bar{\mathbf{x}})$$

  7. If not converged, return to step 2.

ASM

- Apply one iteration of ASM algorithm

1. Examine a region of the image around each point x_i to find the best nearby match for the point x_i':

$$x_i' = x_i + dx_i$$

2. Update the parameters (T, s, R, $\mathbf{b}$) to best fit the new found points $\mathbf{x}'$ using the algorithm in page 79

3. Apply constraints to the parameters, $\mathbf{b}$, to ensure plausible shapes (e.g. limit so $|b_i| < 3\sqrt{\lambda_i}$).

4. Repeat until convergence.

ASM

- So now we have a vector of **b** for the new face
- Classify facial expression using **b**

Classification

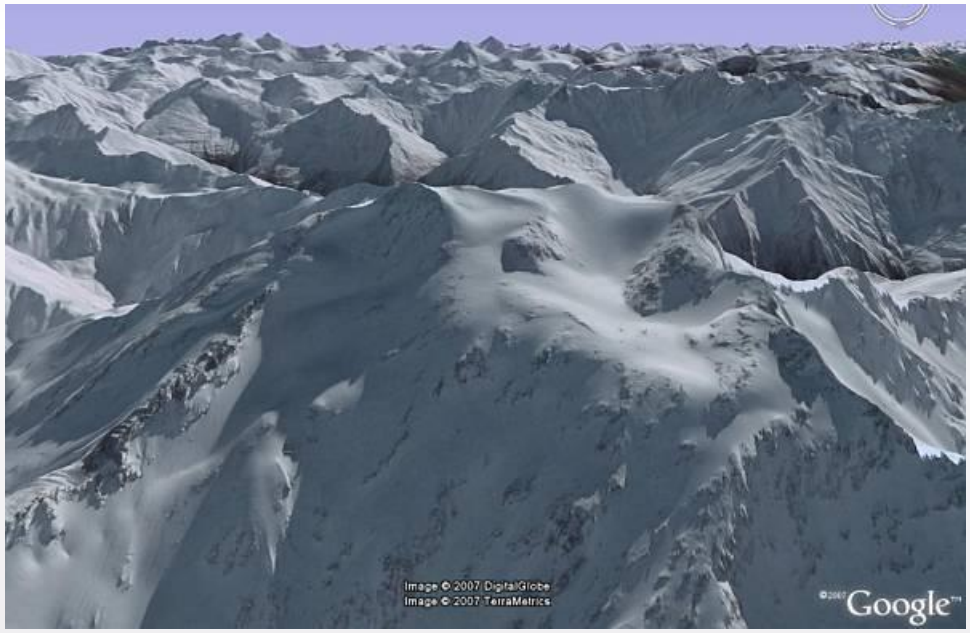
- Training
  - Compute **b** for each training faces
  - Training a classifier (e.g. SVM, Neural Network) using $\{\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_N\}$ and the corresponding labels
- Test
  - Using the trained classifier to classify a new mode vector $\mathbf{b}_{new}$ of a new face

Outline

- Introduction
- Facial expression recognition
  - Appearance-based vs. model-based
- Active appearance model (AAM)
  - Pre-requisite
  - Principle component analysis
  - Procrustes analysis
  - ASM
  - Delaunay triangulation
  - AAM

Delaunay triangulation

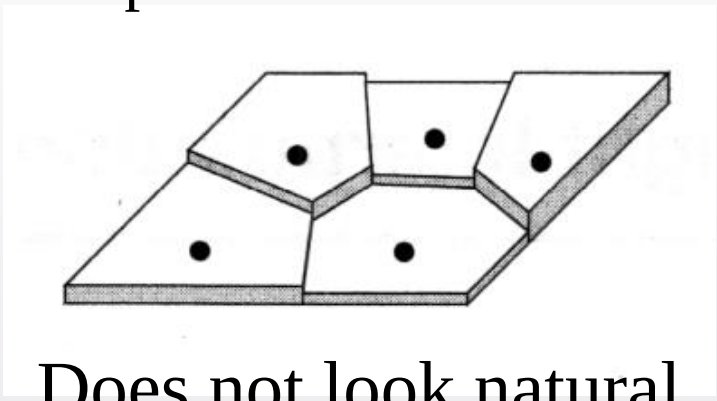
- Terrains by interpolation
- To build a model of the terrain surface, we can start with a number of sample points where we know the height.



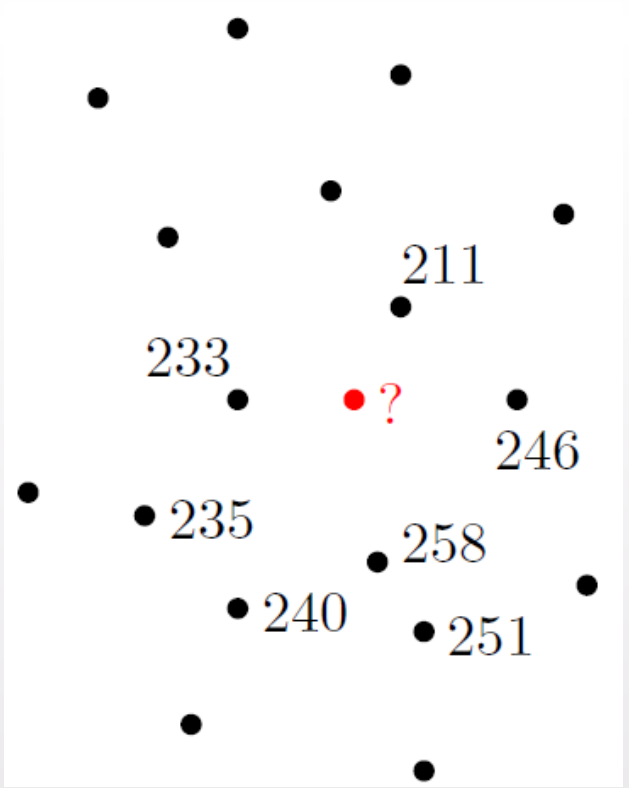
Delaunay triangulation

- How do we interpolate the height at other points?
  - Height $f(p)$ defined at each point p in P
  - How can we most naturally approximate height of points not in P?

