UNIVERSITÄT BONN

Juergen Gall

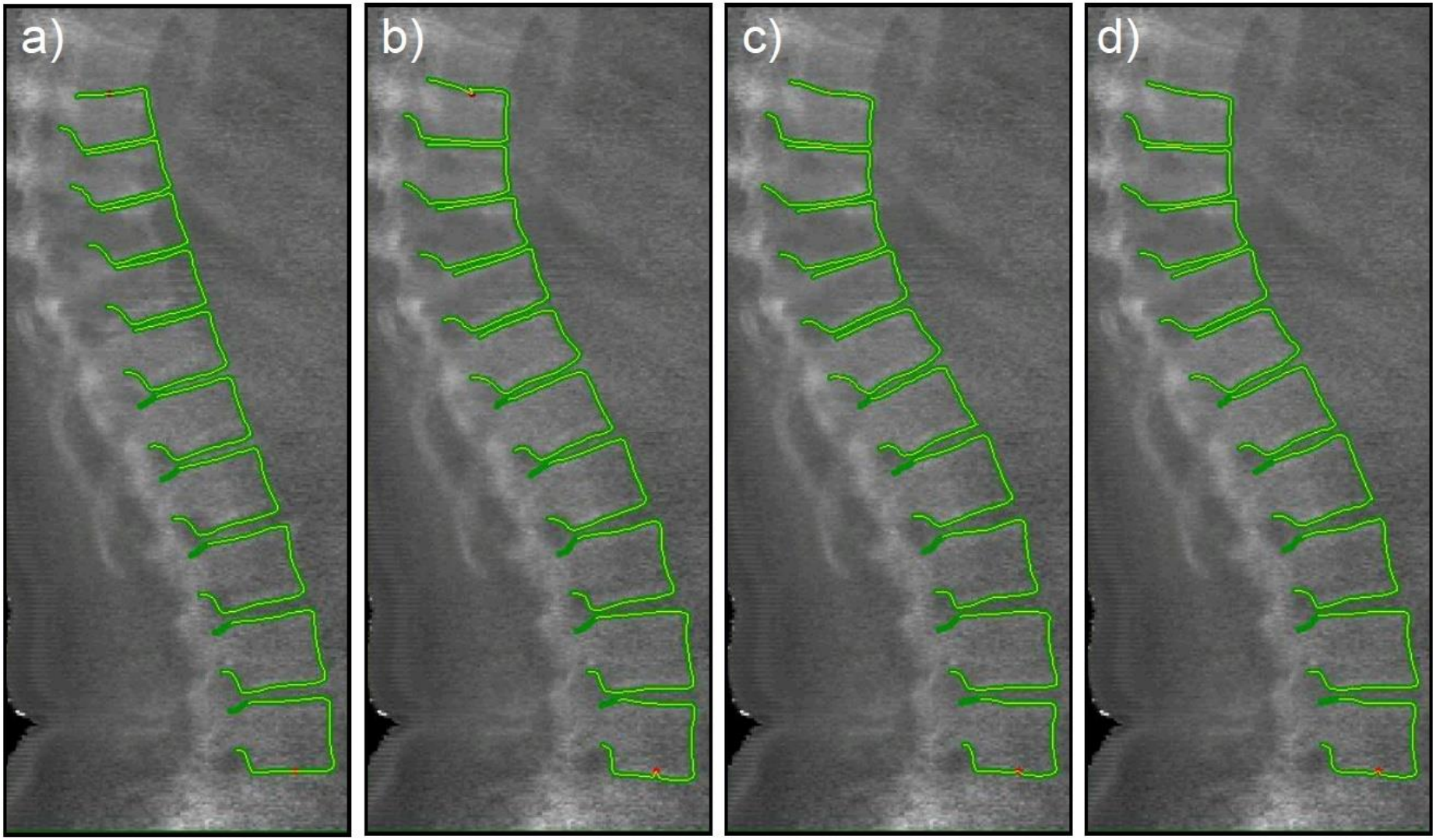
Transformations and Shapes
MA-INF 2201 - Computer Vision
WS24/25

Literature

Chapter 17 Models for shape

S. Prince. **Computer Vision: Models, Learning, and Inference.** Cambridge University Press 2012

Motivation: fitting shape model



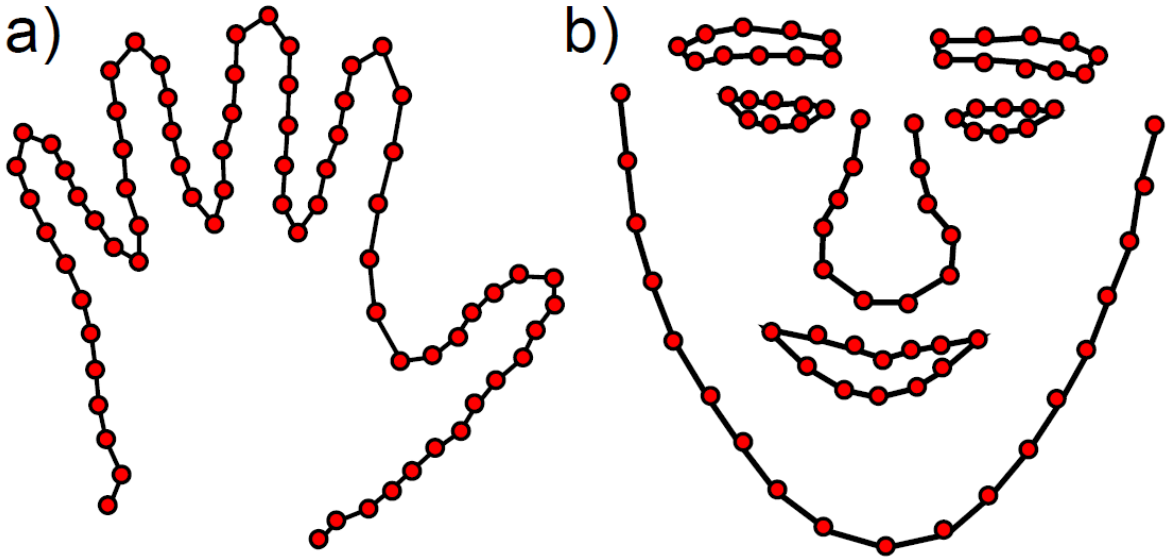
Source: T. Cootes

What is shape?

- Kendall (1984) – Shape “is all the geometrical information that remains when location, scale and rotational effects are filtered out from an object”
- In other words, it is whatever is invariant to a similarity transformation

Source: S. Prince

Landmark Points



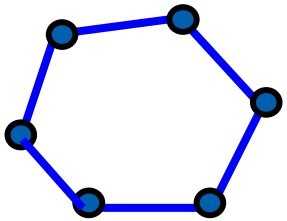
- Landmark points can be thought of as discrete samples from underlying contour
  - Ordered (single continuous contour)
  - Ordered with wrapping (closed contour)
  - More complex organisation (collection of closed and open)

Source: S. Prince

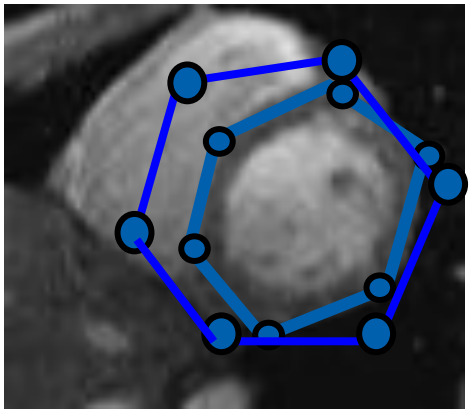
Recap: Snakes

UNIVERSITÄT BONN

- A simple elastic snake is defined by:
  - A set of *n* points,
  - An internal energy term (tension, bending, plus optional shape prior)
  - An external energy term (gradient-based)



- To use to segment an object:
  - Initialize in the vicinity of the object
  - Modify the points to minimize the total energy



Source: K. Grauman

Probability or not?

UNIVERSITÄT BONN

Snakes: minimize cost function

$$E_{total} = E_{internal} + E_{external}$$

Probabilistic view:

$$\hat{\mathbf{W}} = \underset{\mathbf{W}}{\operatorname{argmax}} \left[Pr(\mathbf{W}|\mathbf{x}) \right] = \underset{\mathbf{W}}{\operatorname{argmax}} \left[Pr(\mathbf{x}|\mathbf{W}) Pr(\mathbf{W}) \right]$$

$$= \underset{\mathbf{W}}{\operatorname{argmax}} \left[\log[Pr(\mathbf{x}|\mathbf{W})] + \log[Pr(\mathbf{W})] \right]$$

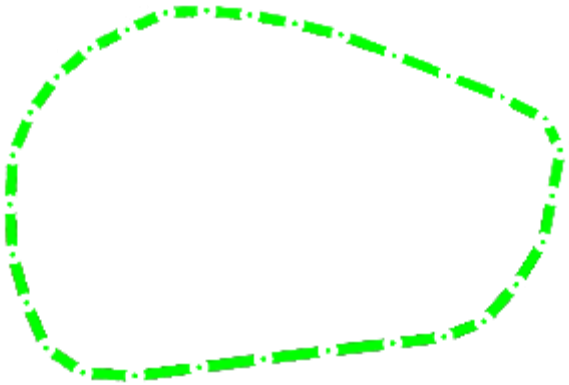
likelihood prior

Weak priors: smoothness etc…

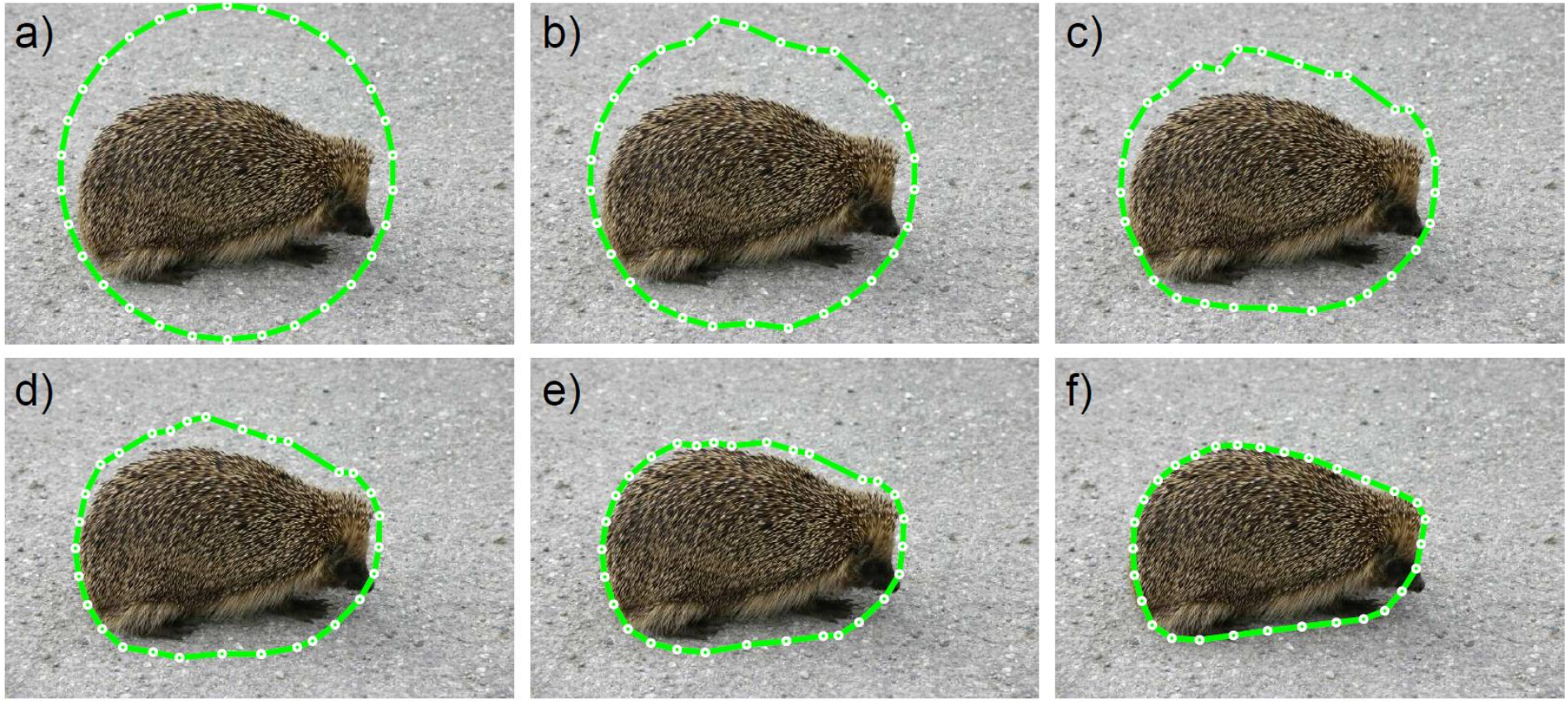
Strong prior: shape

Snakes

What is the name of the animal?



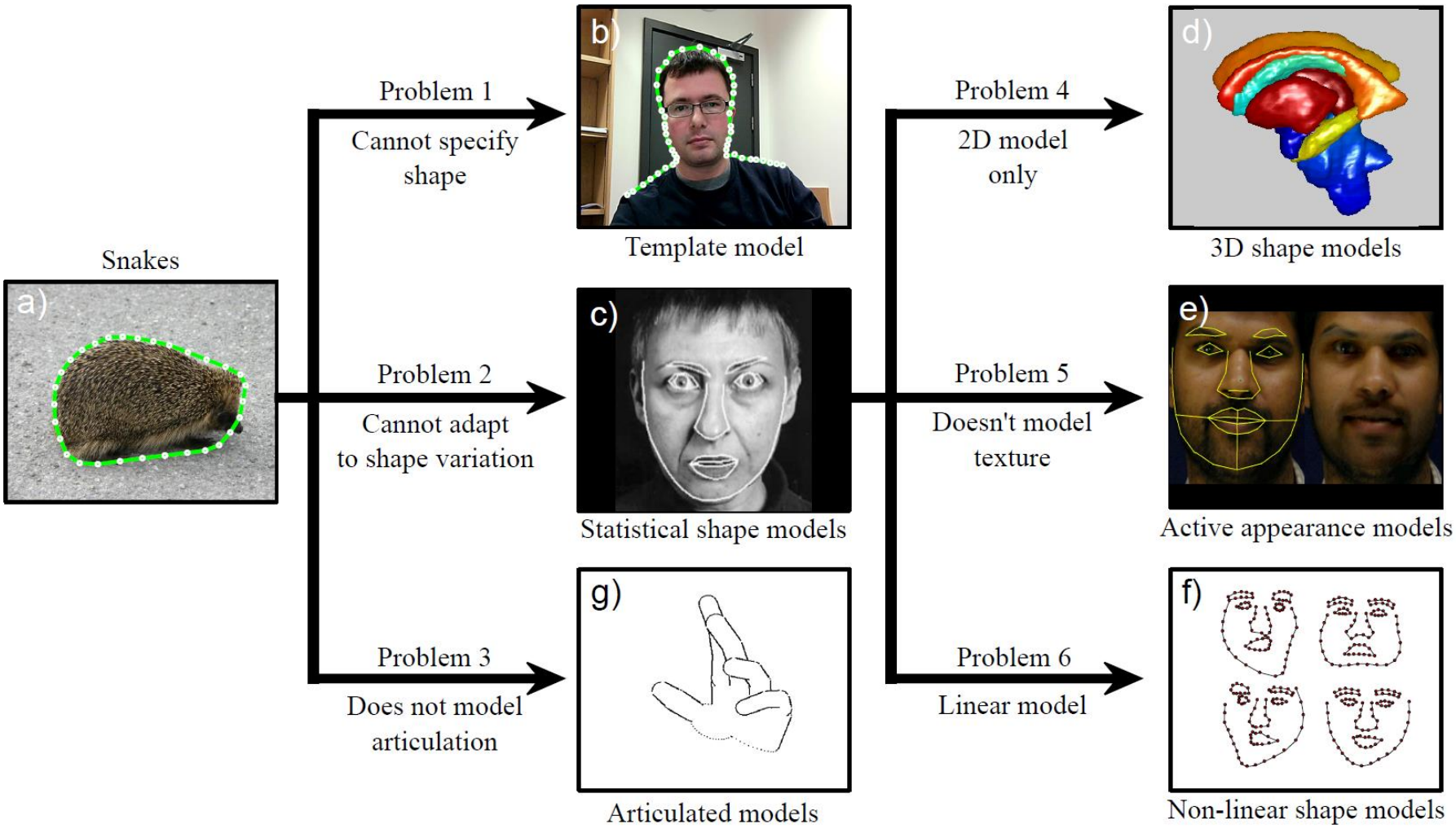
Snakes



Notice failure at nose – falls between points.
Resulting shape does not represent hedgehog

Source: S. Prince

Relationships between models



Source: S. Prince

Shape template model

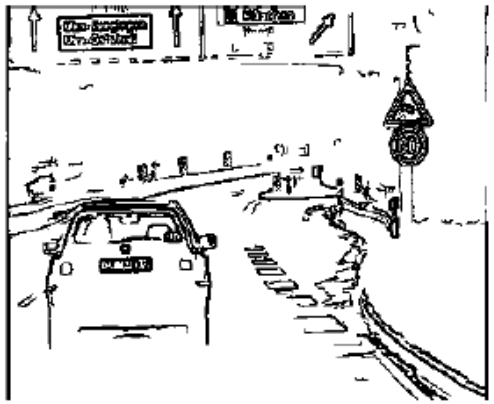
- Shape based on landmark points $\mathbf{W} = \{\mathbf{w}_n\}_{n=1}^{N}$
- These points are assumed known
- Mapped into the image by transformation $\mathbf{trans}[\mathbf{w}, \boldsymbol{\Psi}]$
- What is left is to find parameters of transformation $\boldsymbol{\Psi}$
- Likelihood is based on distance transform:

$$Pr(\mathbf{x}|\mathbf{W}, \boldsymbol{\Psi}) \propto \prod_{n=1}^{N} \exp\left[-\left(\text{dist}\left[\mathbf{x}, \mathbf{trans}[\mathbf{w}_n, \boldsymbol{\Psi}]\right]\right)^2\right]$$

- No prior on parameters (but could have one)

Source: S. Prince

Recap: Chamfer distance



Edge image **Distance transform image**

Source: D. Gavrila

Inference

- Use maximum likelihood approach

$$\hat{\boldsymbol{\Psi}} = \underset{\boldsymbol{\Psi}}{\operatorname{argmax}}\, L = \underset{\boldsymbol{\Psi}}{\operatorname{argmax}} \left[\log \left[Pr(\mathbf{x}|\mathbf{W}, \boldsymbol{\Psi})\right]\right]$$

$$= \underset{\boldsymbol{\Psi}}{\operatorname{argmax}} \left[\sum_{n=1}^{N} -\left(\operatorname{dist}\left[\mathbf{x}, \mathbf{trans}[\mathbf{w}_n, \boldsymbol{\Psi}]\right]\right)^2\right]$$

- No closed form solution
- Must use non-linear optimization or brute-force search
- Use chain rule to compute derivatives

$$\frac{\partial L}{\partial \boldsymbol{\Psi}} = -\sum_{n=1}^{N}\sum_{j=1}^{2} \frac{\partial (\operatorname{dist}[\mathbf{x}, \mathbf{w}'_n])^2}{\partial w'_{jn}} \frac{\partial w'_{jn}}{\partial \boldsymbol{\Psi}} \quad \text{where } \mathbf{w}'_n = \mathbf{trans}[\mathbf{w}_n, \boldsymbol{\Psi}]$$

Source: S. Prince

Derivative

UNIVERSITÄT BONN

Pre-compute Euclidean distance transform D:

Pre-compute gradient by central difference:

$$\frac{\partial}{\partial w'_{jn}} \frac{1}{2} (\mathrm{dist}[x, w'_n])^2 \approx D(w'_n) G_j(w'_n) = D(w'_n) \frac{1}{2} \left(D(w'_n + e_j) - D(w'_n - e_j) \right)$$

Interpolate w' from nearby pixels

Linear Transformations

UNIVERSITÄT BONN

- Transformations Ψ

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

- Linear 2D transformations can be represented with a 2x2 matrix and are combinations of …
  - Scale
  - Rotation,
  - Shear
  - Mirror

Examples

2D Scaling

$$x' = s_x * x$$

$$y' = s_y * y$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

2D Rotate around (0,0)

$$x' = \cos\Theta * x - \sin\Theta * y$$
$$y' = \sin\Theta * x + \cos\Theta * y$$

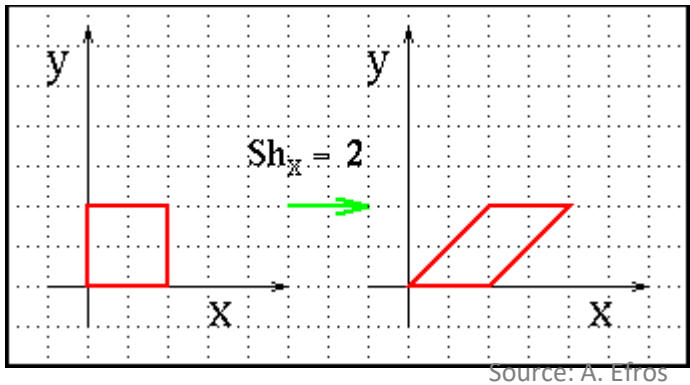
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\Theta & -\sin\Theta \\ \sin\Theta & \cos\Theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

2D Shear

$$x' = x + sh_x * y$$

$$y' = sh_y * x + y$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & sh_x \\ sh_y & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$



Source: A. Efros

Examples

2D Mirror about Y axis

$$x' = -x$$
$$y' = y$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

2D Mirror over (0,0)

$$x' = -x$$
$$y' = -y$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

2D Translation

$$\boldsymbol{x}' = \boldsymbol{x} + \boldsymbol{t}_x$$
$$\boldsymbol{y}' = \boldsymbol{y} + \boldsymbol{t}_y$$

NO!

