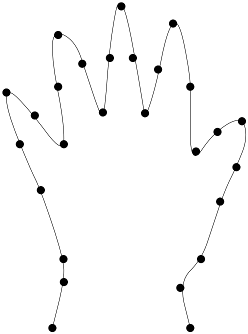
# Point Distribution Models

Jan Kybic

winter semester 2007

# Point distribution models

*(Cootes et al., 1992)*



- Shape description techniques
- A family of shapes = mean + eigenvectors (eigenshapes)
- Shapes described by points

# Point distribution model procedure

**Input:**

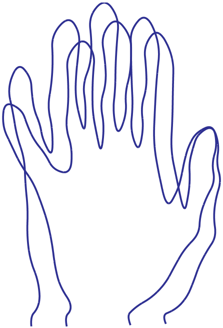
- $M$ training samples
- $N$ points each

$$\mathbf{x}^i = (x_1^i, y_1^i, x_2^i, y_2^i, \ldots, x_N^i, y_N^i)^T$$

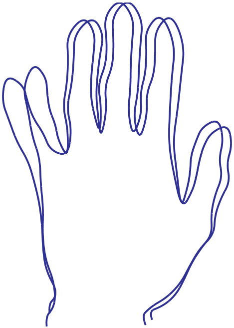
**Procedure:**

- Rigidly align all shapes
- Calculate the mean and the covariance matrix
- PCA (eigen analysis) — find principal modes

# Rigid alignment



before alignment



after alignment

# Aligning two shapes

$$\mathbf{x}^{(1)} = (x_1^{(1)}, y_1^{(1)}, x_2^{(1)}, y_2^{(1)}, \ldots, x_N^{(1)}, y_N^{(1)})^T$$
$$\mathbf{x}^{(2)} = (x_1^{(2)}, y_1^{(2)}, x_2^{(2)}, y_2^{(2)}, \ldots, x_N^{(2)}, y_N^{(2)})^T$$

Find a transformation (rotation, translation, scaling) of $\mathbf{x}^{(2)}$

$$\mathcal{T}(\mathbf{x}^{(2)}) = s\,R \begin{bmatrix} x_i^{(2)} \\ y_i^{(2)} \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix} = \begin{bmatrix} x_i^{(2)} s\cos\theta - y_i^{(2)} s\sin\theta \\ x_i^{(2)} s\sin\theta + y_i^{(2)} s\cos\theta \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

such that a sum of squared distances is minimized

$$E = \sum_{i=1}^{M} w_i \left\| s \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x_i^{(2)} \\ y_i^{(2)} \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix} - \begin{bmatrix} x_i^{(1)} \\ y_i^{(1)} \end{bmatrix} \right\|^2$$

# Aligning two shapes

$$E = \sum_{i=1}^{M} w_i \left\| s \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x_i^{(2)} \\ y_i^{(2)} \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix} - \begin{bmatrix} x_i^{(1)} \\ y_i^{(1)} \end{bmatrix} \right\|^2$$

Minimize $E(\theta, s, t_x, t_y)$ as $\quad \min_\theta \min_{s,t_x,t_y} E_\theta(s, t_x, t_y)$

- **Inner minimization wrt** $s, t_x, t_y$

$$\frac{\partial E}{\partial t_x} = 0\,, \qquad \frac{\partial E}{\partial t_y} = 0\,, \qquad \frac{\partial E}{\partial s} = 0\,,$$

# Aligning two shapes

- **Inner minimization wrt** $s, t_x, t_y$

$$\frac{\partial E}{\partial t_x} = 0\,, \qquad \frac{\partial E}{\partial t_y} = 0\,, \qquad \frac{\partial E}{\partial s} = 0\,,$$

leads to linear equations:

$$s \textstyle\sum_{i=1}^{M} w_i\, q(\quad y_i, -x_i, \theta) - N\, t_x = - \textstyle\sum_{i=1}^{M} w_i\, x_i'$$
$$s \textstyle\sum_{i=1}^{M} w_i\, q(-x_i, -y_i, \theta) - N\, t_y = - \textstyle\sum_{i=1}^{M} w_i\, y_i'$$