Let $f(p)$ = height of nearest point for points not in A

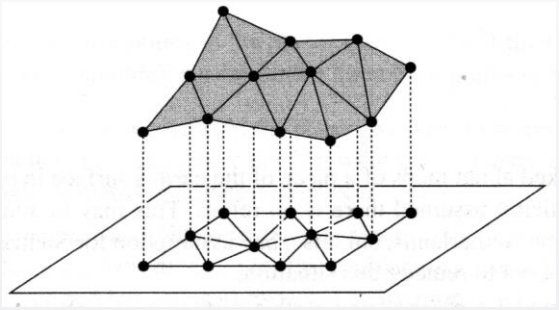


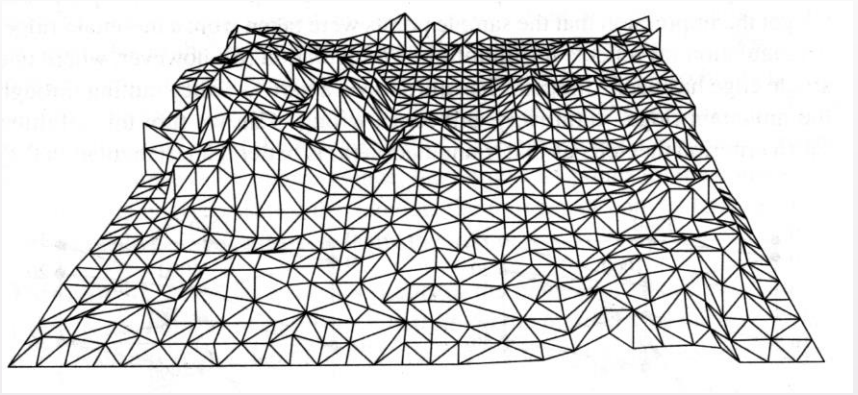
Does not look natural



Delaunay triangulation

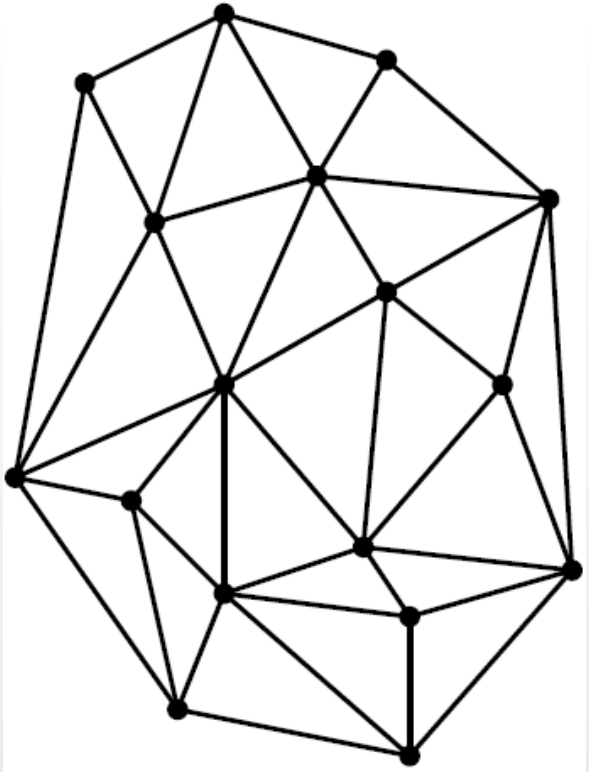
- Better option: triangulation
  - Determine a *triangulation* of P in R^2, then raise points to desired height
  - Triangulation: planar subdivision whose bounded faces are triangles with vertices from P





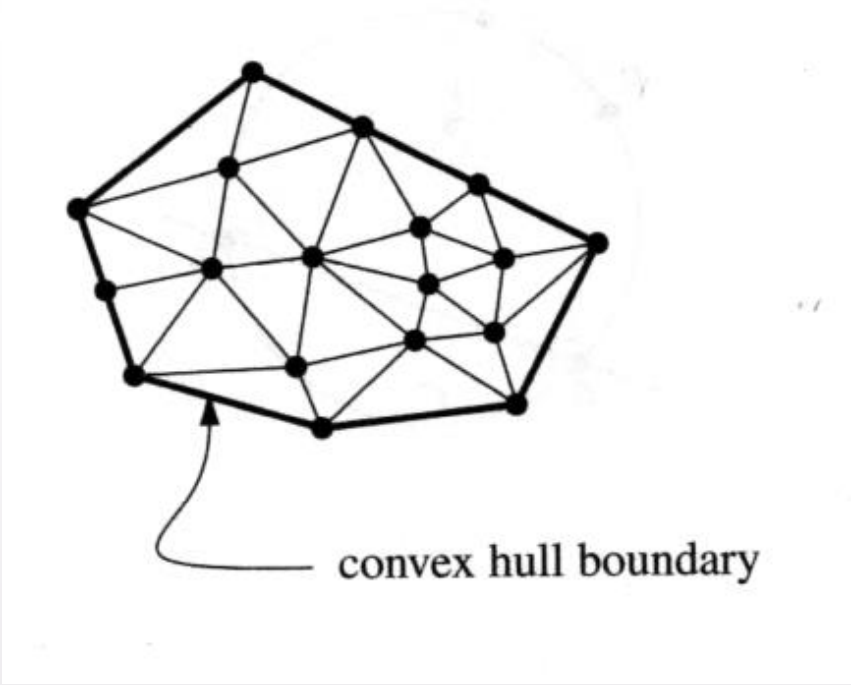
Delaunay triangulation

- Formal definition
  - Let $P = \{p_1,\ldots,p_n\}$ be a point set.
  - *Maximal planar subdivision*: a subdivision S such that no edge connecting two vertices can be added to S without destroying its planarity
  - A triangulation of P is a maximal planar subdivision with vertex set P.



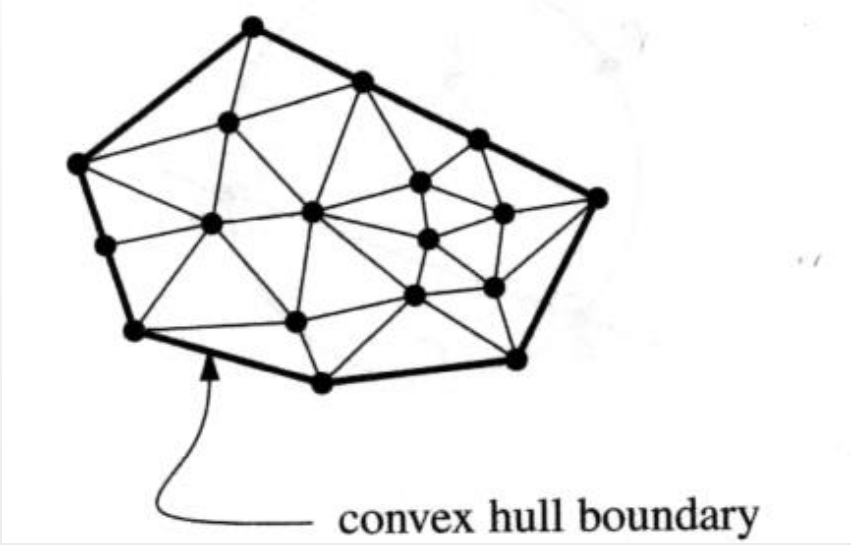
Delaunay triangulation

- Triangulation is made of triangles
  - Outer polygon must be convex hull
  - Internal faces must be triangles, otherwise they could be triangulated further



