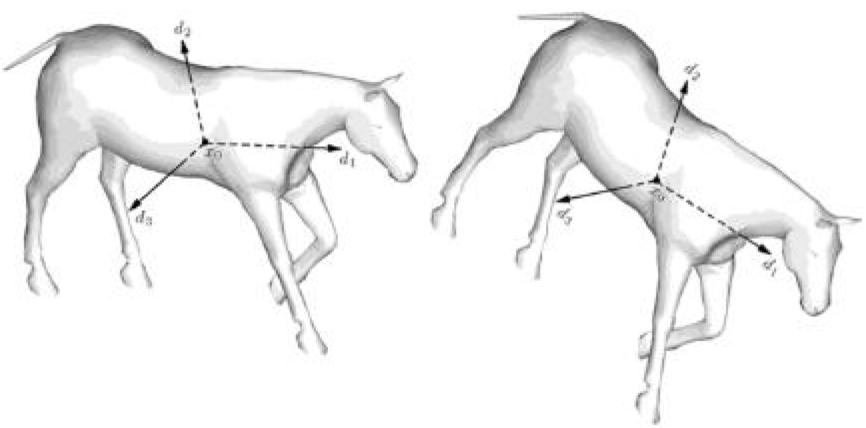
CENG 789 – Digital Geometry Processing

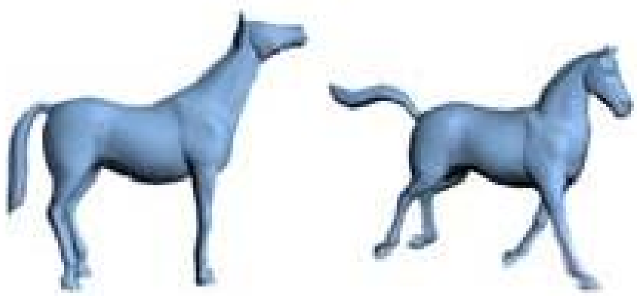
10- Shape Registration
(aka Rigid-Body Alignment)

Prof. Dr. Yusuf Sahillioğlu

Computer Eng. Dept, MIDDLE EAST TECHNICAL UNIVERSITY, Turkey

Rigid vs. Non-rigid



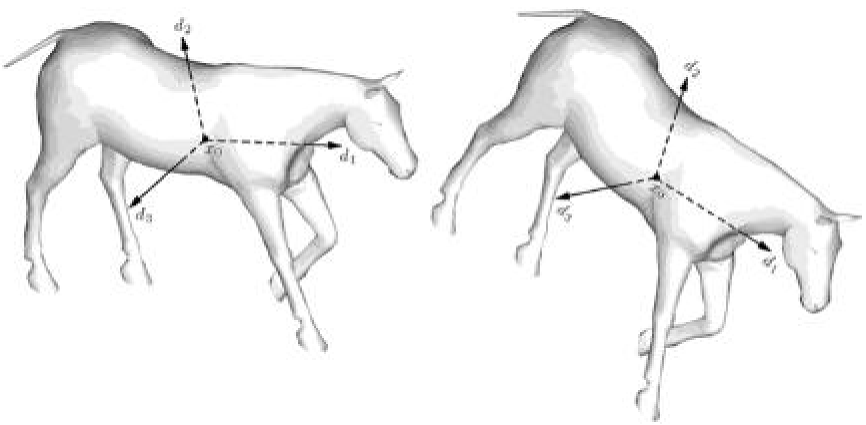


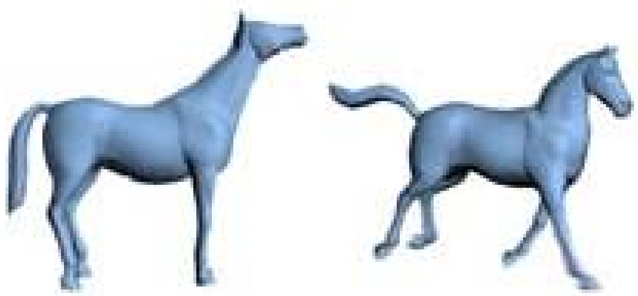
Rigid shapes (mesh, cloud, ..)
(Differ by rigid
transformations
= rotation & translation)

vs.

Non-rigid shapes (mesh, cloud, ..)
(Differ by non-rigid transformatns
= rigid + bending + stretching)
See Deformation lecture.

Rigid vs. Non-rigid



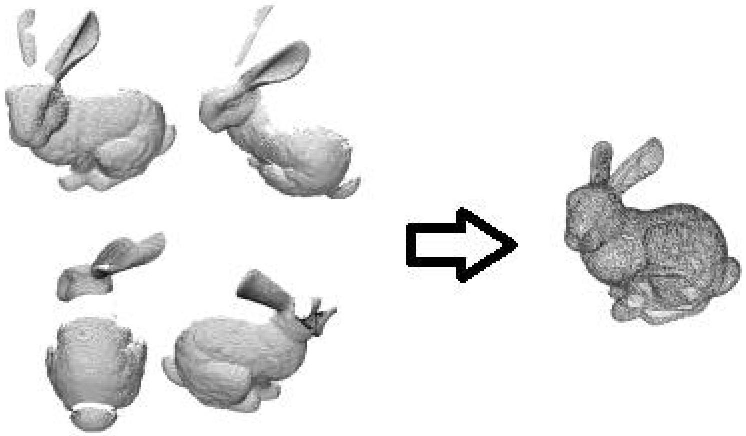


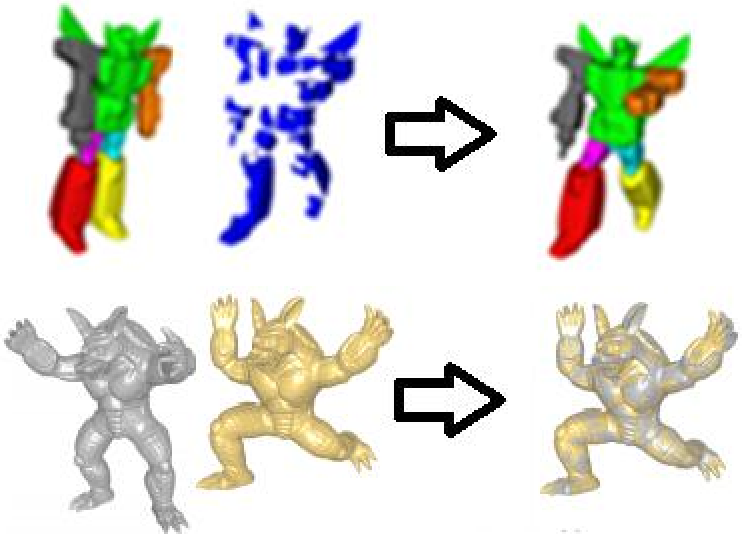
Rigid shapes (mesh, cloud, ..)
(Easier to align/register
due to low degree-
of-freedom)

vs.

Non-rigid shapes (mesh, cloud, ..)
(Harder to align/register
due to high DOF)
See Deformation lecture.