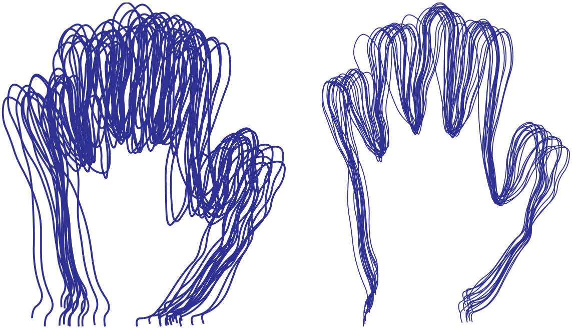
$$s \textstyle\sum_{i=1}^{M} w_i^2 \Big( q^2(y_i, -x_i, \theta) + q^2(x_i, y_i, \theta) \Big) - t_x \textstyle\sum_{i=1}^{M} w_i\, q(y_i, -x_i, \theta)$$
$$- t_y \textstyle\sum_{i=1}^{M} w_i\, q(-x_i, -y_i, \theta)$$
$$= - \sum_{i=1}^{M} w_i\, x_i'\, q(y_i, -x_i, \theta) + \sum_{i=1}^{M} w_i\, y_i'\, q(x_i, -y_i, \theta)$$

where $q(a, b, \theta) = a \sin\theta + b \cos\theta$.

# Aligning two shapes

- **Inner minimization wrt** $s, t_x, t_y$
- **Outer minimization wrt** $\theta$
  One dimensional functional minimization, e.g. Brent's routine or golden section search.

# Aligning all training shapes



Before alignment After alignment

# Aligning all training shapes

- Align each $\mathbf{x}^i$ with $\mathbf{x}^1$, for $i = 2, 3, \dots, M$, obtaining $\{\mathbf{x}^1, \hat{\mathbf{x}}^2, \hat{\mathbf{x}}^3, \dots, \hat{\mathbf{x}}^M\}$.
- Calculate the mean $\bar{\mathbf{x}} = \left[\bar{x}_1, \bar{y}_1, \bar{x}_2, \bar{y}_2, \dots, \bar{x}_N, \bar{y}_N\right]$ of the aligned shapes $\{\mathbf{x}^1, \hat{\mathbf{x}}^2, \hat{\mathbf{x}}^3, \dots, \hat{\mathbf{x}}^M\}$.

$$\bar{x}_j = \frac{1}{M} \sum_{i=1}^{M} \hat{x}_j^i \qquad \text{and} \qquad \bar{y}_j = \frac{1}{M} \sum_{i=1}^{M} \hat{y}_j^i \, .$$

# Aligning all training shapes

- Align each $\mathbf{x}^i$ with $\mathbf{x}^1$, for $i = 2, 3, \ldots, M$, obtaining $\{\mathbf{x}^1, \hat{\mathbf{x}}^2, \hat{\mathbf{x}}^3, \ldots, \hat{\mathbf{x}}^M\}$.
- Calculate the mean $\bar{\mathbf{x}} = \left[\bar{x}_1, \bar{y}_1, \bar{x}_2, \bar{y}_2, \ldots, \bar{x}_N, \bar{y}_N\right]$ of the aligned shapes $\{\mathbf{x}^1, \hat{\mathbf{x}}^2, \hat{\mathbf{x}}^3, \ldots, \hat{\mathbf{x}}^M\}$.
- Align the mean shape $\bar{\mathbf{x}}$ with $\mathbf{x}^1$. *(Necessary for convergence.)*