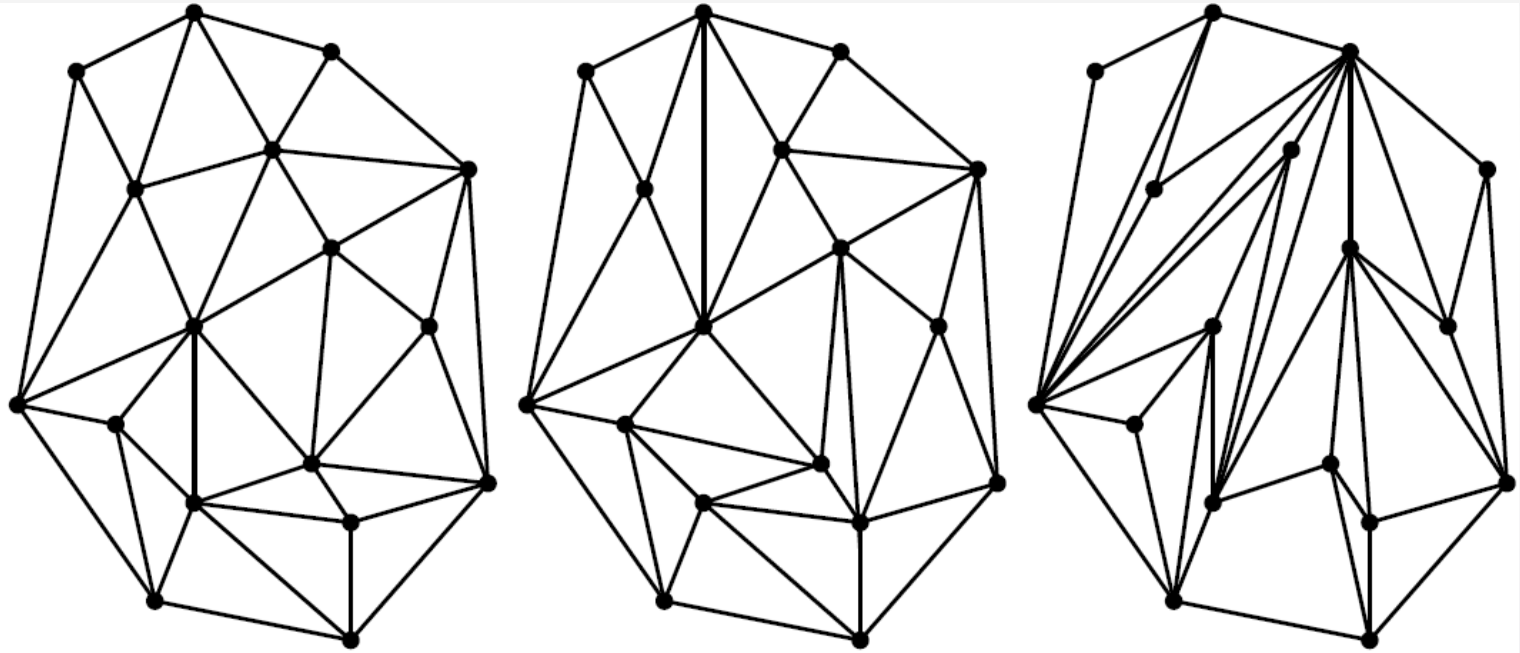
Delaunay triangulation

- For P consisting of n points, all triangulations contain
  - $2n$-2-k triangles
  - $3n$-3-k edges
  - n = number of points in P
  - k = number of points on convex hull of P



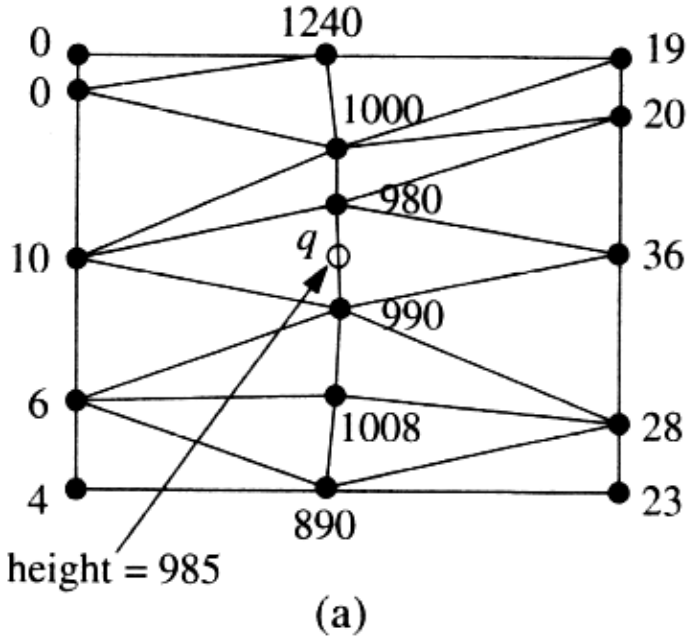
Delaunay triangulation

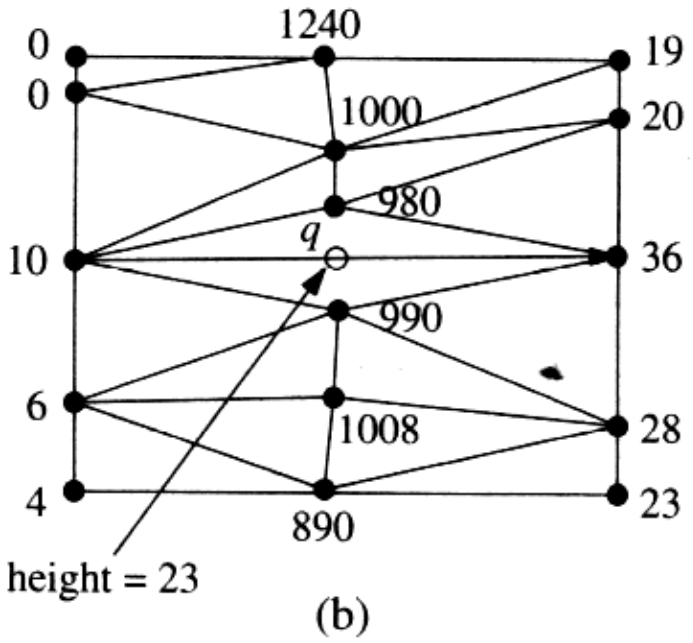
- But which triangulation?



Delaunay triangulation

- Some triangulations are “better” than others
- Avoid skinny triangles, i.e. maximize minimum angle of triangulation



