CSE 554

Lecture 8: Alignment

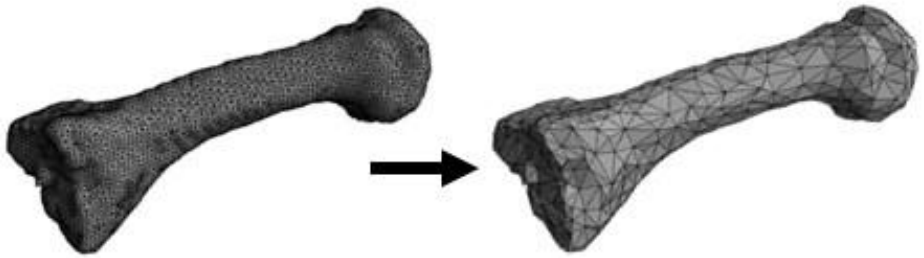
Fall 2016

Review

- Fairing (smoothing)
  - Relocating vertices to achieve a smoother appearance
  - Method: centroid averaging

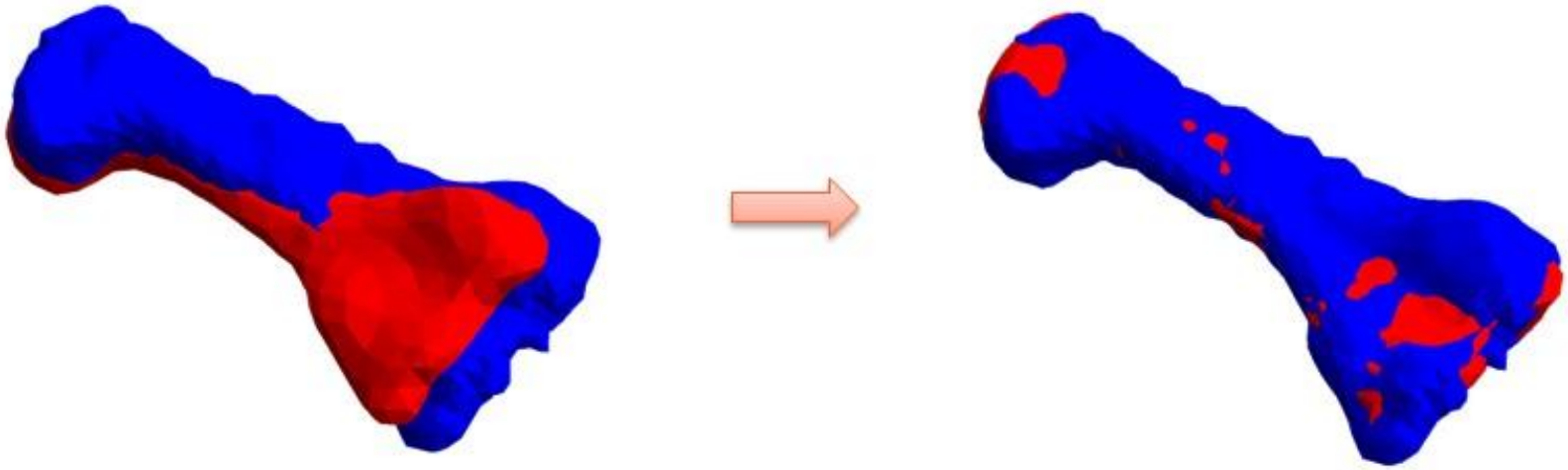


- Simplification
  - Reducing vertex count
  - Method: edge collapsing



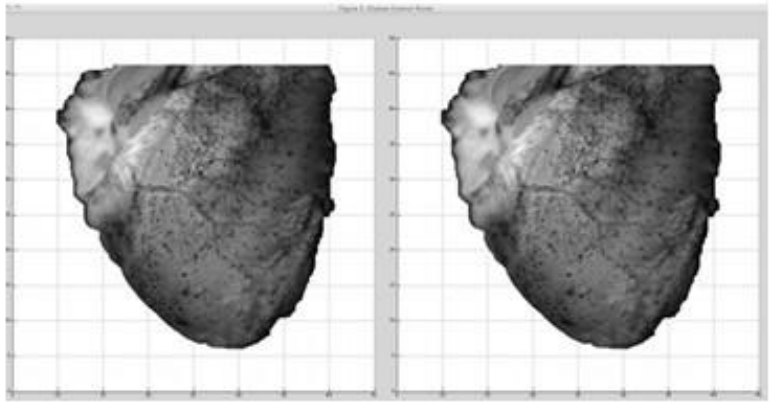
Registration

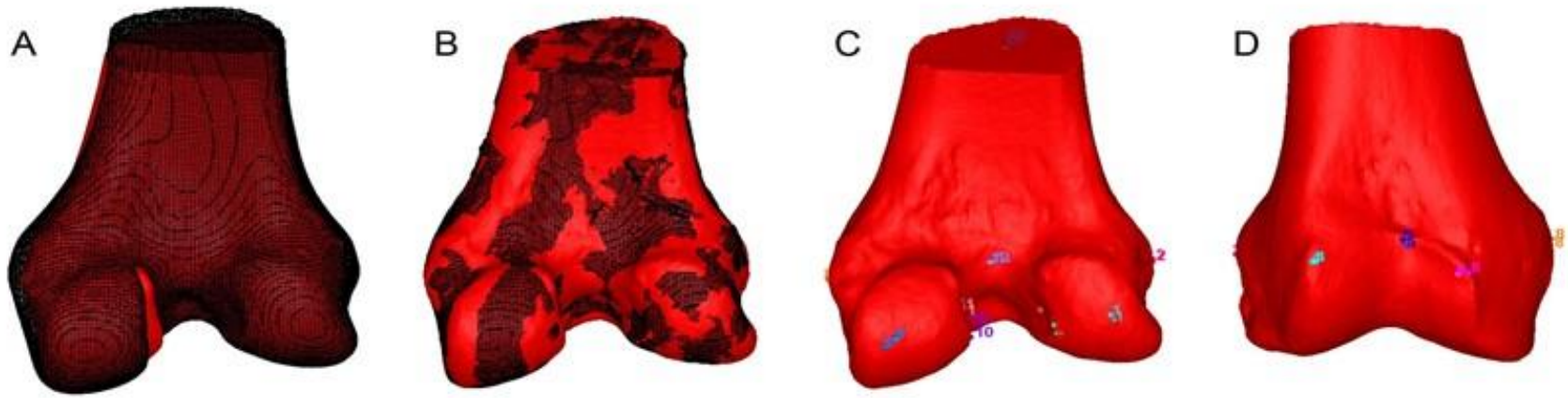
- Fitting one model to match the shape of another



Registration

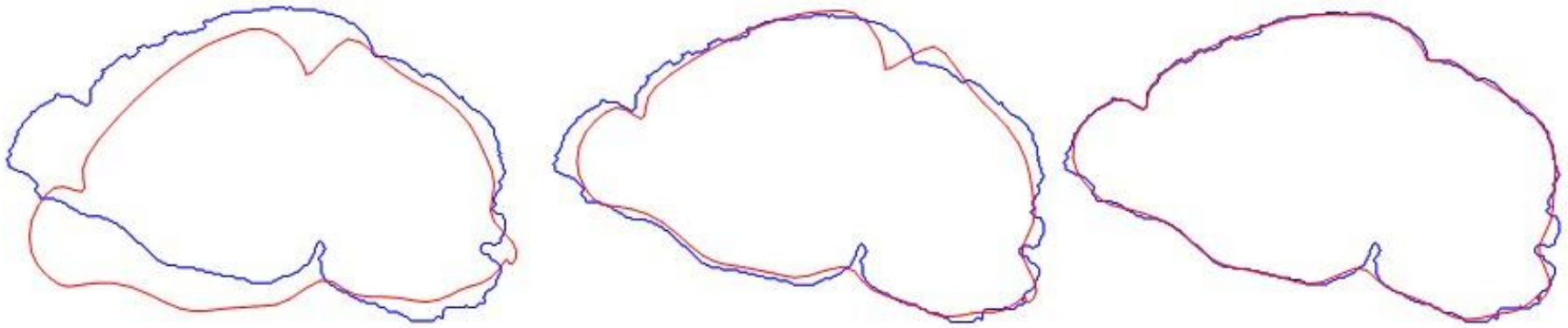
- Applications
  - Tracking and motion analysis
  - Automated annotation





Registration

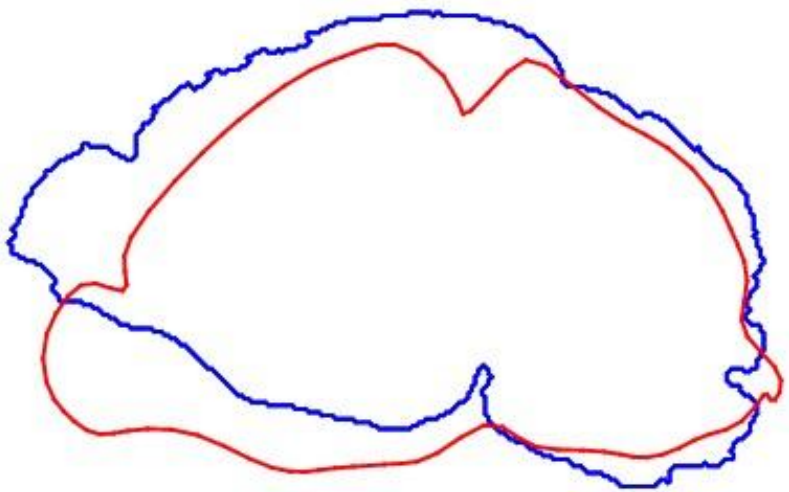
- Challenges: global and local shape differences
  - Imaging causes global shifts and tilts
    - Requires alignment
  - The shape of the organ or tissue differs in subjects and evolve over time
    - Requires deformation