# Aligning all training shapes

- Align each $\mathbf{x}^i$ with $\mathbf{x}^1$, for $i = 2, 3, \ldots, M$, obtaining $\{\mathbf{x}^1, \hat{\mathbf{x}}^2, \hat{\mathbf{x}}^3, \ldots, \hat{\mathbf{x}}^M\}$.
- Calculate the mean $\bar{\mathbf{x}} = \left[\bar{x}_1, \bar{y}_1, \bar{x}_2, \bar{y}_2, \ldots, \bar{x}_N, \bar{y}_N\right]$ of the aligned shapes $\{\mathbf{x}^1, \hat{\mathbf{x}}^2, \hat{\mathbf{x}}^3, \ldots, \hat{\mathbf{x}}^M\}$.
- Align the mean shape $\bar{\mathbf{x}}$ with $\mathbf{x}^1$. *(Necessary for convergence.)*
- Align $\hat{\mathbf{x}}^2, \hat{\mathbf{x}}^3, \ldots, \hat{\mathbf{x}}^M$ to the adjusted mean.

# Aligning all training shapes

- Align each $\mathbf{x}^i$ with $\mathbf{x}^1$, for $i = 2, 3, \ldots, M$, obtaining $\{\mathbf{x}^1, \hat{\mathbf{x}}^2, \hat{\mathbf{x}}^3, \ldots, \hat{\mathbf{x}}^M\}$.
- Calculate the mean $\bar{\mathbf{x}} = \left[\bar{x}_1, \bar{y}_1, \bar{x}_2, \bar{y}_2, \ldots, \bar{x}_N, \bar{y}_N\right]$ of the aligned shapes $\{\mathbf{x}^1, \hat{\mathbf{x}}^2, \hat{\mathbf{x}}^3, \ldots, \hat{\mathbf{x}}^M\}$.
- Align the mean shape $\bar{\mathbf{x}}$ with $\mathbf{x}^1$. *(Necessary for convergence.)*
- Align $\hat{\mathbf{x}}^2, \hat{\mathbf{x}}^3, \ldots, \hat{\mathbf{x}}^M$ to the adjusted mean.
- Repeat until convergence.

We have obtained $M$ (mutually aligned) boundaries $\hat{\mathbf{x}}^1, \hat{\mathbf{x}}^2, \ldots, \hat{\mathbf{x}}^M$ and the mean $\bar{\mathbf{x}}$.

# Deriving the model

We have $M$ boundaries $\hat{\mathbf{x}}^1, \hat{\mathbf{x}}^2, \ldots, \hat{\mathbf{x}}^M$ and the mean $\bar{\mathbf{x}}$.

- Variation from the mean for each training shape

$$\delta\mathbf{x}^i = \hat{\mathbf{x}}^i - \bar{\mathbf{x}}\,.$$

- Covariance matrix $\mathbf{S}$ $(2N \times 2N)$

$$\mathbf{S} = \frac{1}{M}\sum_{i=1}^{M} \delta\mathbf{x}^i \,(\delta\mathbf{x}^i)^T$$

# Deriving the model

We have $M$ boundaries $\hat{\mathbf{x}}^1, \hat{\mathbf{x}}^2, \ldots, \hat{\mathbf{x}}^M$ and the mean $\bar{\mathbf{x}}$.

- Variation from the mean for each training shape

$$\delta \mathbf{x}^i = \hat{\mathbf{x}}^i - \bar{\mathbf{x}} \,.$$

- Covariance matrix $\mathbf{S}$ $(2N \times 2N)$

$$\mathbf{S} = \frac{1}{M} \sum_{i=1}^{M} \delta \mathbf{x}^i \, (\delta \mathbf{x}^i)^T$$

- Principal component analysis

# Principal component analysis

- Eigen decomposition

$$\mathbf{S}\mathbf{p}_i = \lambda_i \mathbf{p}_i$$
$$P = \left[\mathbf{p}^1 \mathbf{p}^2 \mathbf{p}^3 \dots \mathbf{p}^{2N}\right]$$