Source: A. Efros

Homogeneous coordinates

UNIVERSITÄT BONN

To convert to homogeneous coordinates:

$$(x, y) \Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

homogeneous image coordinates

Converting *from* homogeneous coordinates

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} \Rightarrow (x/w, y/w)$$

Source: K. Grauman

Homogeneous Coordinates

- Q: How can we represent 2d translation as a 3x3 matrix using homogeneous coordinates?

$$x' = x + t_x$$

$$y' = y + t_y$$

- A: Using the rightmost column:

$$\mathbf{Translation} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

Source: A. Efros

Basic 2D Transformations

- Basic 2D transformations as 3x3 matrices

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Translate

$$\begin{bmatrix} \boldsymbol{x}' \\ \boldsymbol{y}' \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{s}_x & 0 & 0 \\ 0 & \boldsymbol{s}_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \\ 1 \end{bmatrix}$$

Scale

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\Theta & -\sin\Theta & 0 \\ \sin\Theta & \cos\Theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Rotate

$$\begin{bmatrix} \boldsymbol{x}' \\ \boldsymbol{y}' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & \boldsymbol{sh}_x & 0 \\ \boldsymbol{sh}_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \\ 1 \end{bmatrix}$$

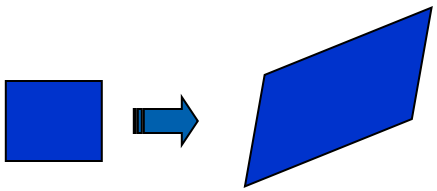
Shear

Source: A. Efros

2D Affine Transformations

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

- Affine transformations are combinations of
  - Linear transformations, and
  - Translations
- Parallel lines remain parallel



Source: K. Grauman

Inference

- Use maximum likelihood approach

$$\hat{\boldsymbol{\Psi}} = \underset{\boldsymbol{\Psi}}{\operatorname{argmax}}\, L = \underset{\boldsymbol{\Psi}}{\operatorname{argmax}} \left[\log \left[Pr(\mathbf{x}|\mathbf{W}, \boldsymbol{\Psi})\right]\right]$$

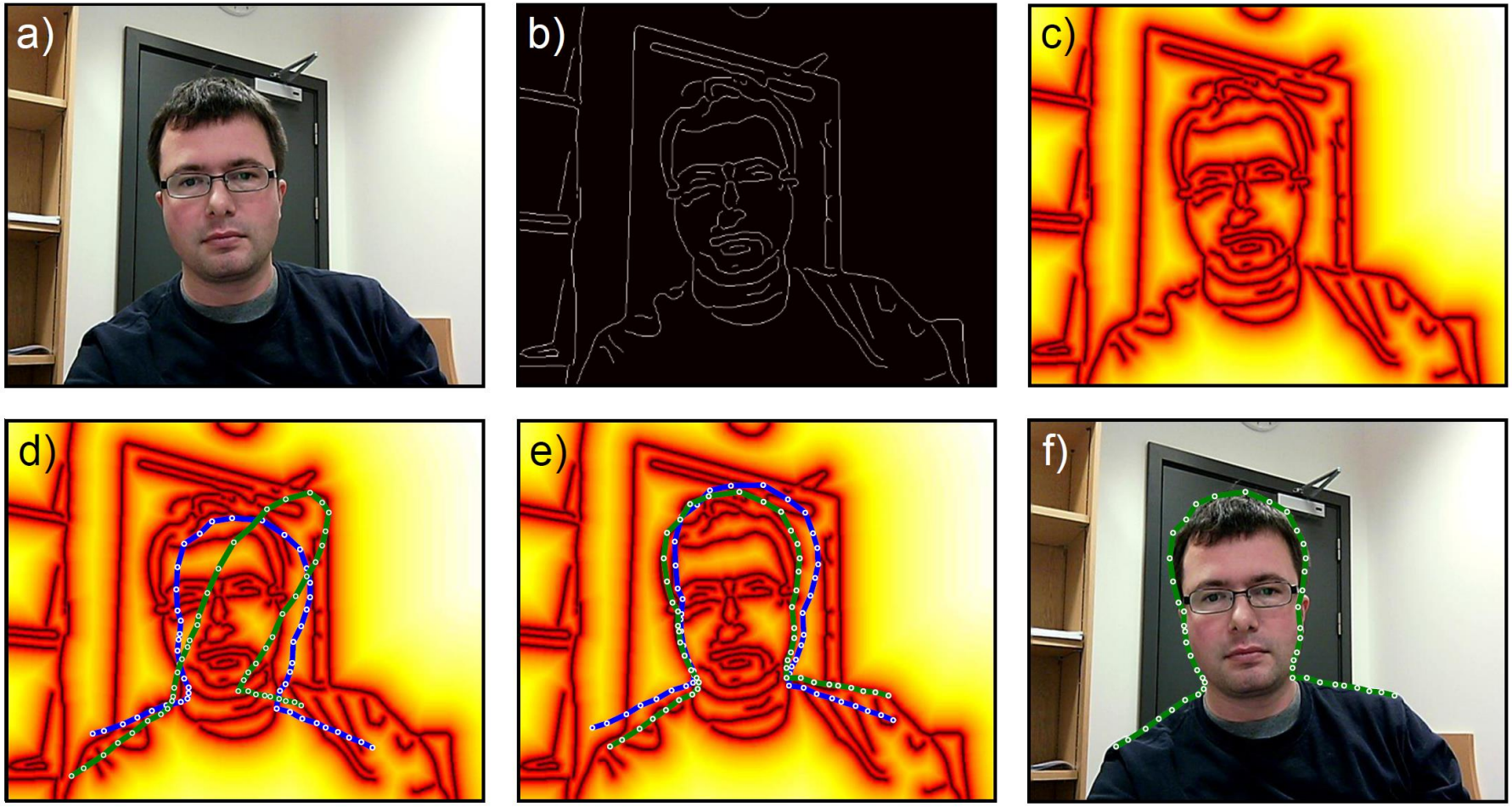
$$= \underset{\boldsymbol{\Psi}}{\operatorname{argmax}} \left[\sum_{n=1}^{N} -\left(\operatorname{dist}\left[\mathbf{x}, \mathbf{trans}[\mathbf{w}_n, \boldsymbol{\Psi}]\right]\right)^2\right]$$

- No closed form solution
- Must use non-linear optimization or brute-force search
- Use chain rule to compute derivatives

$$\frac{\partial L}{\partial \boldsymbol{\Psi}} = -\sum_{n=1}^{N}\sum_{j=1}^{2} \frac{\partial (\operatorname{dist}[\mathbf{x}, \mathbf{w}'_n])^2}{\partial w'_{jn}} \frac{\partial w'_{jn}}{\partial \boldsymbol{\Psi}} \quad \text{where } \mathbf{w}'_n = \mathbf{trans}[\mathbf{w}_n, \boldsymbol{\Psi}]$$

Source: S. Prince

Shape template model



Local optimum

Global optimum

Blue initialization, green estimate

Source: S. Prince

Iterative closest points

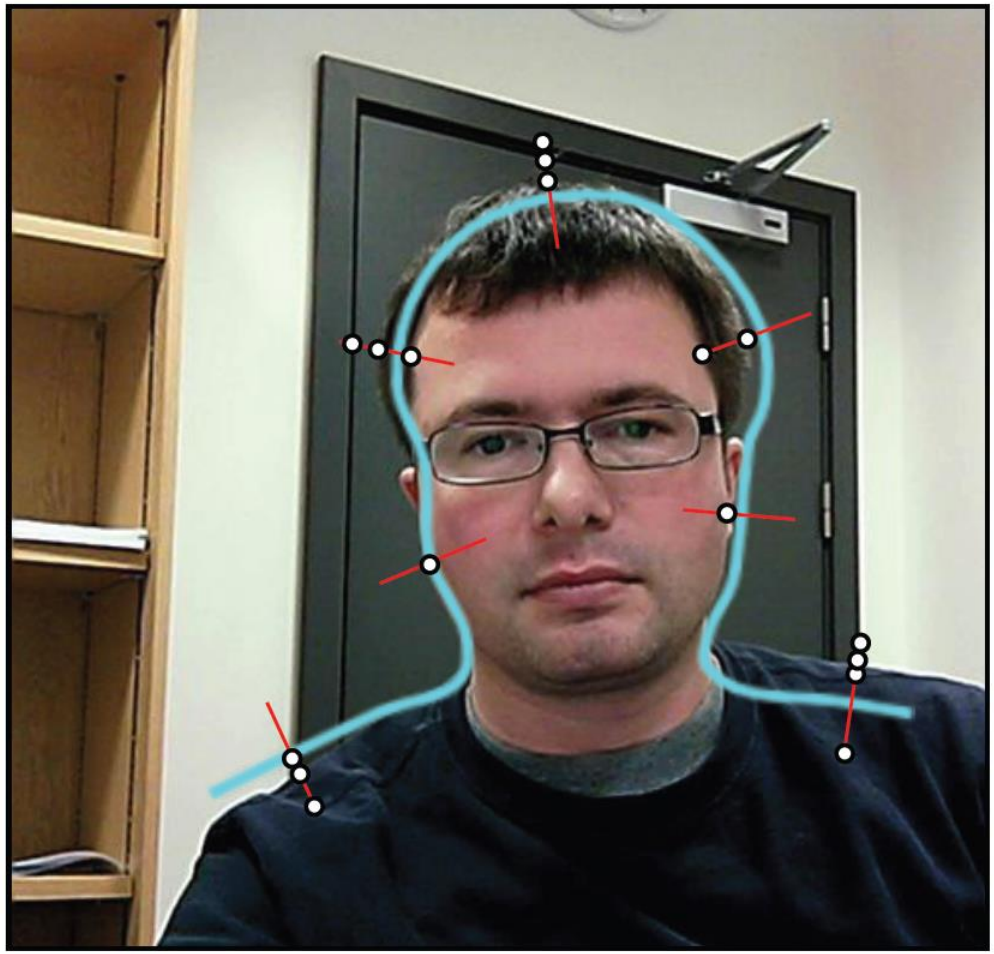
UNIVERSITÄT BONN

- Maximum likelihood approach

$$\frac{\partial L}{\partial \mathbf{\Psi}} = -\sum_{n=1}^{N}\sum_{j=1}^{2} \frac{\partial(\mathrm{dist}[\mathbf{x}, \mathbf{w}'_n])^2}{\partial w'_{jn}} \frac{\partial w'_{jn}}{\partial \mathbf{\Psi}} \qquad \text{where } \mathbf{w}'_n = \mathbf{trans}[\mathbf{w}_n, \mathbf{\Psi}]$$

- Iterative closest point:
  - Find closest edge point or line search along normal
  - Estimate Ψ from correspondences (Closed form solution exist)

Iterative closest points



- Find nearest edge point to each landmark point
- Compute transformation in closed form
- Repeat

Source: S. Prince

Correspondences

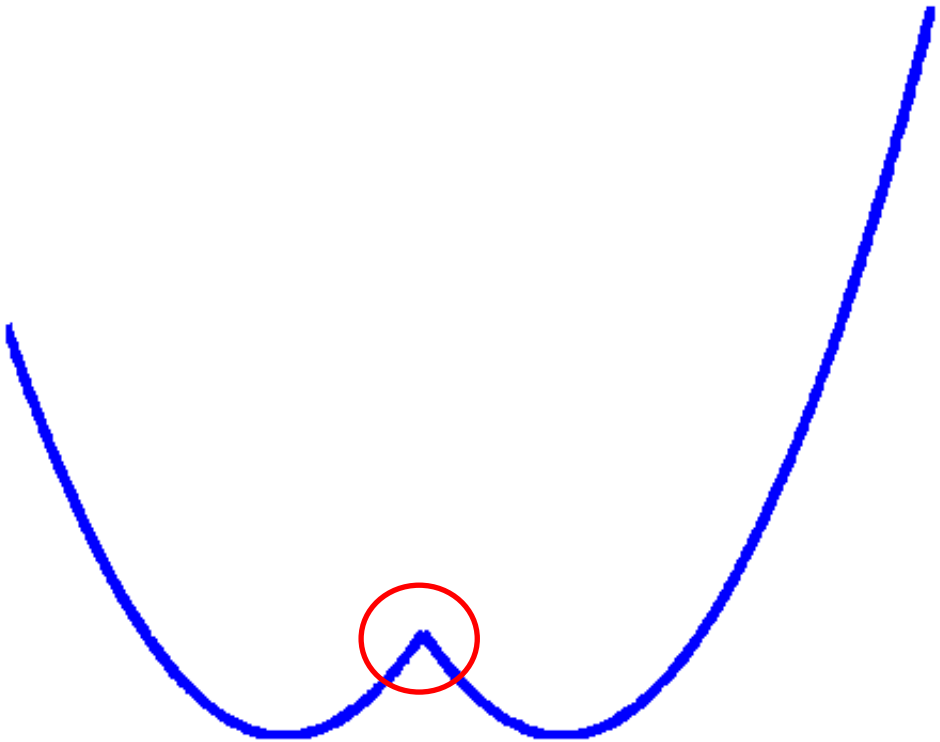
- Line search
- Closest point based on distance transform D:

$$x_n = f(w'_n) = w'_n - \frac{D(w'_n)}{\sqrt{G_x(w'_n)^2 + G_y(w'_n)^2}}(G_x(w'_n)G_y(w'_n))$$

- 1D example:
  - w: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
  - E: 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0
  - D: 4 3 2 1 0 1 2 3 2 1 0 1 2 3 4
  - G: -1 -1 -1 -1 0 1 1 0 -1 -1 0 1 1 1 1
  - x: 4 4 4 4 4 4 4 7 10 10 10 10 10 10 10

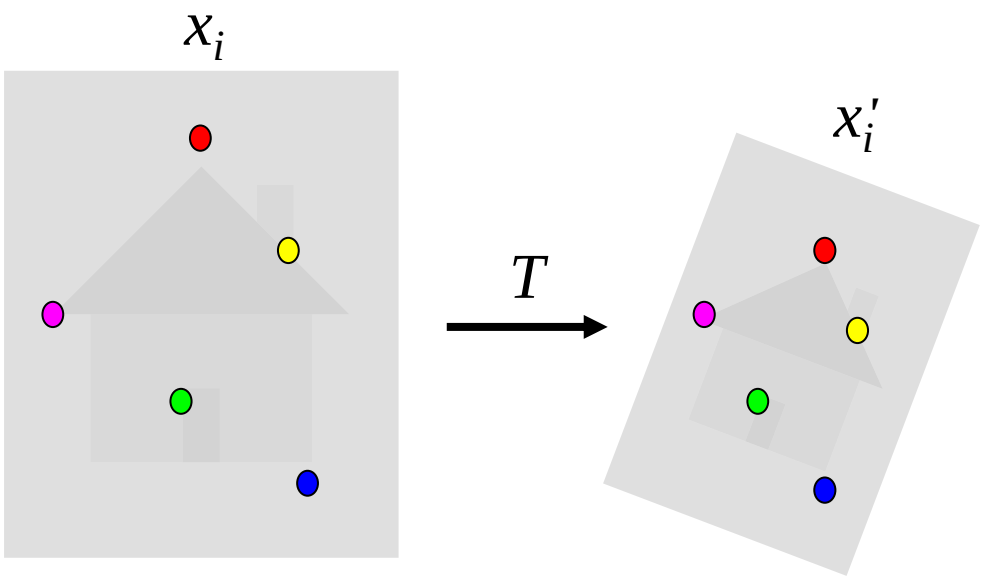
Derivative

Function is not everywhere differentiable



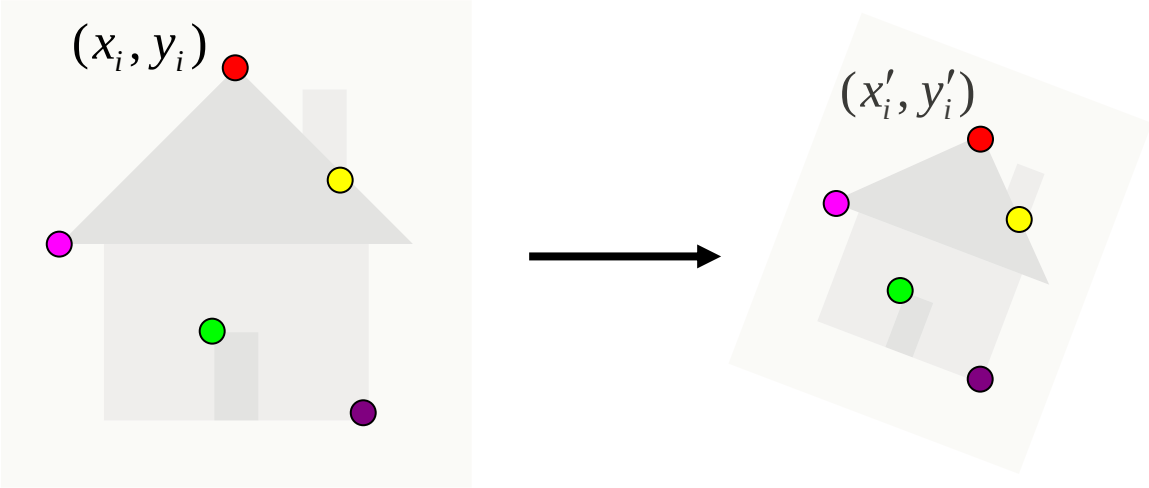
Alignment problem

Fit the parameters of some **transformation** according to a set of matching feature pairs (“correspondences”).



Source: K. Grauman

Fitting an affine transformation



$$\begin{bmatrix} x_i' \\ y_i' \end{bmatrix} = \begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} + \begin{bmatrix} t_1 \\ t_2 \end{bmatrix}$$

$$\begin{bmatrix} \\ \\ \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ t_1 \\ t_2 \end{bmatrix} = \begin{bmatrix} \\ \\ \end{bmatrix}$$

Source: K. Grauman

Least squares problem

A *m x n* matrix of rank *n*:

$$Ax = b$$

Solution with pseudo-inverse:

$$A^T(Ax - b) = 0$$

$$x = (A^T A)^{-1} A^T b$$

Pseudo-inverse $\mathbf{A}^+$

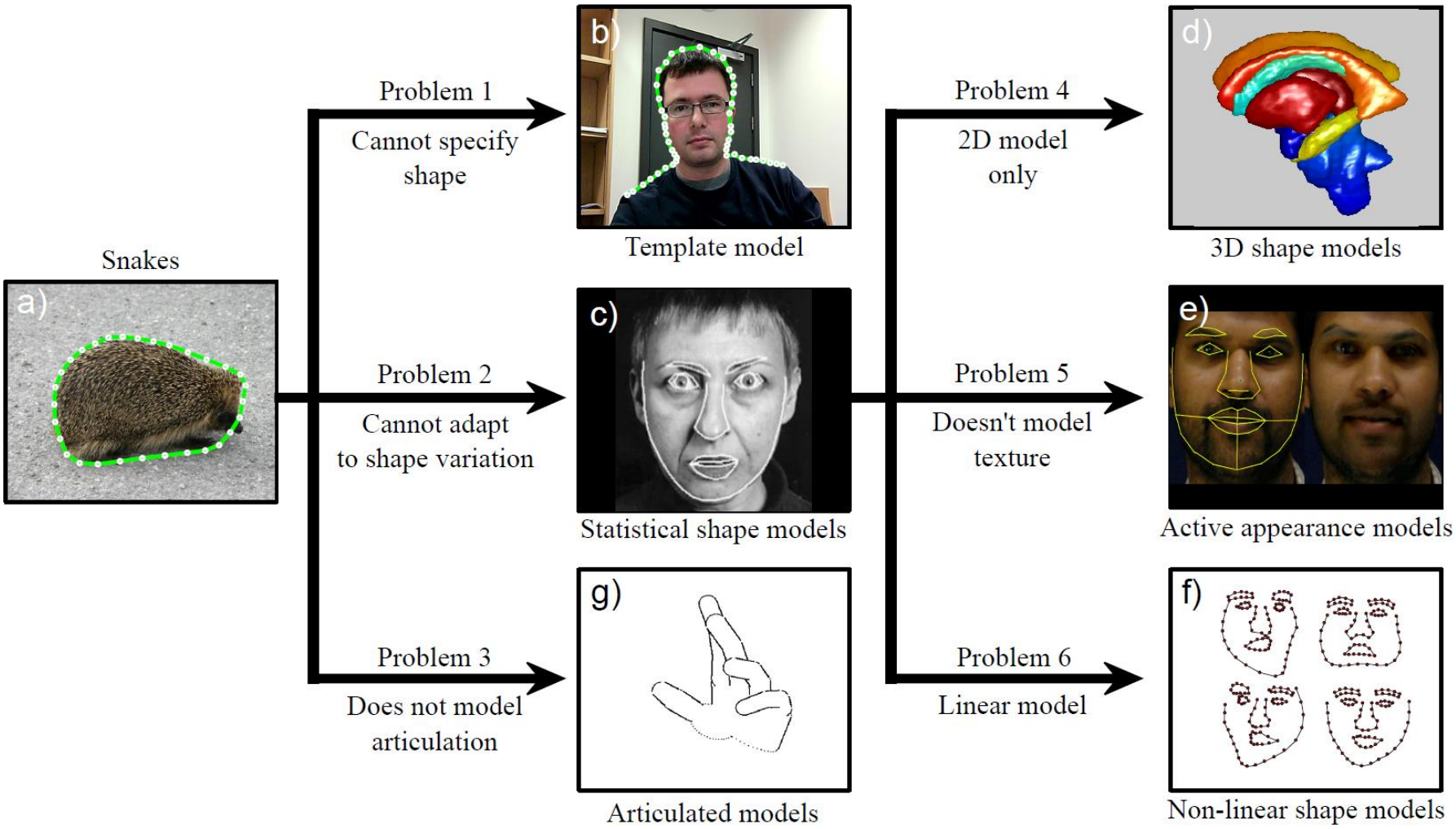
Solution with SVD:

- SVD: $\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{V}^T$; $\mathbf{A}^+ = \mathbf{V}\mathbf{D}^{-1}\mathbf{U}^T$
- $\mathbf{b}' = \mathbf{U}^T\mathbf{b}$
- $y_i = b'_i / D_{ii}$
- $\mathbf{x} = \mathbf{V}\mathbf{y}$

There are different ways to solve the problem. The optimal method depends on size and sparseness of **A**

Specific C++ linear algebra libraries exist!

