Leow Wee Kheng

CS4243 Computer Vision and Pattern Recognition

Active Shape

Similar yet different shapes abound

Dealing with Shape

- How to represent normal shape variations?
- How to change shape?
- How to recognise shape?

Active Shape Model

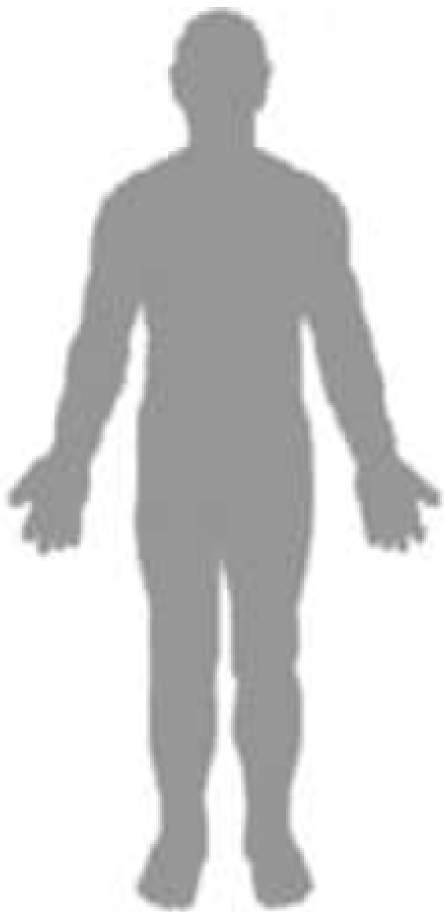
- Represent shape model as distribution of points
  - Point distribution model
- Use PCA to identify major variations
  - Eigen shape model

Model Construction

1. Collect a set of training samples.
2. Mark corresponding landmark points, collect sample shape vectors.
3. Perform spatial alignment.
4. Apply PCA to identify major components.

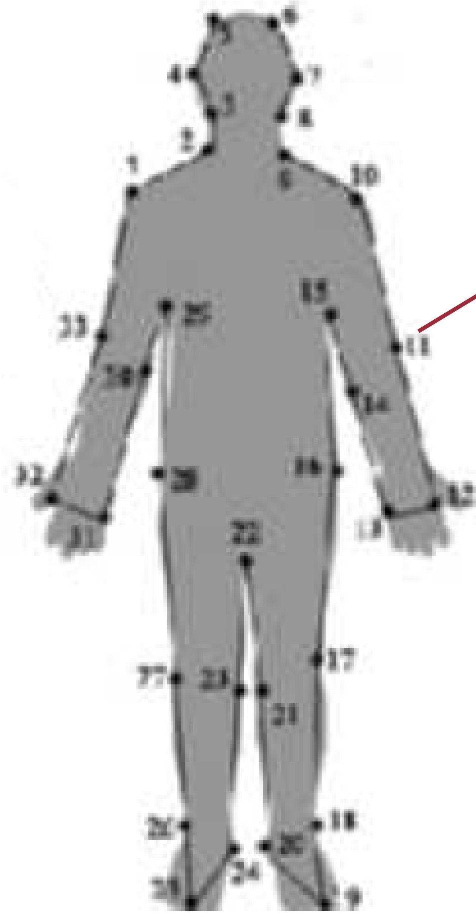
Step 1

- Collect training samples



Step 2

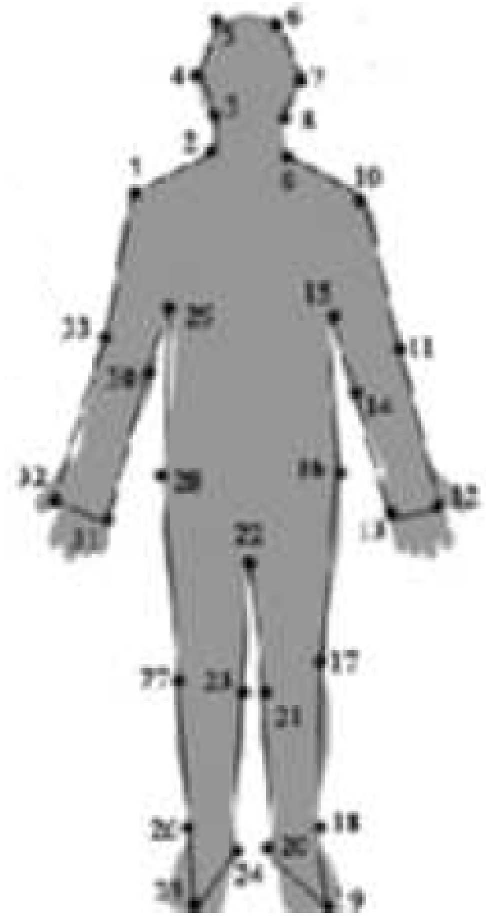
- Mark landmark points consistently on each sample