Rigid vs. Non-rigid



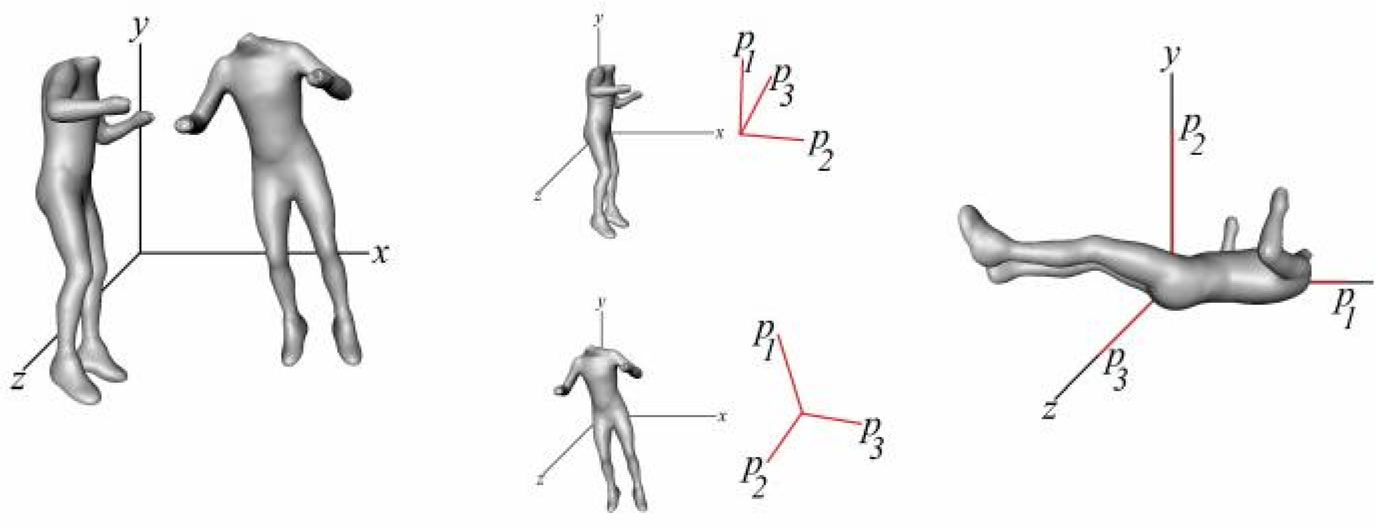


Rigid shapes (mesh, cloud, ..)
(Main app: 3D scan registration) vs.

Non-rigid shapes (mesh, cloud, ..)
(Main apps: shape completion & information transfer via shape correspondence)

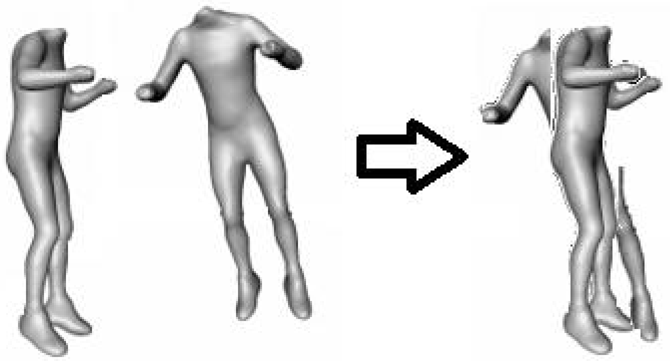
Rigid Alignment via PCA

- ✓ We will do rigid-body alignment this lecture.
- ✓ Easier scenario: Alignment of 2 *complete* shapes.



Rigid Alignment via PCA

- ✓ We will do rigid-body alignment this lecture.
- ✓ Easier scenario: Alignment of 2 *complete* shapes.
- ✓ Translation disambiguation handled by moving centers to the origin.

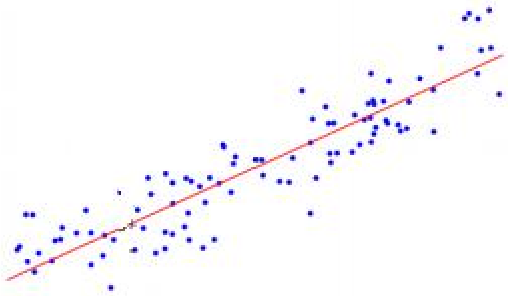


Rigid Alignment via PCA

- ✓ We will do rigid-body alignment this lecture.
- ✓ Easier scenario: Alignment of 2 *complete* shapes.
- ✓ Rotation disambiguation handled by PCA.
  - ✓ PCA on covariance matrix C that encodes an indication of whether 2 attributes, e.g., x-, y-coords, change together or in opposite directions.

$$\mathrm{cov}(X,Y) = \frac{1}{n}\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})$$

$$C = \begin{bmatrix} \sum_{i=1}^{n} x_i^2 & \sum_{i=1}^{n} x_i y_i & \sum_{i=1}^{n} x_i z_i \\ \sum_{i=1}^{n} x_i y_i & \sum_{i=1}^{n} y_i^2 & \sum_{i=1}^{n} y_i z_i \\ \sum_{i=1}^{n} x_i z_i & \sum_{i=1}^{n} y_i z_i & \sum_{i=1}^{n} z_i^2 \end{bmatrix}$$



$$\mathrm{Cov}(X,Y) = E[\underbrace{(X - E[X])(Y - E[Y])}_{1 \cdot -1}]$$

$X = 1 \quad E[X] = 0$

$Y = 3 \quad E[Y] = 4$

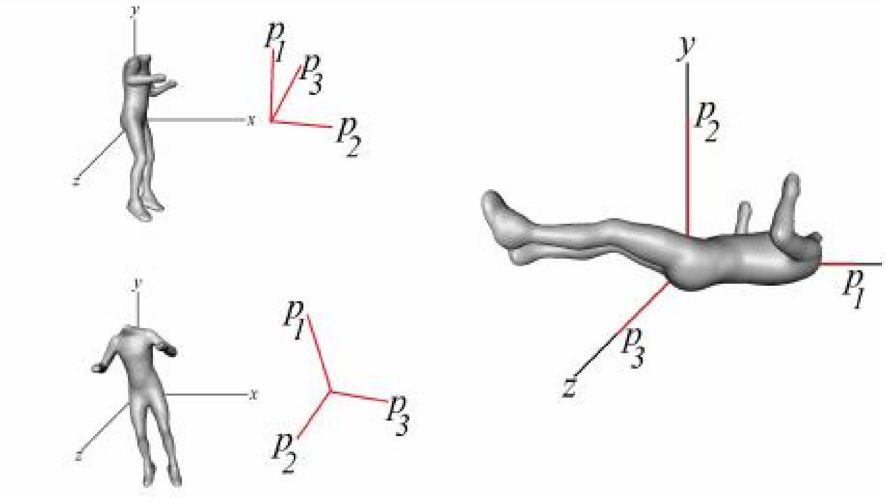
x=1 is above average (0) → increasing

y=3 is below average (4) → decreasing

➔ change in opposite directions (-ve)

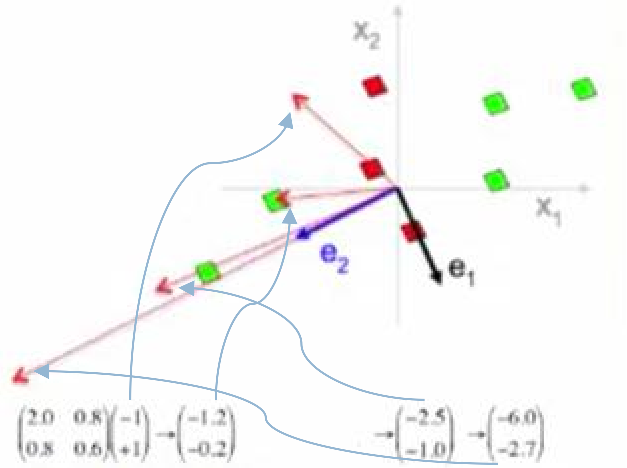
Rigid Alignment via PCA

- ✓ We will do rigid-body alignment this lecture.
- ✓ Easier scenario: Alignment of 2 *complete* shapes.
- ✓ Rotation disambiguation handled by PCA.
  - ✓ PCA on covariance matrix C that encodes an indication of whether 2 attributes, e.g., x-, y-coords, change together or in opposite directions.
  - ✓ Eigenvectors of C provide principal directions (of variations) of the shape (why? next slide), which are then aligned w/ the Cartesian coordinate axes.
  - ✓ Same alignment is applied to the 2^{nd} shape.



Eigendecomposition of Covariance Matrix

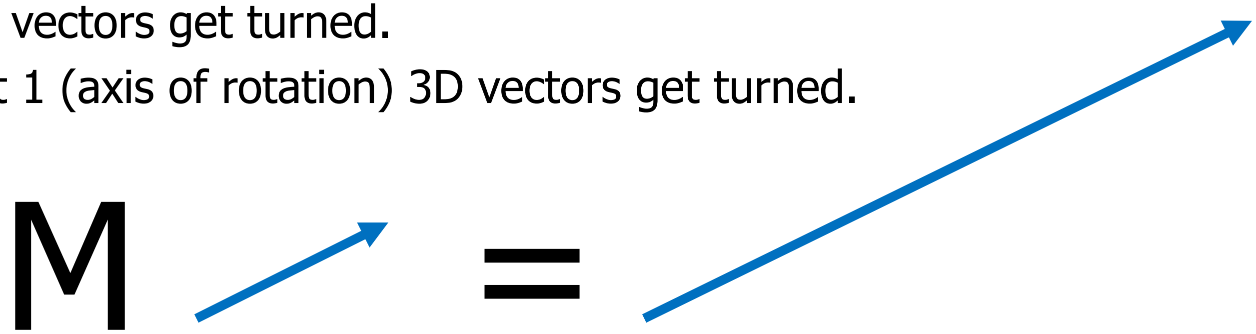
- ✓ Why do the eigenvectors of the covariance matrix C give the desired principal directions of variations, i.e., principal components?