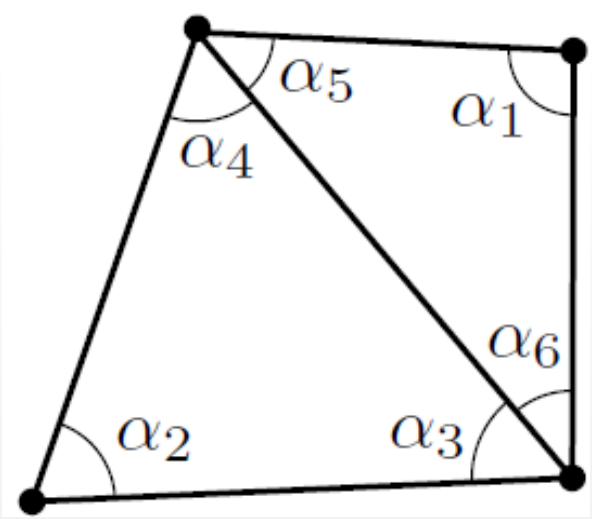


Delaunay triangulation

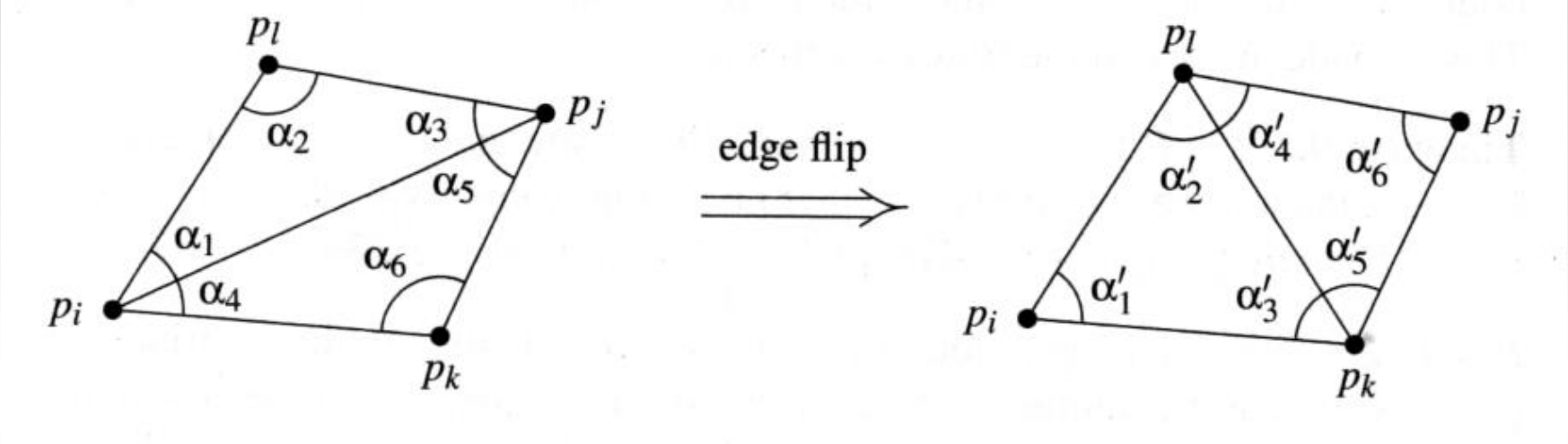
- Let $\mathcal{T}$ be a triangulation of P with m triangles. Its angle vector is $A(\mathcal{T}) = (\alpha_1, \ldots, \alpha_{3m})$ where $\alpha_1, \ldots, \alpha_{3m}$ are the angles of $\mathcal{T}$ sorted by increasing value.

- Let $\mathcal{T}'$ be another triangulation of P. $A(\mathcal{T})$ is larger then $A(\mathcal{T}')$ iff there exists an i such that $\alpha_j = \alpha'_j$ for all $j < i$ and $\alpha_i > \alpha'_i$

- $\mathcal{T}$ is angle optimal if $A(\mathcal{T}) > A(\mathcal{T}')$ for all triangulations $\mathcal{T}'$ of P



Delaunay triangulation



- If the two triangles form a convex quadrilateral, we could have an alternative triangulation by performing an *edge flip* on their shared edge.
  - The edge $e = \overline{P_i P_j}$ is illegal if $\min_{1 \le i \le 6} \alpha_i \le \min_{1 \le i \le 6} \alpha'_i$
  - Flipping an illegal edge increases the angle vector

Delaunay triangulation

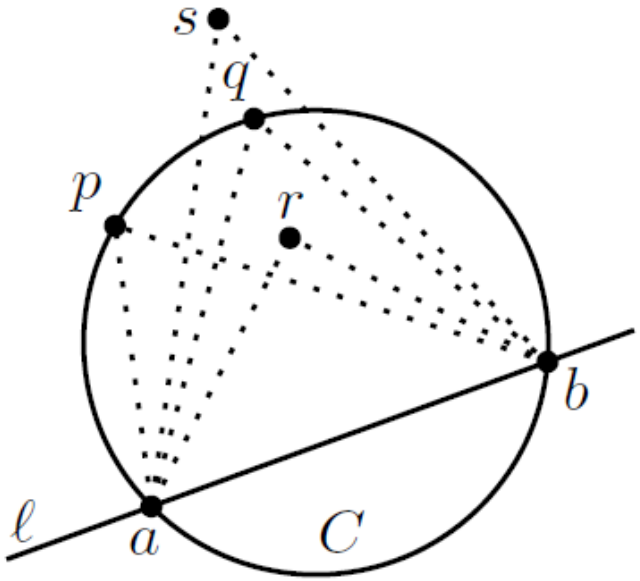
- If triangulation $\mathcal{T}$ contains an illegal edge *e*, we can make A($\mathcal{T}$) larger by flipping *e*.
- In this case, $\mathcal{T}$ is an *illegal triangulation*.

Delaunay triangulation

- We can use *Thale's Theorem* to test if an edge is legal without calculating angles

Theorem: Let C be a circle, ℓ a line intersecting C in points a and b, and p, q, r, s points lying on the same side of ℓ. Suppose that p, q lie on C, r lies inside C, and s lies outside C. Then

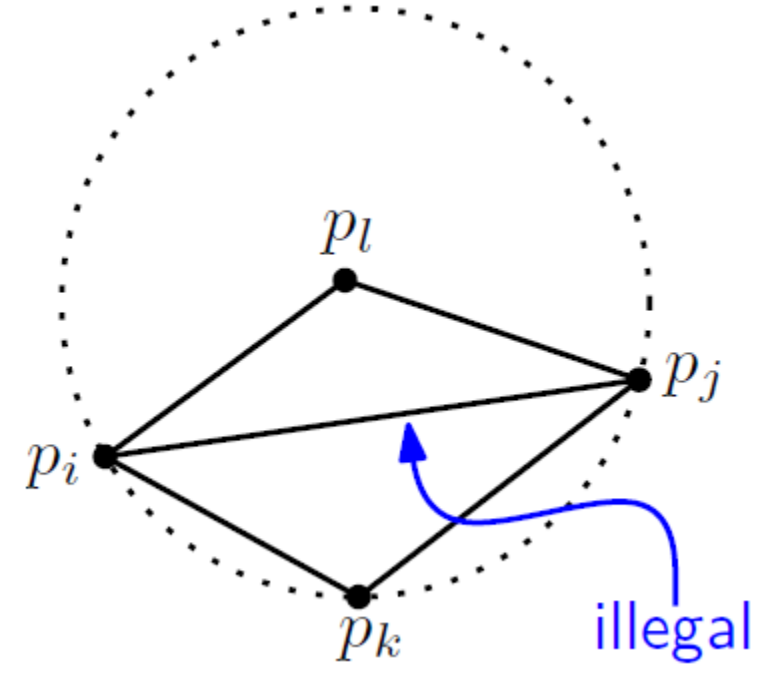
$$\measuredangle arb > \measuredangle apb = \measuredangle aqb > \measuredangle asb$$



Delaunay triangulation

- If p_i, p_j, p_k, p_l form a convex quadrilateral and do not lie on a common circle, exactly one of p_ip_j and p_kp_l is an illegal edge.

Lemma: The edge $\overline{P_iP_j}$ is illegal iff p_l lies in the interior of the circle C.