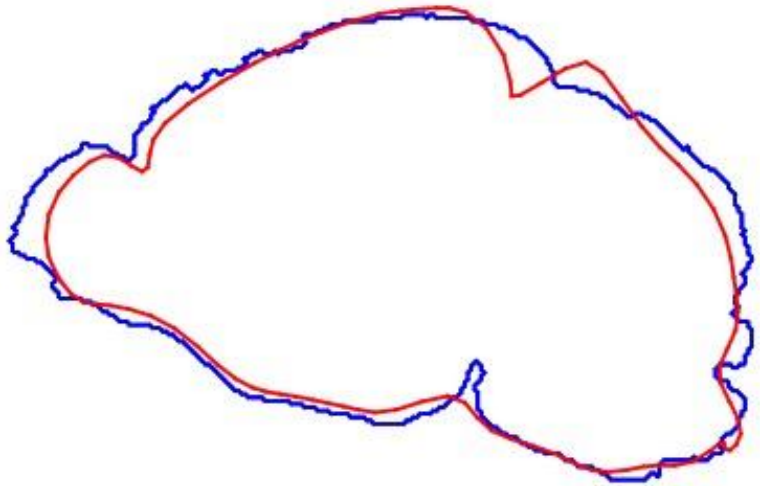
Brain outlines of two mice | After alignment | After deformation

Alignment

- Registration by translation or rotation
  - The structure stays "rigid" under these two transformations
    - Called rigid-body or isometric (distance-preserving) transformations
  - Mathematically, they are represented as matrix/vector operations



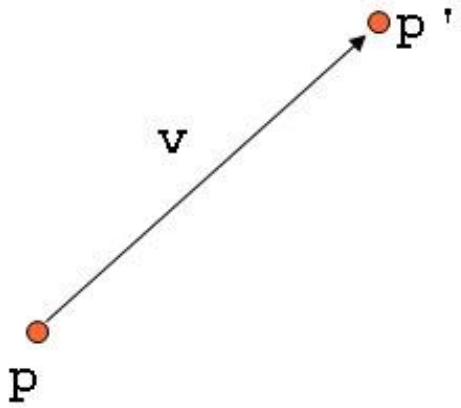
Before alignment



After alignment

Transformation Math

- Translation
  - Vector addition: $\mathbf{p}' = \mathbf{v} + \mathbf{p}$
  - 2D: $\begin{pmatrix} p_x' \\ p_y' \end{pmatrix} = \begin{pmatrix} v_x \\ v_y \end{pmatrix} + \begin{pmatrix} p_x \\ p_y \end{pmatrix}$
  - 3D: $\begin{pmatrix} p_x' \\ p_y' \\ p_z' \end{pmatrix} = \begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix} + \begin{pmatrix} p_x \\ p_y \\ p_z \end{pmatrix}$



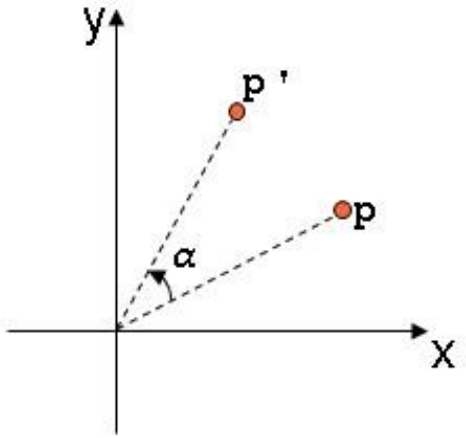
Transformation Math

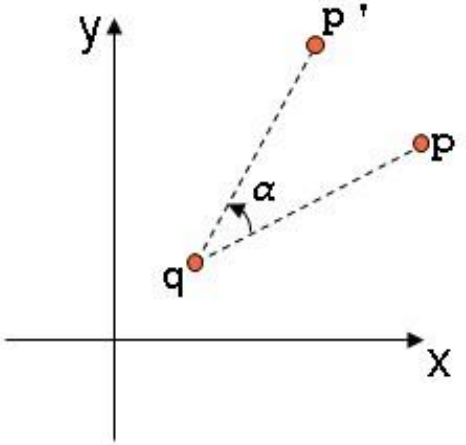
- Rotation
  - Matrix product: $\mathbf{p' = R \cdot p}$
  - 2D: $\begin{pmatrix} p_x' \\ p_y' \end{pmatrix} = R \cdot \begin{pmatrix} p_x \\ p_y \end{pmatrix}$

$$R = \begin{pmatrix} \mathrm{Cos}[\alpha] & -\mathrm{Sin}[\alpha] \\ \mathrm{Sin}[\alpha] & \mathrm{Cos}[\alpha] \end{pmatrix}$$

    - Rotate around the origin!
    - To rotate around another point q:

$$p' = R \cdot (p - q) + q$$





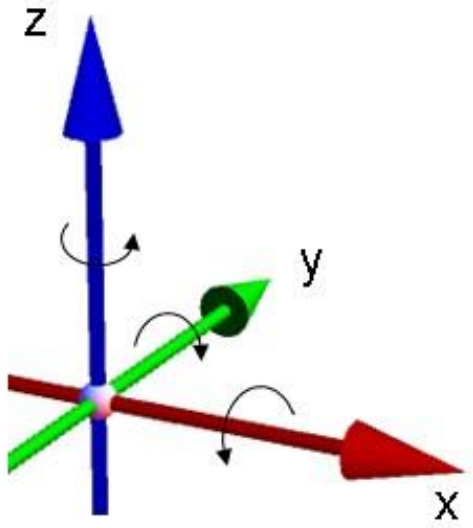
Transformation Math

- Rotation
  - Matrix product: $\mathbf{p}' = \mathbf{R} \cdot \mathbf{p}$
  - 3D: $\begin{pmatrix} p_x' \\ p_y' \\ p_z' \end{pmatrix} = R \cdot \begin{pmatrix} p_x \\ p_y \\ p_z \end{pmatrix}$

Around X axis: $R_x = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \text{Cos}[\alpha] & -\text{Sin}[\alpha] \\ 0 & \text{Sin}[\alpha] & \text{Cos}[\alpha] \end{pmatrix}$

Around Y axis: $R_y = \begin{pmatrix} \text{Cos}[a] & 0 & \text{Sin}[a] \\ 0 & 1 & 0 \\ -\text{Sin}[a] & 0 & \text{Cos}[a] \end{pmatrix}$

Around Z axis: $R_z = \begin{pmatrix} \text{Cos}[a] & -\text{Sin}[a] & 0 \\ \text{Sin}[a] & \text{Cos}[a] & 0 \\ 0 & 0 & 1 \end{pmatrix}$



Any arbitrary 3D rotation can be composed from these three rotations

Transformation Math

- Properties of an arbitrary rotational matrix
  - Orthonormal (orthogonal and normal): $R \cdot R^T = I$
    - Examples:

$$\begin{pmatrix} \text{Cos}[\alpha] & -\text{Sin}[\alpha] \\ \text{Sin}[\alpha] & \text{Cos}[\alpha] \end{pmatrix} \cdot \begin{pmatrix} \text{Cos}[\alpha] & \text{Sin}[\alpha] \\ -\text{Sin}[\alpha] & \text{Cos}[\alpha] \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & \text{Cos}[\alpha] & -\text{Sin}[\alpha] \\ 0 & \text{Sin}[\alpha] & \text{Cos}[\alpha] \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 \\ 0 & \text{Cos}[\alpha] & \text{Sin}[\alpha] \\ 0 & -\text{Sin}[\alpha] & \text{Cos}[\alpha] \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

  - Easy to invert: $R^{-1} = R^T$

Transformation Math

- Properties of an arbitrary rotational matrix
  - Any orthonormal 3x3 matrix represents a rotation around some axis (not limited to X,Y,Z)
    - The angle of rotation can be calculated from the trace of the matrix
      - Trace: sum of diagonal entries
      - 2D: The trace equals 2 Cos(a), where a is the rotation angle
      - 3D: The trace equals 1 + 2 Cos(a)

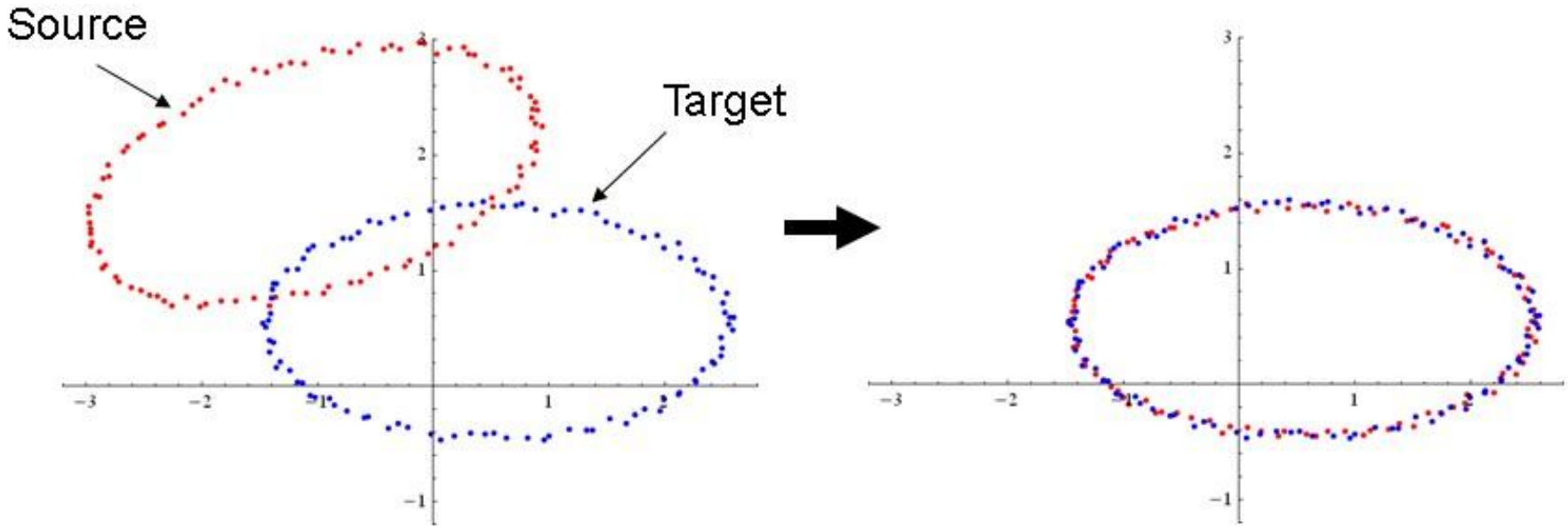
Examples:

$$\begin{pmatrix} \text{Cos}[\alpha] & -\text{Sin}[\alpha] \\ \text{Sin}[\alpha] & \text{Cos}[\alpha] \end{pmatrix} \qquad \begin{pmatrix} 1 & 0 & 0 \\ 0 & \text{Cos}[\alpha] & -\text{Sin}[\alpha] \\ 0 & \text{Sin}[\alpha] & \text{Cos}[\alpha] \end{pmatrix}$$