- ✓ Take any vector, e.g., [-1 1]^T, transform it via C, get [-1.2 -0.2]^T, repeat.
- ✓ Turns out, multiplying by C spins the initial vector towards direction of the greatest variance. It is also scaled along the way but it's OK.
- ✓ So, look for vectors that do not get turned when multiplied by C, hence look for eigenvectors of C.
- ✓ Eigenvector of a transformation M is a vector that has not changed at all after transformation M except by a scalar known as the eigenvalue: $Mv = \lambda v$, where v is an eigenvector of M w/ the eigenvalue λ.

Eigenvector Pop Quiz

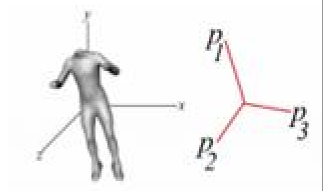
- ✓ How many eigenvectors does a 2D reflection matrix have, and geometric meaning?
  - ✓ 2: along the reflection line and perpendicular to the reflection line.
- ✓ Same for i) 3D uniform scaling mtrx? ii) 2D rotation mtr? iii) 3D rot. m?
  - ✓ Infinitely many: every vector preserves direction and changes magnitude after uniform scaling matrix is applied.
  - ✓ 0: all 2D vectors get turned.
  - ✓ 1: all but 1 (axis of rotation) 3D vectors get turned.

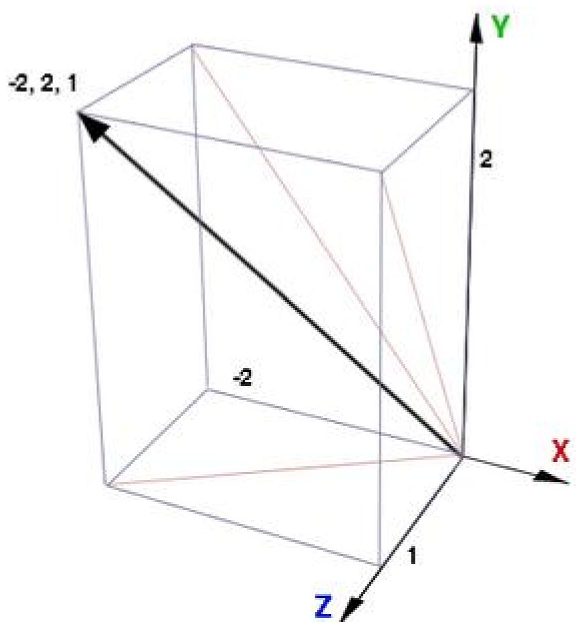


- ✓ Eigenvector of a transformation M is a vector that has not changed at all after transformation M except by a scalar known as the eigenvalue: $Mv = \lambda v$, where v is an eigenvector of M w/ the eigenvalue λ. In the example above, $\lambda = 3$.

Rigid Alignment via PCA

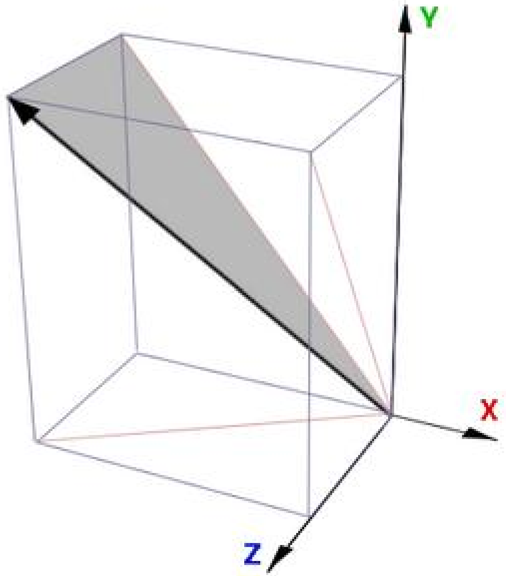
- ✓ How to align principal axes (ready now) w/ Cartesian axes?
- ✓ Easier scenario: Alignment of 2 *complete* shapes.
- ✓ Rotation disambiguation handled by PCA.
  - ✓ Align principal axes w/ the Cartesian axes; e.g., align black w/ y-axis below.



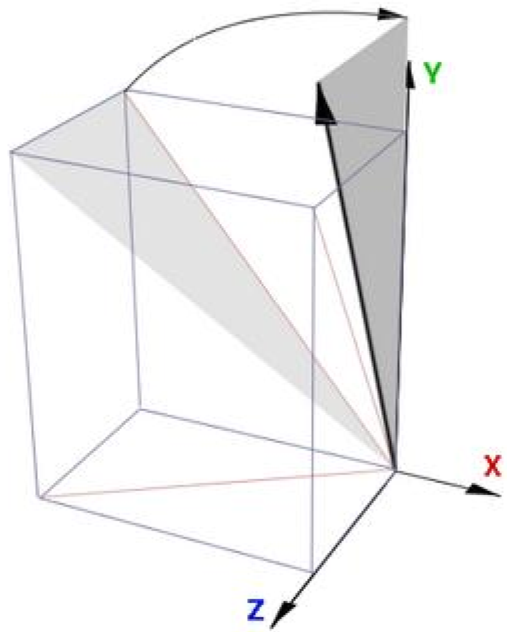


Rigid Alignment via PCA

- ✓ How to align principal axes (ready now) w/ Cartesian axes?
- ✓ Easier scenario: Alignment of 2 *complete* shapes.
- ✓ Rotation disambiguation handled by PCA.
  - ✓ Align principal axes w/ the Cartesian axes; e.g., align black w/ y-axis below.



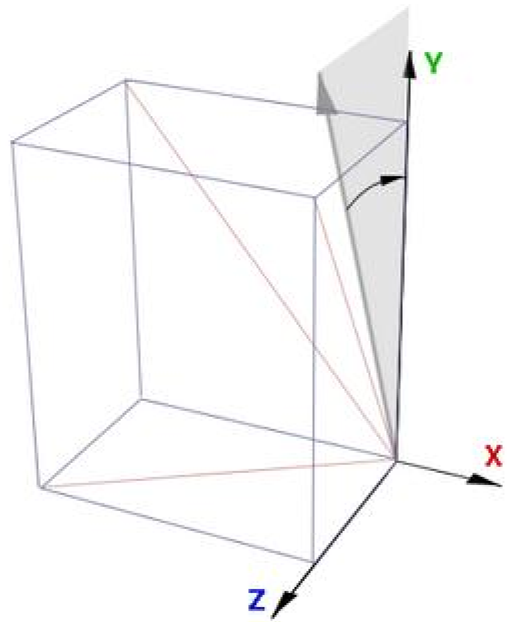
vector forms a triangle



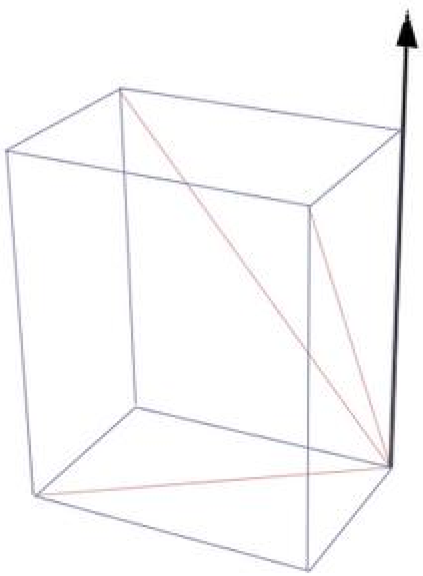
rotation around the z-axis

Rigid Alignment via PCA

- ✓ How to align principal axes (ready now) w/ Cartesian axes?
- ✓ Easier scenario: Alignment of 2 *complete* shapes.
- ✓ Rotation disambiguation handled by PCA.
  - ✓ Align principal axes w/ the Cartesian axes; e.g., align black w/ y-axis below.