Delaunay triangulation

- A legal triangulation is a triangulation that does not contain any illegal edge.
- Compute Legal Triangulations

1. Compute a triangulation of input points P.
2. Flip illegal edges of this triangulation until all edges are legal.

  - Algorithm terminates because there is a finite number of triangulations.
  - *Too slow to be interesting…*

Delaunay triangulation

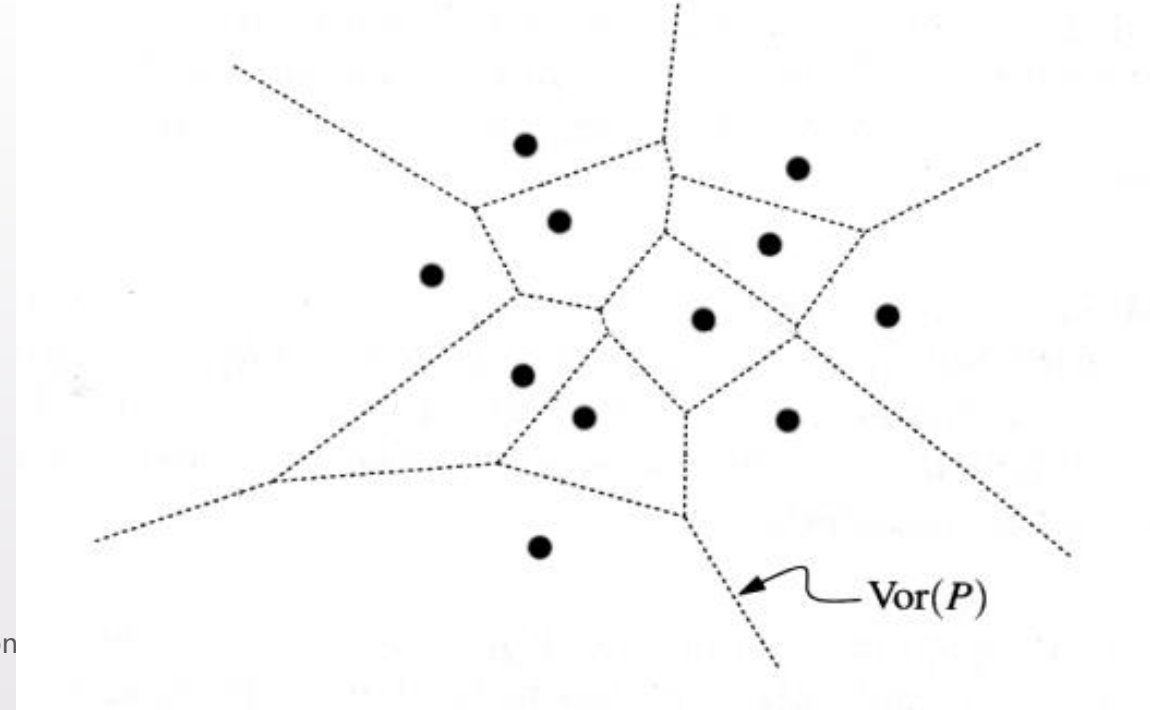
- Before we can understand an interesting solution to the terrain problem, we need to understand Delaunay Graphs.
- Delaunay Graph of a set of points *P* is the dual graph of the Voronoi diagram of *P*

Delaunay triangulation

- Voronoi Diagram and Delaunay Graph
  - Let P be a set of n points in the plane
  - The Voronoi diagram Vor(P) is the subdivision of the plane into Voronoi cells $\mathcal{V}(p)$ for all $p \in P$
  - Let $\mathcal{G}$ be the dual graph of Vor(P)
  - The Delaunay graph $\mathcal{DG}(P)$ is the straight line embedding of $\mathcal{G}$

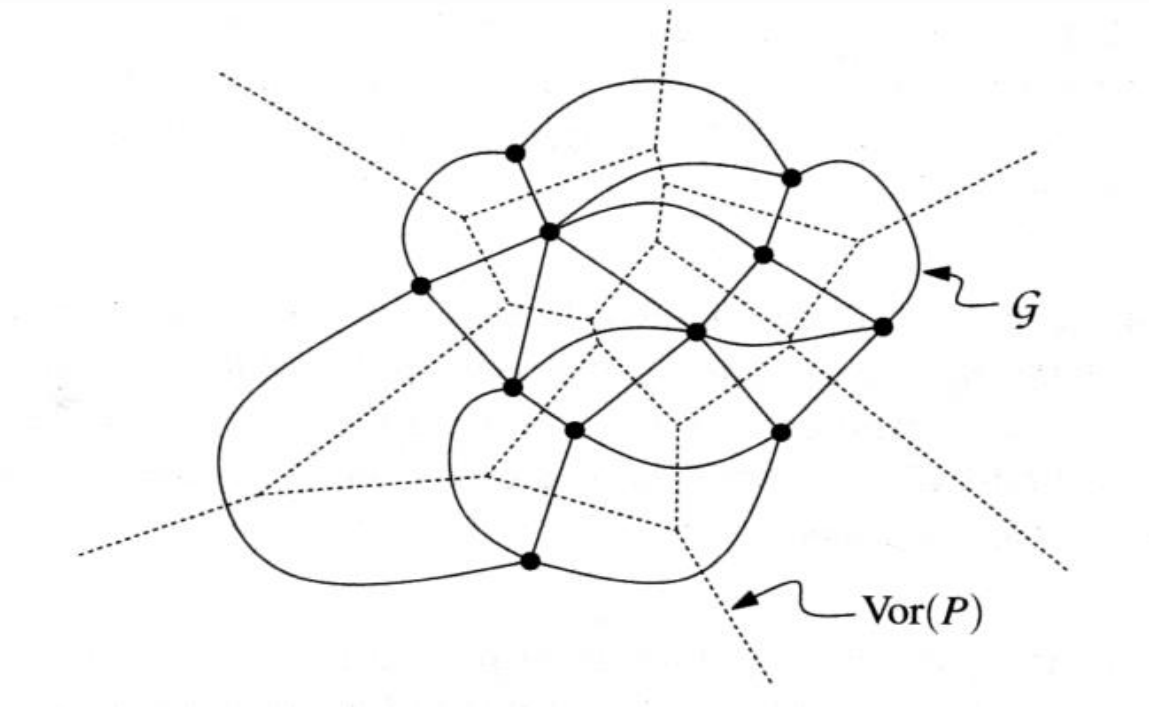
Delaunay triangulation

- Voronoi Diagram and Delaunay Graph
  - Calculate Vor(P)
  - Place one vertex in each site of the Vor(P)