Relationships between models



Source: S. Prince

Statistical shape models

- Also called
  - Point distribution models
  - Active shape models (as they adapt to the image)
- Likelihood:

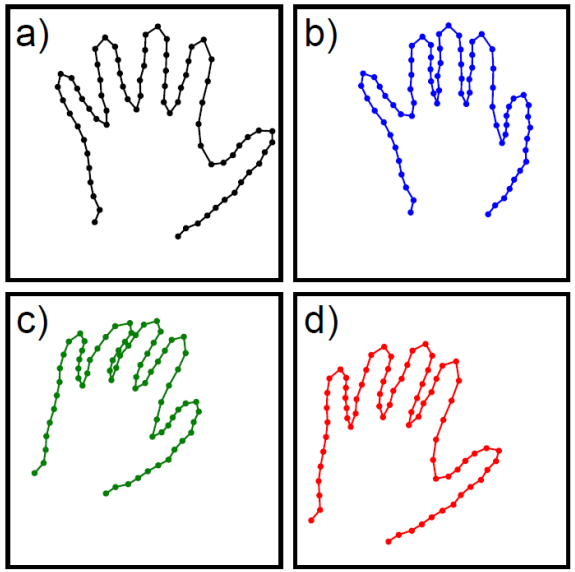
$$Pr(\mathbf{x}_i|\mathbf{w}_i) \propto \prod_{n=1}^{N} \exp\left[-(\text{dist}\left[\mathbf{x}_i, \mathbf{trans}[\mathbf{w}_{in}, \boldsymbol{\Psi}_i]\right])^2\right]$$

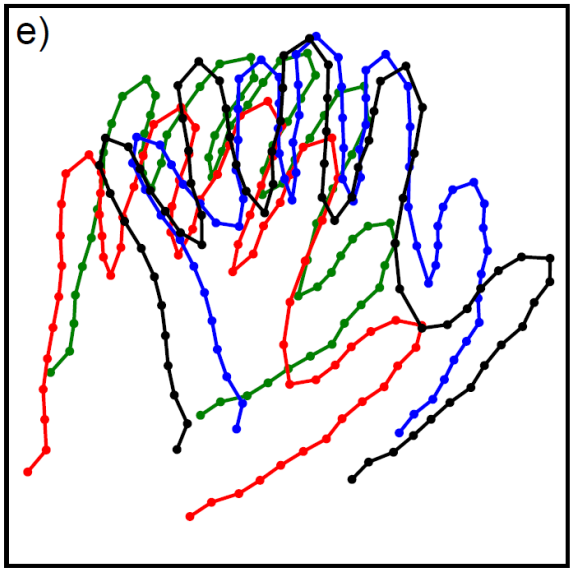
- Prior:

$$Pr(\mathbf{w}_i) = \text{Norm}_{\mathbf{w}_i}[\boldsymbol{\mu}, \boldsymbol{\Sigma}]$$

Source: S. Prince

Learning

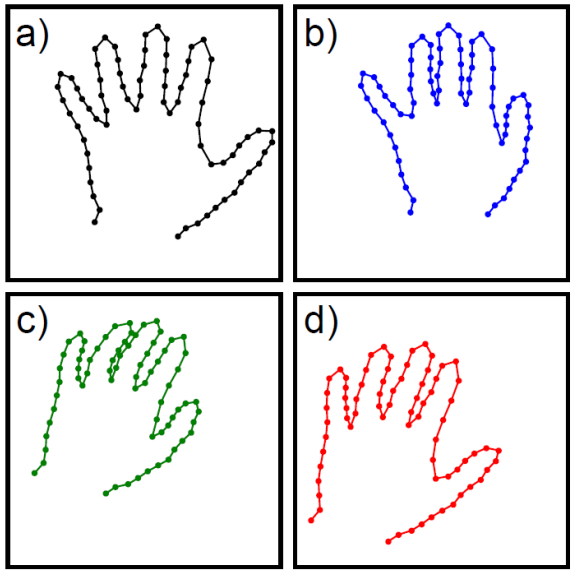




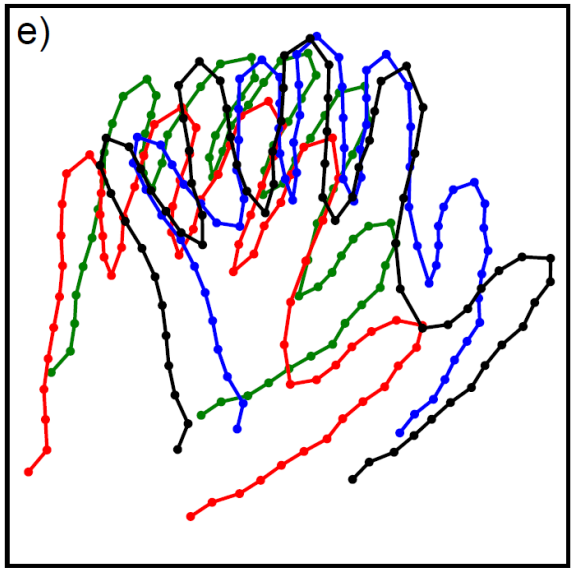
Training data

Source: S. Prince

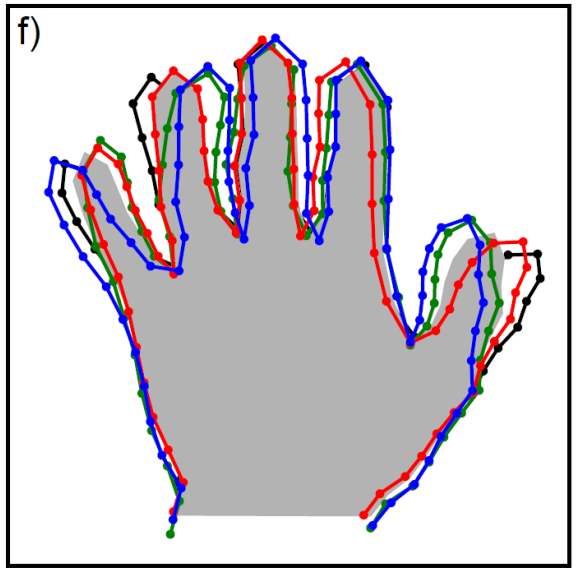
Generalized Procrustes analysis



Training data



Before alignment

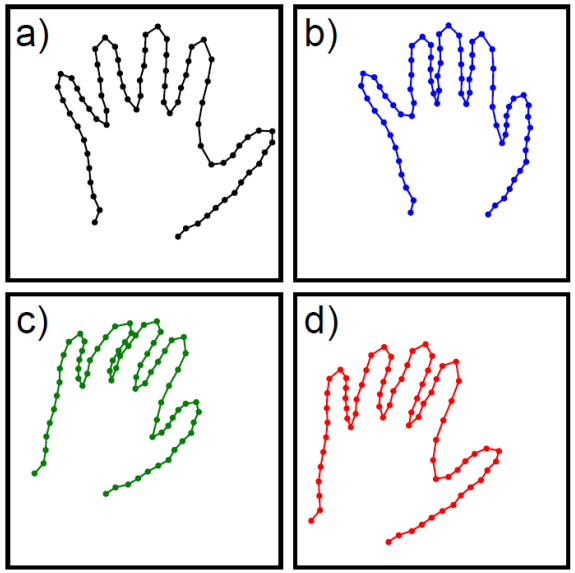


After alignment

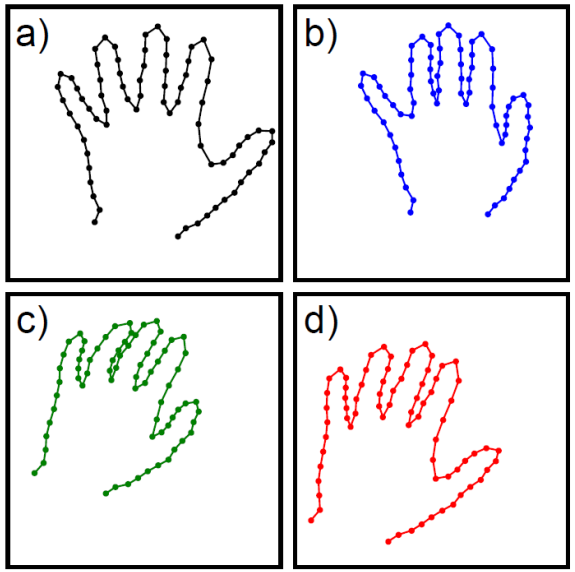
Source: S. Prince

Generalized Procrustes analysis

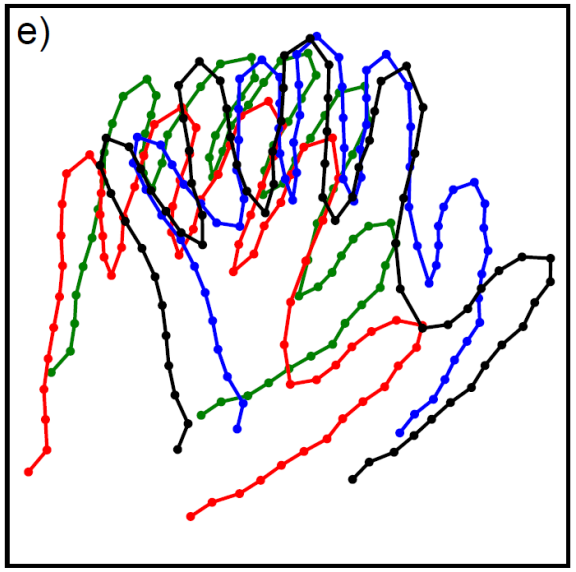
- Given I shapes with N landmarks
- Correspondences between shapes are known (landmarks)
- Take one shape as mean shape
- Iterate:
  - Solve for affine transformation ψ_i for each shape i to the mean $A_i x = b_i$
  - Apply transformation ψ_i to each shape i
  - Compute the mean μ_n of each landmark (mean shape)



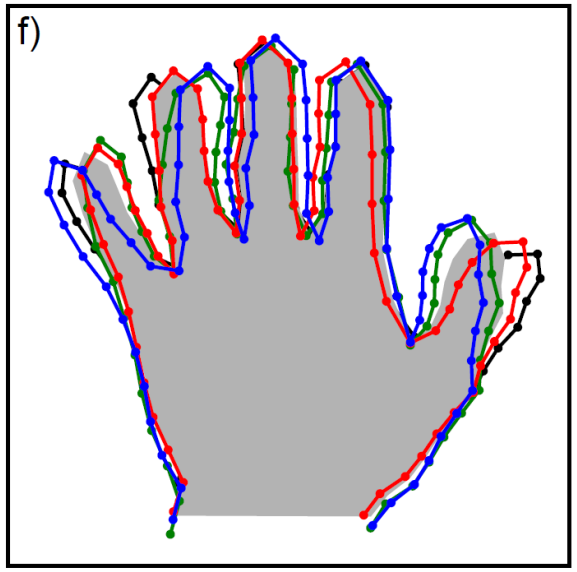
Generalized Procrustes analysis



Training data



Before alignment



After alignment

Source: S. Prince

Inference

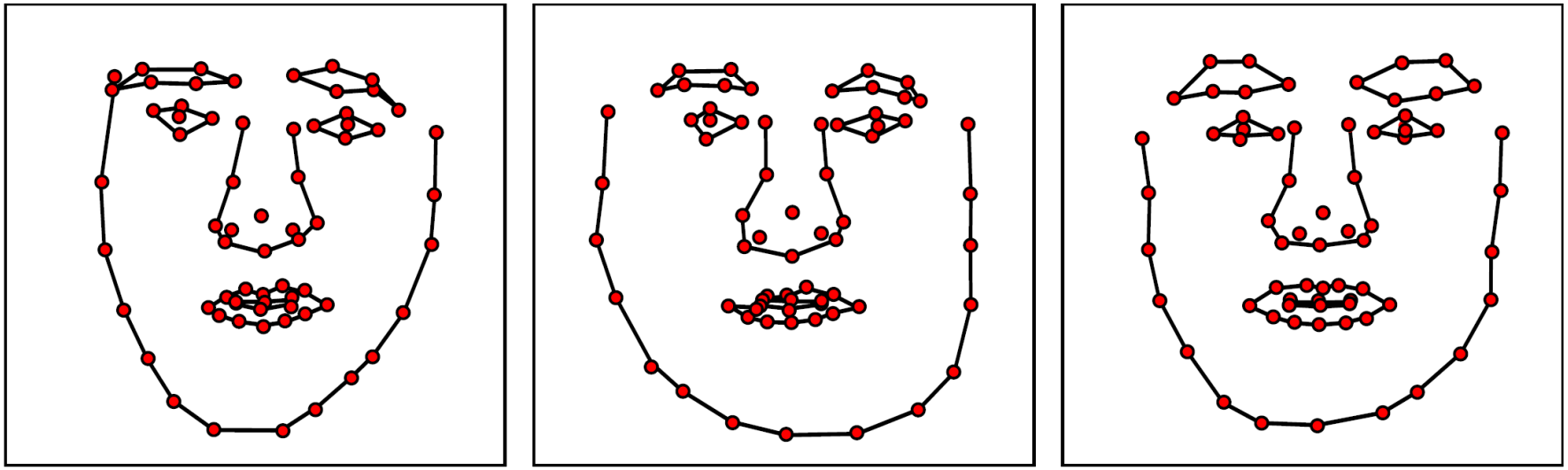
- Map inference:

$$\hat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmax}} \left[\max_{\boldsymbol{\Psi}} \left[\sum_{n=1}^{N} -\left(\operatorname{dist}\left[\mathbf{x}_i, \mathbf{trans}[\mathbf{w}_n, \boldsymbol{\Psi}]\right]\right)^2 + \log[\operatorname{Norm}_{\mathbf{w}}[\boldsymbol{\mu}, \boldsymbol{\Sigma}]] \right] \right]$$

- No closed form solution
- Use non-linear optimisation
- Or use ICP approach
- However, many parameters, and not clear they are all needed
- More efficient to use subspace model

Source: S. Prince

Face model



Three samples from learned model for faces

Source: S. Prince

Subspace shape model

- Generate data from model:

$$\mathbf{w}_i = \boldsymbol{\mu} + \boldsymbol{\Phi}\mathbf{h}_i + \epsilon_i$$

  - $\boldsymbol{\mu}$ is the mean shape
  - the matrix $\boldsymbol{\Phi} = [\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \ldots, \boldsymbol{\phi}_K]$ contains K basis functions in it columns
  - ϵ_i is normal noise with covariance $\sigma^2\mathbf{I}$
- Can alternatively write

$$\mathbf{w}_i = \boldsymbol{\mu} + \sum_{k=1}^{K} \boldsymbol{\phi}_k h_{ik} + \epsilon_i$$

Source: S. Prince

Approximating with subspace

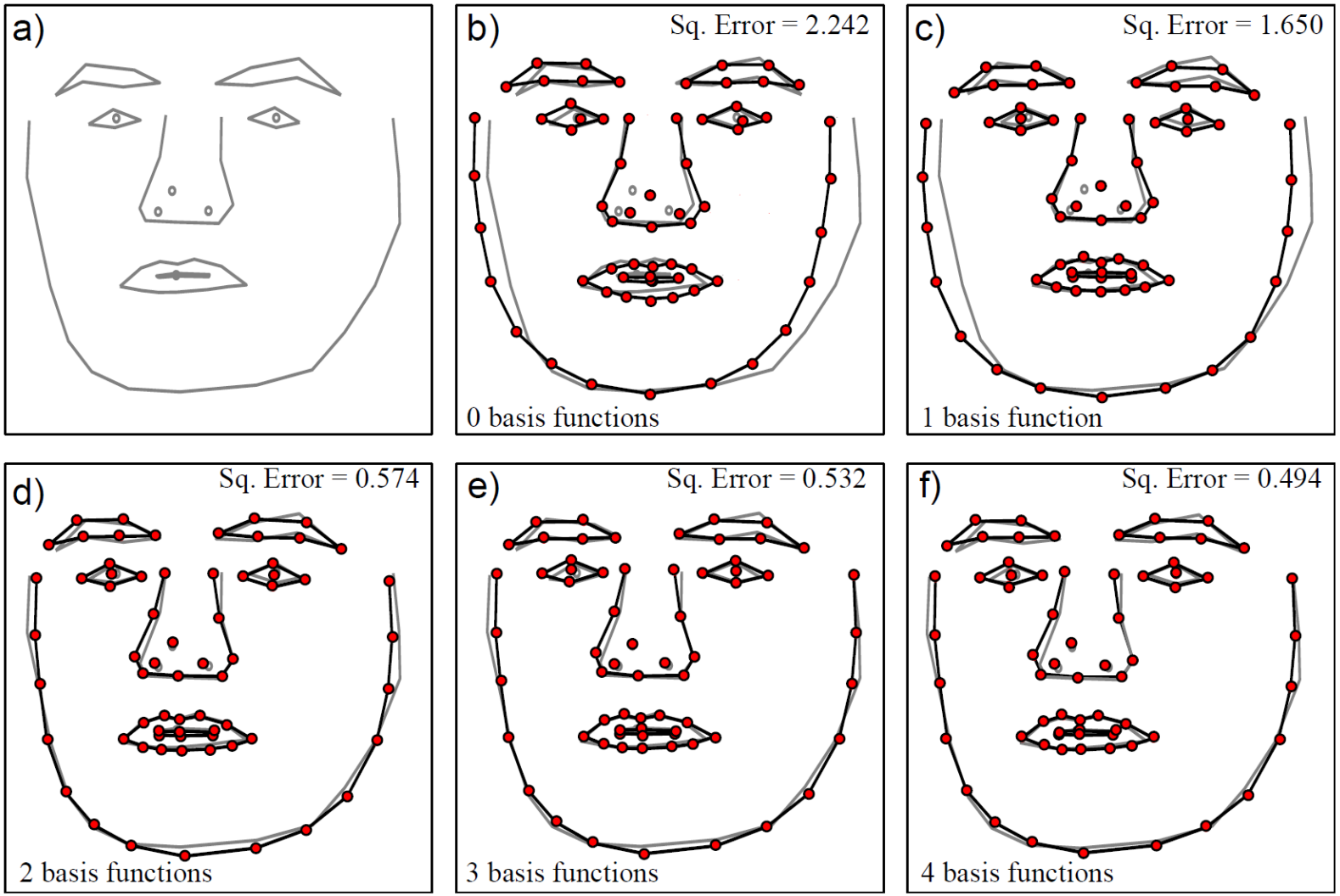
Can approximate a vector **w** with a weighted sum of the basis functions

$$\mathbf{w}_i \approx \boldsymbol{\mu} + \sum_{k=1}^{K} \phi_k h_{ik}$$

Surprising how well this works even with a small number of basis functions

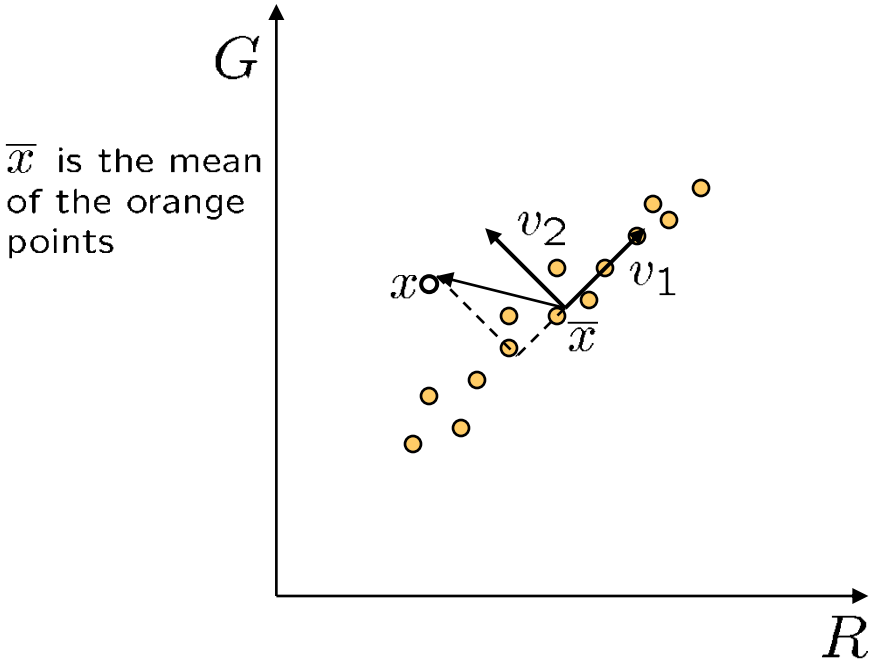
Source: S. Prince

Subspace shape model



Source: S. Prince

Linear subspaces



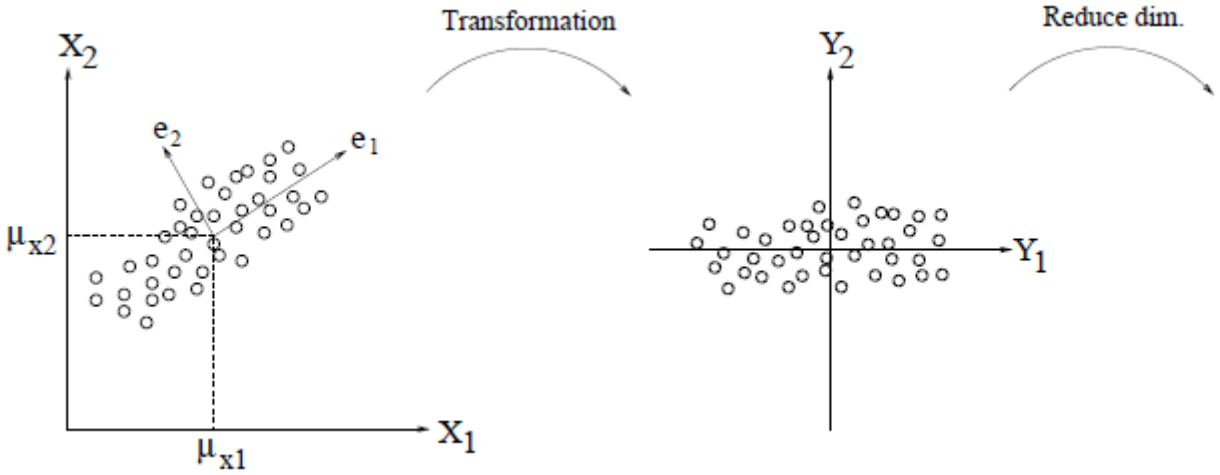
- convert **x** into $\mathbf{v_1}$, $\mathbf{v_2}$ coordinates

$$\mathbf{x} \rightarrow ((\mathbf{x} - \overline{x}) \cdot \mathbf{v_1}, (\mathbf{x} - \overline{x}) \cdot \mathbf{v_2})$$

- $\mathbf{v}_2$ is distance to line
- $\mathbf{v}_1$ is position along line

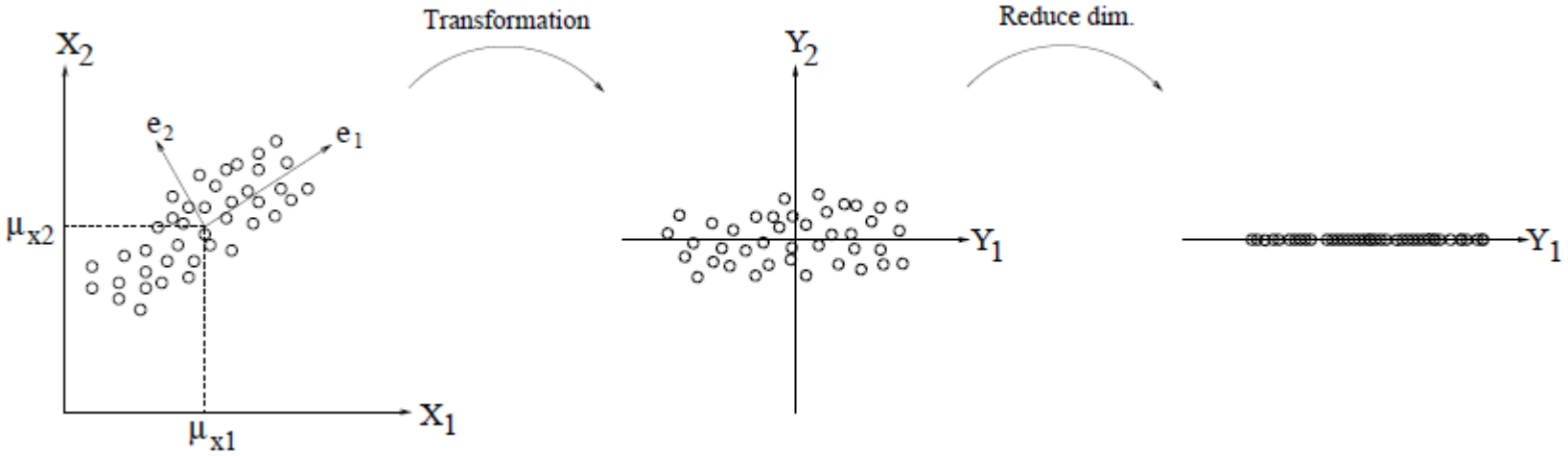
- Represent high-dimensional space by linear subspace
- Idea: Estimate new coordinate system from data points

Dimensionality reduction



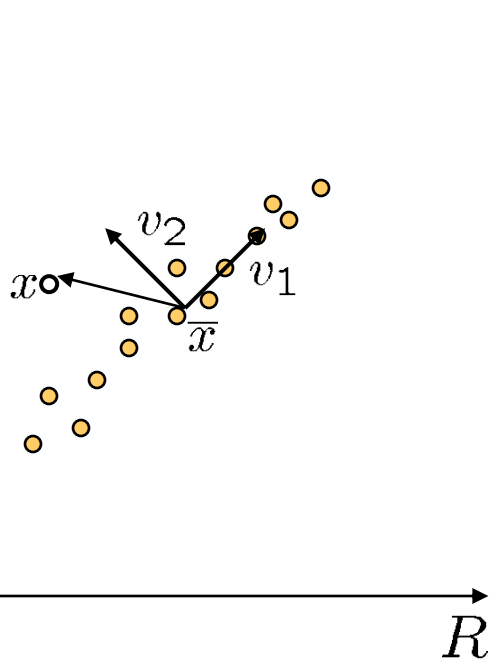
Source: T. Moeslund

Dimensionality reduction



Source: T. Moeslund

Linear subspaces



Consider the variation along direction **v** among all of the orange points:

$$var(\mathbf{v}) = \sum_{\text{orange point } \mathbf{x}} \|(\mathbf{x} - \overline{\mathbf{x}})^{\mathrm{T}} \cdot \mathbf{v}\|^2$$

$$\begin{aligned} var(\mathbf{v}) &= \sum_{\mathbf{x}} \|(\mathbf{x} - \overline{\mathbf{x}})^{\mathrm{T}} \cdot \mathbf{v}\|^2 \\ &= \sum_{\mathbf{x}} \mathbf{v}^{\mathrm{T}}(\mathbf{x} - \overline{\mathbf{x}})(\mathbf{x} - \overline{\mathbf{x}})^{\mathrm{T}}\mathbf{v} \\ &= \mathbf{v}^{\mathrm{T}}\left[\sum_{\mathbf{x}}(\mathbf{x} - \overline{\mathbf{x}})(\mathbf{x} - \overline{\mathbf{x}})^{\mathrm{T}}\right]\mathbf{v} \\ &= \mathbf{v}^{\mathrm{T}}\mathbf{A}\mathbf{v} \quad \text{where } \mathbf{A} = \sum_{\mathbf{x}}(\mathbf{x} - \overline{\mathbf{x}})(\mathbf{x} - \overline{\mathbf{x}})^{\mathrm{T}} \end{aligned}$$