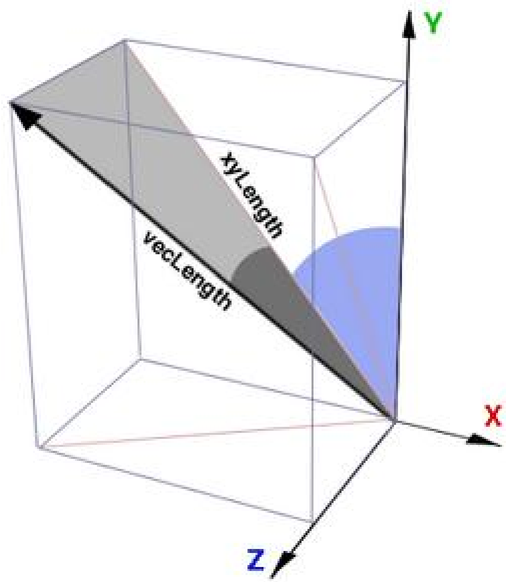
rotation around the x-axis



vector aligned with the y-axis

Rigid Alignment via PCA

- ✓ How to align principal axes (ready now) w/ Cartesian axes?
- ✓ Easier scenario: Alignment of 2 *complete* shapes.
- ✓ Rotation disambiguation handled by PCA.
  - ✓ Align principal axes w/ the Cartesian axes; e.g., align black w/ y-axis below.
  - ✓ Blue (zAngle) and dark (xAngle) angles?

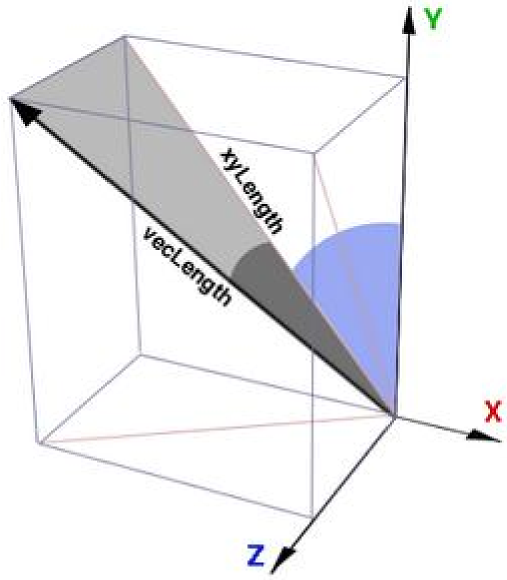



```
xyLength = sqrt(x * x + y * y);
xyLength = sqrt(-2 * -2 + 2 * 2);
xyLength = 2.83;

zAngle = acos(y / xyLength);
zAngle = acos(2.0 / 2.83);
zAngle = 0.785;
```

Rigid Alignment via PCA

- ✓ How to align principal axes (ready now) w/ Cartesian axes?
- ✓ Easier scenario: Alignment of 2 *complete* shapes.
- ✓ Rotation disambiguation handled by PCA.
  - ✓ Align principal axes w/ the Cartesian axes; e.g., align black w/ y-axis below.
  - ✓ Blue (zAngle) and dark (xAngle) angles?



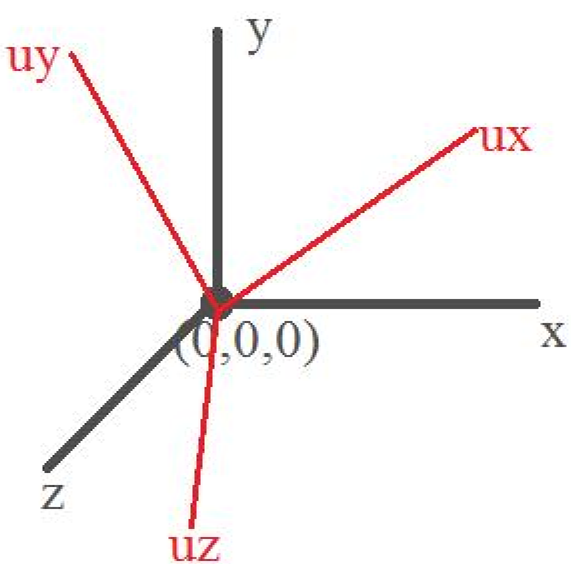

```
xAngle = acos(xyLength / vecLength);
xAngle = acos(2.83 / 3.0);
xAngle = 0.338;
```

Rigid Alignment via PCA

- ✓ How to align principal axes (ready now) w/ Cartesian axes?
- ✓ Easier scenario: Alignment of 2 *complete* shapes.
- ✓ Rotation disambiguation handled by PCA.
  - ✓ Align principal axes w/ the Cartesian coordinate axes.
  - ✓ This was cool; but not exactly solving our alignment problem because
    - ✓ It rather moves object from one position to another within a single reference frame.
    - ✓ In our case we want switching coordinates from one system (principles) to another (Cartesian); like this:
    - ✓ Tables, chairs, and other furniture is defined in a local (modeling) coordinate system.
    - ✓ They can be placed into a room, defined in another coordinate system, by transforming furniture (modeling) coordinates to room (world) coordinates.

Rigid Alignment via PCA

- ✓ How to align principal axes (ready now) w/ Cartesian axes?
- ✓ Easier scenario: Alignment of 2 *complete* shapes.
- ✓ Rotation disambiguation handled by PCA.
  - ✓ Align principal axes w/ the Cartesian coordinate axes.
  - ✓ Switching coordinates from one system (principles) to another (Cartesian).

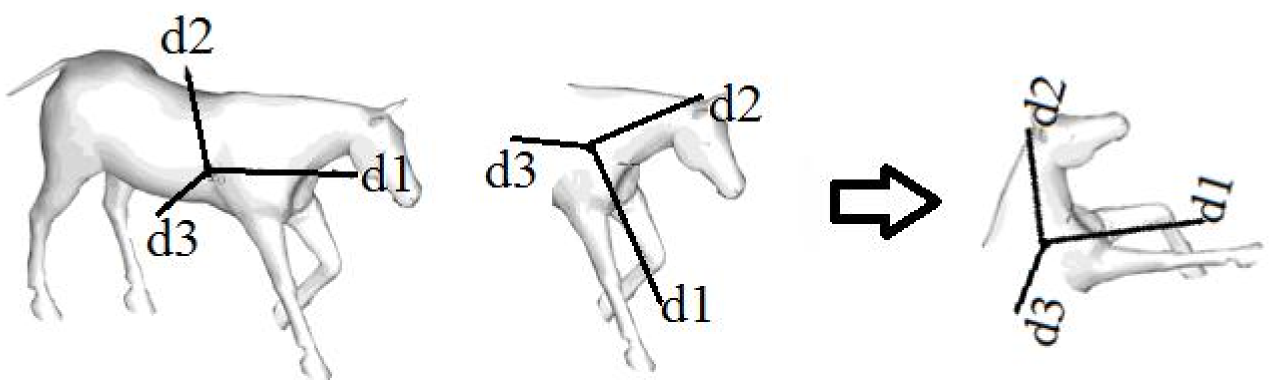


$$R = \begin{bmatrix} ux1 & ux2 & ux3 \\ uy1 & uy2 & uy3 \\ uz1 & uz2 & uz3 \end{bmatrix}$$

R rotates unit vectors ui onto xyz axes.
That is, apply R to all vertices/objs in the ui-coordinate frame: v ' = R * v