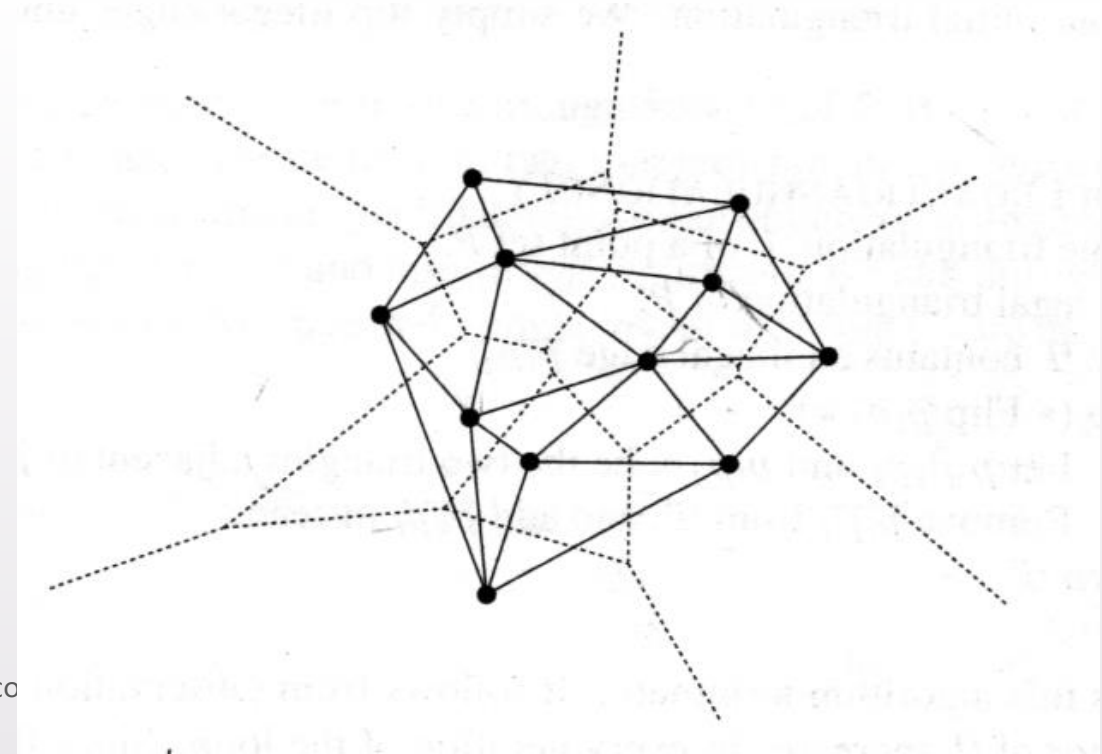
Delaunay triangulation

- Voronoi Diagram and Delaunay Graph
  - If two sites s_i and s_j share an edge (s_i and s_j are adjacent), create an arc between v_i and v_j, the vertices located in sites s_i and s_j



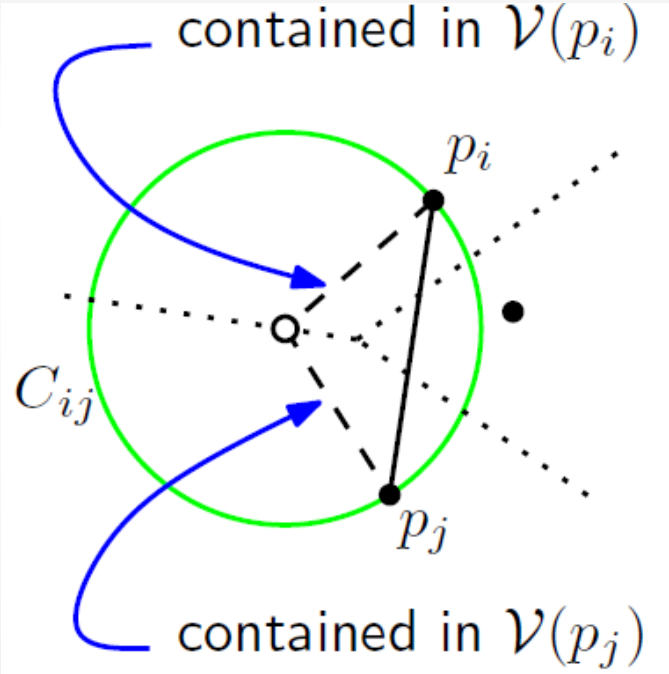
Delaunay triangulation

- Voronoi Diagram and Delaunay Graph
  - Finally, straighten the arcs into line segments. The resultant graph is $\mathcal{DG}(P)$.



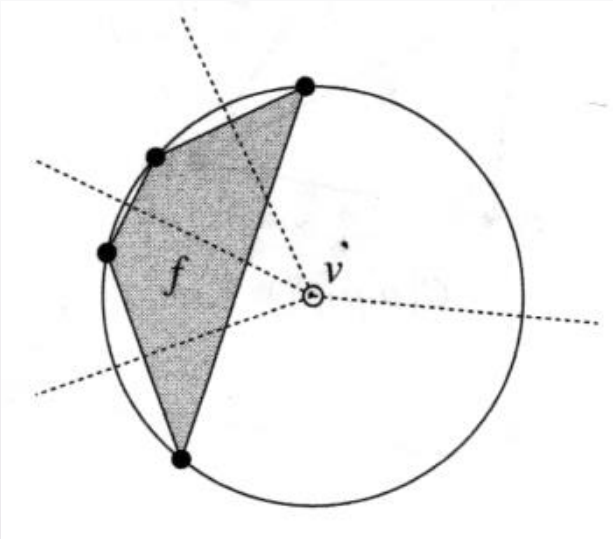
Delaunay triangulation

- Properties of Delaunay Graphs
  - No two edges cross; $\mathcal{DG}(P)$ is a planar graph.



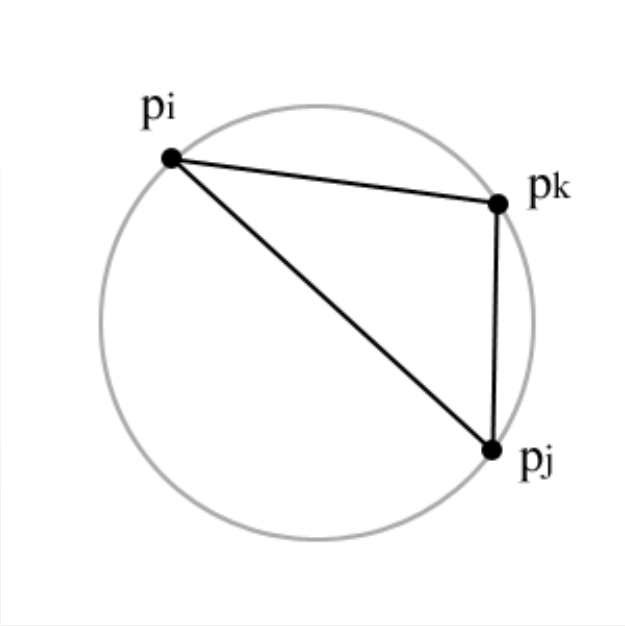
Delaunay triangulation

- Some sets of more than 3 points of Delaunay graph may lie on the same circle.
- These points form empty convex polygons, which can be triangulated.
- *Delaunay Triangulation* is a triangulation obtained by adding 0 or more edges to the Delaunay Graph.