    - The larger the trace, the smaller the rotation angle

Alignment

- Input: two models represented as point sets
  - Source and target
- Output: locations of the translated and rotated source points

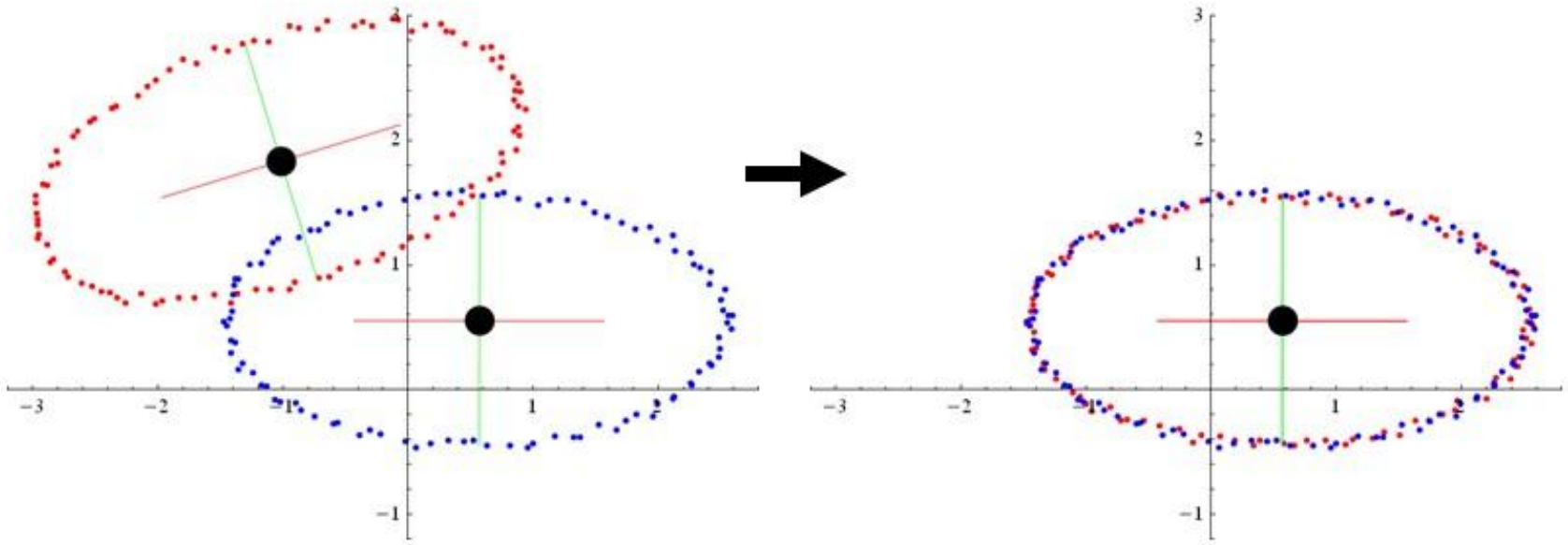


Alignment

- Method 1: Principal component analysis (PCA)
  - Aligning principal directions
- Method 2: Singular value decomposition (SVD)
  - Optimal alignment given prior knowledge of correspondence
- Method 3: Iterative closest point (ICP)
  - An iterative SVD algorithm that computes correspondences as it goes

Method 1: PCA

- Compute a shape-aware coordinate system for each model
  - Origin: Centroid of all points
  - Axes: Directions in which the model varies most or least
- Transform the source to align its origin/axes with the target



Basic Math

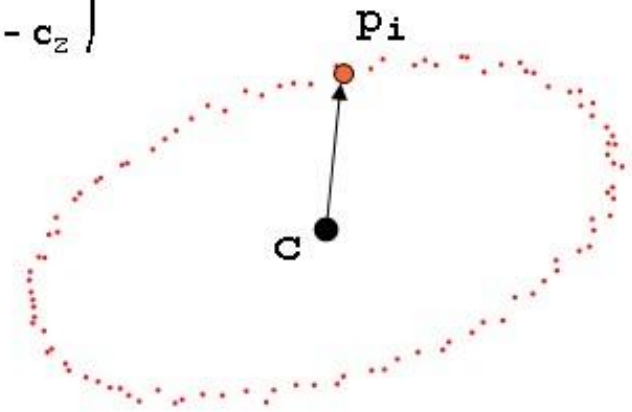
- Eigenvectors and eigenvalues
  - Let M be a square m-by-m matrix, v is an eigenvector and λ is an eigenvalue if:

$$\mathbf{M} \cdot \mathbf{v} = \lambda * \mathbf{v}$$

  - Any scalar multiples of an eigenvector is also an eigenvector (with the same eigenvalue).
    - So an eigenvector should be treated as a "line", rather than a vector
  - There are at most m non-zero eigenvectors
    - If M is symmetric, its eigenvectors are pairwise orthogonal

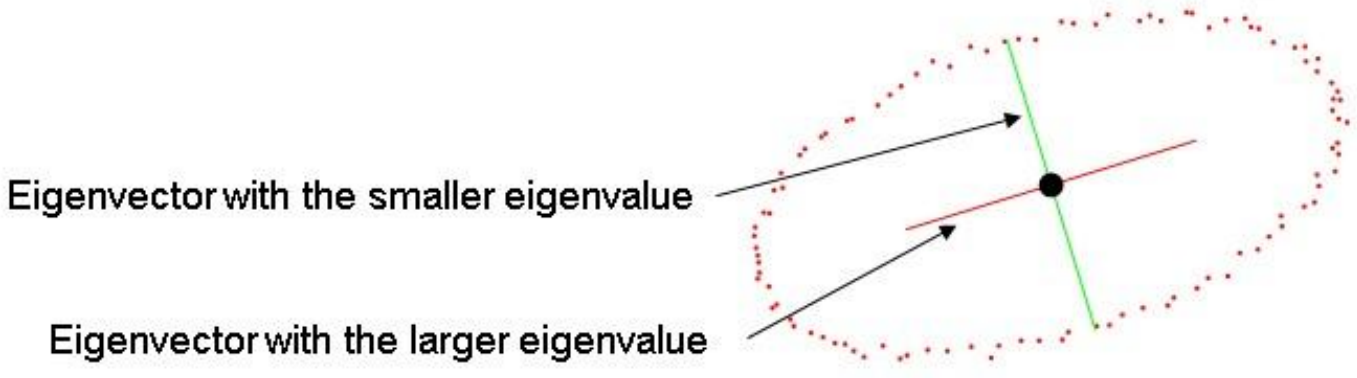
Method 1: PCA

- Computing axes: Principal Component Analysis (PCA)
  - Consider a set of points $p_1, \ldots, p_n$ with centroid location c
    - Construct matrix P whose *i*-th column is vector $p_i - c$
      - 2D (2 by n): $P = \begin{pmatrix} p_{1_x} - c_x & p_{2_x} - c_x & \cdots & p_{n_x} - c_x \\ p_{1_y} - c_y & p_{2_y} - c_y & \cdots & p_{n_y} - c_y \end{pmatrix}$
      - 3D (3 by n): $P = \begin{pmatrix} p_{1_x} - c_x & p_{2_x} - c_x & \cdots & p_{n_x} - c_x \\ p_{1_y} - c_y & p_{2_y} - c_y & \cdots & p_{n_y} - c_y \\ p_{1_z} - c_z & p_{2_z} - c_z & \cdots & p_{n_z} - c_z \end{pmatrix}$
    - Build the covariance matrix: $M = P \cdot P^T$
      - 2D: a 2 by 2 matrix
      - 3D: a 3 by 3 matrix
      - Symmetric!