landmark point

Step 2

- Collect landmark points into shape vector



i-th sample shape vector

$$\mathrm{s}'_i = (x'_{i0}, y'_{i0}, x'_{i1}, y'_{i1}, \ldots, x'_{in}, y'_{in})^\top$$

position of *j*-th landmark point

$$(x'_{ij}, y'_{ij})$$

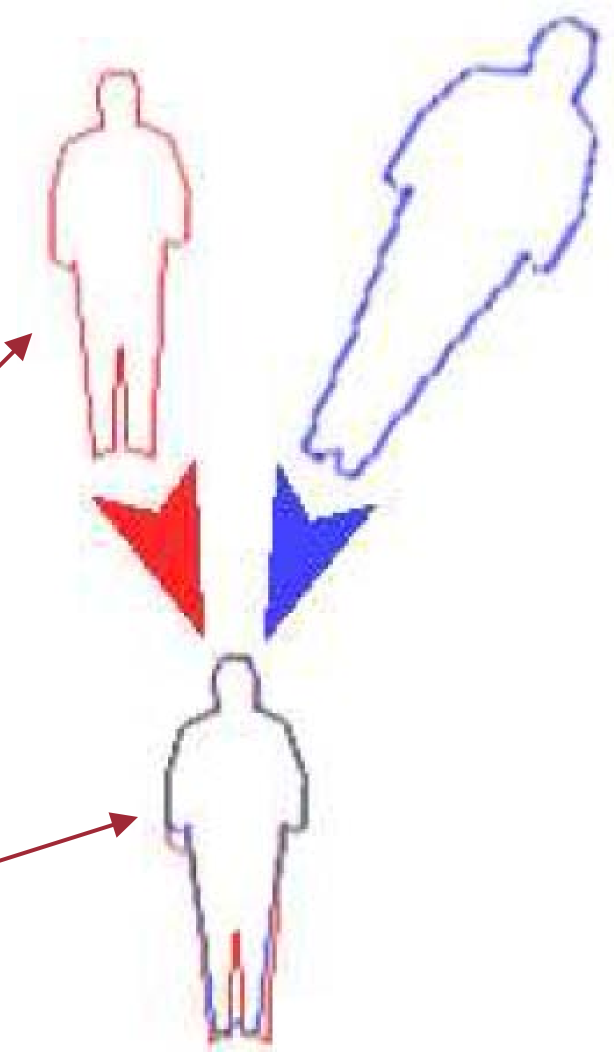
Step 3

- Perform spatial alignment
  - Translate to consistent position (e.g., align centroid)
  - Scale to same size
  - Rotate to same orientation

$$\mathbf{s}'_i = (x'_{i0}, y'_{i0}, x'_{i1}, y'_{i1}, \ldots, x'_{in}, y'_{in})^\top$$

$$\mathbf{s}_i = (x_{i0}, y_{i0}, x_{i1}, y_{i1}, \ldots, x_{in}, y_{in})^\top$$

aligned shape vector



Step 4

- Apply PCA to identify major components
  - Apply PCA

$$\mathbf{Q} = [\mathbf{q}_1, \ldots, \mathbf{q}_m]$$

  - Keep top k compoments

$$\mathbf{Q} = [\mathbf{q}_1 \quad \ldots \quad \mathbf{q}_k]$$

  - Mapping between aligned shape and eigenshape

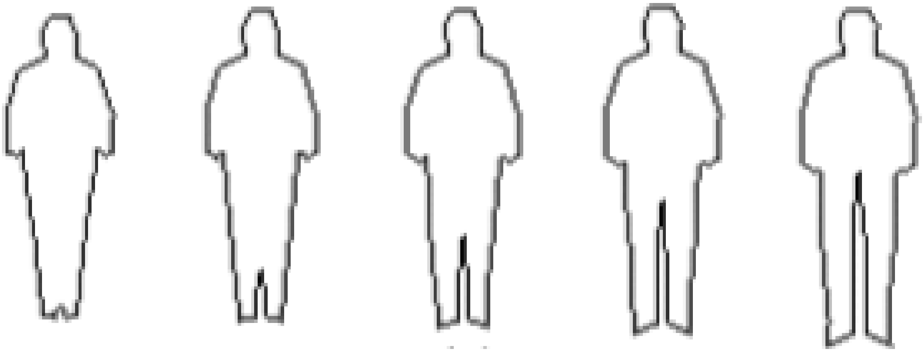
$$\mathbf{e}_i = \mathbf{Q}^\top (\mathbf{s}_i - \overline{\mathbf{s}}) \qquad \mathbf{s}_i = \overline{\mathbf{s}} + \mathbf{Q}\,\mathbf{e}_i$$