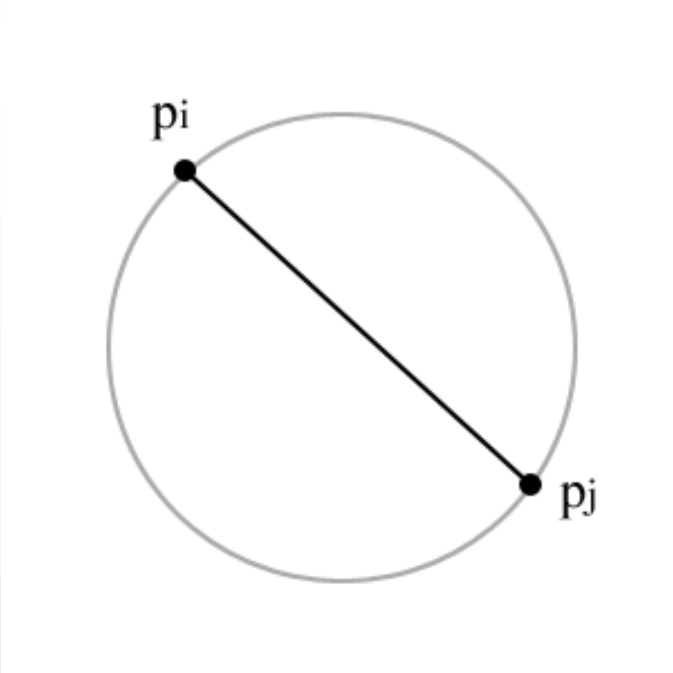
Delaunay triangulation

- From the properties of Voronoi Diagrams…
  - Three points $p_i, p_j, p_k \in P$ are vertices of the same face of the $\mathcal{DG}(P)$ iff the circle through p_i, p_j, p_k contains no point of P on its interior.



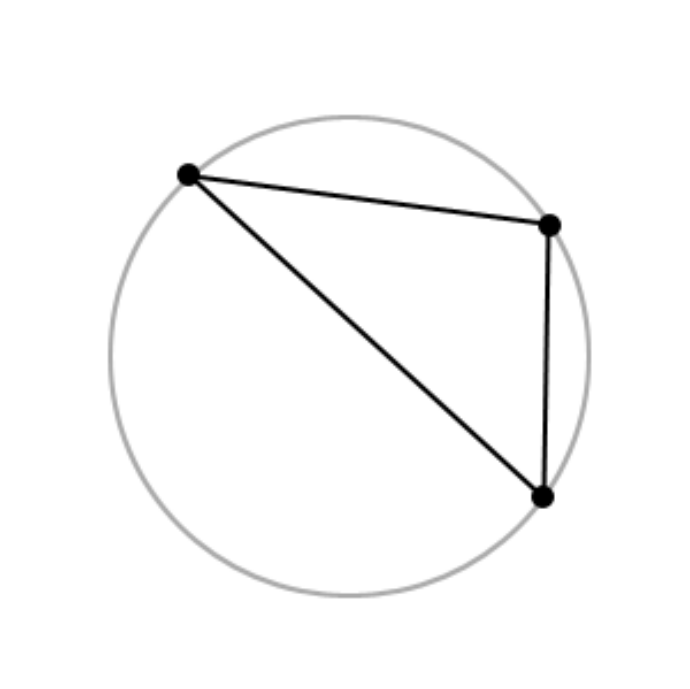
Delaunay triangulation

- From the properties of Voronoi Diagrams…
  - Two points $p_i, p_j \in P$ form an edge of $\mathcal{DG}(P)$ iff there is a closed disc C that contains p_i and p_j on its boundary and does not contain any other point of P.



Delaunay triangulation

- From the previous two properties…
  - A triangulation $\mathcal{T}$ of P is a $\mathcal{DT}(P)$ iff the circumcircle of any triangle of $\mathcal{T}$ does not contain a point of P in its interior.



Delaunay triangulation

- A triangulation $\mathcal{T}$ of *P* is legal iff $\mathcal{T}$ is a $\mathcal{DT}$(P).
  - DT $\rightarrow$ Legal: Empty circle property and Thale's Theorem implies that all $\mathcal{DT}$ are legal
  - Legal $\rightarrow$ DT
- Let *P* be a set of points in the plane. Any angle-optimal triangulation of *P* is a Delaunay triangulation of *P*.
- Furthermore, any Delaunay triangulation of *P* maximizes the minimum angle over all triangulations of *P*.

Delaunay triangulation

- Therefore, the problem of finding a triangulation that maximizes the minimum angle is reduced to the problem of finding a Delaunay Triangulation.

So how do we find the Delaunay Triangulation?

Delaunay triangulation

- How do we compute $\mathcal{DT}(P)$?
- We could compute $\mathcal{V}or(P)$ then dualize into $\mathcal{DT}(P)$.
- Instead, we will compute $\mathcal{DT}(P)$ using a randomized incremental method.

Delaunay triangulation

- Incremental method
  1. Initialize triangulation $\mathcal{T}$ with a "big enough" helper bounding triangle that contains all points P.
  2. Randomly choose a point p_r from P.
  3. Find the triangle Δ that p_r lies in.
  4. Subdivide Δ into smaller triangles that have p_r as a vertex.
  5. Flip edges until all edges are legal.
  6. Repeat steps 2-5 until all points have been added to $\mathcal{T}$.

Delaunay triangulation

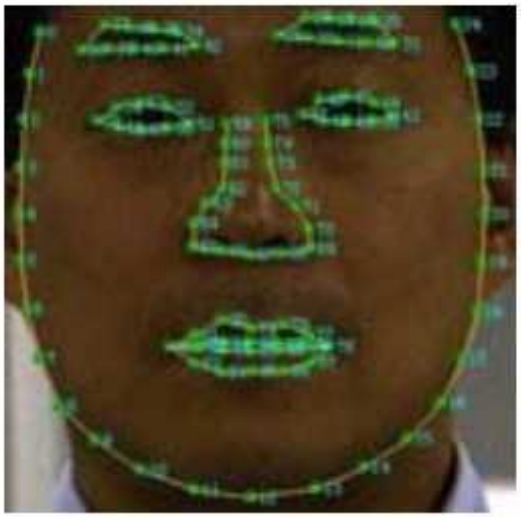
- In matlab, try delaunay(X,Y) function

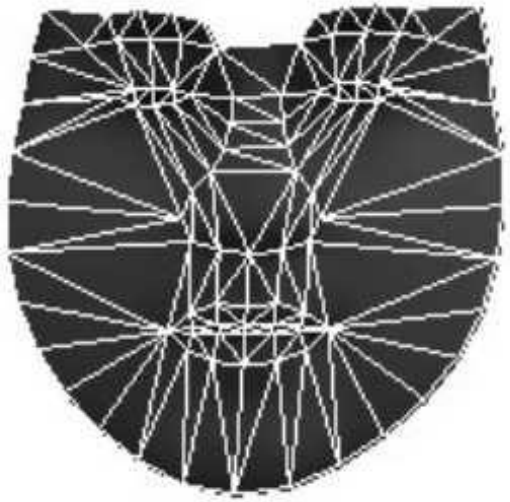
Outline

- Introduction
- Facial expression recognition
  - Appearance-based vs. model-based
- Active appearance model (AAM)
  - Pre-requisite
  - Principle component analysis
  - Procrustes analysis
  - ASM
  - Delaunay triangulation
  - AAM

AAM

- Steps of constructing AAM
  1. Apply triangulation algorithm on the training faces;