Method 1: PCA

- Computing axes: Principal Component Analysis (PCA)
  - Eigenvectors of the covariance matrix represent principal directions of shape variation (2 in 2D; 3 in 3D)
  - Eigenvalues indicate amount of variation along each eigenvector
    - Eigenvector with largest (smallest) eigenvalue is the direction where the model shape varies the most (least)



Method 1: PCA

- PCA-based alignment
  - Let c_S, c_T be centroids of source and target.
  - First, translate source to align c_S with c_T:

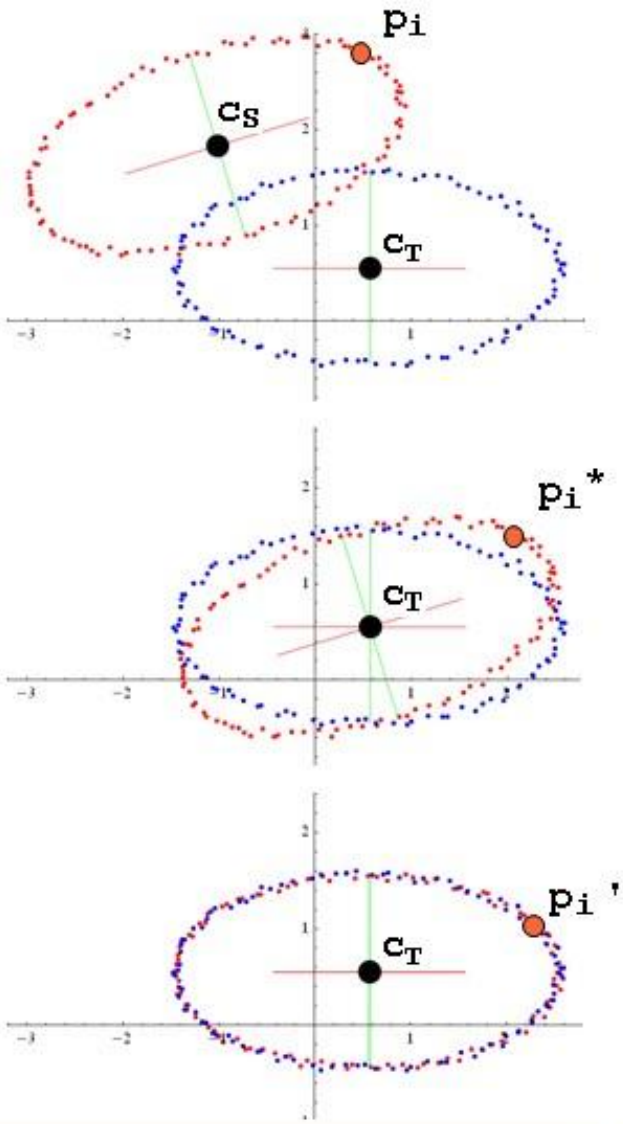
$$\mathbf{p_i}^* = \mathbf{p_i} + (\mathbf{c_T} - \mathbf{c_S})$$

  - Next, find rotation R that aligns two sets of PCA axes, and rotate source around c_T:

$$\mathbf{p_i}' = \mathbf{c_T} + \mathbf{R} \cdot (\mathbf{p_i}^* - \mathbf{c_T})$$

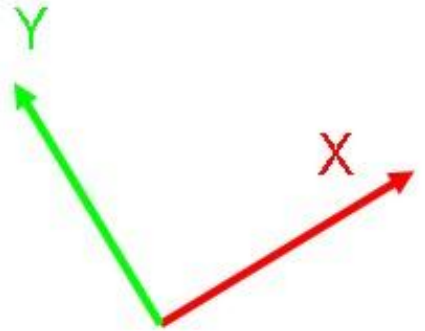
  - Combined:

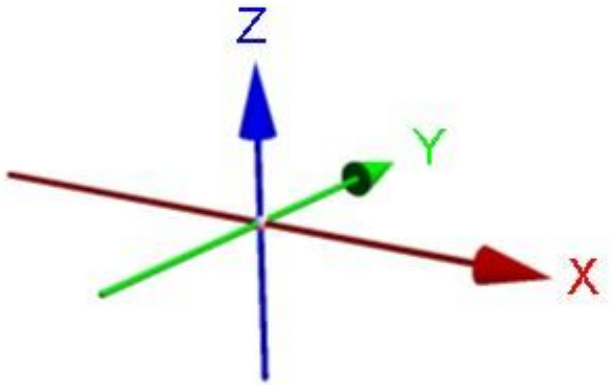
$$\mathbf{p_i}' = \mathbf{c_T} + \mathbf{R} \cdot (\mathbf{p_i} - \mathbf{c_S})$$



Method 1: PCA

- Oriented axes
  - 2D: Y is ccw from X
  - 3D: X,Y,Z follow right-handed rule





Method 1: PCA

- Finding rotation between two sets of oriented axes
  - Let A, B be two matrices whose columns are the axes
    - The axes are orthogonal and normalized (i.e., both A and B are orthonormal)
  - We wish to compute a rotation matrix R such that:

$$R \cdot A = B$$

  - Notice that A and B are orthonormal, so we have:

$$R = B \cdot A^{-1} = B \cdot A^{T}$$



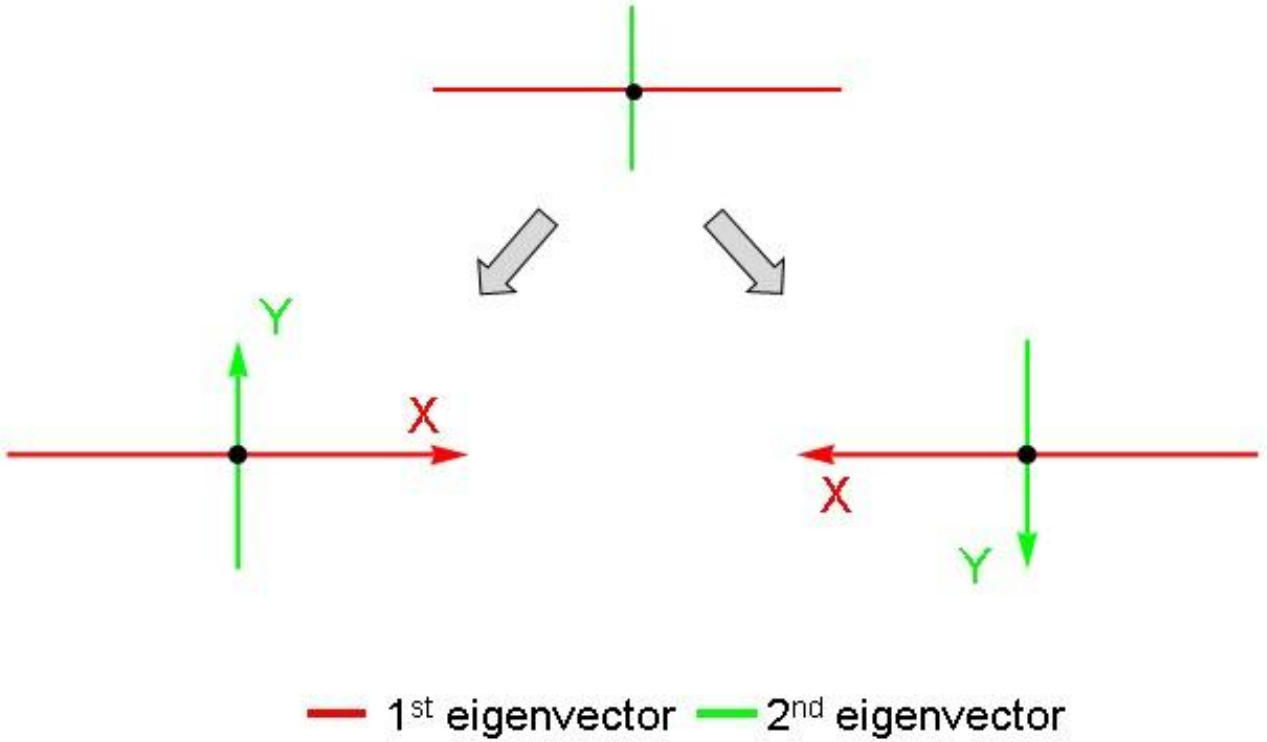
$$A = \begin{pmatrix} \frac{\sqrt{3}}{2} & -\frac{1}{2} \\ \frac{1}{2} & \frac{\sqrt{3}}{2} \end{pmatrix} \quad \begin{matrix} X1 & Y1 \end{matrix}$$

$$B = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad \begin{matrix} X2 & Y2 \end{matrix}$$

$$R = \begin{pmatrix} \frac{\sqrt{3}}{2} & \frac{1}{2} \\ -\frac{1}{2} & \frac{\sqrt{3}}{2} \end{pmatrix}$$

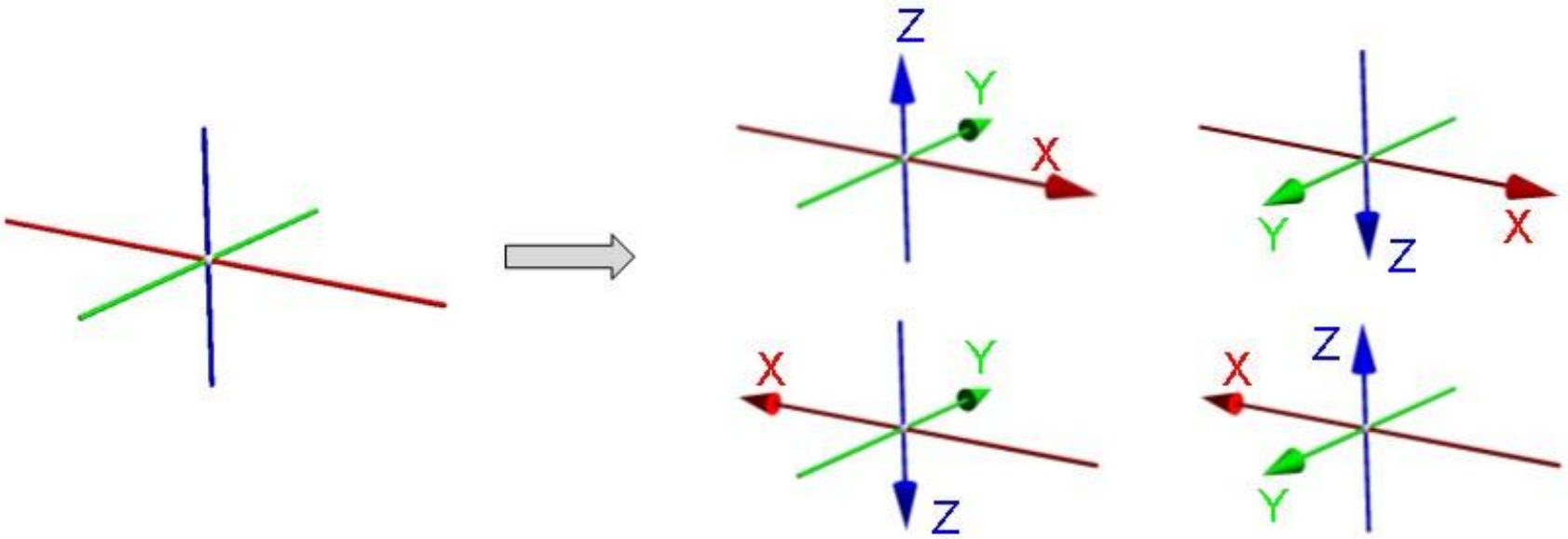
Method 1: PCA

- How to get oriented axes from eigenvectors?
  - In 2D, two eigenvectors can define 2 sets of oriented axes (whose X,Y correspond to 1st and 2nd eigenvectors)