We know eigenvalues $\lambda_i$ are real because $\mathbf{S}$ is symmetric, positive definite. Eigenvectors (principal components) $\mathbf{p}_i$ are orthogonal, so $\mathbf{P}$ is a basis and any vector $\mathbf{x}$ can be represented as

$$\mathbf{x} = \bar{\mathbf{x}} + \mathbf{P}\,\mathbf{b}$$

# Principal component analysis

- Eigen decomposition

$$\mathbf{S}\mathbf{p}_i = \lambda_i \mathbf{p}_i$$
$$P = \left[\mathbf{p}^1 \mathbf{p}^2 \mathbf{p}^3 \ldots \mathbf{p}^{2N}\right]$$

  We know eigenvalues $\lambda_i$ are real because $\mathbf{S}$ is symmetric, positive definite. Eigenvectors (principal components) $\mathbf{p}_i$ are orthogonal, so $\mathbf{P}$ is a basis and any vector $\mathbf{x}$ can be represented as

$$\mathbf{x} = \bar{\mathbf{x}} + \mathbf{P}\,\mathbf{b}$$

- Order eigenvectors $\mathbf{p}_i$ and eigenvalues $\lambda_i$ such that $\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq \ldots \lambda_{2N}$. Most changes are then described by the first few eigenvectors.

# Principal component analysis

- Eigen decomposition

$$\mathbf{S}\mathbf{p}_i = \lambda_i \mathbf{p}_i$$
$$P = \left[\mathbf{p}^1 \mathbf{p}^2 \mathbf{p}^3 \ldots \mathbf{p}^{2N}\right]$$

$$\mathbf{x} = \bar{\mathbf{x}} + \mathbf{P}\,\mathbf{b}$$

- Order eigenvectors $\mathbf{p}_i$ and eigenvalues $\lambda_i$ such that $\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq \ldots \lambda_{2N}$. Most changes are then described by the first few eigenvectors.
- Consider only $K$ largest eigenvalues.

$$\begin{aligned} \text{Approximation} \quad & \mathbf{x} \approx \bar{\mathbf{x}} + \mathbf{P}_K\,\mathbf{b}_K \\ \text{with} \quad & \mathbf{P}_K = \left[\mathbf{p}^1 \mathbf{p}^2 \mathbf{p}^3 \ldots \mathbf{p}^K\right] \\ & \mathbf{b}_t = \left[b_1, b_2, \ldots, b_K\right]^T \end{aligned}$$

Choose the smallest $K$, such that $\sum_{i=1}^{K} \lambda_i \geq \alpha \sum_{i=1}^{N} \lambda_i$.

# Point distribution model

- **Input:** $M$ non-aligned boundaries $\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^M$.
- **Output:** mean $\bar{\mathbf{x}}$ and reduced eigvector matrix $\mathbf{P}_K$

# Point distribution model

- **Input:** $M$ non-aligned boundaries $\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^M$.
- **Output:** mean $\bar{\mathbf{x}}$ and reduced eigvector matrix $\mathbf{P}_K$
- **New shape generation:**

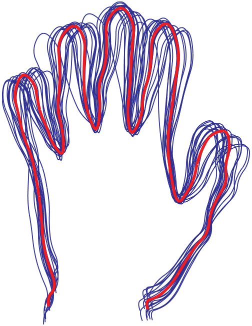
$$\widetilde{\mathbf{x}} = \bar{\mathbf{x}} + P_K \, \mathbf{b}_K$$

For "well-behaved' shapes

$$-3\sqrt{\lambda_i} \leq b_i \leq 3\sqrt{\lambda_i}$$

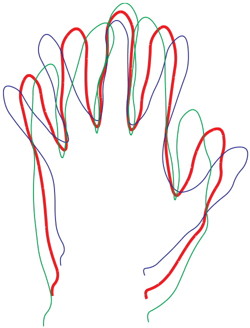
# PDM example

Before alignment

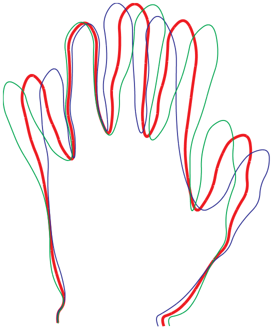


After alignment, mean shape

# PDM example



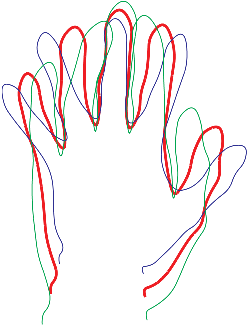
First mode



Second mode

The mean shape is in red, the shape corresponding to $-3\sqrt{\lambda}$ in blue and the shape corresponding to $+3\sqrt{\lambda}$ in green.