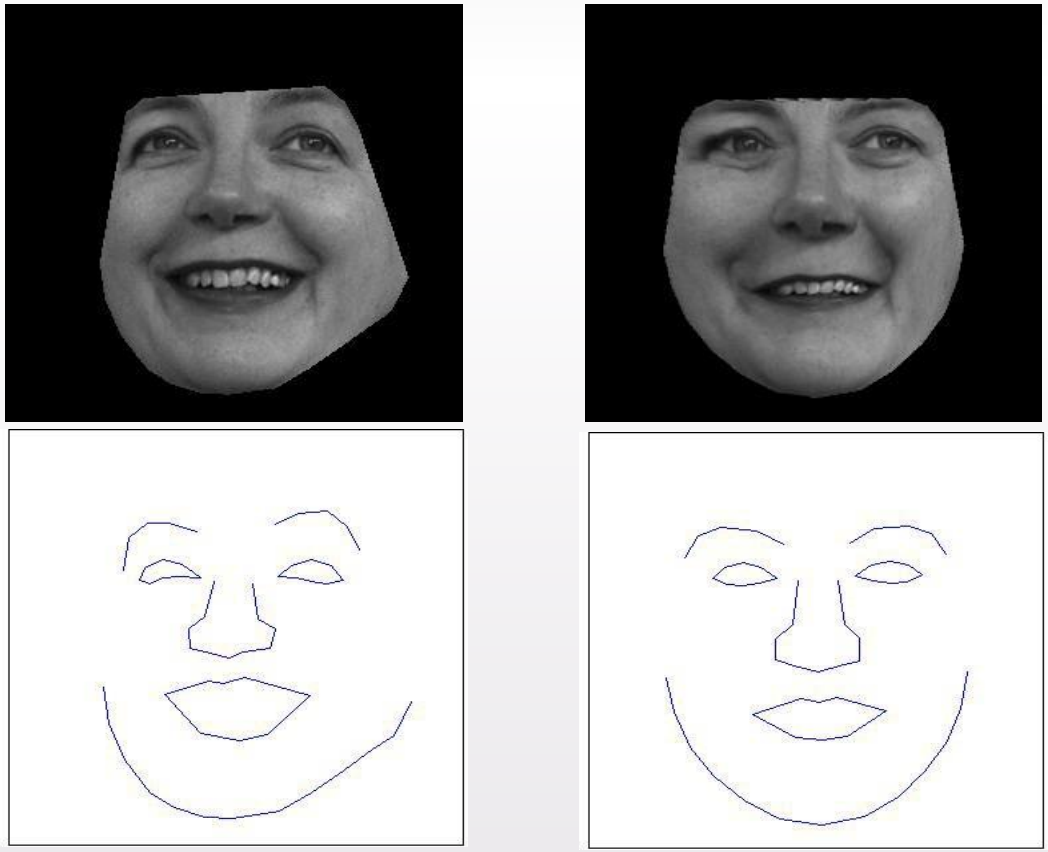




AAM

- Steps of constructing AAM

2. Warp the training face to the mean face by matching the corresponding triangles



AAM

- Steps of constructing AAM

3. Sample the grey level information $\mathbf{g}_{im}$ from the shape-normalized image over the region covered by the mean face

   To minimize the effect of global lighting variation, normalize the faces

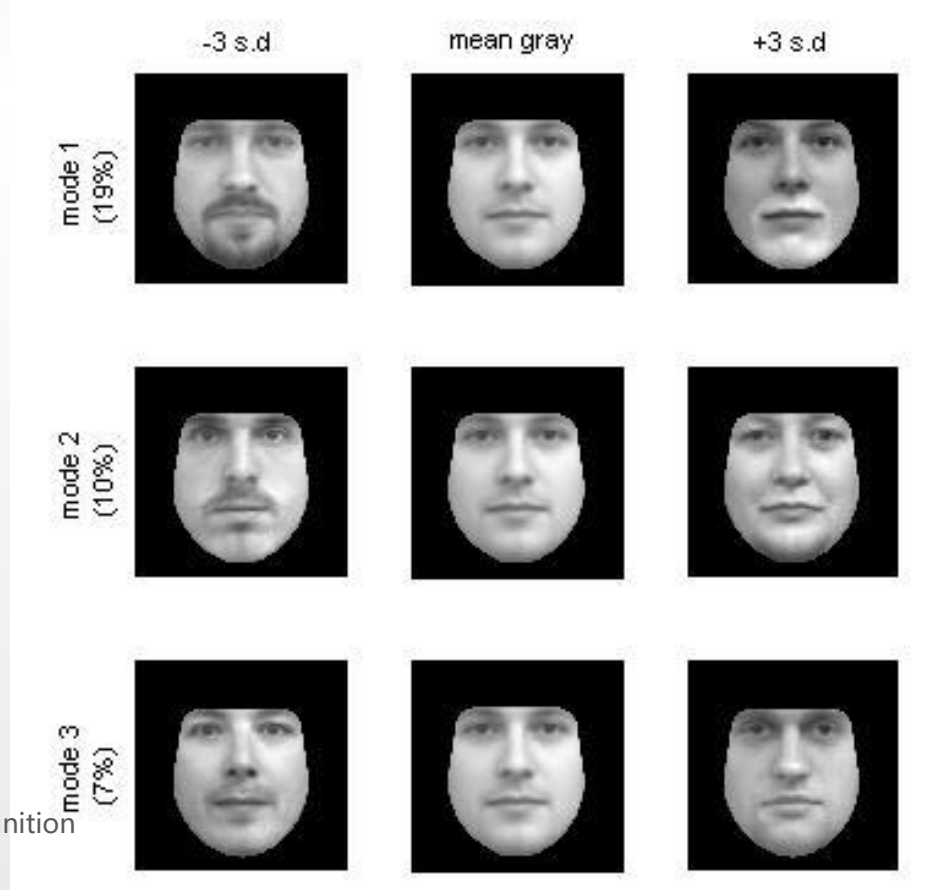
$$\mathbf{g} = (\mathbf{g}_{im} - \beta\mathbf{1})/\alpha$$

$$\alpha = \mathbf{g}_{im} \cdot \bar{\mathbf{g}}, \beta = (\mathbf{g}_{im} \cdot \mathbf{1})/n$$

AAM

- Steps of constructing AAM
  4. Apply PCA to the normalized data

$$\mathbf{g} = \bar{\mathbf{g}} + \mathbf{P}_g \mathbf{b}_g$$



AAM

- Appearance vector:

$$\mathbf{b} = \begin{pmatrix} \mathbf{W}_s \mathbf{b}_s \\ \mathbf{b}_g \end{pmatrix} = \begin{pmatrix} \mathbf{W}_s \mathbf{P}_s^T (\mathbf{x} - \bar{\mathbf{x}}) \\ \mathbf{P}_g^T (\mathbf{g} - \bar{\mathbf{g}}) \end{pmatrix}$$

- Apply a further PCA

$$\mathbf{b} = \mathbf{Q}\mathbf{c}$$

  - $\mathbf{c}$ is a vector of *appearance* parameters

- We can express the face as functions of $\mathbf{c}$

$$\mathbf{x} = \bar{\mathbf{x}} + \mathbf{P}_s \mathbf{W}_s \mathbf{Q}_s \mathbf{c}, \mathbf{g} = \bar{\mathbf{g}} + \mathbf{P}_g \mathbf{Q}_g \mathbf{c} \qquad \mathbf{Q} = \begin{pmatrix} \mathbf{Q}_s \\ \mathbf{Q}_g \end{pmatrix}$$