  - Change shape parameters $\mathbf{e}$ gives different shape

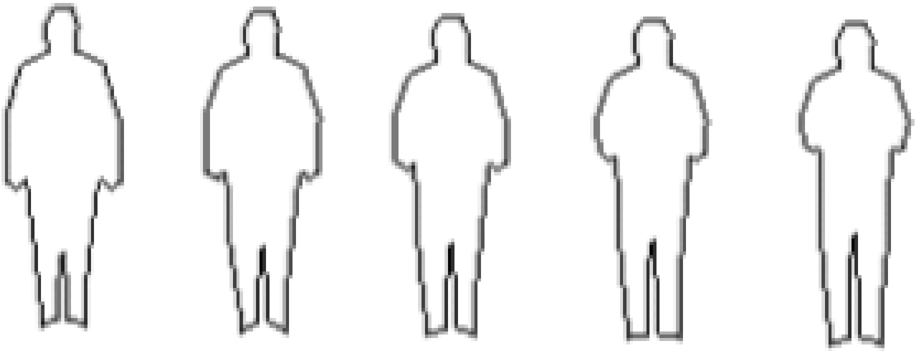
$$\mathbf{s} = \overline{\mathbf{s}} + \mathbf{Q}\,\mathbf{e}$$

Step 4

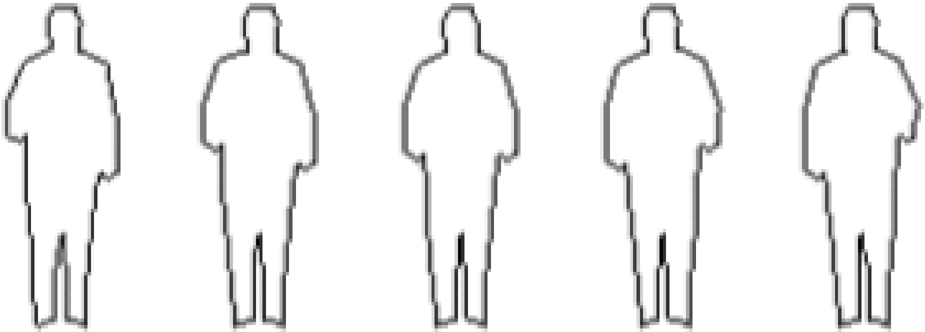
- Vary e_1



- Vary e_2



- Vary e_3



Model Matching

- Generate a shape from shape model to match input shape.
- Can be used for
  - Shape matching: is it a known shape?
  - Shape detection: where is known shape?
  - Shape segmentation: get outline of known shape
  - Shape recognition: what is input shape?

Basic Steps

1. Extract features from input image.
2. Initialise shape parameters e.
3. Repeat until convergence
   - Generate new shape s with shape parameters e.
   - Align s to input features: rigid registration.
   - Compare s with input features: difference measure.
   - Use difference to update shape parameters e.

Basic Steps

1. Extract features from input image.
2. Initialise shape parameters e.
3. Repeat until convergence
   - Generate new shape s with shape parameters e.
   - Align s to input features: rigid registration.
   - Compare s with input features: difference measure.
   - Use difference to update shape parameters e.

Image features

- Many possible features
  - Local intensity maximal or inimal
  - Edges
  - Corners
  - Region boundaries

Image features

- Local intensity maximal / minimal
  - Apply non-maximum/minimum suppression.
- Edges
  - Use edge detectors, e.g., Canny's edge detector.
- Corners
  - Corner detectors, e.g., Harris corner.
- Region boundaries
  - Need to first perform image segmentation to identity homogeneous regions.