Recall: Eigenvalue Problem

UNIVERSITÄT BONN

- Maximize objective J (Rayleigh Quotient):

$$J(\mathbf{w}) = \frac{\mathbf{w}^T S_B \mathbf{w}}{\mathbf{w}^T S_W \mathbf{w}}$$

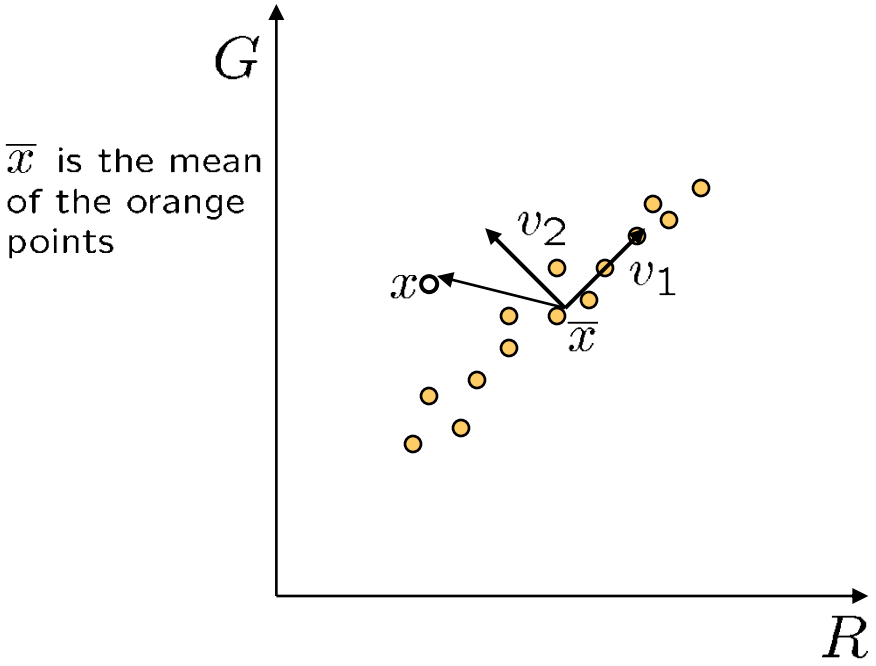
- Equivalent to

$$\begin{aligned} \min_{\mathbf{w}} \quad & -\frac{1}{2}\mathbf{w}^T S_B \mathbf{w} \\ \text{s.t.} \quad & \mathbf{w}^T S_W \mathbf{w} = 1 \end{aligned}$$

- Generalized eigenvalue problem:

$$S_B \mathbf{w} = \lambda S_W \mathbf{w}$$

Linear subspaces



Consider the variation along direction **v** among all of the orange points:

$$var(\mathbf{v}) = \sum_{\text{orange point } \mathbf{x}} \|(\mathbf{x} - \overline{\mathbf{x}})^{\mathrm{T}} \cdot \mathbf{v}\|^2$$

$$\begin{aligned} var(\mathbf{v}) &= \sum_{\mathbf{x}} \|(\mathbf{x} - \overline{\mathbf{x}})^{\mathrm{T}} \cdot \mathbf{v}\|^2 \\ &= \sum_{\mathbf{x}} \mathbf{v}^{\mathrm{T}}(\mathbf{x} - \overline{\mathbf{x}})(\mathbf{x} - \overline{\mathbf{x}})^{\mathrm{T}}\mathbf{v} \\ &= \mathbf{v}^{\mathrm{T}} \left[\sum_{\mathbf{x}} (\mathbf{x} - \overline{\mathbf{x}})(\mathbf{x} - \overline{\mathbf{x}})^{\mathrm{T}}\right] \mathbf{v} \\ &= \mathbf{v}^{\mathrm{T}}\mathbf{A}\mathbf{v} \quad \text{where } \mathbf{A} = \sum_{\mathbf{x}} (\mathbf{x} - \overline{\mathbf{x}})(\mathbf{x} - \overline{\mathbf{x}})^{\mathrm{T}} \end{aligned}$$

Solution: $\mathbf{v}_1$ is eigenvector of **A** with *largest* eigenvalue
$\mathbf{v}_2$ is eigenvector of **A** with *smallest* eigenvalue

Principal component analysis (PCA)

UNIVERSITÄT BONN

- Suppose each data point is N-dimensional
- Same procedure applies:

$$\begin{aligned} var(\mathbf{v}) &= \sum_{\mathbf{x}} \|(\mathbf{x}-\overline{\mathbf{x}})^{\mathrm{T}} \cdot \mathbf{v}\| \\ &= \mathbf{v}^{\mathrm{T}}\mathbf{A}\mathbf{v} \quad \text{where } \mathbf{A} = \sum_{\mathbf{x}} (\mathbf{x}-\overline{\mathbf{x}})(\mathbf{x}-\overline{\mathbf{x}})^{\mathrm{T}} \end{aligned}$$

- The eigenvectors of **A** define a new coordinate system
  - eigenvector with largest eigenvalue captures the most variation among training vectors **x**
  - eigenvector with smallest eigenvalue has least variation
- We can compress the data by only using the top few eigenvectors
  - corresponds to choosing a “linear subspace”
    - represent points on a line, plane, or “hyper-plane”
  - these eigenvectors are known as the ***principal components***

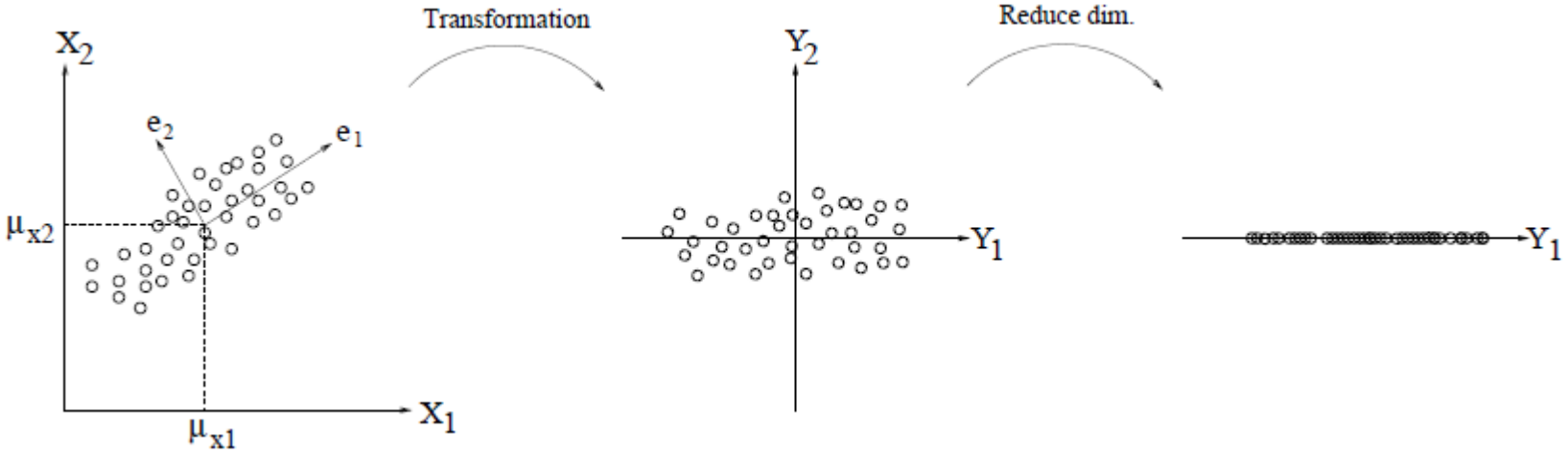
Principal component analysis (PCA)

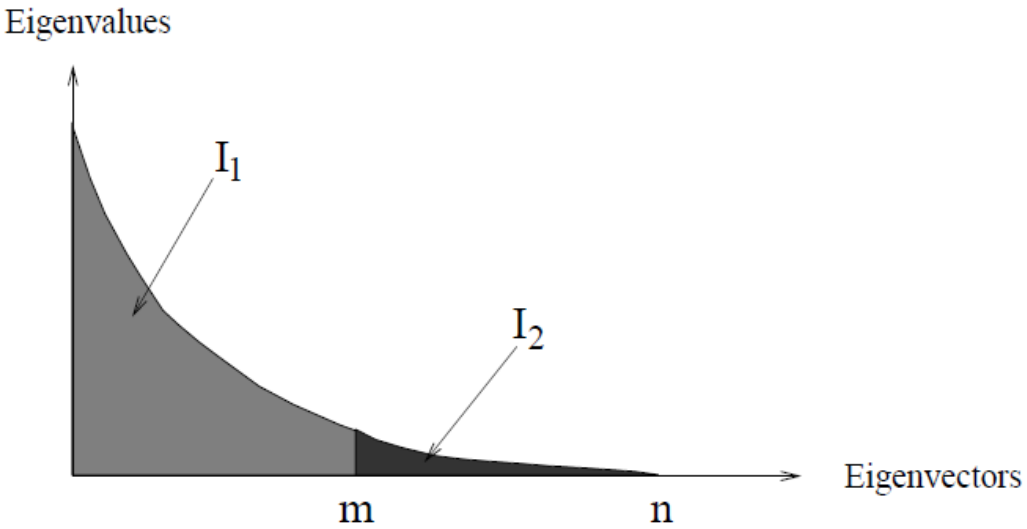
UNIVERSITÄT BONN

- Given data points w_i
- Compute mean μ
- $W = (w_1 - \mu, \ldots, w_n - \mu)$
- Eigenvalue decomposition:
  $WW^T = U\Sigma V^T V\Sigma U^T = UL^2U^T$ (u_i basis vectors)
- SVD: $W = U\Sigma V^T$

Principal component analysis (PCA)

- Dimensionality reduction



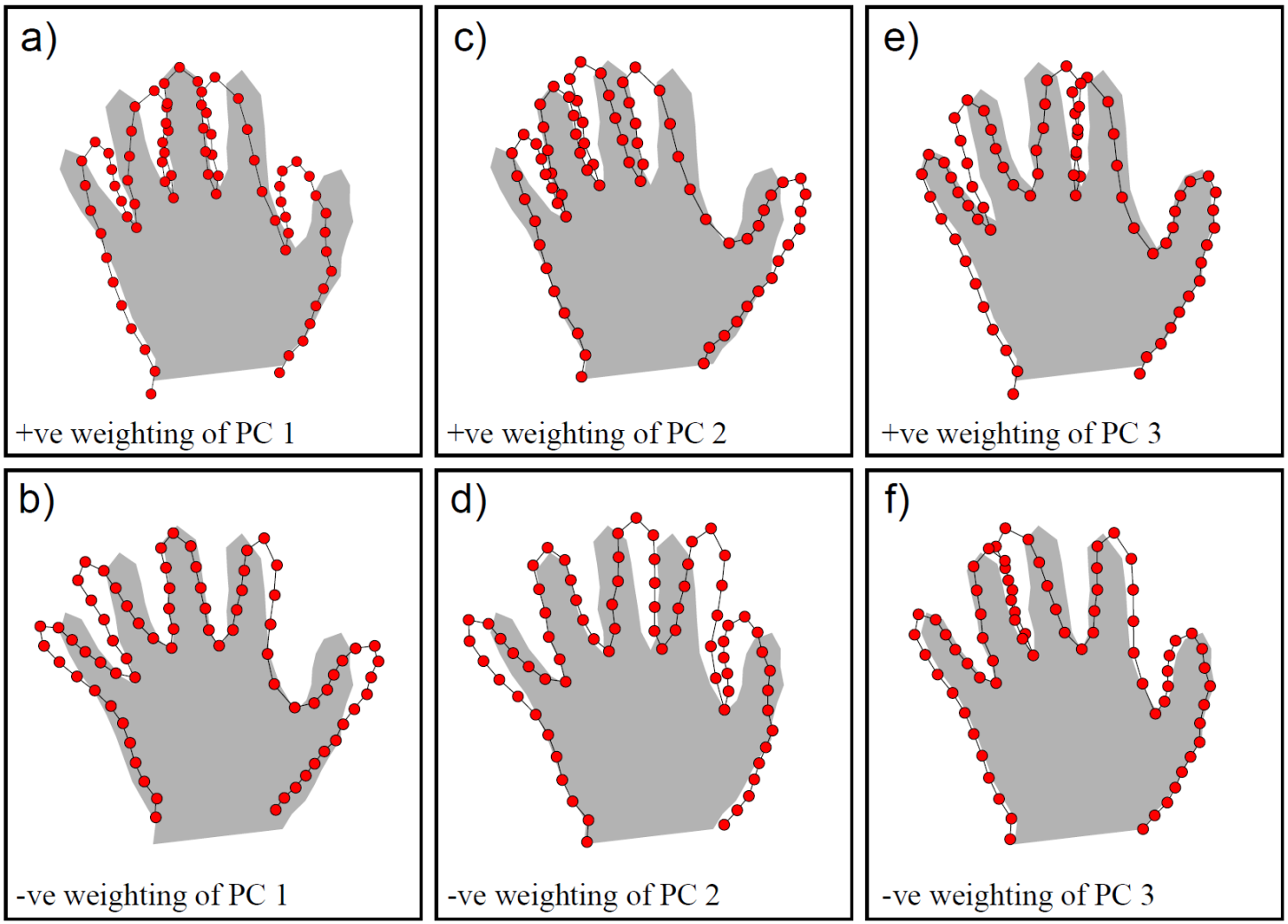


$$\frac{\sum_{i=1}^{i=K} L_{ii}^2}{\sum_{i=1}^{i=D} L_{ii}^2} > 1 - \epsilon$$

Source: T. Moeslund

Learned hand model

$$\boldsymbol{\mu} + \sum_{k=1}^{K} \phi_k h_{ik}$$



Source: S. Prince

Probabilistic PCA

Generative eq: $\mathbf{w}_i = \boldsymbol{\mu} + \boldsymbol{\Phi}\mathbf{h}_i + \epsilon_i$

Probabilistic version:

$$Pr(\mathbf{w}_i|\mathbf{h}_i, \boldsymbol{\mu}, \boldsymbol{\Phi}, \sigma^2) = \mathrm{Norm}_{\mathbf{w}_i}[\boldsymbol{\mu} + \boldsymbol{\Phi}\mathbf{h}_i, \sigma^2\mathbf{I}]$$

Add prior: $Pr(\mathbf{h}_i) = \mathrm{Norm}_{\mathbf{h}_i}[\mathbf{0}, \mathbf{I}]$

Density:

$$\begin{aligned} Pr(\mathbf{w}_i) &= \int Pr(\mathbf{w}_i|\mathbf{h}_i)Pr(\mathbf{h}_i)d\mathbf{h}_i \\ &= \int \mathrm{Norm}_{\mathbf{w}_i}[\boldsymbol{\mu} + \boldsymbol{\Phi}\mathbf{h}_i, \sigma^2\mathbf{I}]\mathrm{Norm}_{\mathbf{h}_i}[\mathbf{0}, \mathbf{I}]d\mathbf{h}_i \\ &= \mathrm{Norm}_{\mathbf{w}_i}[\boldsymbol{\mu}, \boldsymbol{\Phi}\boldsymbol{\Phi}^T + \sigma^2\mathbf{I}]. \end{aligned}$$

Source: S. Prince

Probabilistic PCA

$$\int \mathrm{Norm}_{w_i}[\mu + \Phi h_i, \sigma^2 I]\,\mathrm{Norm}_{h_i}[0, I]dh_i$$

$$\text{using} \quad w = \mu + \Phi h_i, \quad \Phi^{-1}(w - \mu) = h_i$$

$$= \int \mathrm{Norm}_{w_i}[\mu + \Phi\Phi^{-1}(w - \mu), \sigma^2 I]\,\mathrm{Norm}_{w}[\Phi 0 + \mu, \Phi I \Phi^T]dw$$

$$= \int \mathrm{Norm}_{w_i}[w, \sigma^2 I]\,\mathrm{Norm}_{w}[\mu, \Phi\Phi^T]dw$$

$$\text{using} \quad \int \mathrm{Norm}_x[a, A]\,\mathrm{Norm}_x[b, B]dx = \mathrm{Norm}_a[b, A + B]$$

$$= \mathrm{Norm}_{w_i}[\mu, \sigma^2 I + \Phi\Phi^T]$$

Learning PPCA

Learn parameters $\boldsymbol{\mu}, \boldsymbol{\Phi}$ and σ^2 from data $\{\mathbf{w}_i\}_{i=1}^{I}$ where $\mathbf{w}_i = [\mathbf{w}_{i1}^T, \mathbf{w}_{i2}^T \ldots, \mathbf{w}_{iN}^T]^T$.

Learn mean: $\boldsymbol{\mu} = \frac{\sum_{i=1}^{I} \mathbf{w}_i}{I}$

Then set $\mathbf{W} = [\mathbf{w}_1 - \boldsymbol{\mu}, \mathbf{w}_2 - \boldsymbol{\mu}, \ldots, \mathbf{w}_I - \boldsymbol{\mu}]$ and compute eigen-decomposition

$$\mathbf{W}\mathbf{W}^T = \mathbf{U}\mathbf{L}^2\mathbf{U}^T$$

Choose parameters

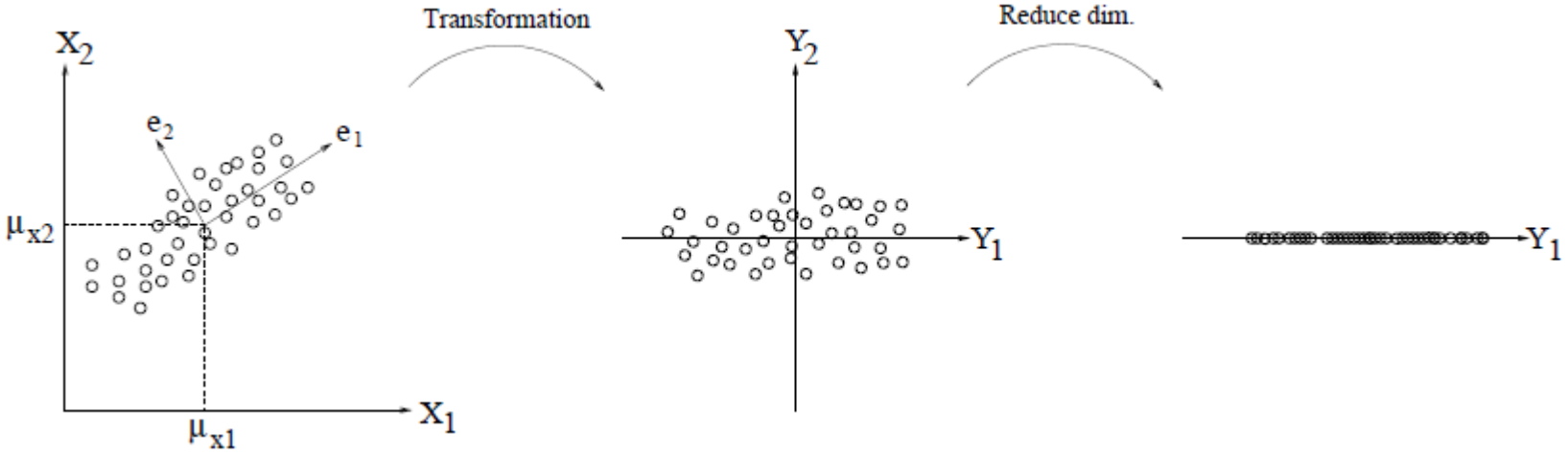
$$\hat{\sigma}^2 = \frac{1}{D-K} \sum_{j=K+1}^{D} L_{jj}^2$$

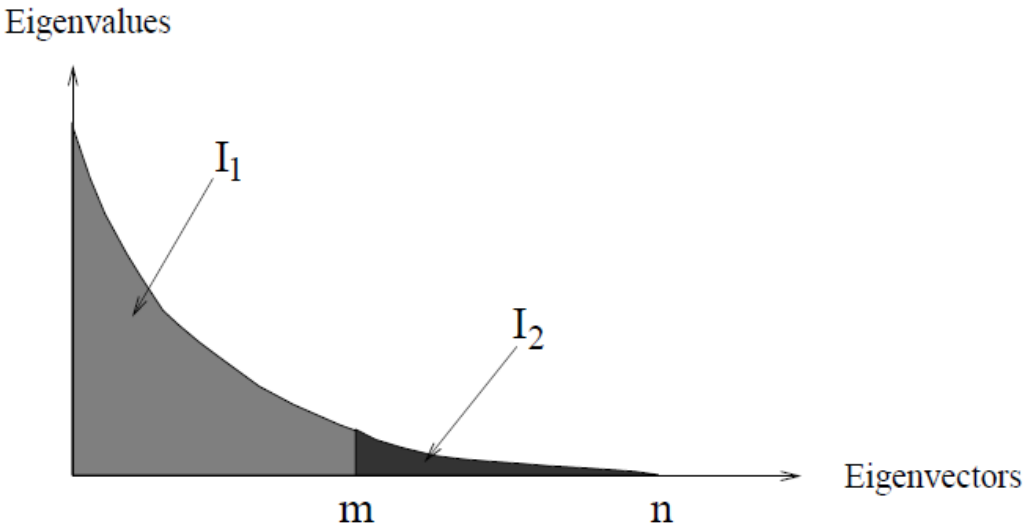
$$\hat{\boldsymbol{\Phi}} = \mathbf{U}_K(\mathbf{L}_K^2 - \hat{\sigma}^2\mathbf{I})^{1/2}$$

Source: S. Prince

Principal component analysis (PCA)

- Dimensionality reduction





$$\frac{\sum_{i=1}^{i=K} L_{ii}^2}{\sum_{i=1}^{i=D} L_{ii}^2} > 1 - \epsilon$$

Source: T. Moeslund

Properties of basis functions

Learning of parameters based on eigen-decomposition:

$$\mathbf{W}\mathbf{W}^T = \mathbf{U}\mathbf{L}^2\mathbf{U}^T$$

Parameters for representing D-dim. Data by K principal components:

$$\hat{\sigma}^2 = \frac{1}{D-K}\sum_{j=K+1}^{D} L_{jj}^2$$ Estimate noise

$$\hat{\mathbf{\Phi}} = \mathbf{U}_K(\mathbf{L}_K^2 - \hat{\sigma}^2\mathbf{I})^{1/2}$$ As PCA but remove noise