# Active shape models



PDM

Image to fit

Fit a learned point distribution model (PDM) to a given image.

# Pose and shape parameters

- Point distribution model (PDM) consists of
  - mean $\bar{\mathbf{p}}$
  - eigenvectors $\mathbf{P}$

# Pose and shape parameters

- Point distribution model (PDM) consists of
  - mean $\bar{\mathbf{p}}$
  - eigenvectors $\mathbf{P}$
- Fitted model given by:
  - pose parameters: $\theta$, $s$, $t_x$, $t_y$
  - shape parameters: $\mathbf{b}$

$$\widetilde{\mathbf{p}} = \mathbf{Pb} + \bar{\mathbf{p}}$$

$$\widetilde{\mathbf{p}} = \begin{bmatrix} \widetilde{\mathbf{p}}_1 & \widetilde{\mathbf{p}}_2 & \dots & \widetilde{\mathbf{p}}_N \end{bmatrix}$$

$$\widetilde{\mathbf{p}} = \begin{bmatrix} \tilde{x}_1 & \tilde{y}_1 & \tilde{x}_2 & \tilde{y}_2 & \dots & \tilde{x}_N & \tilde{y}_N \end{bmatrix}$$

$$\begin{bmatrix} x_i \\ y_i \end{bmatrix} = s \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & -\cos\theta \end{bmatrix} \begin{bmatrix} \tilde{x}_i \\ \tilde{y}_i \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

$$\mathbf{p} = \begin{bmatrix} x_1 & y_1 & x_2 & y_2 & \dots & x_N & y_N \end{bmatrix}$$

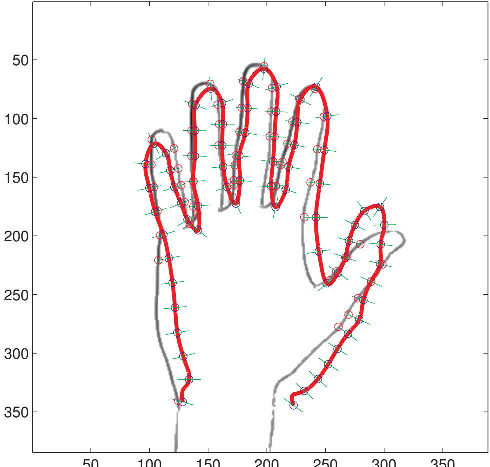
# Fitting

$$\mathbf{p} = T_{s,\theta,t_x,t_y}(\widetilde{\mathbf{p}}) = s\,\mathbf{Q}_\theta \widetilde{\mathbf{p}} + \mathbf{r}_{t_x,t_y} \quad \text{where} \quad \widetilde{\mathbf{p}} = \mathbf{P}\mathbf{b} + \bar{\mathbf{p}}$$

- Calculate an edge map of the image
- For each landmark $\mathbf{p}_i$ we find a line normal to the shape contour.
- New position $\mathbf{p}_i'$ is the maximum of the edge map on the line. (If maximum too weak, no change.)

# Fitting

- New position $\mathbf{p}'_i$ is the maximum of the edge map on the line. (If maximum too weak, no change.)



# Fitting

$$\mathbf{p} = T_{s,\theta,t_x,t_y}(\widetilde{\mathbf{p}}) = s\,\mathbf{Q}_\theta \widetilde{\mathbf{p}} + \mathbf{r}_{t_x,t_y} \quad \text{where} \quad \widetilde{\mathbf{p}} = \mathbf{P}\mathbf{b} + \bar{\mathbf{p}}$$

- Calculate an edge map of the image
- For each landmark $\mathbf{p}_i$ we find a line normal to the shape contour.
- New position $\mathbf{p}_i'$ is the maximum of the edge map on the line. (If maximum too weak, no change.)
- Adjust pose parameters $\theta$, $s$, $t_x$, $t_y$ by the alignment algorithm.