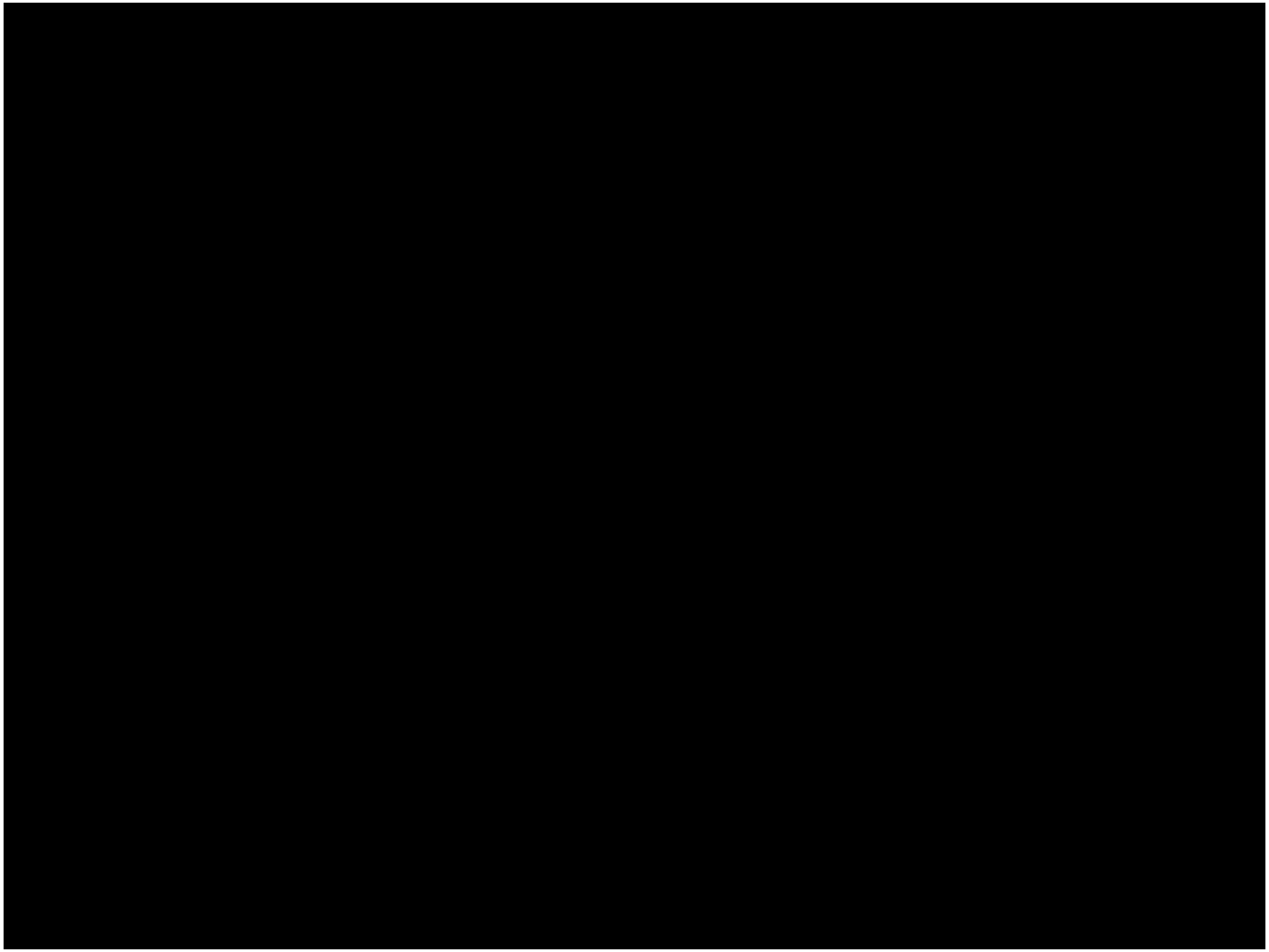
Basic Steps

1. Extract features from input image.
2. Initialise shape parameters e.
3. Repeat until convergence
   - Generate new shape s with shape parameters e.
   - Align s to input features: rigid registration.
   - Compare s with input features: difference measure.
   - Use difference to update shape parameters e.

Initialisation

- Depends on problem, no fixed method.
- Basic ideas:
  - Start with $\mathbf{e} = \mathbf{0}$; get mean shape $\bar{\mathbf{s}}$.
  - Use simple methods to get approximate solution.
  - Use approximate solution as initial setting.

Initialisation



Basic Steps

1. Extract features from input image.
2. Initialise shape parameters e.
3. Repeat until convergence
   - Generate new shape s with shape parameters e.
   - Align s to input features: rigid registration.
   - Compare s with input features: difference measure.
   - Use difference to update shape parameters e.

Generate Shape

- Generate shape $\mathbf{s}$ with parameters $\mathbf{e}$

$$\mathbf{s} = \bar{\mathbf{s}} + \mathbf{Q}\,\mathbf{e}$$

- Be careful; otherwise, can get weird shape!
  - Bad example 1:

$$\mathbf{e} = -\mathbf{Q}^{\mathrm{T}}\bar{\mathbf{s}}$$

  - Bad example 2:

$$\mathbf{e} = [1000, 0, 0, \ldots, 0]$$

- Need to constrain $\mathbf{e}$
  - Example, for all k

$$|e_k| < 3\sigma_k = 3\sqrt{\lambda_k}$$

Basic Steps

1. Extract features from input image.
2. Initialise shape parameters e.
3. Repeat until convergence
   - Generate new shape s with shape parameters e.
   - Align s to input features: rigid registration.
   - Compare s with input features: difference measure.
   - Use difference to update shape parameters e.