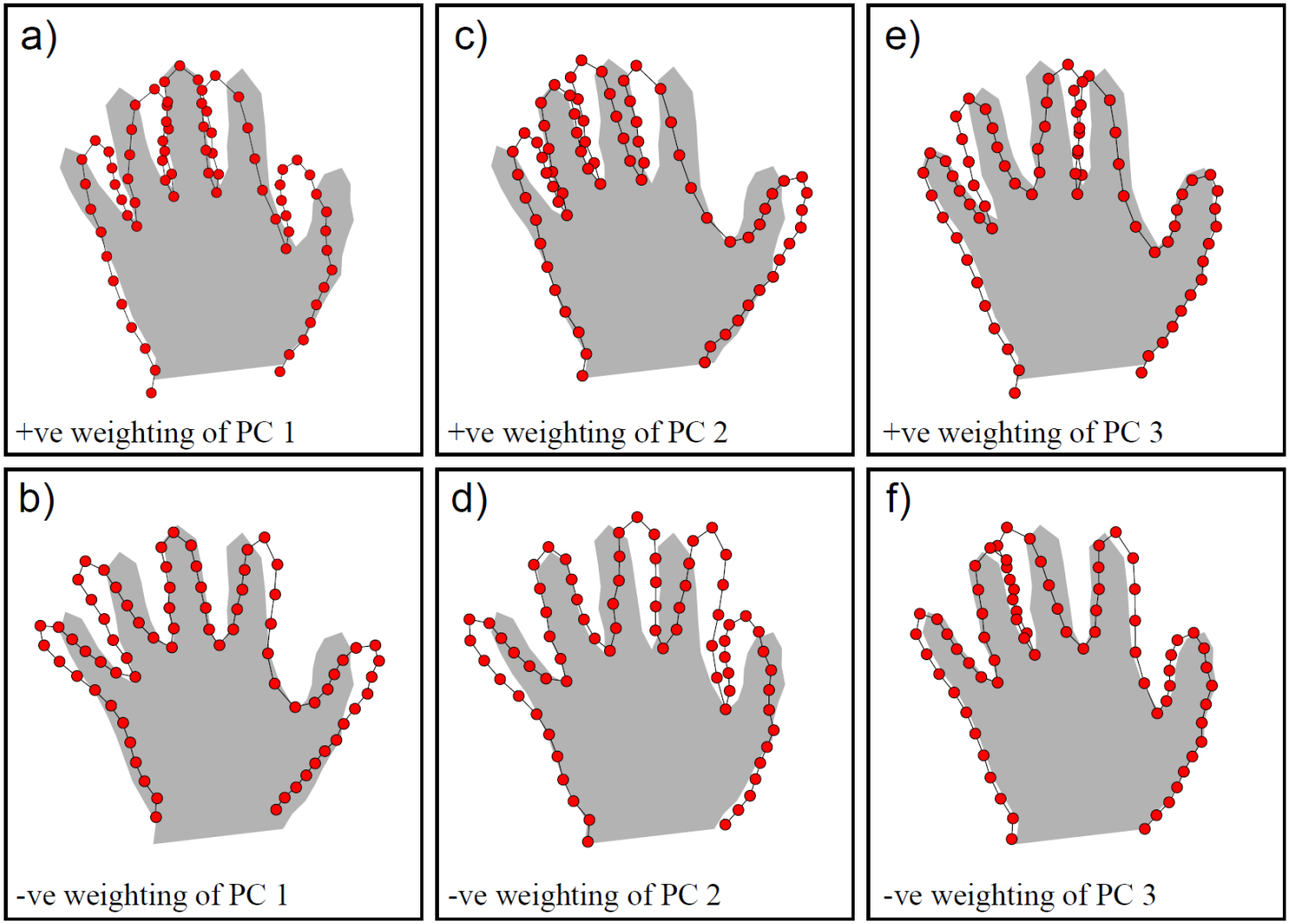
Notice that:

- Basis functions in $\mathbf{\Phi}$ are orthogonal
- Basis functions in $\mathbf{\Phi}$ are ordered

Source: S. Prince

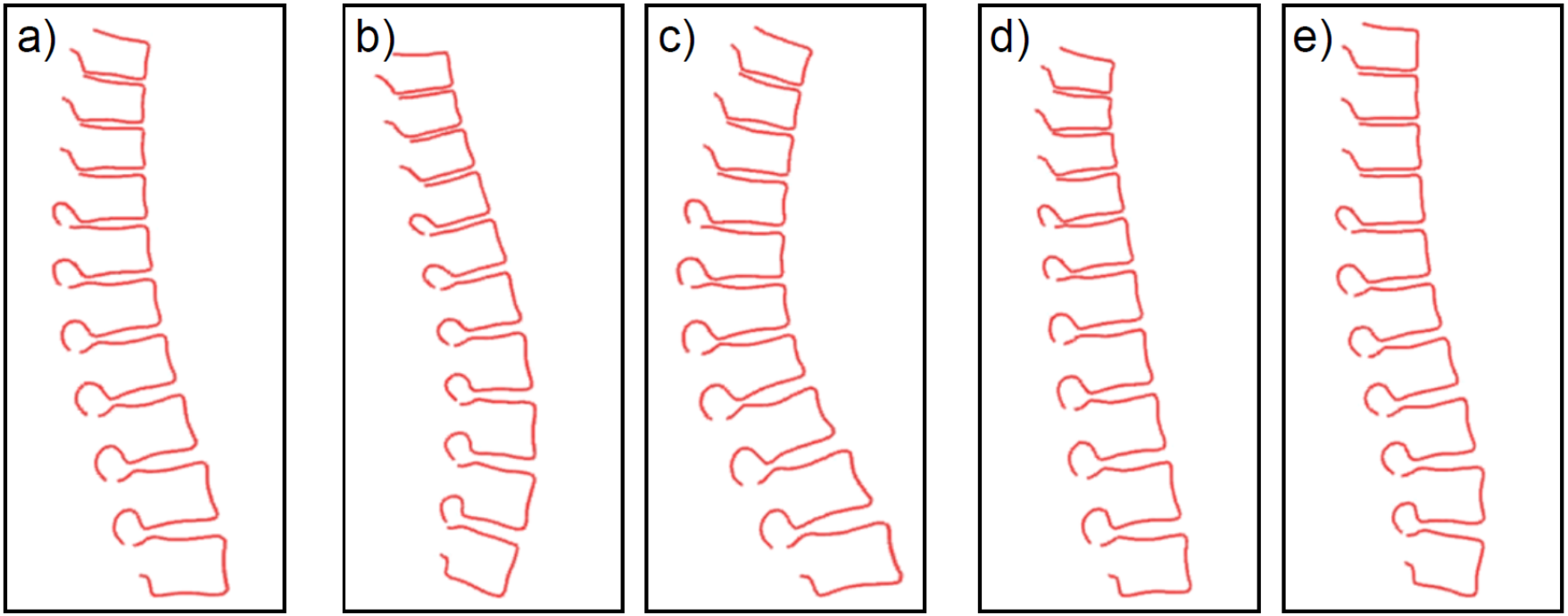
Learned hand model

$$\boldsymbol{\mu} + \sum_{k=1}^{K} \phi_k h_{ik}$$



Source: S. Prince

Learned spine model



Mean

Manipulating first principal component

Manipulating second principal component

Source: T. Cootes

Inference

To fit model to an image:

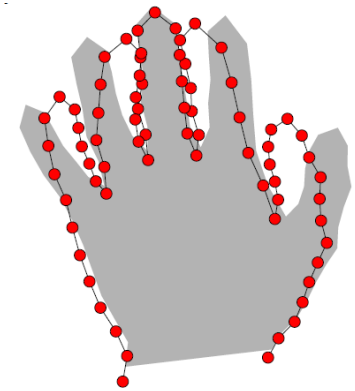
$$\hat{\mathbf{h}} = \underset{\mathbf{h}}{\operatorname{argmax}} \left[\max_{\boldsymbol{\Psi}} \left[\sum_{n=1}^{N} \left(-\frac{\left(\operatorname{dist}\left[\mathbf{x}_i, \mathbf{trans}[\boldsymbol{\mu}_n + \boldsymbol{\Phi}_n \mathbf{h}, \boldsymbol{\Psi}]\right]\right)^2}{\sigma^2} \right) + \log[\operatorname{Norm}_{\mathbf{h}}[\mathbf{0}, \mathbf{I}]] \right] \right]$$

likelihood

prior

ICP Approach:

- Find closest points to current prediction
- Update weightings $\mathbf{h}$
- Find closest points to current prediction
- Update transformation parameters $\boldsymbol{\psi}$



Source: S. Prince

Inference

1. Update weightings $\mathbf{h}$

$$\begin{aligned}\hat{\mathbf{h}} &= \underset{\mathbf{h}}{\operatorname{argmax}} \left[\sum_{n=1}^{N} \log[Pr(\mathbf{y}_n|\mathbf{h}), \boldsymbol{\Psi}] + \log[Pr(\mathbf{h})]\right] \\ &= \underset{\mathbf{h}}{\operatorname{argmax}} \left[\sum_{n=1}^{N} -\left(\mathbf{y}_n - \operatorname{trans}[\boldsymbol{\mu}_n + \boldsymbol{\Phi}_n\mathbf{h}, \boldsymbol{\Psi}]\right)^2/\sigma^2 - \log[\mathbf{h}^T\mathbf{h}]\right]\end{aligned}$$

If transformation parameters can be represented as $\mathbf{Aw} + \mathbf{b}$

$$\hat{\mathbf{h}} = \left(\sigma^2\mathbf{I} + \sum_{n=1}^{N} \boldsymbol{\Phi}_n^T\mathbf{A}^T\mathbf{A}\boldsymbol{\Phi}_n\right)^{-1} \sum_{n=1}^{N} \boldsymbol{\Phi}_n^T\mathbf{A}^T(\mathbf{y}_n - \mathbf{A}\boldsymbol{\mu}_n - \mathbf{b})$$

2. Update transformation parameters $\boldsymbol{\psi}$
   - Using one of the closed form solutions

Source: S. Prince

Simple version with PCA

UNIVERSITÄT BONN

1. Initialise the shape parameters, $\mathbf{b}$, to zero (the mean shape).
2. Generate the model point positions using $\mathbf{x} = \bar{\mathbf{x}} + \mathbf{P}\mathbf{b}$
3. Find the pose parameters (X_t, Y_t, s, θ) which best align the model points $\mathbf{x}$ to the current found points $\mathbf{Y}$

$$T_{X_t,Y_t,s,\theta}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} X_t \\ Y_t \end{pmatrix} + \begin{pmatrix} s\cos\theta & -s\sin\theta \\ s\sin\theta & s\cos\theta \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix}$$

4. Project $\mathbf{Y}$ into the model co-ordinate frame by inverting the transformation T:

$$\mathbf{y} = T^{-1}_{X_t,Y_t,s,\theta}(\mathbf{Y})$$

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

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

7. If not converged, return to step 2.

T. Cootes. An Introduction to Active Shape Models.

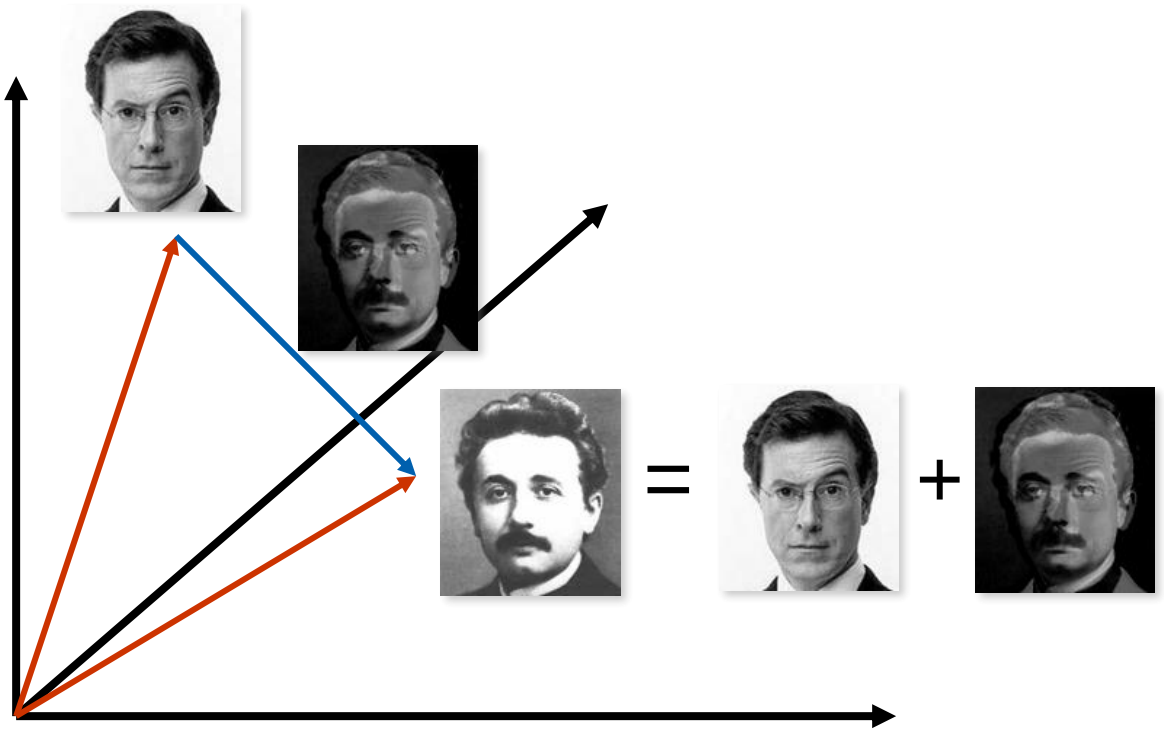
Fitting model

Figure provided by Tim Cootes

Much better to use statistical classifier instead of just distance from edges

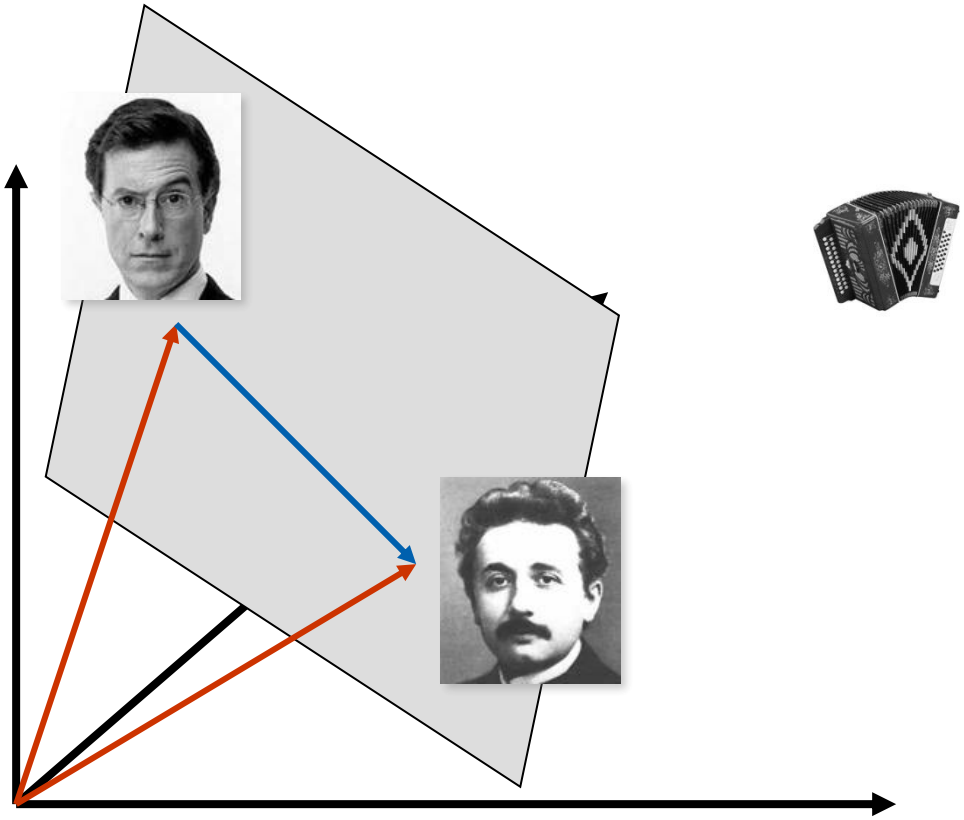
Source: S. Prince

The space of faces



- An image is a point in a high dimensional space
  - An N x M intensity image is a point in R^{NM}
  - We can define vectors in this space as we did in the 2D case

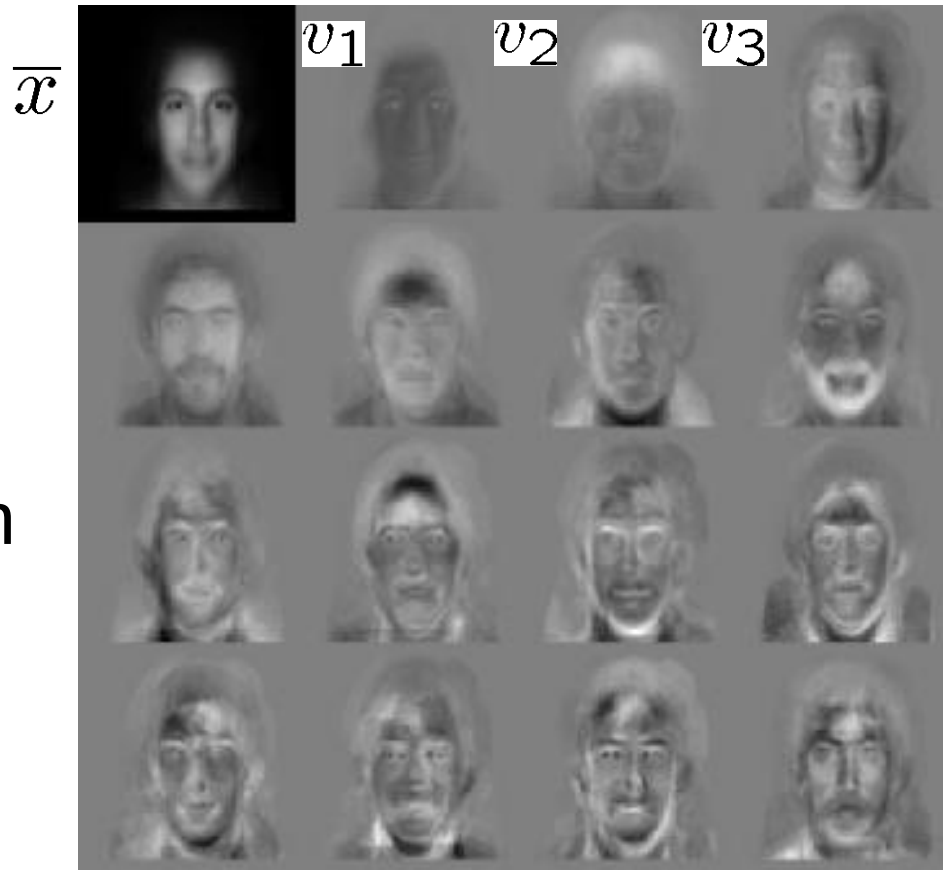
Dimensionality reduction



- The set of faces is a “subspace” of the set of images
  - Suppose it is K dimensional
  - We can find the best subspace using PCA
  - This is like fitting a “hyper-plane” to the set of faces
    - spanned by vectors $\mathbf{v_1}$, $\mathbf{v_2}$, ..., $\mathbf{v_K}$
    - any face $\mathbf{x} \approx \overline{\mathbf{x}} + a_1\mathbf{v}_1 + a_2\mathbf{v}_2 + \ldots + a_k\mathbf{v_k}$

Eigenfaces

- PCA extracts the eigenvectors of **A**
  - Gives a set of vectors **v_1**, **v_2**, **v_3**, ...
  - Each one of these vectors is a direction in face space
    - what do these look like?

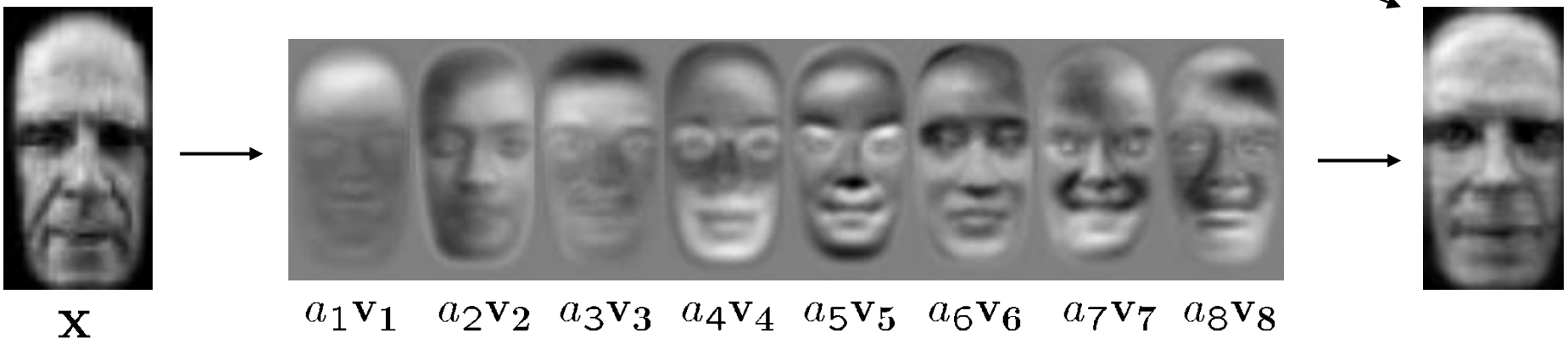


Projecting onto the eigenfaces

- The eigenfaces $\mathbf{v_1}$, ..., $\mathbf{v_K}$ span the space of faces
- A face is converted to eigenface coordinates by

$$\mathbf{x} \rightarrow (\underbrace{(\mathbf{x}-\overline{\mathbf{x}})\cdot\mathbf{v_1}}_{a_1}, \underbrace{(\mathbf{x}-\overline{\mathbf{x}})\cdot\mathbf{v_2}}_{a_2}, \ldots, \underbrace{(\mathbf{x}-\overline{\mathbf{x}})\cdot\mathbf{v_K}}_{a_K})$$

$$\mathbf{x} \approx \overline{\mathbf{x}} + a_1\mathbf{v_1} + a_2\mathbf{v_2} + \ldots + a_K\mathbf{v_K}$$



Detection and recognition with eigenfaces

UNIVERSITÄT BONN

Algorithm

1. Process the image database (set of images with labels)
   - Run PCA—compute eigenfaces
   - Calculate the K coefficients for each image
2. Given a new image (to be recognized) **x**, calculate K coefficients