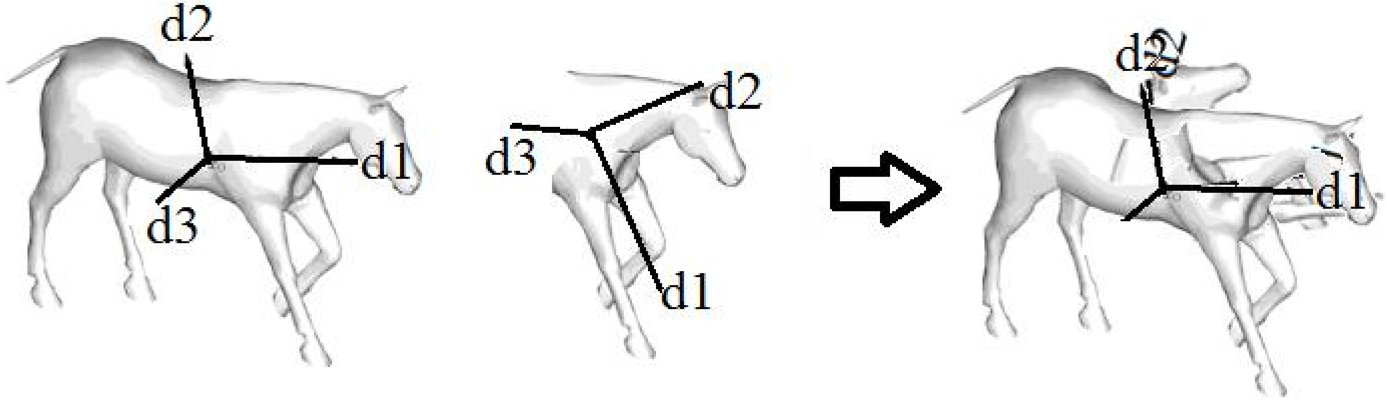
Rigid Alignment via PCA Drawback

- ✓ We have a problem w/ this PCA-based scheme.
- ✓ Harder scenario: Alignment of 2 *partial/incomplete* shapes.
- ✓ PCA-based solution is a global approach.
  - ✓ Based on variations on coordinates.
  - ✓ Two partially overlapped shapes have different variations; so, PCA fails.



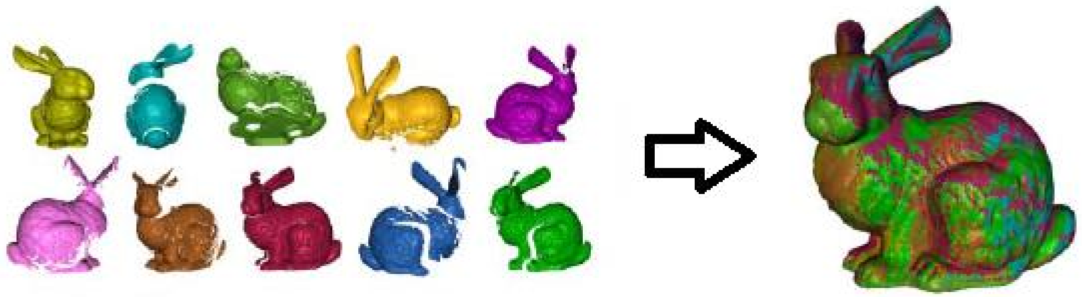
Rigid Alignment via PCA Drawback

- ✓ We have a problem w/ this PCA-based scheme.
- ✓ Harder scenario: Alignment of 2 *partial* shapes.
- ✓ PCA-based solution is a global approach.
  - ✓ Two partially overlapped shapes have different variations; so, PCA fails.
  - ✓ Depending on the amount of overlap, it may still give a good initialization though, e.g., for ICP (upcoming).



Rigid Alignment with Partial Overlap

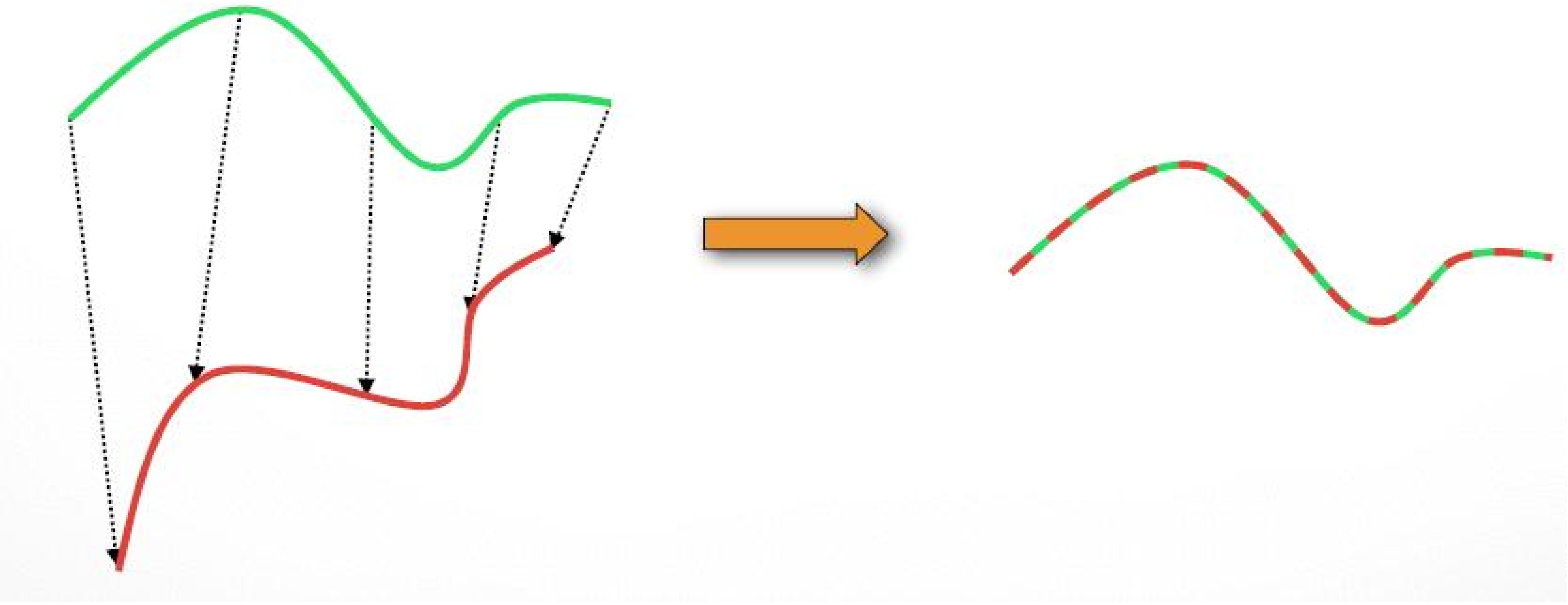
- ✓ Harder scenario: Alignment of 2 *partial* shapes.
- ✓ Partial shapes arise frequently in life: 3D scans.
- ✓ Handle them using
  - ✓ Direct rotation computation (when map b/w 2 shapes known).
  - ✓ Iterative Closest Point algorithm (when map unknown).
  - ✓ RANSAC (when map unknown).



- ✓ These methods perform pairwise alignment. Generally have >2 scans.
  - ✓ Align all other scans to an anchor (impractical-to-find) scan.
  - ✓ Greedily align each new scan to accumulated alignments (order-dependent).
  - ✓ Distributing accumulated error is an issue.

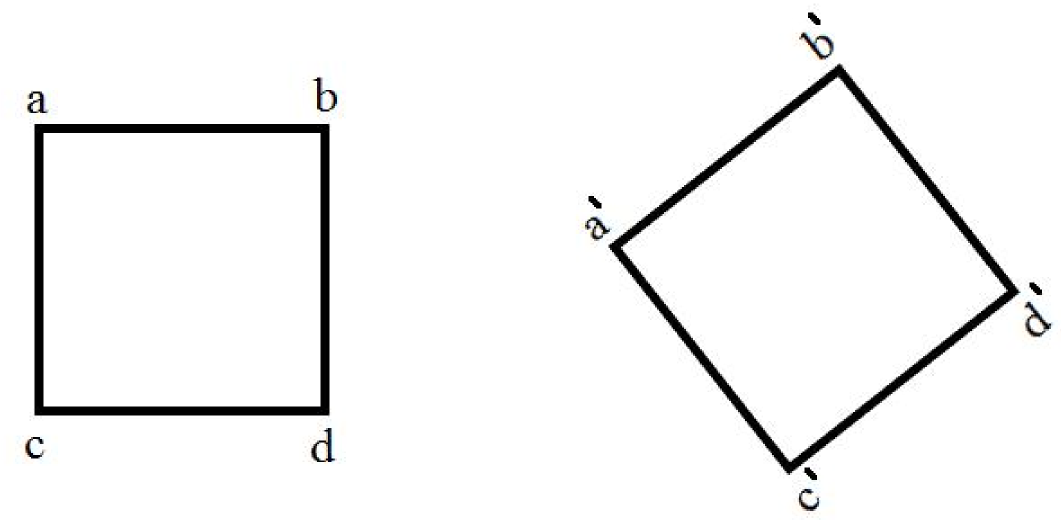
Rigid Alignment w/ the Perfect Map

✓ If map/correspondences are known, we can derive rotation and translation that aligns the 1st shape with the 2nd.