Rigid Registration

- Two sets of points with known correspondence: $\{ \mathbf{p}_i \}, \{ \mathbf{p}'_i \}, i = 1,\ldots, n.$
- Align two point sets without shape change.
- Possible transformations:
  - Scaling s
  - Rotation $\mathbf{R}$
  - Translation $\mathbf{T}$
- That is, find s, $\mathbf{R}$, $\mathbf{T}$ so that

$$\mathbf{p}'_i = s\,\mathbf{R}\,\mathbf{p}_i + \mathbf{T}$$

Step 1

- Remove translation
  - Compute centroids

$$\overline{\mathrm{p}} = \frac{1}{n}\sum_{i=1}^{n} \mathrm{p}_i\,, \quad \overline{\mathrm{p}}' = \frac{1}{n}\sum_{i=1}^{n} \mathrm{p}'_i$$

  - Subtract centroids from points

$$\mathrm{r}_i = \mathrm{p}_i - \overline{\mathrm{p}}\,, \quad \mathrm{r}'_i = \mathrm{p}'_i - \overline{\mathrm{p}}'$$

  - Now, both centroids are at origin, i.e., no translation.

Step 2

- Compute scaling factor
  - Compute ratio of variance

$$s^2 = \frac{\sum_{i=1}^{n} \|\mathbf{r}'_i\|^2}{\sum_{i=1}^{n} \|\mathbf{r}_i\|^2}$$

  - Scaling factor s is square-root of ratio.

Step 3

- Compute rotation
  - Form matrix $\mathbf{M}$, compute $\mathbf{Q}$

$$\mathbf{M} = \sum_{i=1}^{n} \mathbf{r}'_i \, \mathbf{r}_i^\top \qquad\qquad \mathbf{Q} = \mathbf{M}^\top \mathbf{M}$$

  - Perform eigen-decomposition of $\mathbf{Q}$

$$\mathbf{Q} = \mathbf{V} \mathbf{\Lambda} \mathbf{V}^\top$$

$$\mathbf{V} = [\mathbf{v}_1 \; \mathbf{v}_2 \; \mathbf{v}_3], \quad \mathbf{\Lambda} = \mathrm{diag}(\lambda_1, \lambda_2, \lambda_3)$$

  - Compute inverse square-root of $\mathbf{Q}$

$$\mathbf{Q}^{-1/2} = \mathbf{V} \, \mathrm{diag}\left(\frac{1}{\sqrt{\lambda_1}}, \frac{1}{\sqrt{\lambda_2}}, \frac{1}{\sqrt{\lambda_3}} \right) \mathbf{V}^\top$$

  - Compute rotation matrix $\mathbf{R}$

$$\mathbf{R} = \mathbf{M} \mathbf{Q}^{-1/2}$$

Step 4

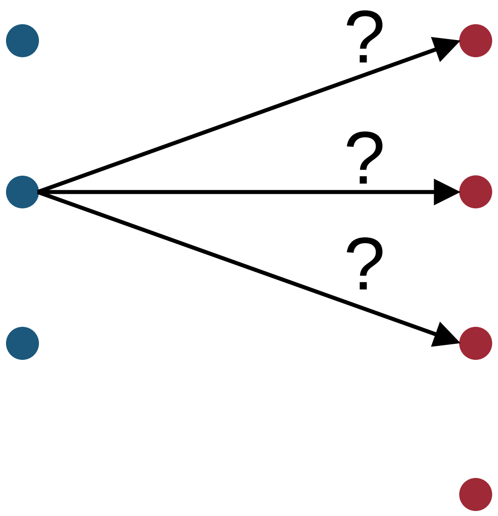
- Compute translation
  - Use computed s and $\mathbf{T}$

$$\mathrm{T} = \overline{\mathrm{p}}' - s\,\mathrm{R}\,\overline{\mathrm{p}}$$

- Computed s, $\mathbf{R}$, $\mathbf{T}$ are best-fitting (min. error)