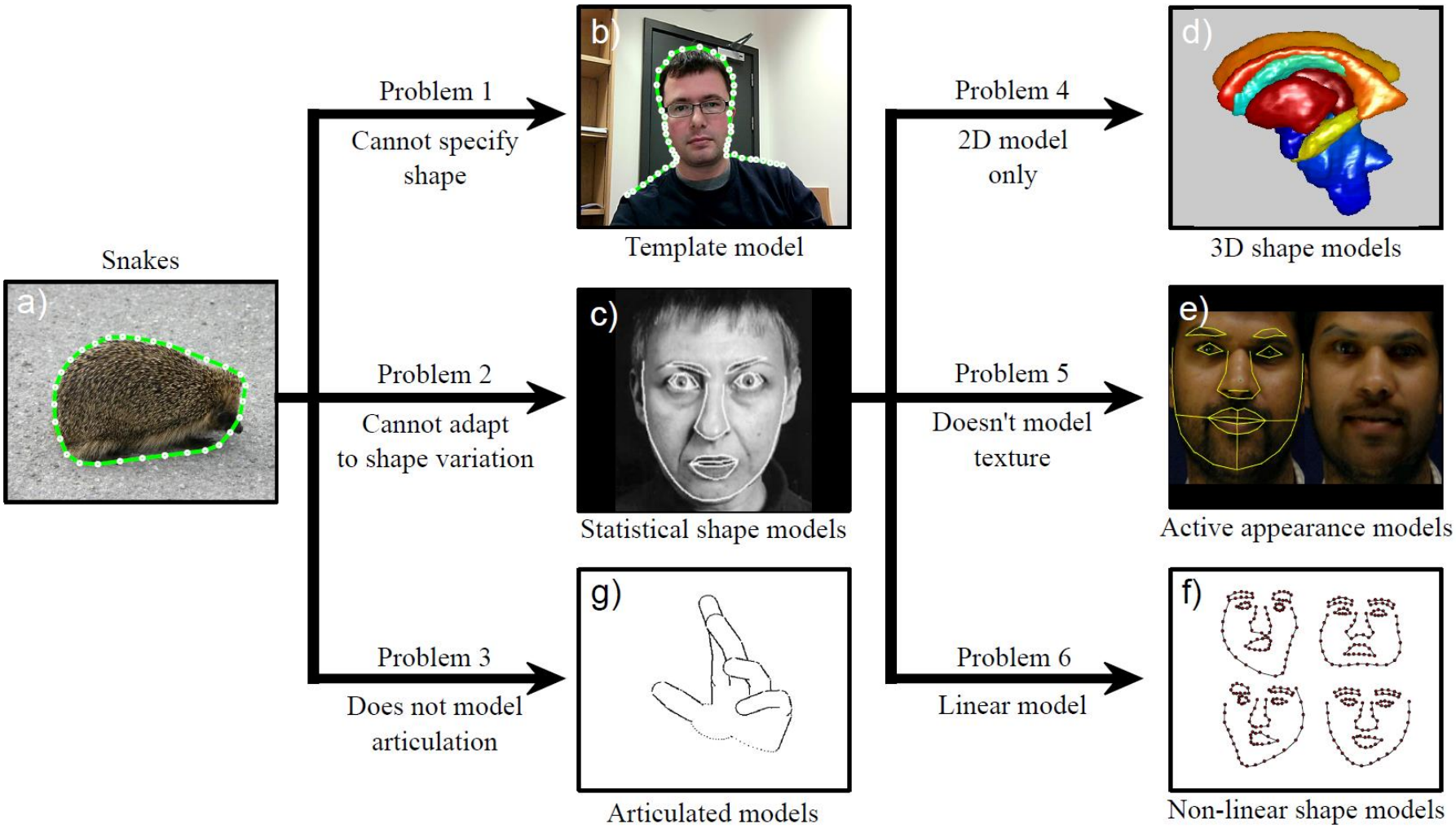
$$\mathbf{x} \rightarrow (a_1, a_2, \ldots, a_K)$$

3. Detect if x is a face

$$\|\mathbf{x} - (\bar{\mathbf{x}} + a_1\mathbf{v_1} + a_2\mathbf{v_2} + \ldots + a_K\mathbf{v_K})\| < \text{threshold}$$

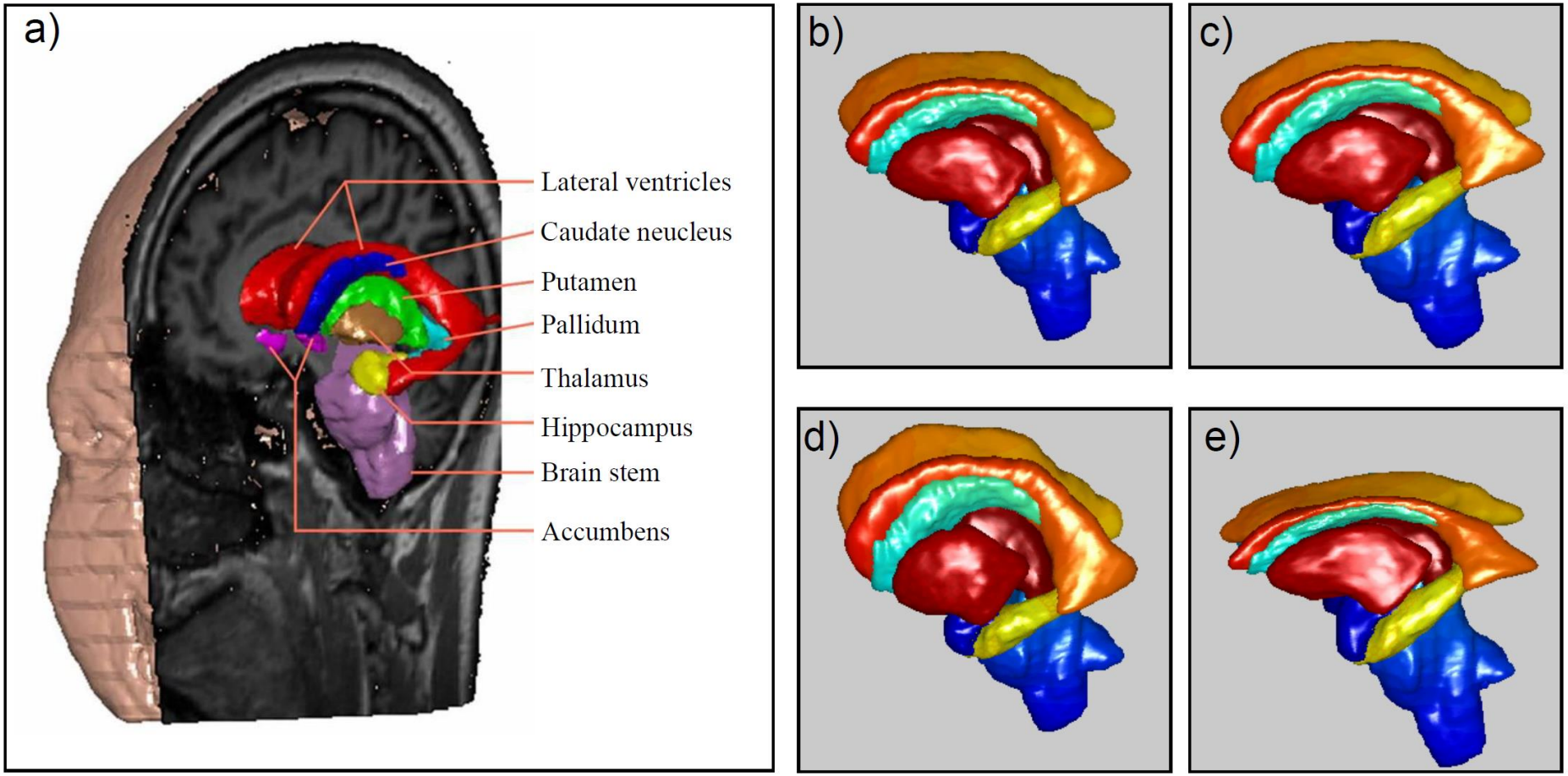
4. If it is a face, who is it?
   - Find closest labeled face in database (nearest-neighbor in K-dimensional space)

Relationships between models



Source: S. Prince

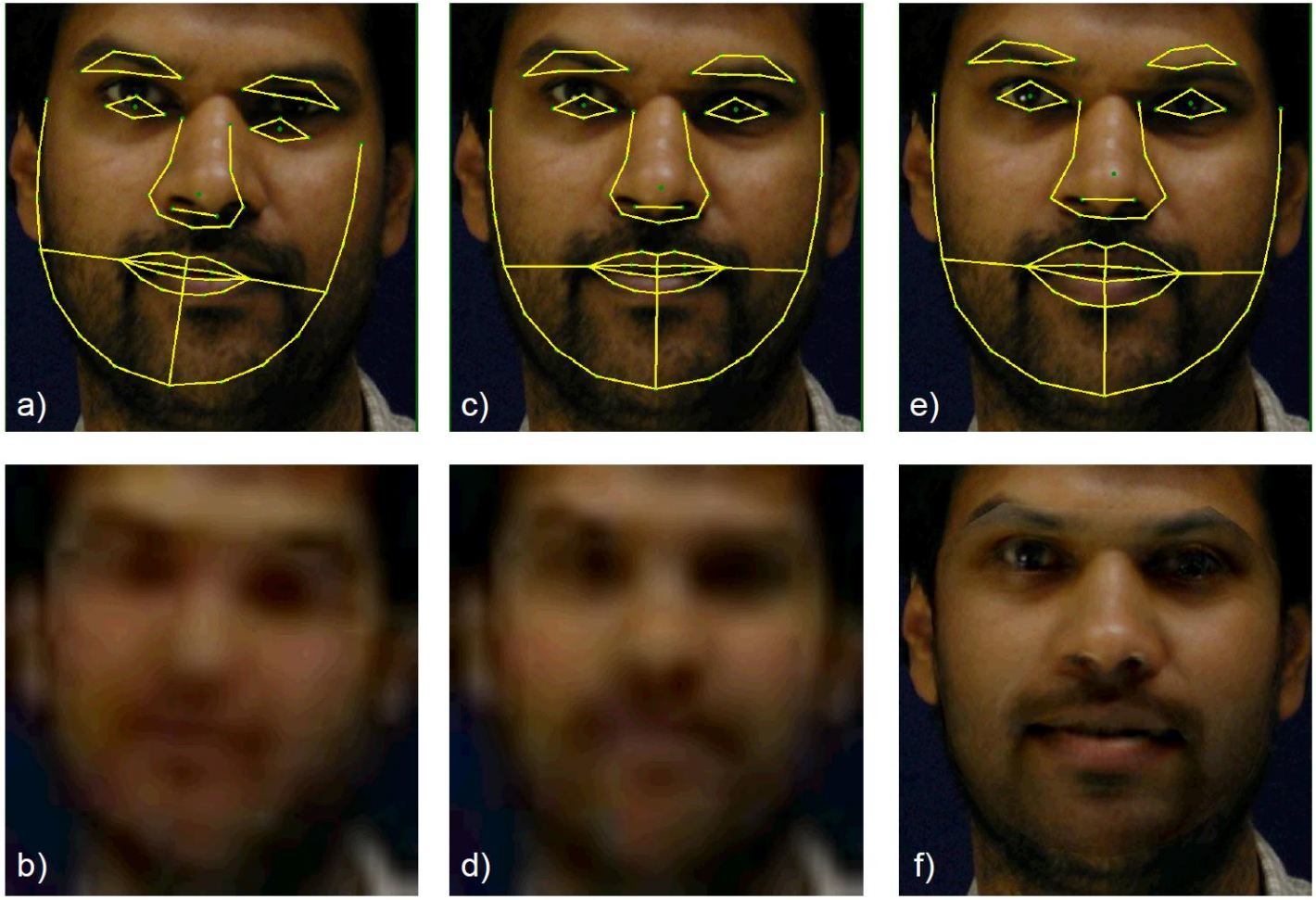
3D shape models



Babalola *et al.* (2008)

Source: S. Prince

Statistical models for shape and appearance



Source: T. Cootes

Statistical models for shape and appearance

$$
\begin{aligned}
Pr(\mathbf{h}_i) &= \text{Norm}_{\mathbf{h}_i}[\mathbf{0}, \mathbf{I}] \\
Pr(\mathbf{w}_i|\mathbf{h}_i) &= \text{Norm}_{\mathbf{w}_i}[\boldsymbol{\mu}_w + \boldsymbol{\Phi}_w \mathbf{h}_i, \sigma_w^2 \mathbf{I}]
\end{aligned}
$$

1. We draw a hidden variable $\mathbf{h}$ from a prior
2. We draw landmark points $\mathbf{w}$ from a subspace model

Source: S. Prince

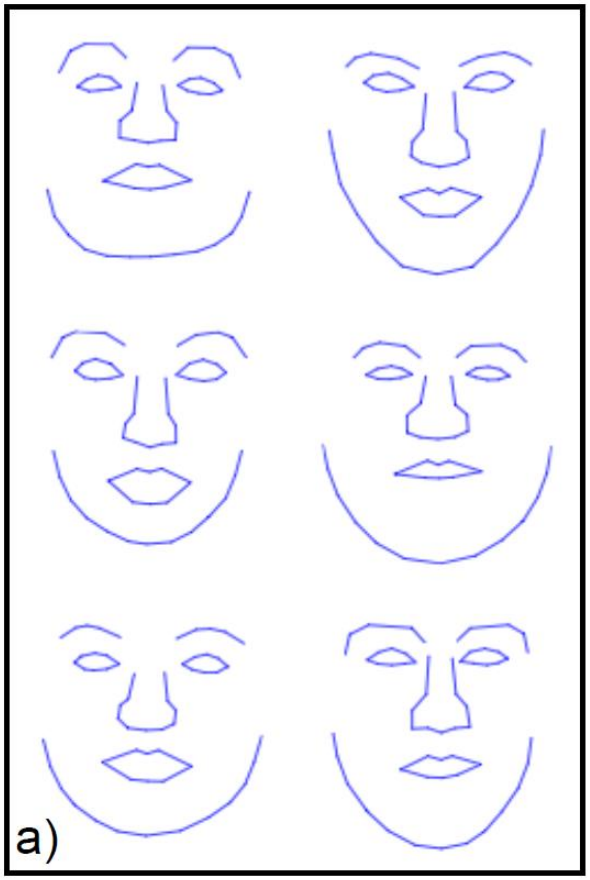
Statistical models for shape and appearance

$$
\begin{aligned}
Pr(\mathbf{h}_i) &= \text{Norm}_{\mathbf{h}_i}[\mathbf{0}, \mathbf{I}] \\
Pr(\mathbf{w}_i|\mathbf{h}_i) &= \text{Norm}_{\mathbf{w}_i}[\boldsymbol{\mu}_w + \boldsymbol{\Phi}_w \mathbf{h}_i, \sigma_w^2 \mathbf{I}] \\
Pr(\mathbf{x}_i|\mathbf{w}_i, \mathbf{h}_i) &= \text{Norm}_{\mathbf{x}_i}[\mathbf{warp}[\boldsymbol{\mu}_x + \boldsymbol{\Phi}_x \mathbf{h}_i, \mathbf{w}_i, \boldsymbol{\Psi}_i], \sigma_x^2 \mathbf{I}]
\end{aligned}
$$

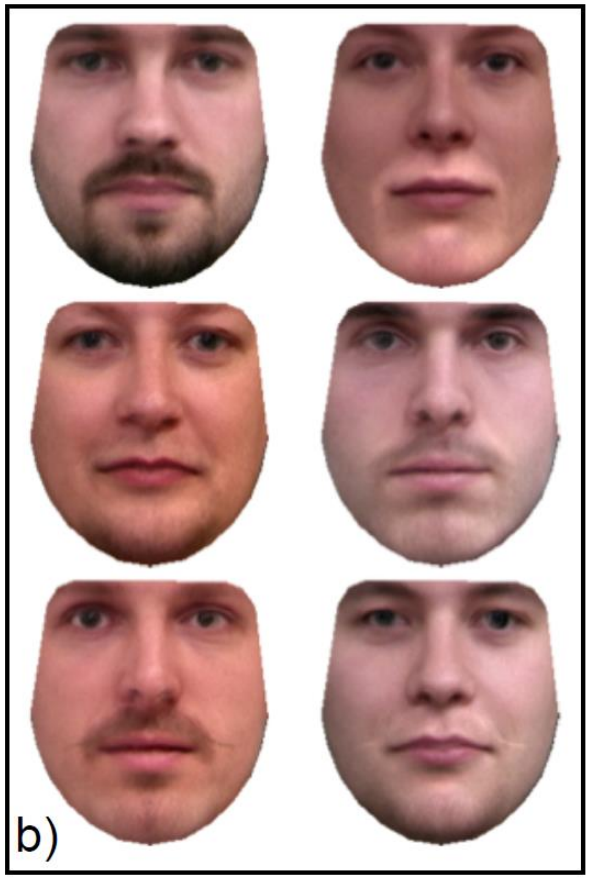
1. We draw a hidden variable $\mathbf{h}$ from a prior
2. We draw landmark points $\mathbf{w}$ from a subspace model
3. We draw image intensities $\mathbf{x}$.
   - Generate image intensities in standard template shape
   - Transform the landmark points (parameters $\boldsymbol{\psi}$)
   - Transform the image to landmark points
   - Add noise

Source: S. Prince

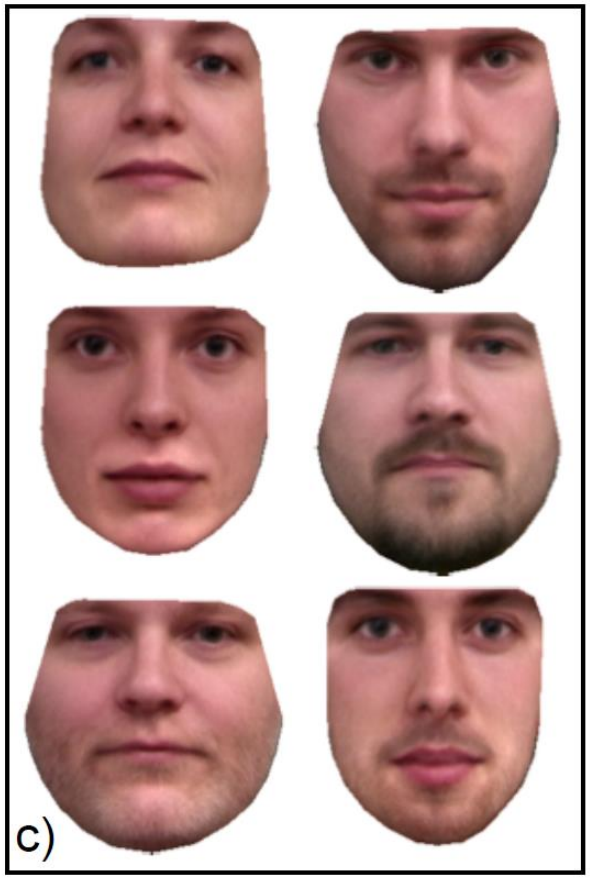
Shape and appearance model



Shape model



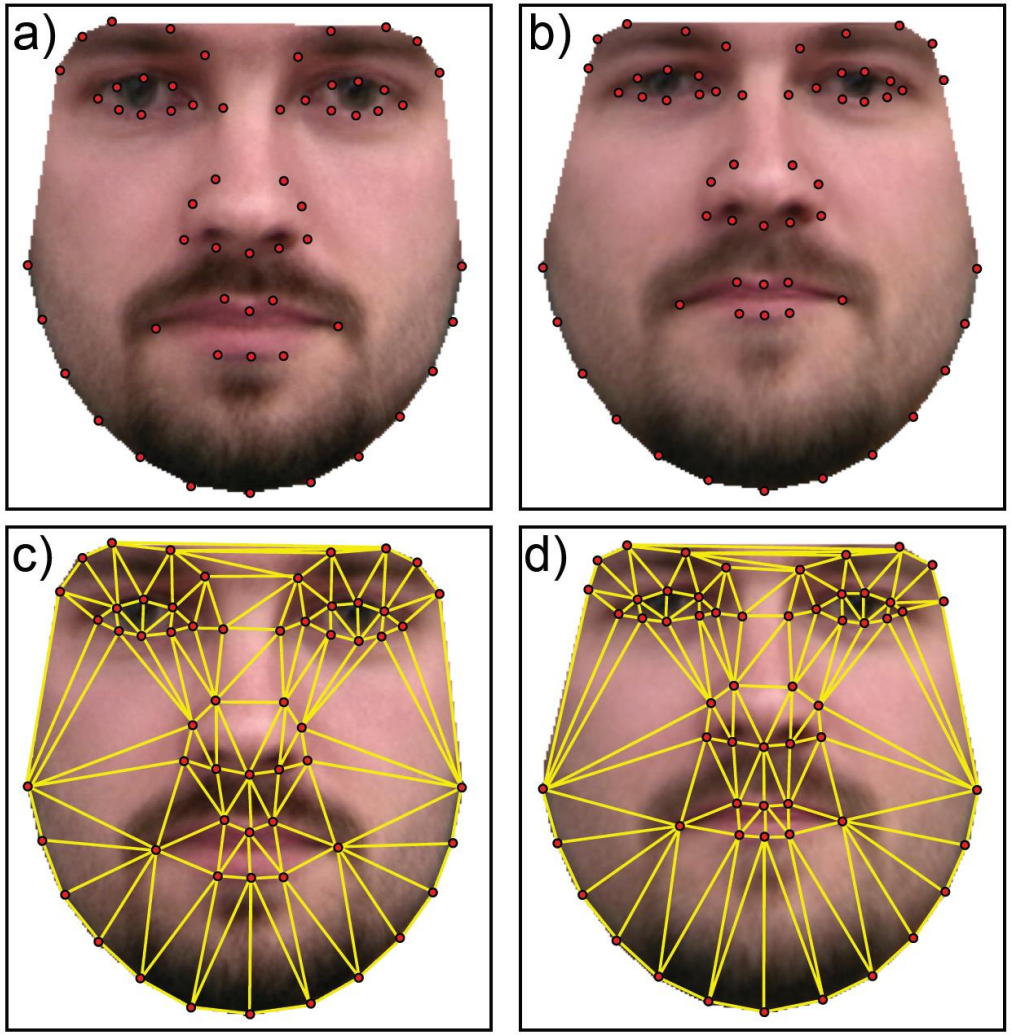
Intensity model



Shape and intensity

Source: M. Stegmann

Warping images



Piecewise affine transformation

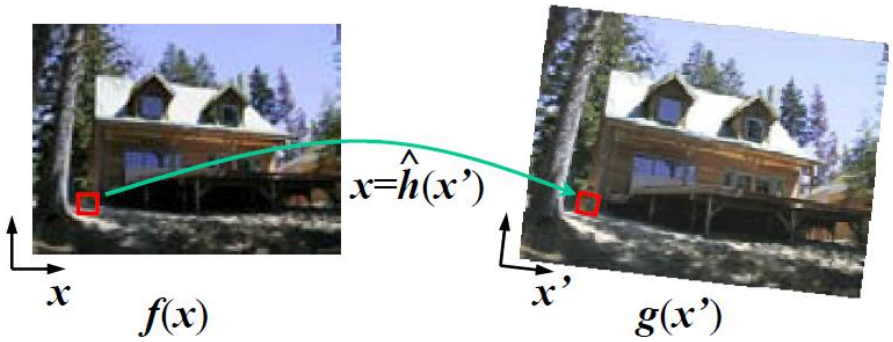
Triangulate image points using Delaunay triangulation.

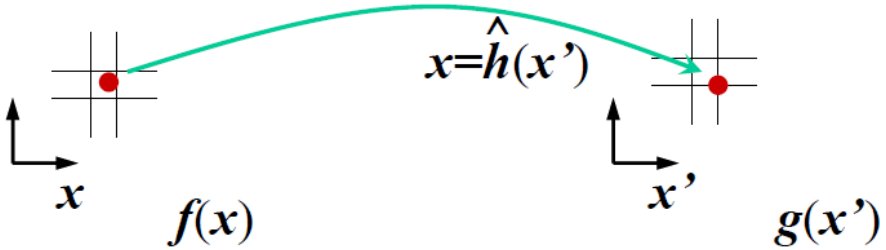
Image in each triangle is warped by an affine transformation.