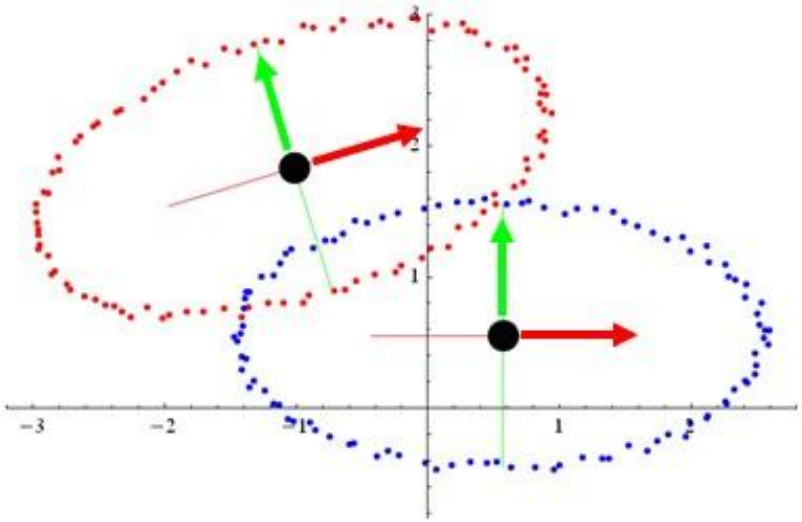
Method 1: PCA

- How to get oriented axes from eigenvectors?
  - In 3D, three eigenvectors can define 4 sets of oriented axes (whose X,Y,Z correspond to 1st, 2nd, 3rd eigenvectors)

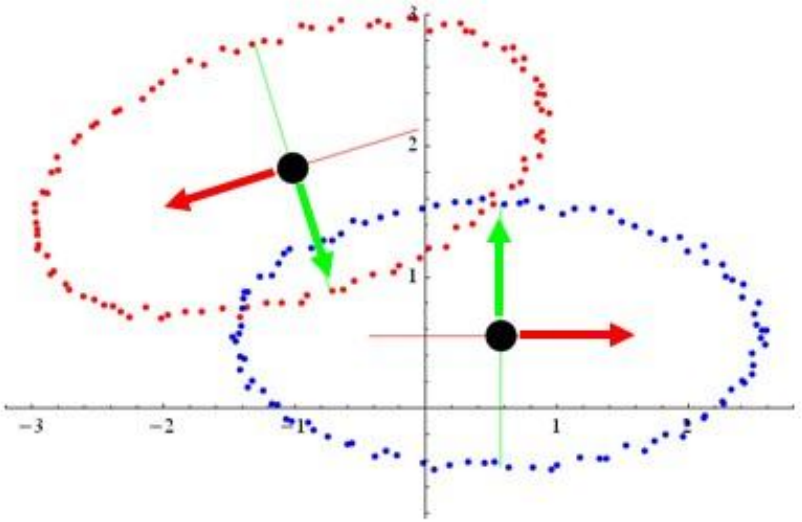


Method 1: PCA

- Finding rotation between two sets of eigenvectors
  - Fix the orientation of the target axes.
  - For each possible orientation of the source axes, compute R
  - Pick the R with smallest rotation angle (by checking the trace of R)
    - *Assuming the source is "close" to the target!*



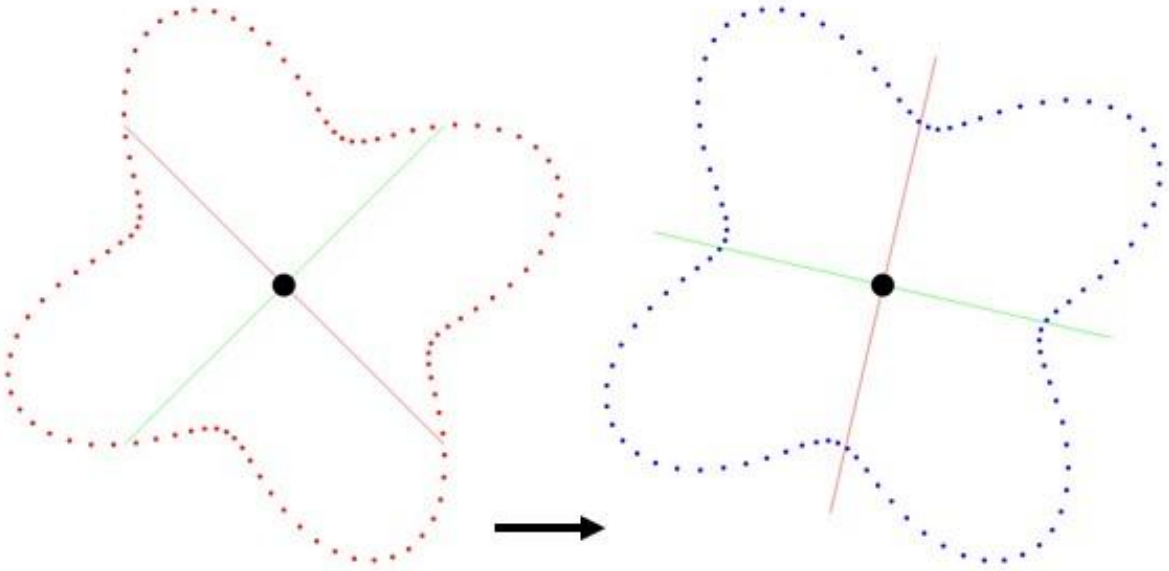
Smaller rotation



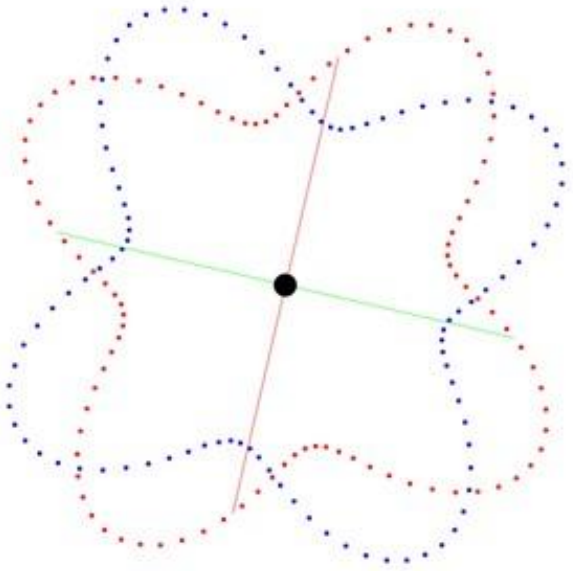
Larger rotation

Method 1: PCA

- Limitations
  - Axes can be unreliable for circular objects
    - Eigenvalues become similar, and eigenvectors become unstable