Unknown Correspondence

- Usually, correspondence is unknown

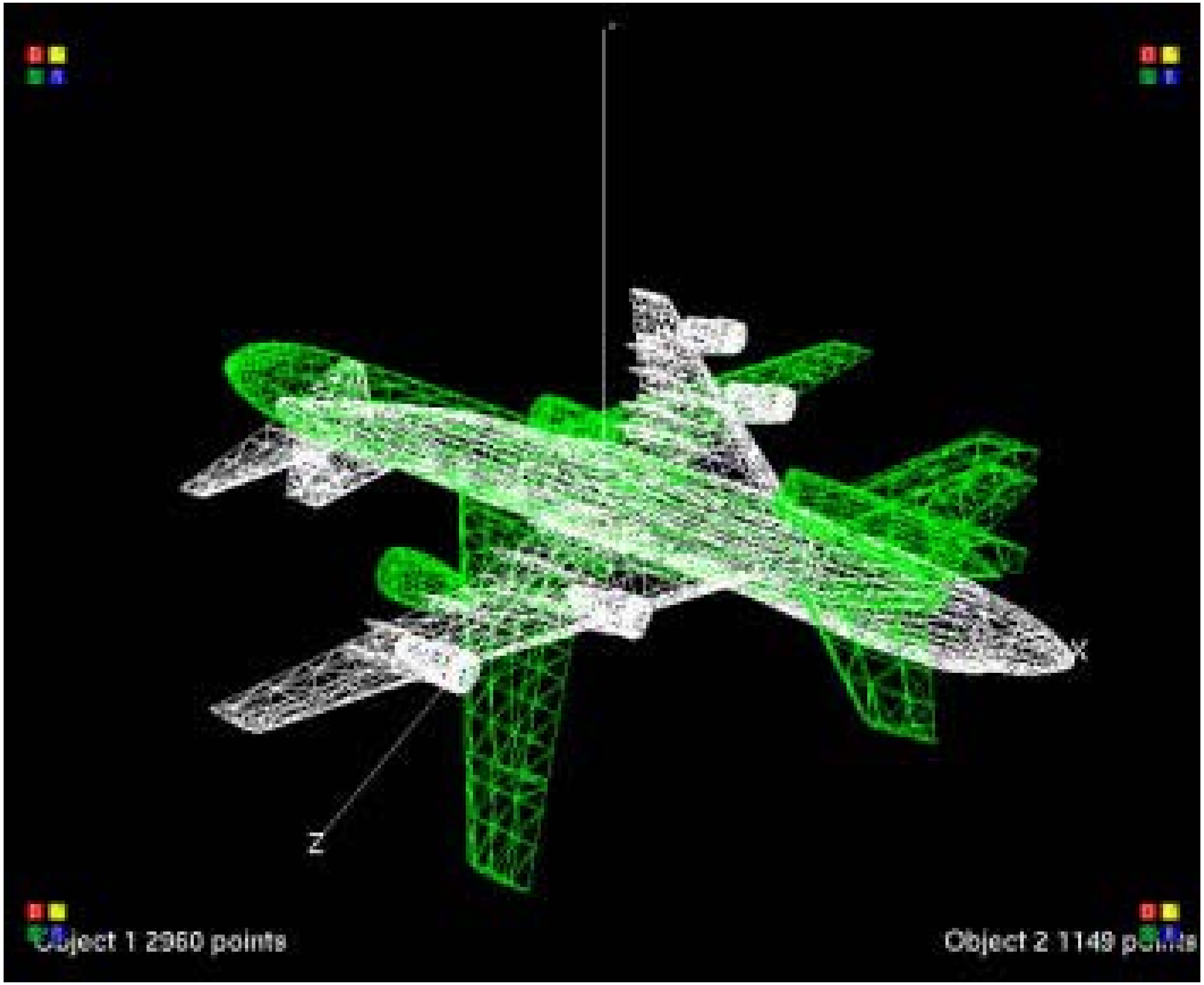


- Number of points are not the same.

Iterative Closest Point

- Make educated guess, then iteratively refine.
- Repeat until convergence
  1. Find closest point $\mathbf{p}'_j$ of each $\mathbf{p}_i$.
  2. Find best s, $\mathbf{R}$, $\mathbf{T}$ that align $\mathbf{p}_i$ to $\mathbf{p}'_j$.
  3. Align $\mathbf{p}_i$ to $\mathbf{p}'_j$.