Source: M. Stegmann

Warping

- GPU/OpenGL rendering
- Inverse map with interpolation:





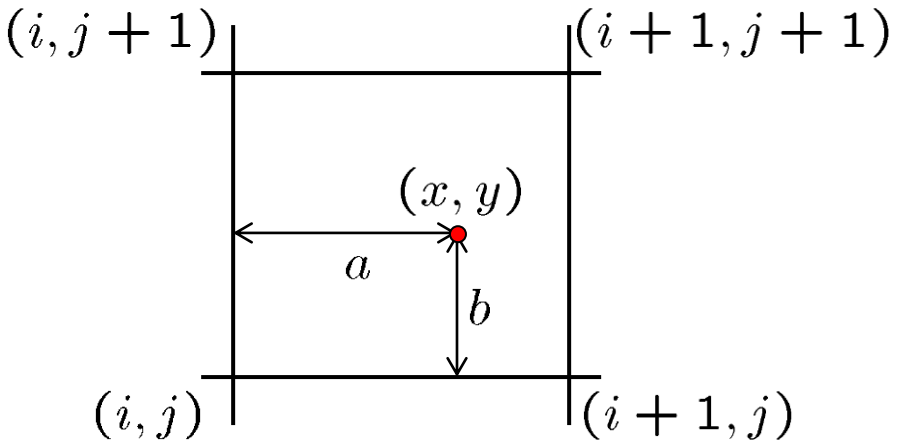
procedure *inverseWarp*$(f, \boldsymbol{h}, \textbf{out}\ g)$:

For every pixel $\boldsymbol{x}'$ in $g(\boldsymbol{x}')$

1. Compute the source location $\boldsymbol{x} = \hat{\boldsymbol{h}}(\boldsymbol{x}')$
2. Resample $f(\boldsymbol{x})$ at location $\boldsymbol{x}$ and copy to $g(\boldsymbol{x}')$

Bilinear interpolation

Sampling at *f(x,y):*



$$
\begin{aligned}
f(x,y) = \quad & (1-a)(1-b) \quad f[i,j] \\
& +a(1-b) \quad\;\; f[i+1,j] \\
& \quad +ab \quad\quad\;\; f[i+1,j+1] \\
& +(1-a)b \quad\;\; f[i,j+1]
\end{aligned}
$$

Slide from Alyosha Efros, CMU

Learning

$$
\begin{aligned}
Pr(\mathbf{h}_i) &= \text{Norm}_{\mathbf{h}_i}[\mathbf{0}, \mathbf{I}] \\
Pr(\mathbf{w}_i|\mathbf{h}_i) &= \text{Norm}_{\mathbf{w}_i}[\boldsymbol{\mu}_w + \boldsymbol{\Phi}_w \mathbf{h}_i, \sigma_w^2 \mathbf{I}] \\
Pr(\mathbf{x}_i|\mathbf{w}_i, \mathbf{h}_i) &= \text{Norm}_{\mathbf{x}_i}[\mathbf{warp}[\boldsymbol{\mu}_x + \boldsymbol{\Phi}_x \mathbf{h}_i, \mathbf{w}_i, \boldsymbol{\Psi}_i], \sigma_x^2 \mathbf{I}]
\end{aligned}
$$

Goal is to learn parameters :

$$\{\boldsymbol{\mu}_w, \boldsymbol{\Phi}_w, \sigma_w^2, \boldsymbol{\mu}_x, \boldsymbol{\Phi}_x, \sigma_x^2\}$$

Problem

- We are given the transformed landmark points
- We are given the warped and transformed images

Solution

- Use Procrustes analysis to un-transform landmark points
- Warp observed images to template shape

Source: S. Prince

Learning

Now have aligned landmark points $\mathbf{w}$, and aligned images $\mathbf{x}$, we can learn the simpler model:

$$\begin{aligned} Pr(\mathbf{h}_i) &= \text{Norm}_{\mathbf{h}_i}[\mathbf{0}, \mathbf{I}] \\ Pr(\mathbf{w}_i|\mathbf{h}_i) &= \text{Norm}_{\mathbf{w}_i}[\boldsymbol{\mu}_w + \boldsymbol{\Phi}_w \mathbf{h}_i, \sigma_w^2 \mathbf{I}] \\ Pr(\mathbf{x}_i|\mathbf{h}_i) &= \text{Norm}_{\mathbf{x}_i}[\boldsymbol{\mu}_x + \boldsymbol{\Phi}_x \mathbf{h}_i, \sigma_x^2 \mathbf{I}]. \end{aligned}$$

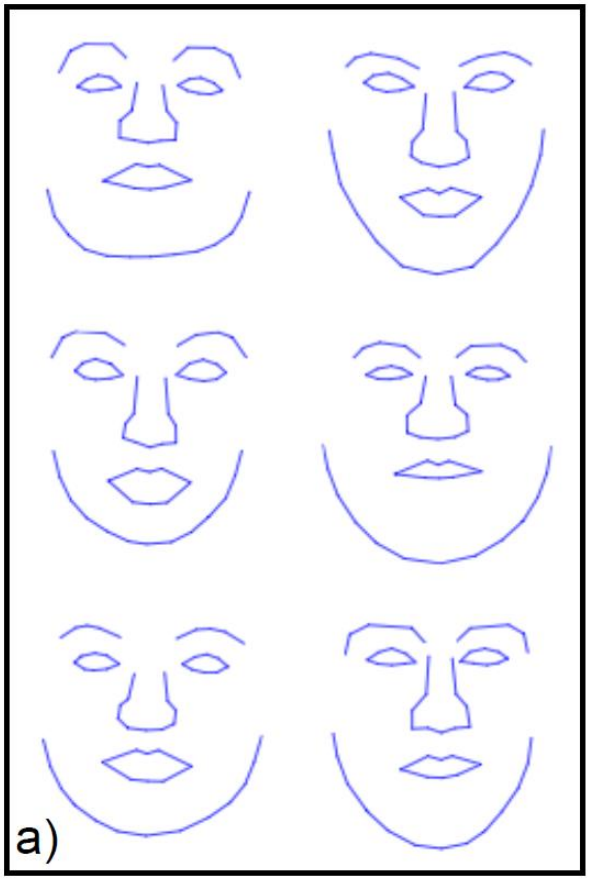
Can write generative equation as:

$$\begin{bmatrix} \mathbf{w}_i \\ \mathbf{x}_i \end{bmatrix} = \begin{bmatrix} \boldsymbol{\mu}_w \\ \boldsymbol{\mu}_x \end{bmatrix} + \begin{bmatrix} \boldsymbol{\Phi}_w \\ \boldsymbol{\Phi}_x \end{bmatrix} \mathbf{h}_i + \begin{bmatrix} \boldsymbol{\epsilon}_{wi} \\ \boldsymbol{\epsilon}_{xi} \end{bmatrix}$$

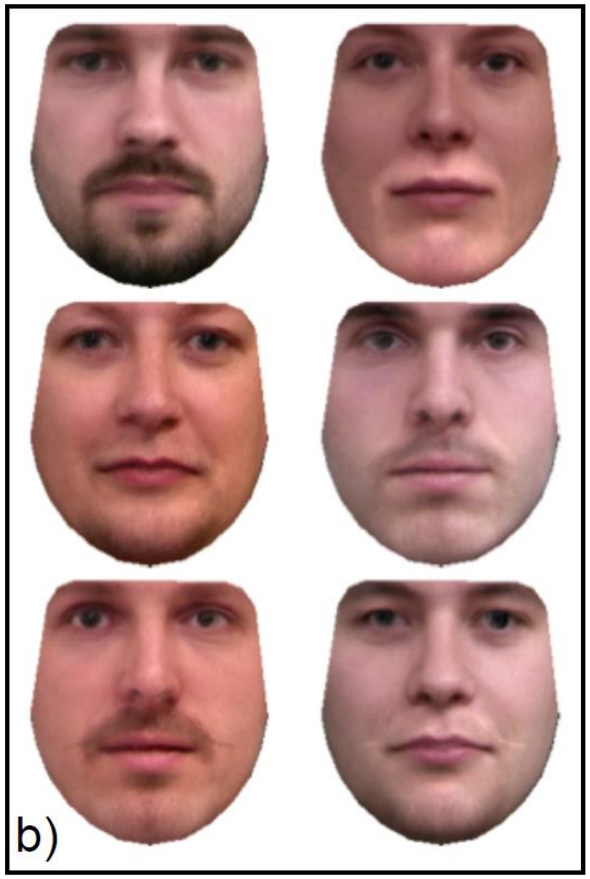
Has the form: $\mathbf{x}' = \boldsymbol{\mu}' + \boldsymbol{\Phi}' \mathbf{h} + \boldsymbol{\epsilon}'$

Source: S. Prince

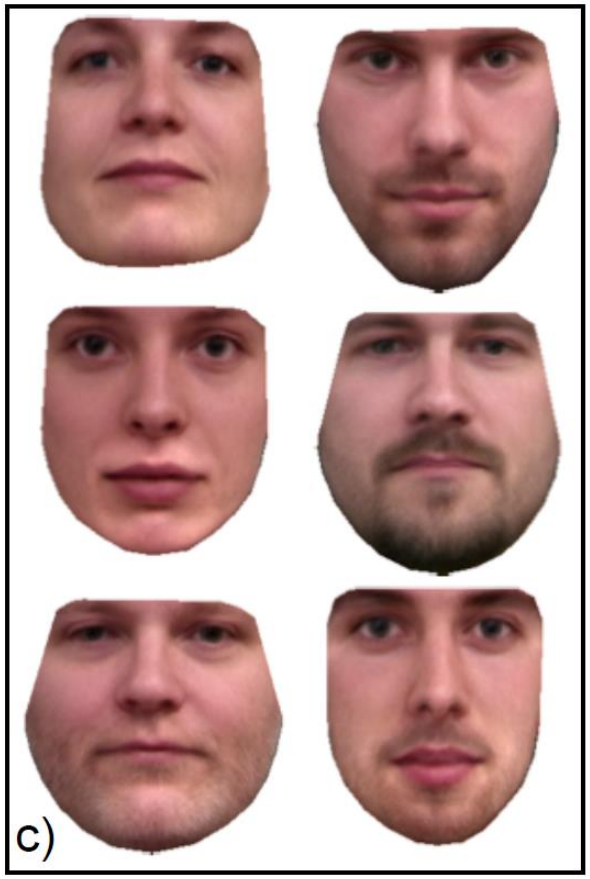
Shape and appearance model



Shape model



Intensity model



Shape and intensity

Source: M. Stegmann

Inference

Likelihood of observed intensities

$$Pr(\mathbf{x}|\mathbf{h}) = \text{Norm}_{\mathbf{x}}[\mathbf{warp}[\boldsymbol{\mu}_x + \boldsymbol{\Phi}_x\mathbf{h}, \boldsymbol{\mu} + \boldsymbol{\Phi}\mathbf{h}, \boldsymbol{\Psi}], \sigma_x^2\mathbf{I}]$$

To fit the model use maximum likelihood

$$\hat{\mathbf{h}}, \hat{\boldsymbol{\Psi}} = \underset{\mathbf{h},\boldsymbol{\Psi}}{\text{argmax}}\left[\log\left[Pr(\mathbf{x}|\mathbf{h})\right]\right]$$

$$= \underset{\mathbf{h},\boldsymbol{\Psi}}{\text{argmin}}\left[(\mathbf{x}-\mathbf{warp}[\boldsymbol{\mu}_x+\boldsymbol{\Phi}_x\mathbf{h}, \boldsymbol{\mu}_w+\boldsymbol{\Phi}_w\mathbf{h}, \boldsymbol{\Psi}])^T\right.$$

$$\left.(\mathbf{x}-\mathbf{warp}[\boldsymbol{\mu}_x+\boldsymbol{\Phi}_x\mathbf{h}, \boldsymbol{\mu}_w+\boldsymbol{\Phi}_w\mathbf{h}, \boldsymbol{\Psi}])\right]$$

This has the least squares form

$$f[\boldsymbol{\theta}] = \mathbf{z}[\boldsymbol{\theta}]^T\mathbf{z}[\boldsymbol{\theta}]$$

Source: S. Prince

Inference

This has the least squares form

$$f[\boldsymbol{\theta}] = \mathbf{z}[\boldsymbol{\theta}]^T \mathbf{z}[\boldsymbol{\theta}]$$

Use Gauss-Newton method or similar

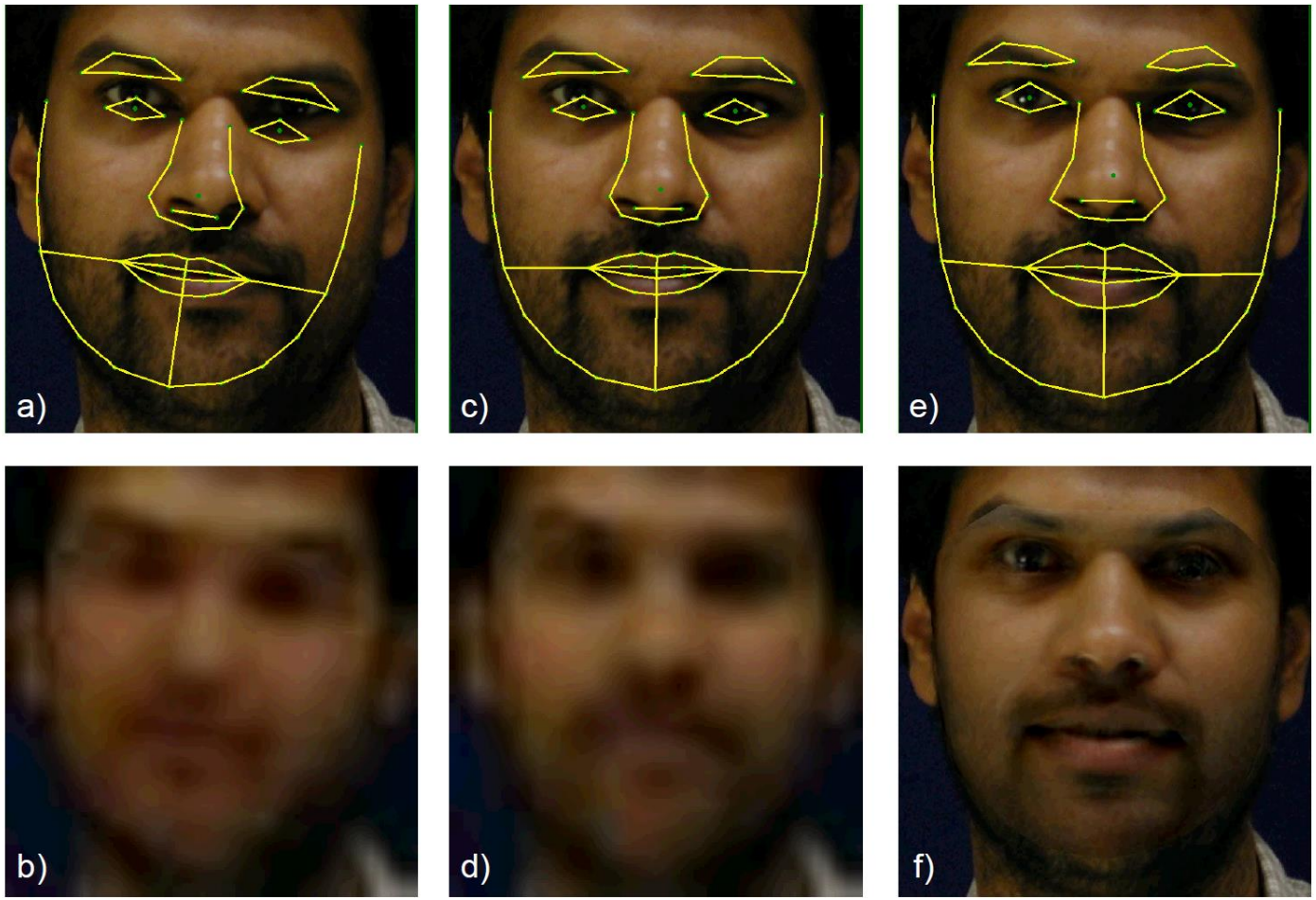
$$\boldsymbol{\theta}^{[t]} = \boldsymbol{\theta}^{[t-1]} + \lambda(\mathbf{J}^T\mathbf{J})^{-1}\frac{\partial f}{\partial \boldsymbol{\theta}}$$

Where the Jacobian $\mathbf{J}$ is a matrix with elements

$$J_{mn} = \frac{\partial z_m}{\partial \theta_n}$$

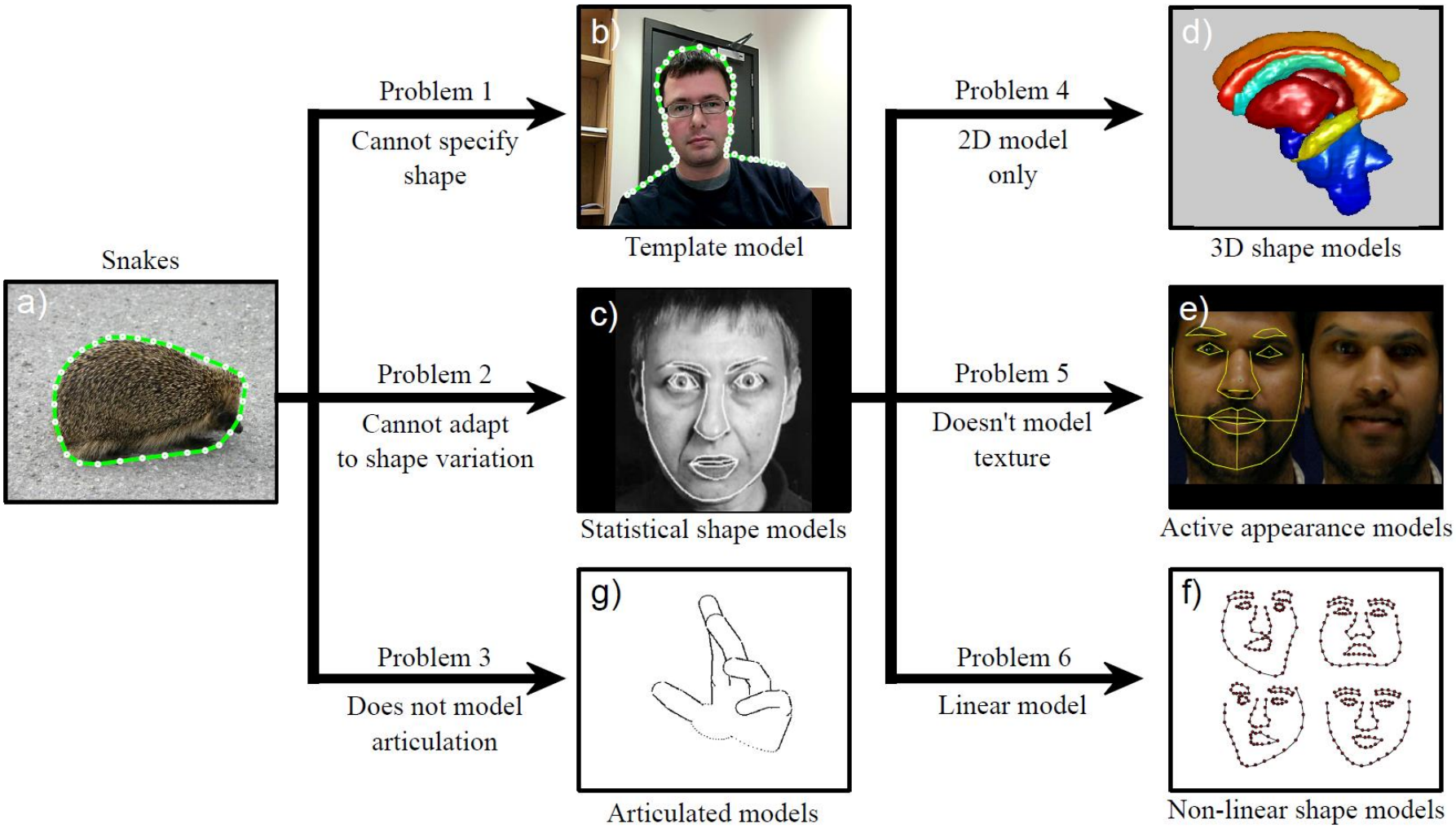
Source: S. Prince

Statistical models for shape and appearance



Source: T. Cootes

Relationships between models



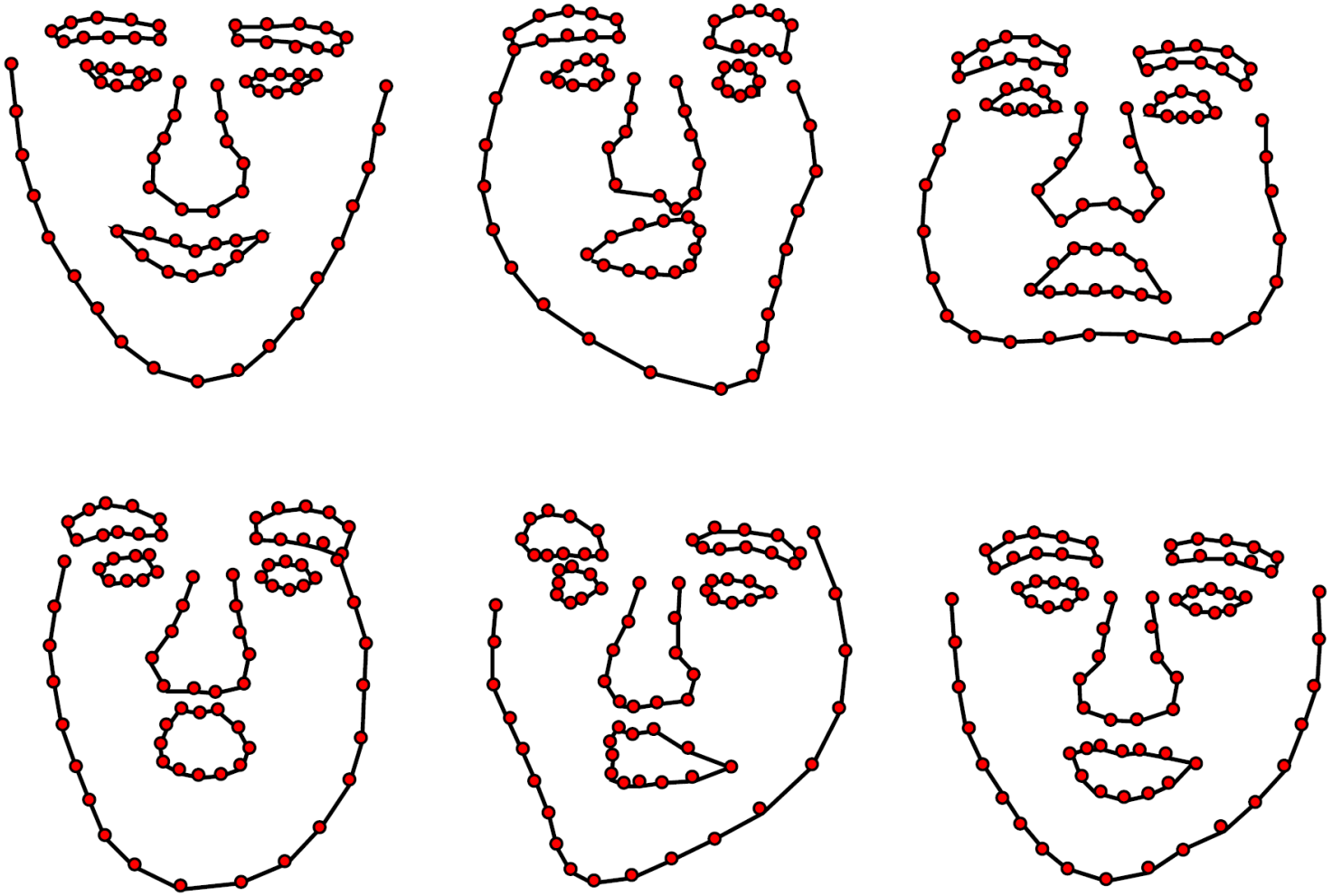
Source: S. Prince

Non-linear models

- The shape and appearance models that we have studied so far are based on the normal distribution
- But more complex shapes might need more complex distributions
  - Could use mixture of PPCAs or similar
  - Or use a non-linear subspace model
- Example: Gaussian process latent variable model (GPLVM) – Computer Vision II