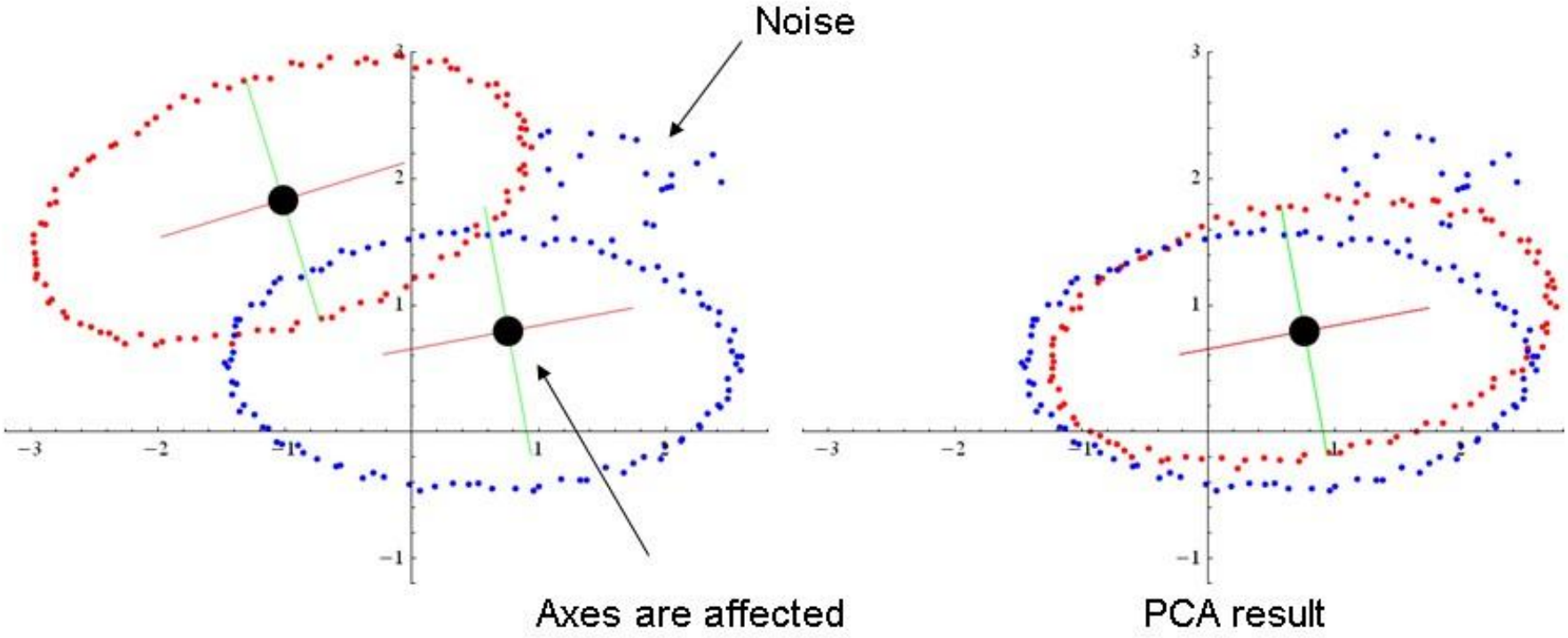
Rotation by a small angle



PCA result

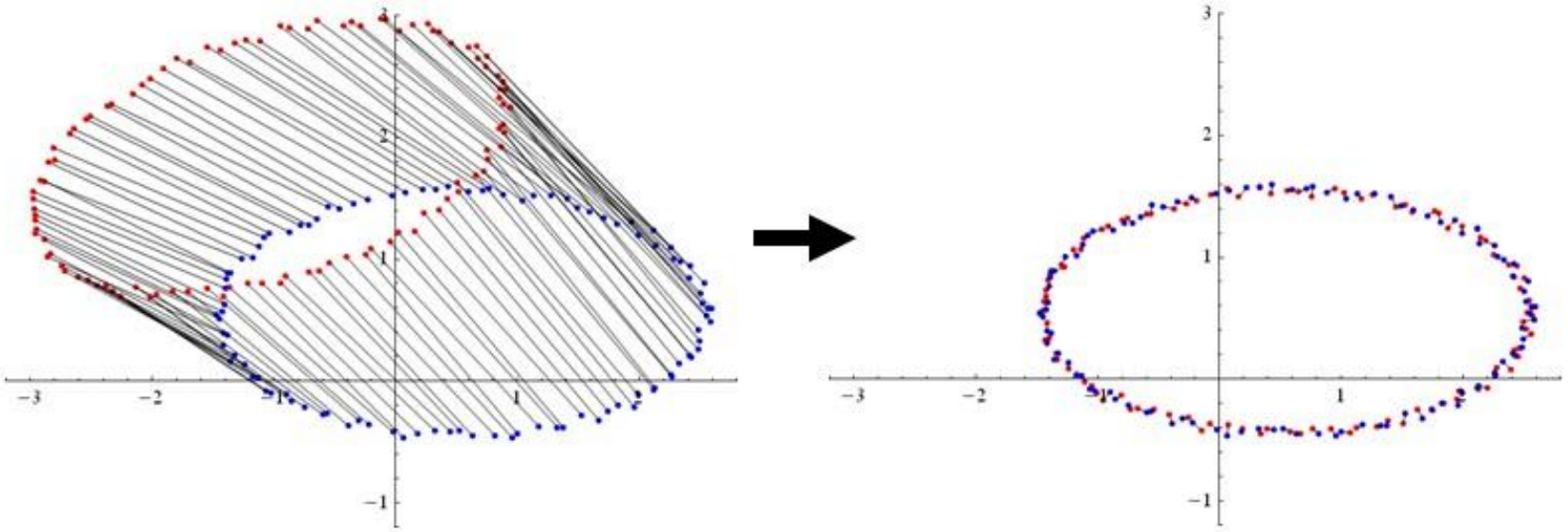
Method 1: PCA

- Limitations
  - Centroid and axes are affected by noise



Method 2: SVD

- Optimal alignment between corresponding points
  - Assuming that for each source point, we know where the corresponding target point is.



Method 2: SVD

- Formulating the problem
  - Source points $p_1, \ldots, p_n$ with centroid location c_S
  - Target points $q_1, \ldots, q_n$ with centroid location c_T
    - q_i is the corresponding point of p_i
  - After centroid alignment and rotation by some R, a transformed source point is located at:

$$\mathbf{p_i}' = \mathbf{c_T} + \mathbf{R} \cdot (\mathbf{p_i} - \mathbf{c_S})$$

  - We wish to find the R that minimizes sum of pair-wise distances:

$$\mathbf{E} = \sum_{i=1}^{n} \|\mathbf{q_i} - \mathbf{p_i}'\|^2$$

Method 2: SVD

- An equivalent formulation
  - Let P be a matrix whose *i*-th column is vector $p_i - c_S$
  - Let Q be a matrix whose *i*-th column is vector $q_i - c_T$
  - Consider the cross-covariance matrix:

$$\mathbf{M} = \mathbf{P} \cdot \mathbf{Q}^T$$

  - Find the orthonormal matrix R that maximizes the trace:

$$\mathbf{Tr}[\mathbf{R} \cdot \mathbf{M}]$$

Method 2: SVD

- Solving the minimization problem
  - Singular value decomposition (SVD) of an m by m matrix M:

$$\mathbf{M} = \mathbf{U} \cdot \mathbf{W} \cdot \mathbf{V}^{\mathbf{T}}$$

    - U,V are m by m orthonormal matrices (i.e., rotations)
    - W is a diagonal m by m matrix with non-negative entries
  - Theorem: the orthonormal matrix (rotation) $\mathtt{R} = \mathtt{V} \cdot \mathtt{U}^{\mathtt{T}}$ is the one that maximizes the trace $\mathtt{Tr}[\mathtt{R} \cdot \mathtt{M}]$
  - SVD is available in *Mathematica* and many Java/C++ libraries

Method 2: SVD

- SVD-based alignment: summary
  - Forming the cross-covariance matrix

$$\mathbf{M} = \mathbf{P} \cdot \mathbf{Q}^T$$

  - Computing SVD

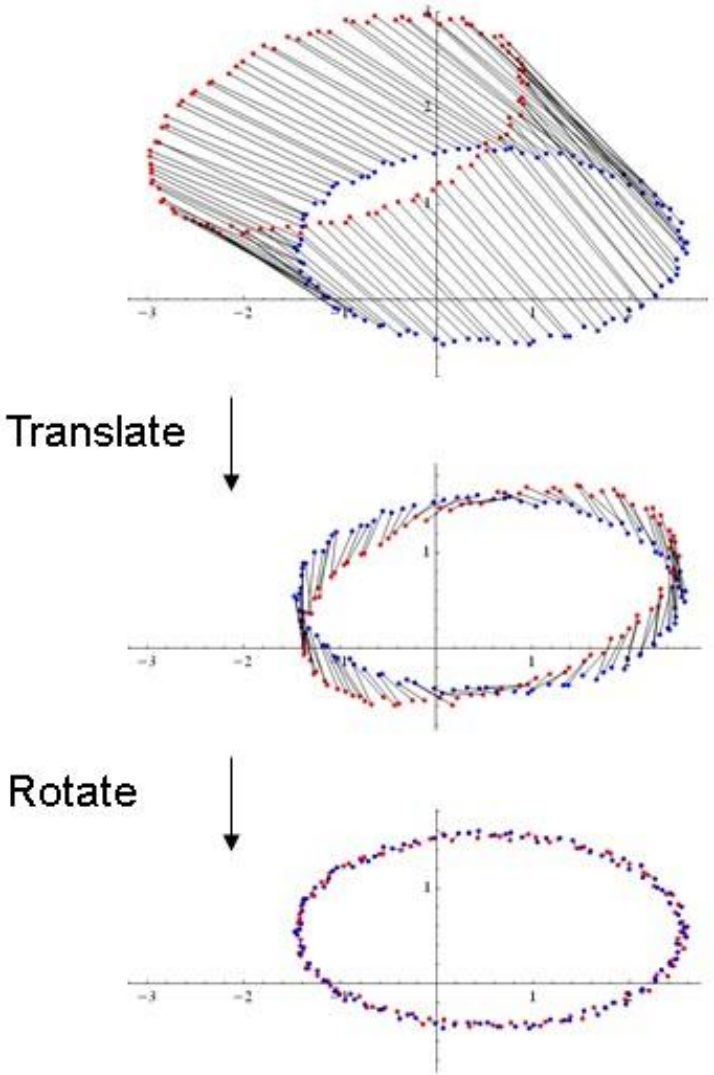
$$\mathbf{M} = \mathbf{U} \cdot \mathbf{W} \cdot \mathbf{V}^T$$

  - The optimal rotation matrix is

$$\mathbf{R} = \mathbf{V} \cdot \mathbf{U}^T$$

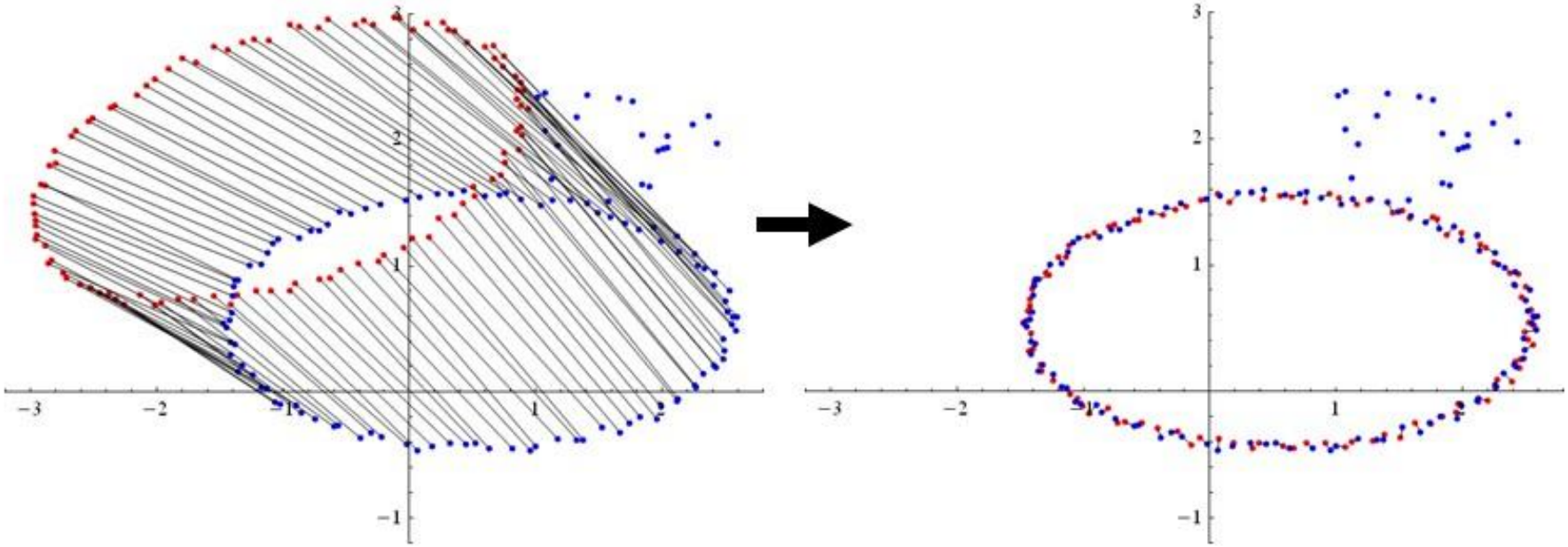
  - Translate and rotate the source:

$$\mathbf{p}_i' = \mathbf{c}_T + \mathbf{R} \cdot (\mathbf{p}_i - \mathbf{c}_S)$$