Iterative Closest Point

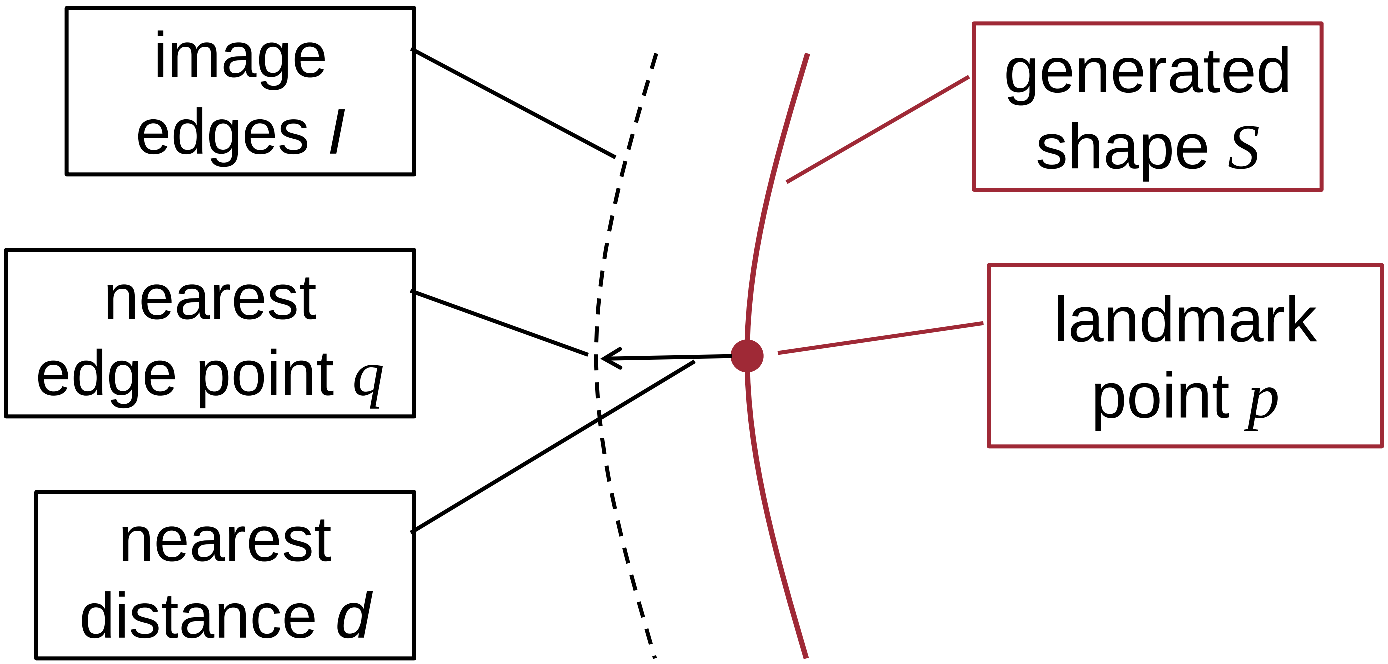


Basic Steps

1. Extract features from input image.
2. Initialise shape parameters e.
3. Repeat until convergence
   - Generate new shape s with shape parameters e.
   - Align s to input features: rigid registration.
   - Compare s with input features: difference measure.
   - Use difference to update shape parameters e.

Difference Measure

- Simplest: nearest distance



  - Sum nearest distance

$$D(S,I)=\sum_{p\in S}\min_{q\in I} d(p,q)$$

Other Difference Measures

- Chamfer distance
  - Average nearest distance

$$M(S,I) = \frac{1}{|S|} \sum_{p \in S} \min_{q \in I}(p,q)$$

  - Symmetric form

$$C(S,I) = M(S,I) + M(I,S)$$

Other Difference Measures

- Hausdorff distance
  - Largest nearest distance

$$L(S, I) = \max_{p \in S} \min_{q \in I} (p, q)$$

  - Symmetric form

$$H(S, I) = \max(L(S, I), L(I, S))$$

Basic Steps

1. Extract features from input image.
2. Initialise shape parameters e.
3. Repeat until convergence
   - Generate new shape s with shape parameters e.
   - Align s to input features: rigid registration.
   - Compare s with input features: difference measure.
   - Use difference to update shape parameters e.

Update Parameters

- Simplest method: Gradient descent
  - Parameters $\mathbf{a} = (a_1, \ldots, a_m)$
  - Difference or error: $E(\mathbf{a})$
  - Compute gradient $\partial E / \partial \mathbf{a}$
  - Update rule:

constant update rate

$$\Delta \mathbf{a} = -\eta \frac{\partial E}{\partial \mathbf{a}} \quad \text{i.e.,} \quad \mathbf{a}(t+1) = \mathbf{a}(t) - \eta \frac{\partial E}{\partial \mathbf{a}}$$

  - If E increases with a_k, $\partial E / \partial a_k$ is positive. Then, decrease $a_k \to$ decrease E.
  - If E decreases with a_k, $\partial E / \partial a_k$ is negative. Then, increase $a_k \to$ decrease E.
  - So, always update $\mathbf{a}$ to decrease E.

- What if difficult to manually differentiate E ?
  - Apply a method that does not require gradient, e.g., Powell's direction set method.
  - Estimate gradient.
    - If you do that really well, you get Powell's method.
  - Use Matlab symbolic differentiation to derive gradient.
  - Perturbation method:
    - Change $\mathbf{e}$ slightly to find direction of decreasing E.
  - Monte Carlo method:
    - Randomly generate $\mathbf{e}$.
  - Others…