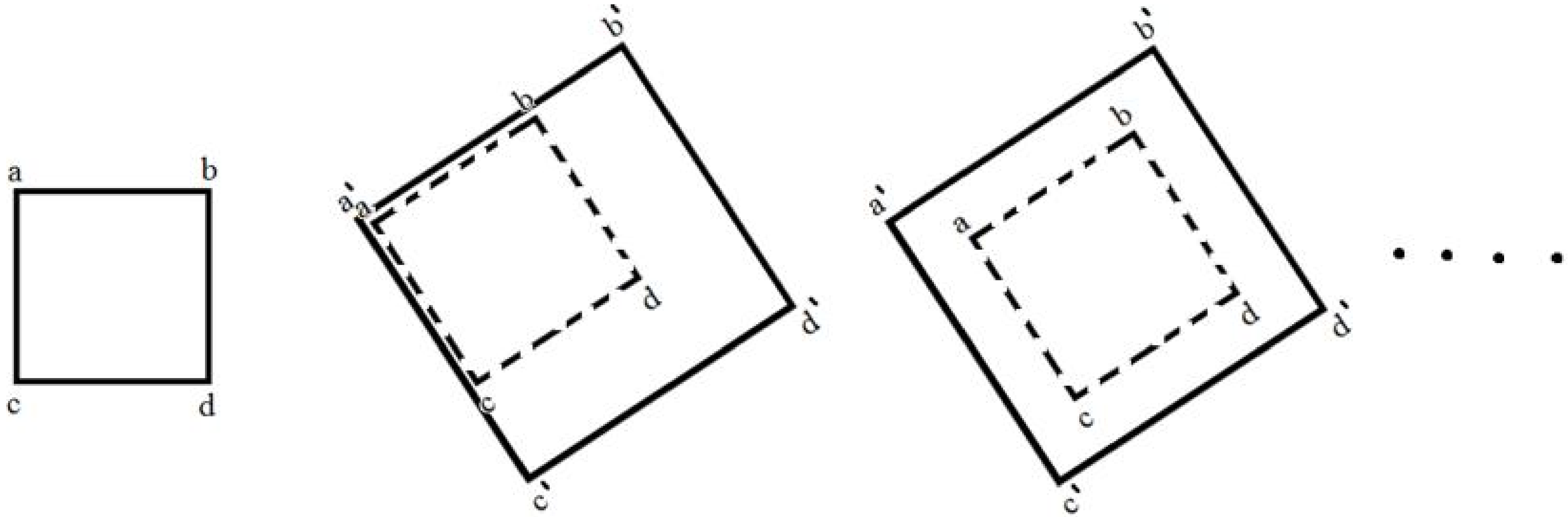
Rigid Alignment w/ the Perfect Map

- ✓ If map/correspondences are known, we can derive rotation and translation that aligns the 1st shape with the 2nd.
- ✓ If correspondences are perfect, you can find the optimal rotation w/ no error.



Rigid Alignment w/ Imperfect Map

- ✓ If map/correspondences are known, we can derive rotation and translation that aligns the 1st shape with the 2nd.
- ✓ If correspondences are not perfect, you can find the optimal rotation by minimizing an error.



Rigid Alignment w/ Imperfect Map

- ✓ If map/correspondences are known, we can derive rotation and translation that aligns the 1st shape with the 2nd.

Let $P = \{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n\}$ and $Q = \{\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_n\}$ be two sets of corresponding points in $\mathbb{R}^d$. We wish to find a rigid transformation that optimally aligns the two sets in the least squares sense, i.e., we seek a rotation R and a translation vector $\mathbf{t}$ such that

$$(R, \mathbf{t}) = \underset{R,\mathbf{t}}{\operatorname{argmin}} \sum_{i=1}^{n} w_i \left\| (R\mathbf{p}_i + \mathbf{t}) - \mathbf{q}_i \right\|^2 ,$$

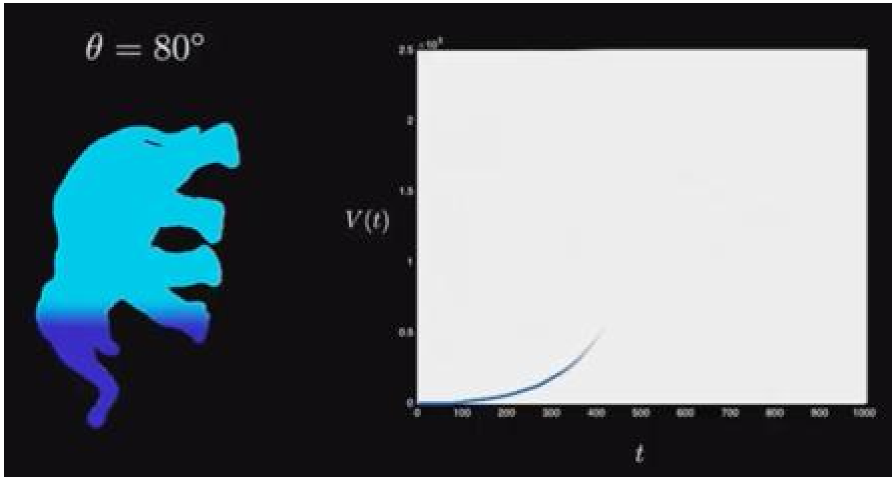
where $w_i > 0$ are weights for each point pair.

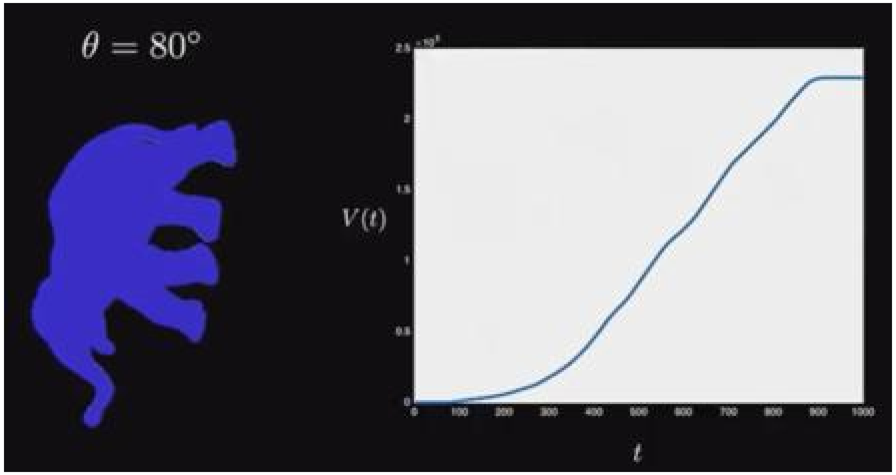
- ✓ Note that if correspondences were perfect, no least-squares minimiztn would be necessary; just solve 2 equations:
  - ✓ R*p1 + t = q1 and R*p2 + t = q2 for the variables R & t.

Derivative

- ✓ If map/correspondences are known, we can derive rotation and translation that aligns the 1st shape with the 2nd.