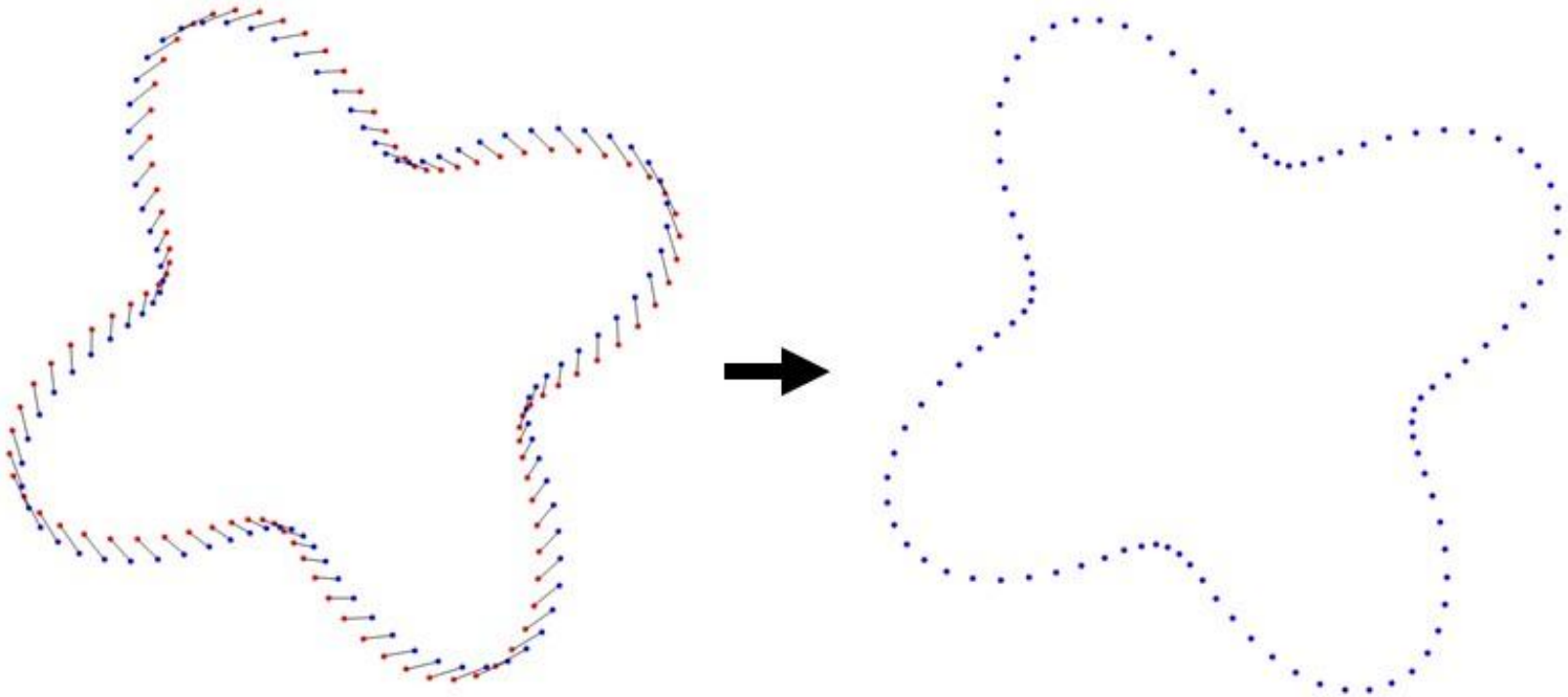
Method 2: SVD

- Advantage over PCA: more stable
  - As long as the correspondences are correct



Method 2: SVD

- Advantage over PCA: more stable
  - As long as the correspondences are correct



Method 2: SVD

- Limitation: requires accurate correspondences
  - Which are usually not available

Method 3: ICP

- Iterative closest point (ICP)
  - Use PCA alignment to obtain an initial alignment
  - Alternate between estimating correspondences (e.g., closest point) and aligning the corresponding points by SVD
    - Repeat until a termination criteria is met.

ICP Algorithm

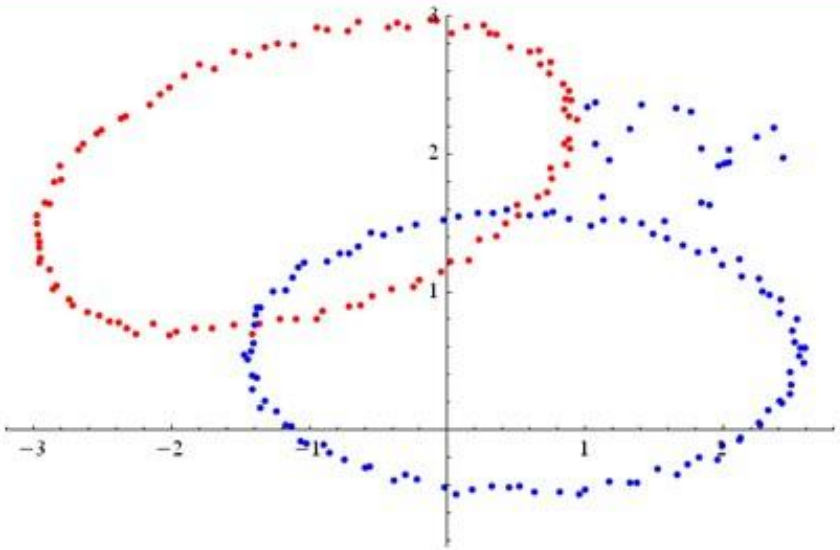
Washington University in St. Louis
School of Engineering & Applied Science

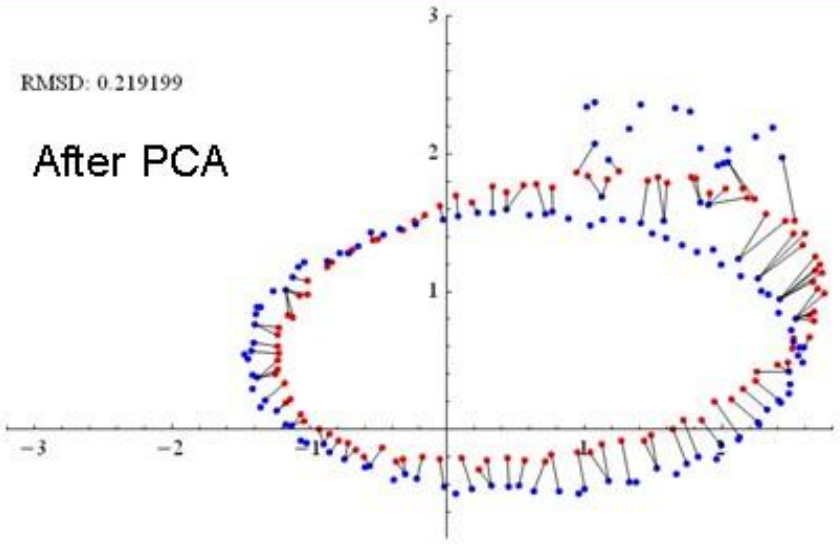
- Termination criteria
  - A user-given maximum iteration is reached
  - The improvement of fitting is small
    - Root Mean Squared Distance (RMSD):

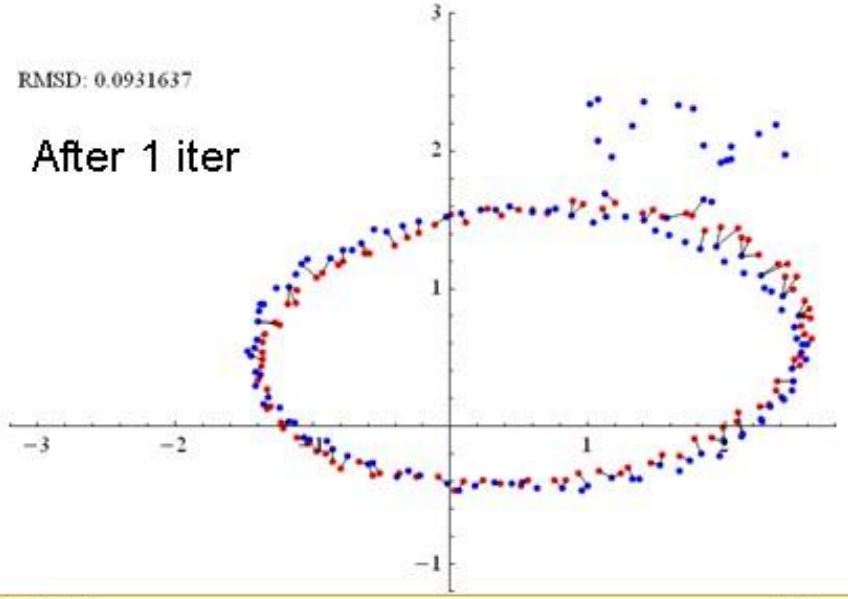
$$\sqrt{\frac{\sum_{i=1}^{n} \|q_i - p_i'\|^2}{n}}$$