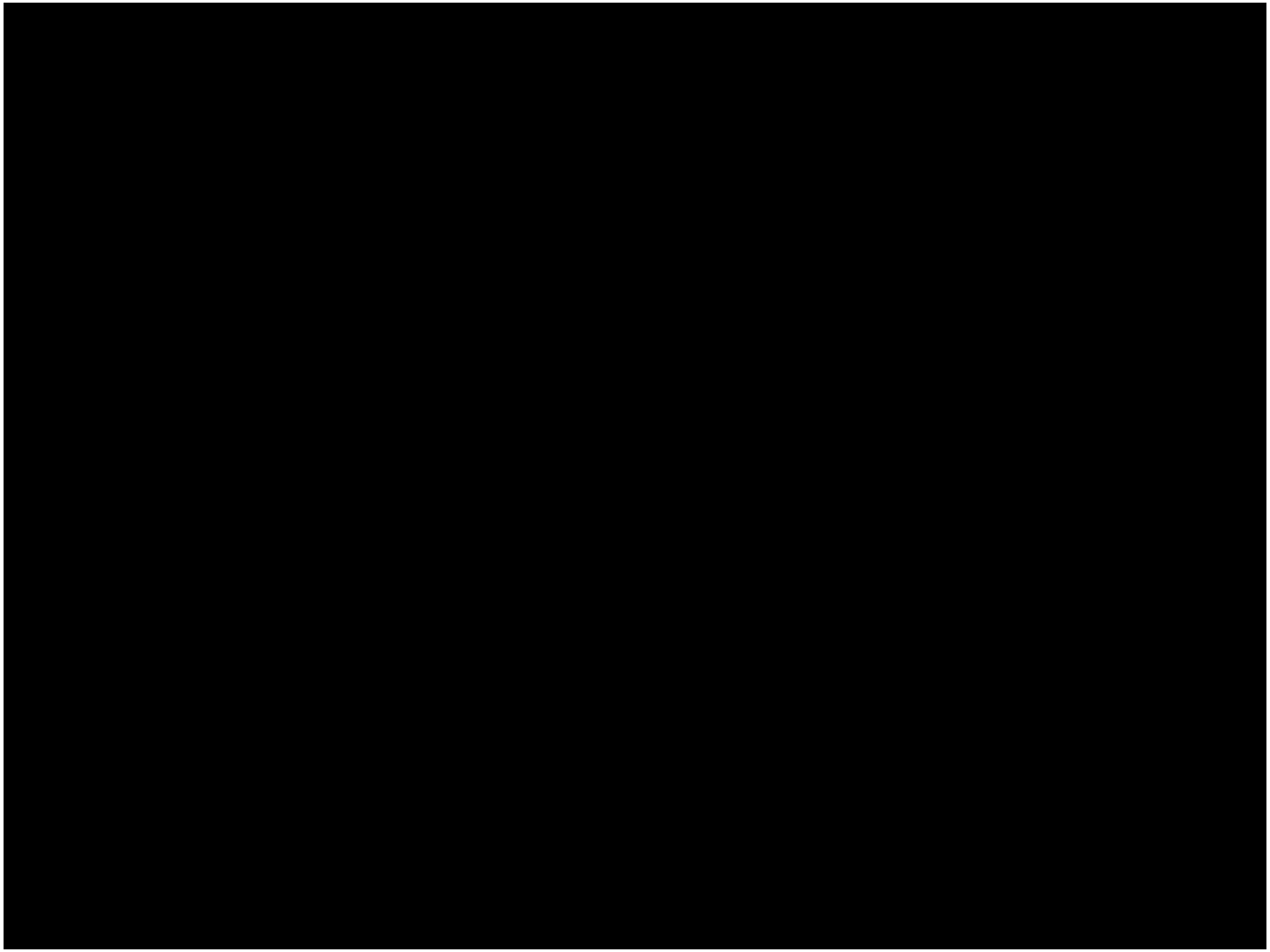
Basic Steps

1. Extract features from input image.
2. Initialise shape parameters e.
3. Repeat until convergence
   - Generate new shape s with shape parameters e.
   - Align s to input features: rigid registration.
   - Compare s with input features: difference measure.
   - Use difference to update shape parameters e.

Convergence

- Several possible criteria
  - When error E is small enough.
  - When change of error ΔE is small enough.
  - After enough iterations.

Example



Summary

- Apply PCA to extract major variations.
- Can generate shapes within normal variations.
- Can be applied to many applications.

Further Reading

- Active shape: [Cootes95]
- Active appearance: [Cootes01]
- Iterative closest point: [Bsel92]
- Robust ICP: [Phillips07]
- Application: Human detection [Setyawan01]

References

- P. J. Besl and N. D. McKay. A method for registration of 3-D shapes. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992.
- T. F. Cootes, C. J. Taylor, D. H. Cooper, and J. Graham. Active Shape Models—Their Training and Application. Computer Vision and Image Understanding. 61(1):38–59, 1995.
- T. F. Cootes, G. J. Edwards, and C. J. Taylor. Active Appearance Models. IEEE Transactions on Pattern Analysis and Machine Intelligence. 23(6):681–685, 2001.
- Phillips J. M., Liu R., and Tomasi C. Outlier robust ICP for minimizing fractional RMSD. In *Proc. of Int. Conf. on 3D Digital Imaging and Modeling* (2007), pp. 427–434.
- H. Setyawan. *Model-Based Human Detection in Images*. M.Sc. Thesis, NUS, 2001.