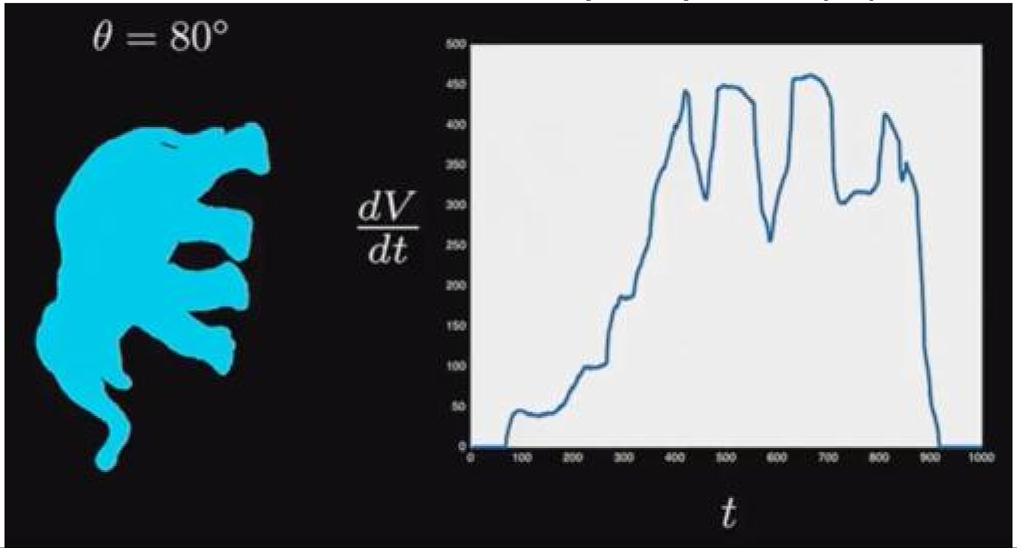
- ✓ Do energy functional minimization using matrix algebra.
  - ✓ Take the derivative of E(x) w.r.t. x and search for its roots.
  - ✓ Derivative – What was that? Here is the geometric intuition.
  - ✓ Volume of the water as we dip the elephant to the tank: monotonically incrsng.





Derivative

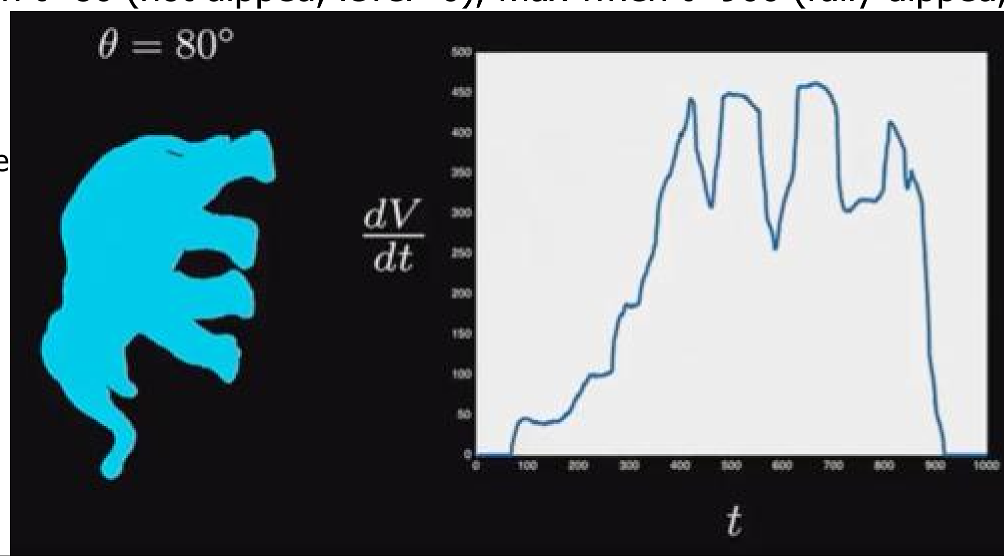
- ✓ If map/correspondences are known, we can derive rotation and translation that aligns the 1st shape with the 2nd.
- ✓ Do energy functional minimization using matrix algebra.
  - ✓ Take the derivative of E(x) w.r.t. x and search for its roots.
  - ✓ Derivative – What was that? Here is the geometric intuition.
  - ✓ Derivative is the Rate of Change in water level; how fast the water increasing.
    - ✓ It never decreases in this scenario so derivative (dV/dt) is always positive.



Derivative

- ✓ If map/correspondences are known, we can derive rotation and translation that aligns the 1st shape with the 2nd.

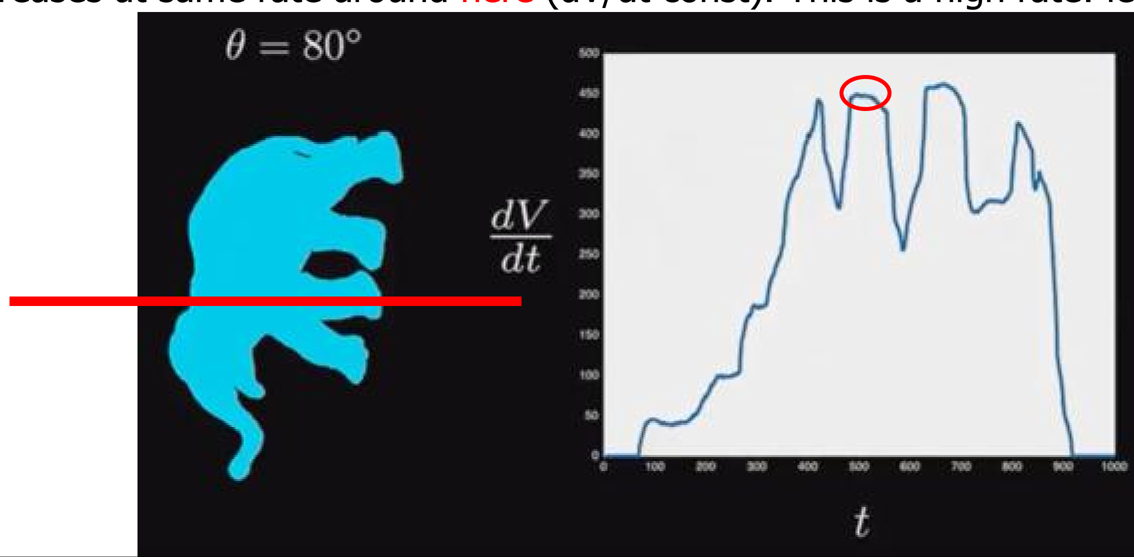
- ✓ Do energy functional minimization using matrix algebra.
  - ✓ Take the derivative of E(x) w.r.t. x and search for its roots.
  - ✓ Derivative – What was that? Here is the geometric intuition.
  - ✓ Derivative is the Rate of Change in water level; how fast the water increasing.
    - ✓ V is min when t<80 (not dipped, level=0), max when t>900 (fully dipped, level=2.3×10^5).
      - ✓ Special places where derivative is zero.



Derivative

- ✓ If map/correspondences are known, we can derive rotation and translation that aligns the 1st shape with the 2nd.

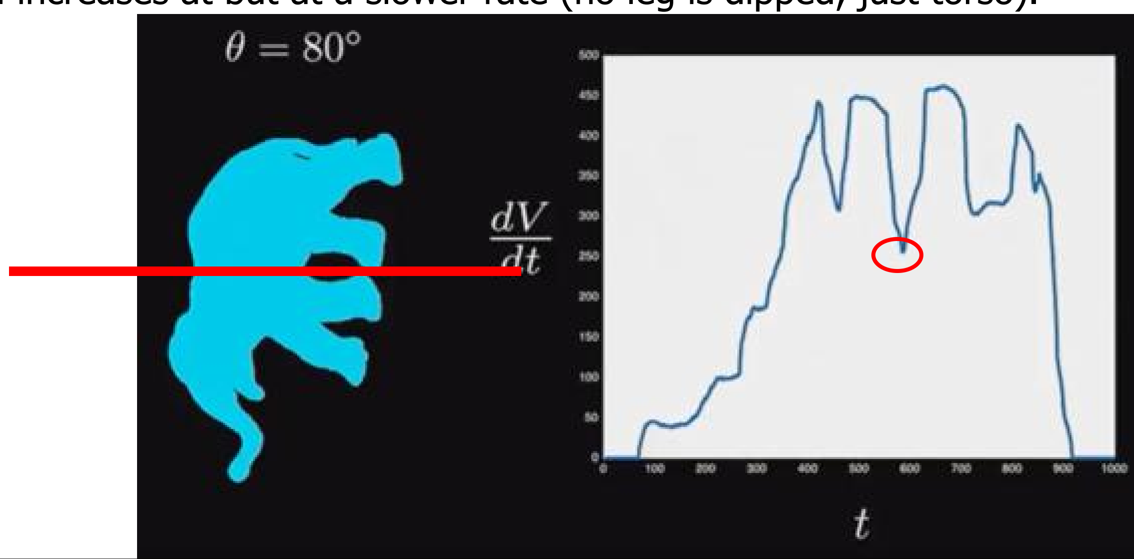
- ✓ Do energy functional minimization using matrix algebra.
  - ✓ Take the derivative of E(x) w.r.t. x and search for its roots.
  - ✓ Derivative – What was that? Here is the geometric intuition.
  - ✓ Derivative is the Rate of Change in water level; how fast the water increasing.
    - ✓ V increases at same rate around here (dV/dt const). This is a high rate: leg + torso dipped.