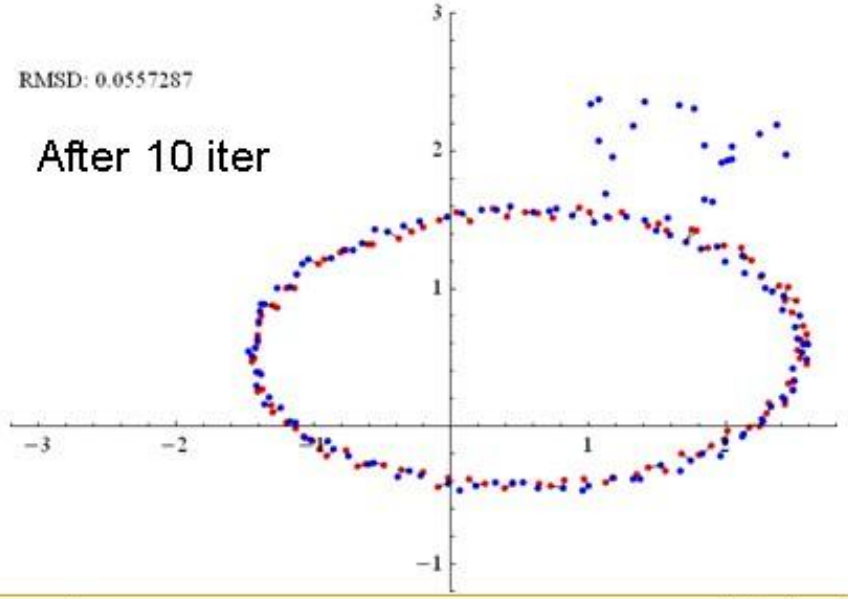
      - Captures average deviation in all corresponding pairs
    - Stops the iteration if the difference in RMSD before and after each iteration falls beneath a user-given threshold

ICP Algorithm

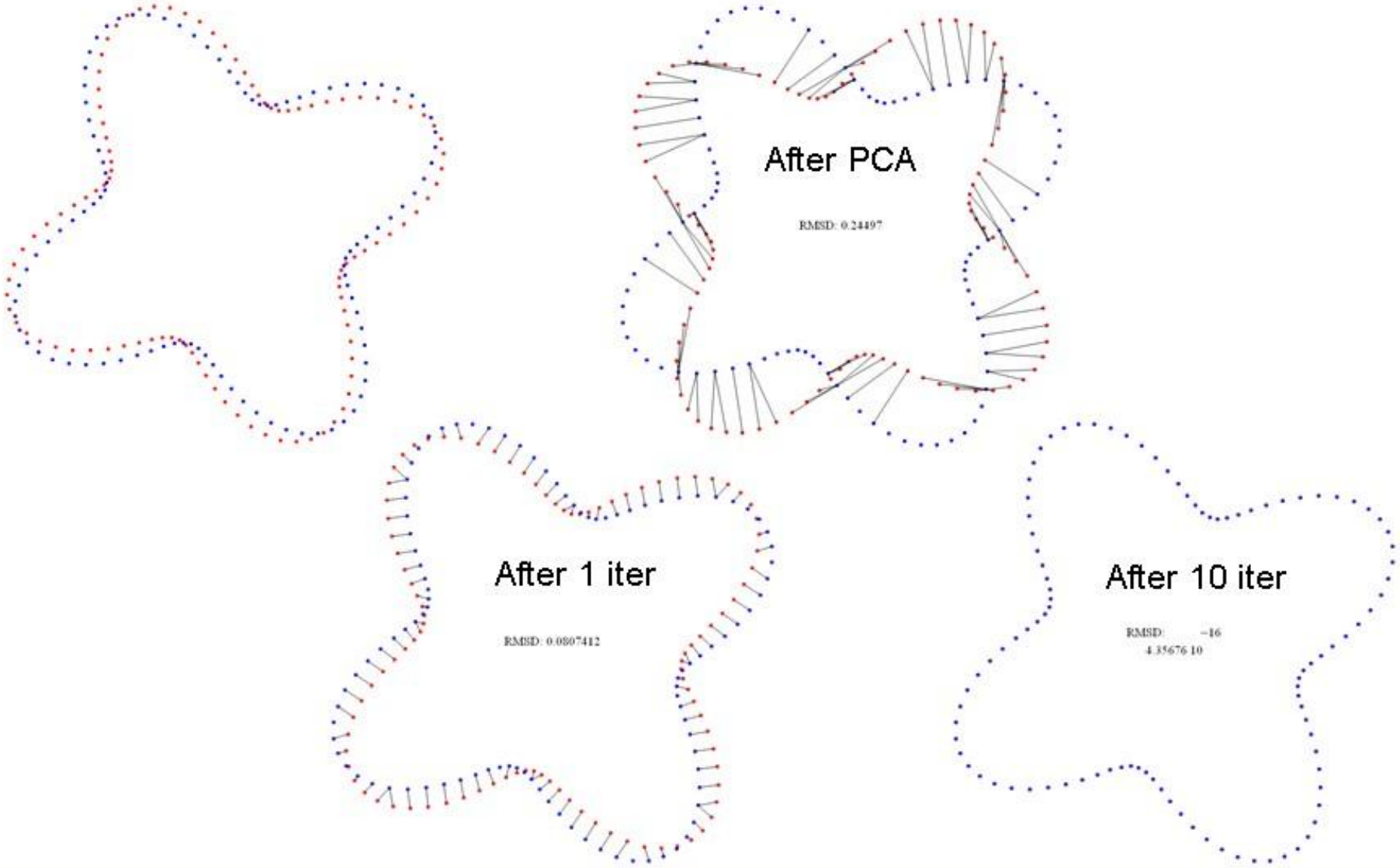








ICP Algorithm

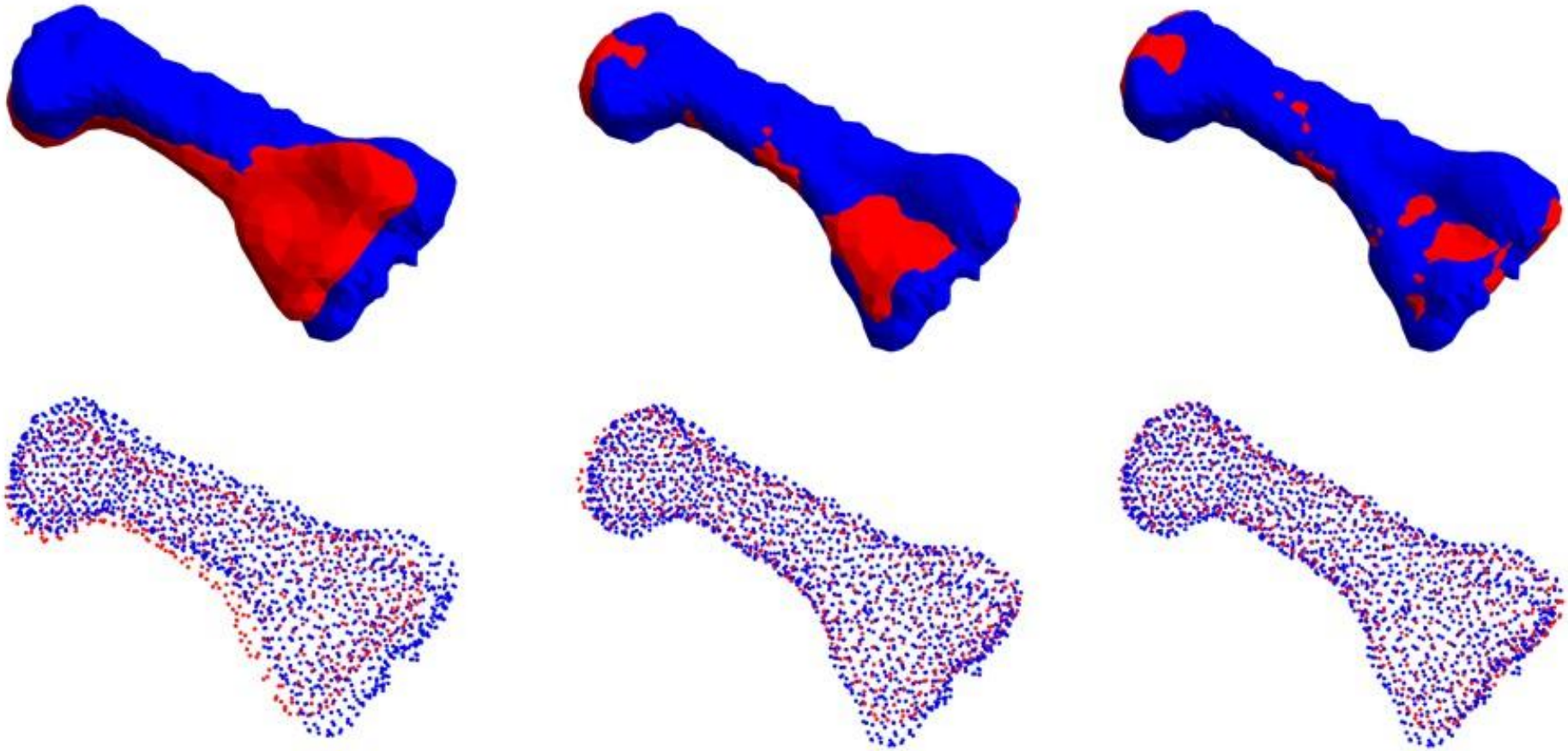


More Examples



More Examples

Washington University in St. Louis
School of Engineering & Applied Science



After PCA

After ICP