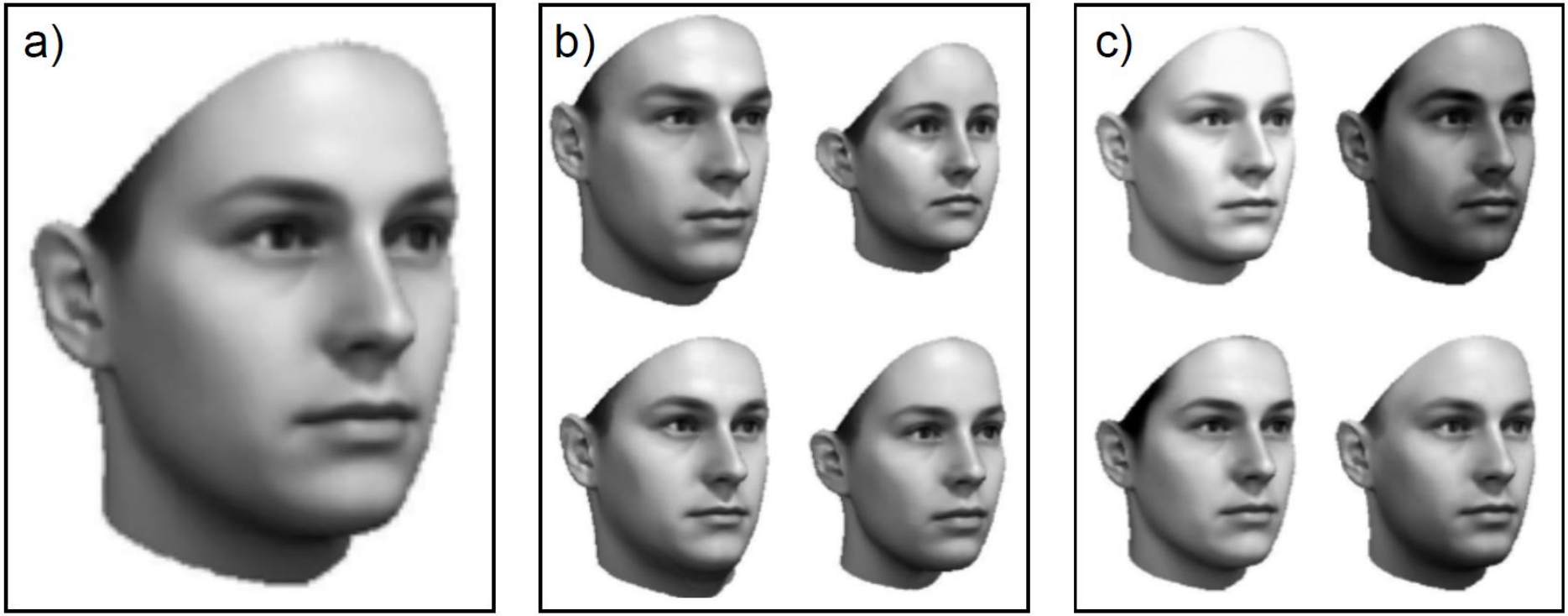
Source: S. Prince

GPLVM Shape models



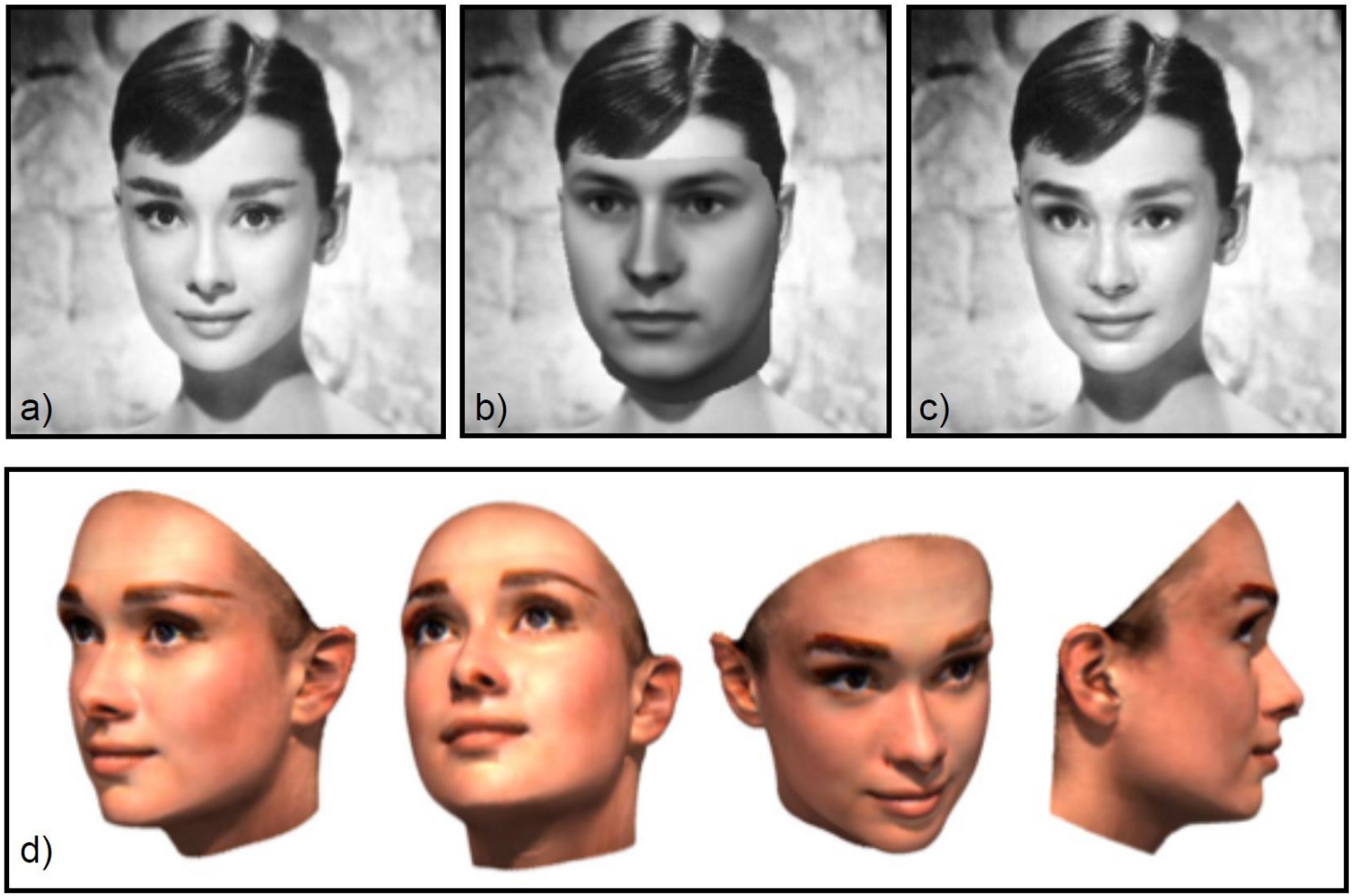
Source: Y. Huang

3D morphable models



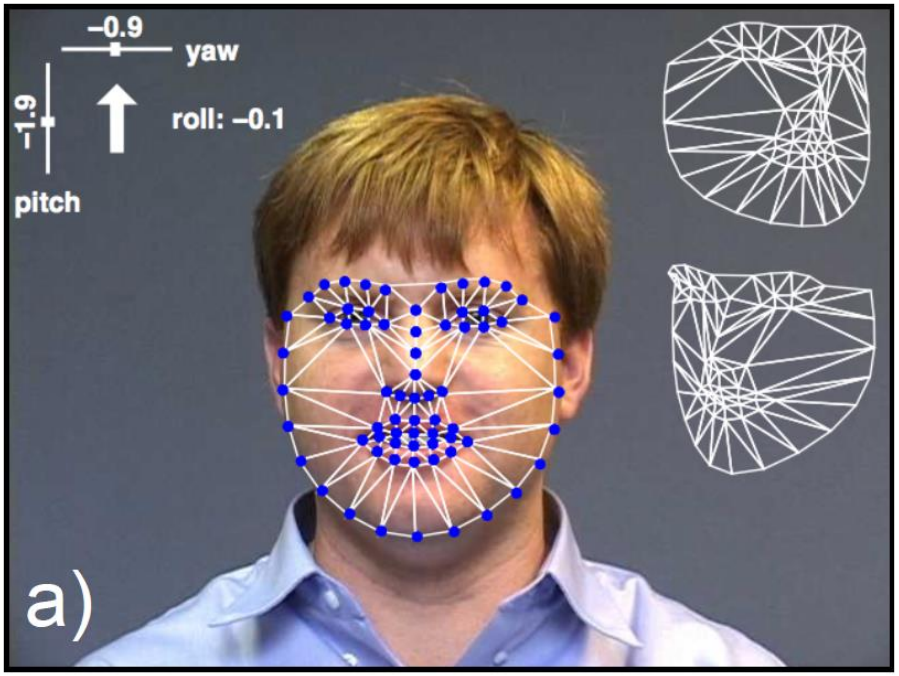
Adapted from Blanz & Vetter (2003). ©2003 IEEE

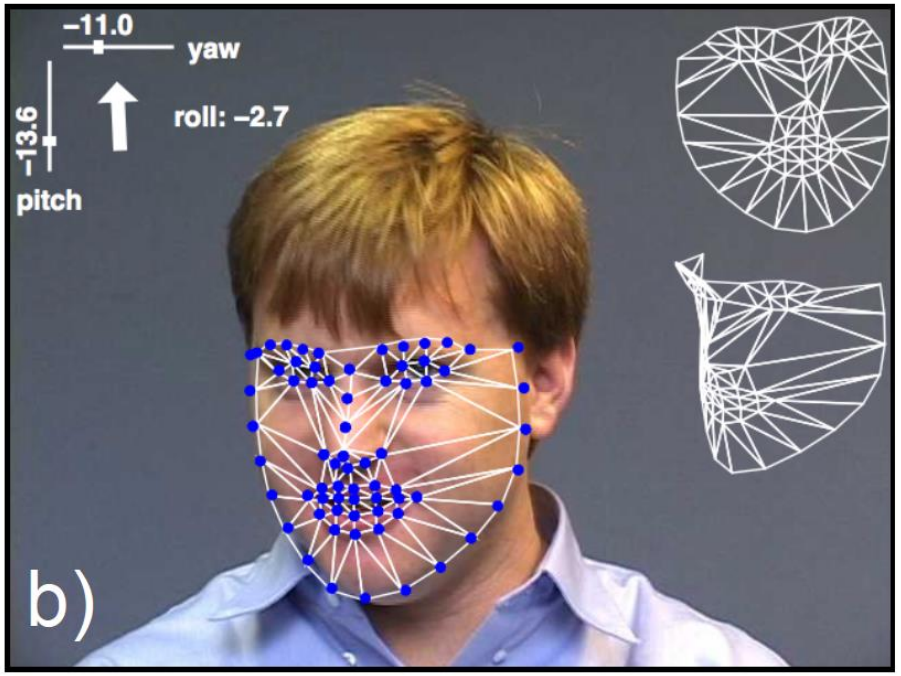
3D morphable models



Adapted from Blanz & Vetter (1999).

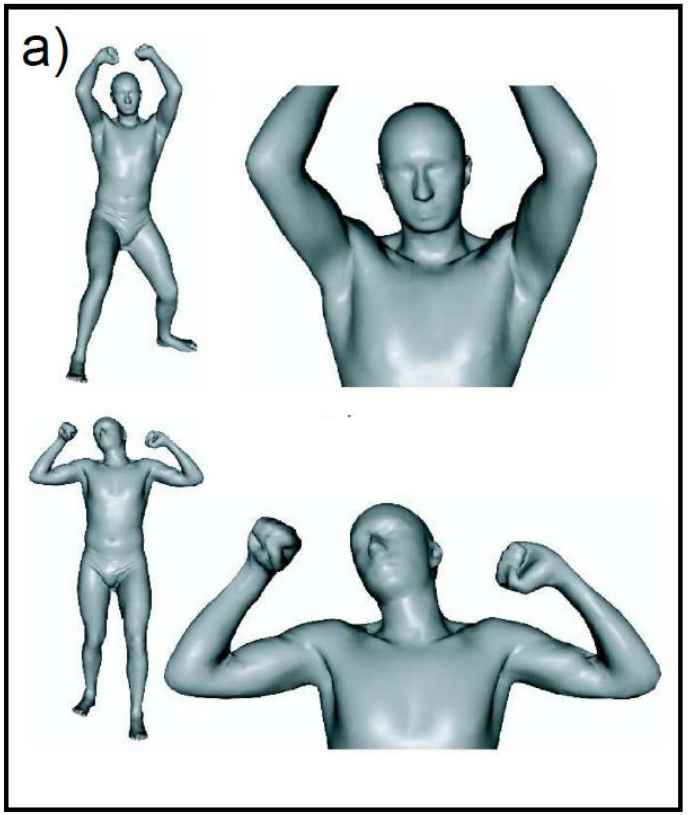
3D morphable models

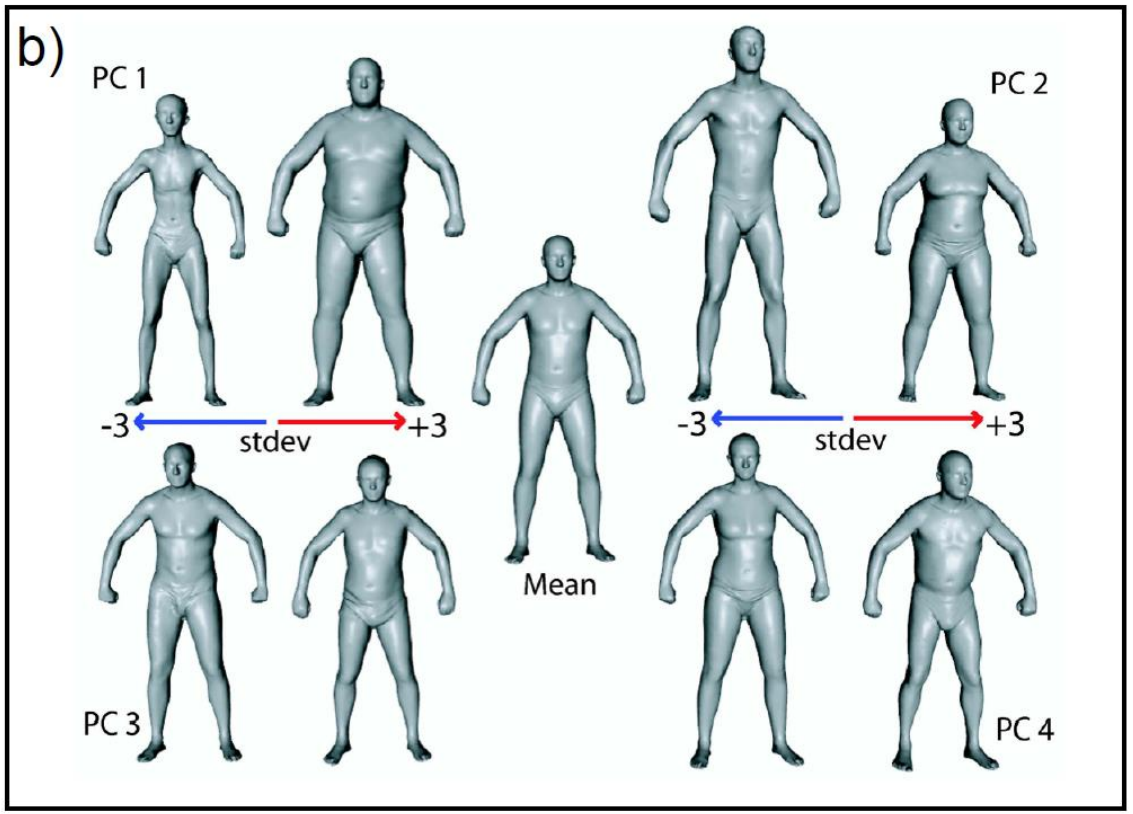




Matthews *et al.* (2007). ©2007 Springer.

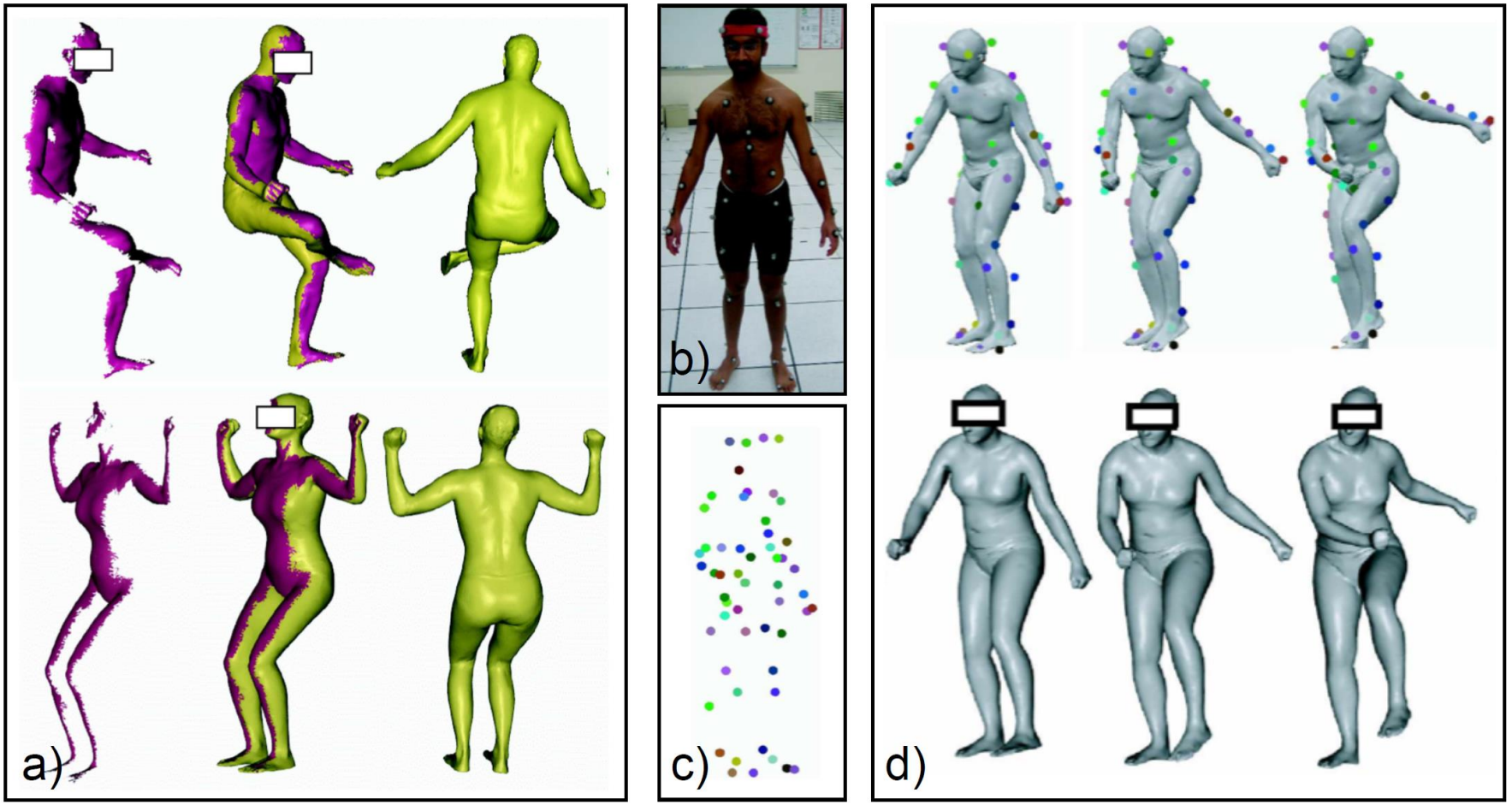
3D body model





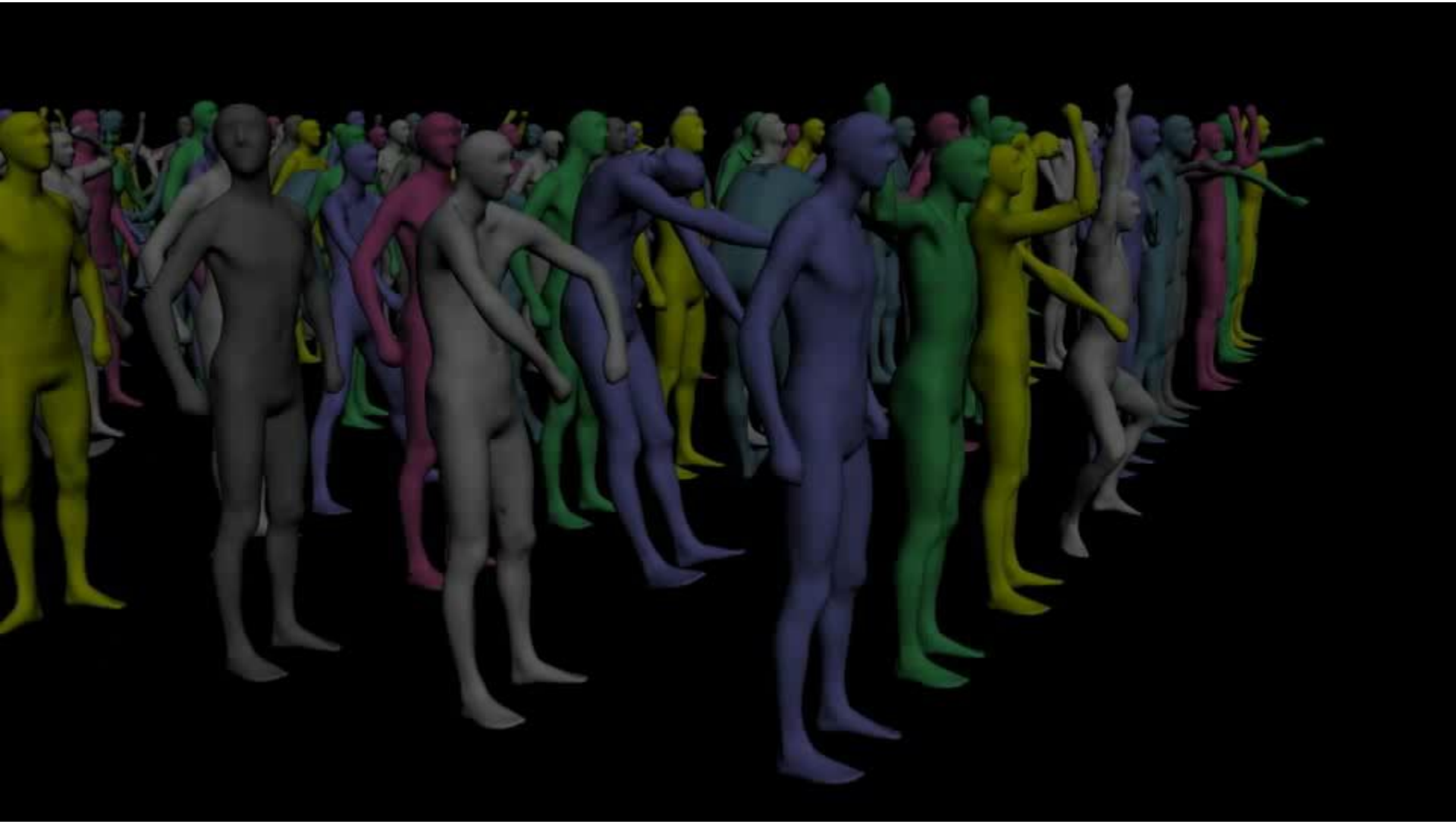
Adapted from Anguelov *et al.* (2005). ©2005 ACM

3D body model applications



Adapted from Anguelov *et al.* (2005). ©2005 ACM

Shape model



Video editing

Jain et al. **MovieReshape: Tracking and Reshaping of Humans in Videos.** ToG 2010.


UNIVERSITÄT BONN