# Fitting

$$\mathbf{p} = T_{s,\theta,t_x,t_y}(\widetilde{\mathbf{p}}) = s\,\mathbf{Q}_\theta \widetilde{\mathbf{p}} + \mathbf{r}_{t_x,t_y} \quad \text{where} \quad \widetilde{\mathbf{p}} = \mathbf{P}\mathbf{b} + \bar{\mathbf{p}}$$

- Calculate an edge map of the image
- For each landmark $\mathbf{p}_i$ we find a line normal to the shape contour.
- New position $\mathbf{p}_i'$ is the maximum of the edge map on the line. (If maximum too weak, no change.)
- Adjust pose parameters $\theta$, $s$, $t_x$, $t_y$ by the alignment algorithm.
- Adjust shape parameters $\mathbf{b}$ as follows:

$$\widetilde{\mathbf{p}}_i' = T^{-1}(\mathbf{p}_i')$$
$$\mathbf{b}' = \mathbf{P}^{-1}(\widetilde{\mathbf{p}}_i' - \bar{\mathbf{p}}) = \mathbf{P}^T(\widetilde{\mathbf{p}}_i' - \bar{\mathbf{p}})$$

# Fitting

$$\mathbf{p} = T_{s,\theta,t_x,t_y}(\widetilde{\mathbf{p}}) = s\,\mathbf{Q}_\theta \widetilde{\mathbf{p}} + \mathbf{r}_{t_x,t_y} \quad \text{where} \quad \widetilde{\mathbf{p}} = \mathbf{Pb} + \bar{\mathbf{p}}$$

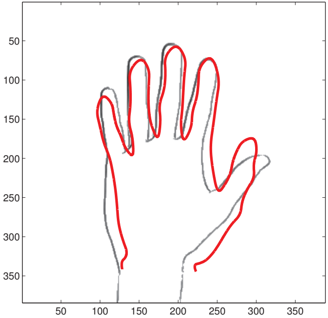
- Calculate an edge map of the image
- For each landmark $\mathbf{p}_i$ we find a line normal to the shape contour.
- New position $\mathbf{p}'_i$ is the maximum of the edge map on the line. (If maximum too weak, no change.)
- Adjust pose parameters $\theta$, $s$, $t_x$, $t_y$ by the alignment algorithm.
- Adjust shape parameters $\mathbf{b}$ as follows:

$$\widetilde{\mathbf{p}}'_i = T^{-1}(\mathbf{p}'_i)$$
$$\mathbf{b}' = \mathbf{P}^{-1}(\widetilde{\mathbf{p}}'_i - \bar{\mathbf{p}}) = \mathbf{P}^T(\widetilde{\mathbf{p}}'_i - \bar{\mathbf{p}})$$

- Repeat until convergence

# Example

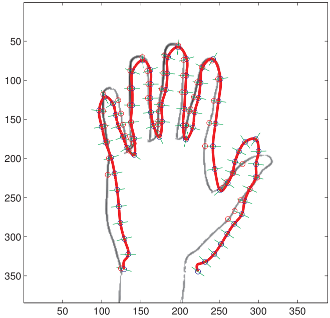
Hand image



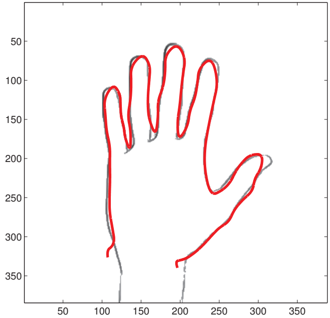

Edge map + initial shape.

The image is smoothed and a gradient magnitude image calculated in each color channel. The edge map is a maximum over the three color channels, thresholded to obtain a clean background.

# Example




First iteration




Final postition