Derivative

- ✓ If map/correspondences are known, we can derive rotation and translation that aligns the 1st shape with the 2nd.

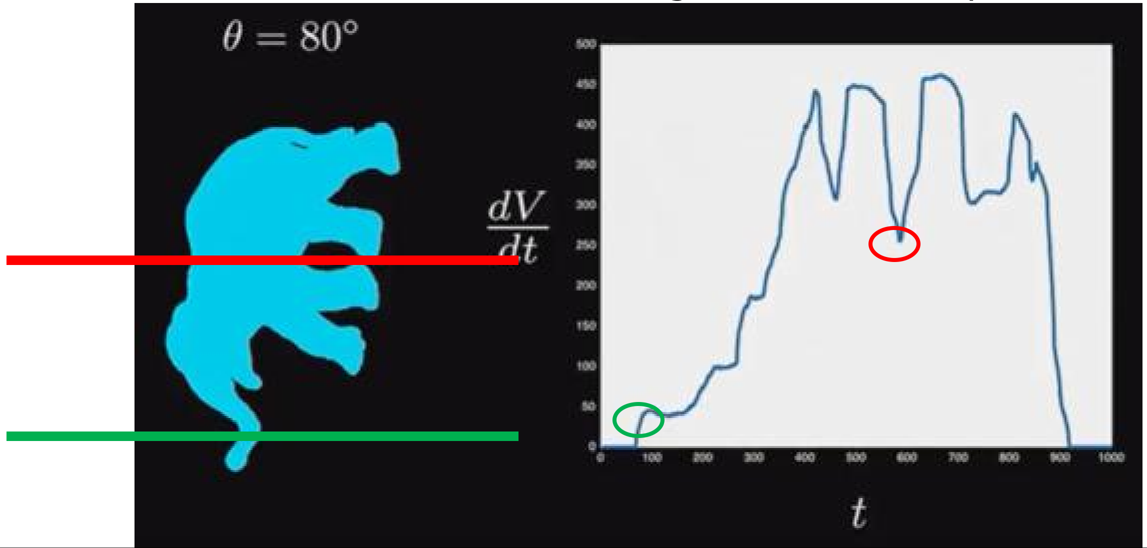
- ✓ Do energy functional minimization using matrix algebra.
  - ✓ Take the derivative of E(x) w.r.t. x and search for its roots.
  - ✓ Derivative – What was that? Here is the geometric intuition.
  - ✓ Derivative is the Rate of Change in water level; how fast the water increasing.
    - ✓ V still increases at but at a slower rate (no leg is dipped, just torso).



Derivative

- ✓ If map/correspondences are known, we can derive rotation and translation that aligns the 1st shape with the 2nd.

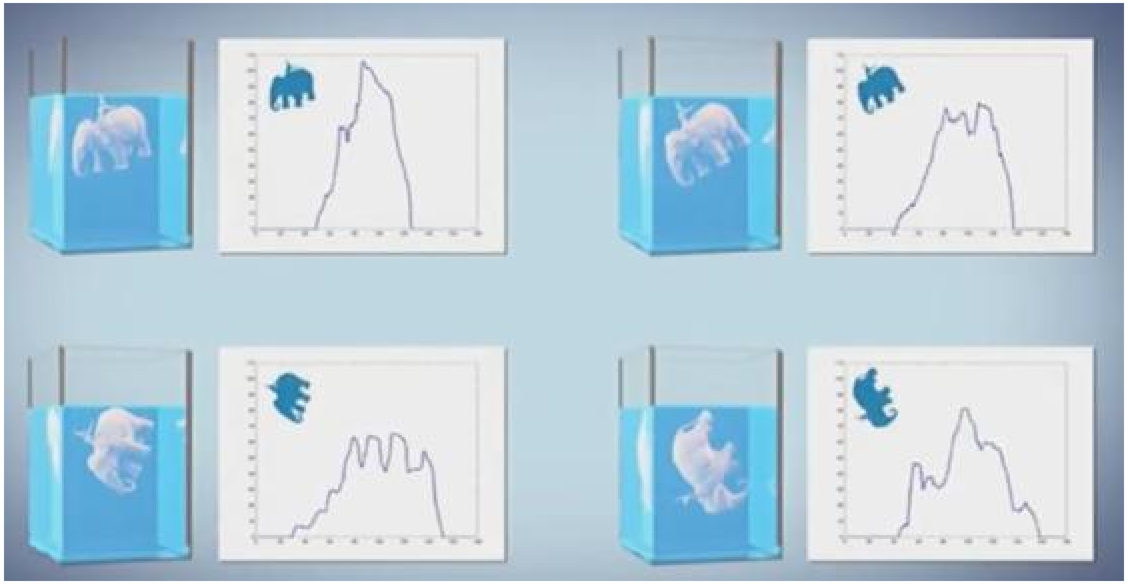
- ✓ Do energy functional minimization using matrix algebra.
  - ✓ Take the derivative of E(x) w.r.t. x and search for its roots.
  - ✓ Derivative – What was that? Here is the geometric intuition.
  - ✓ Derivative is the Rate of Change in water level; how fast the water increasing.
    - ✓ V still increases at but at a slower rate. Still higher than trunk dip 'cos this is for thick torso.



Derivative

- ✓ If map/correspondences are known, we can derive rotation and translation that aligns the 1st shape with the 2nd.

- ✓ Do energy functional minimization using matrix algebra.
  - ✓ Take the derivative of E(x) w.r.t. x and search for its roots.
  - ✓ Derivative – What was that? Here is the geometric intuition.
  - ✓ One can use dV/dt to recognize and reconstruct shape but these are off-topic.



Derivative

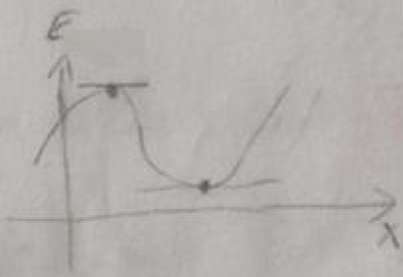
- ✓ If map/correspondences are known, we can derive rotation and translation that aligns the 1st shape with the 2nd.

- ✓ Do energy functional minimization using matrix algebra.
  - ✓ Take the derivative of E(x) w.r.t. x and search for its roots.

Find x that minimizes $E(x)$

Solve $\frac{\partial E}{\partial x} = 0$ Why?, E_{match} is a function on x

This function is min/max at locations where slope (derivative: rate of change of the value of E_{corr} as we move slightly along x) is 0.

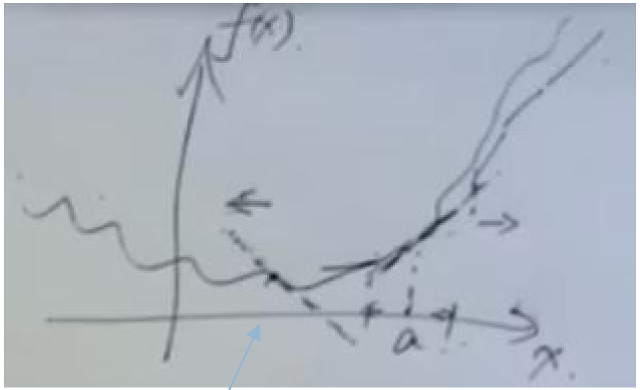


remark When you don't have a graph to look at, the best way to find where slope is 0 is to set the derivative equal to 0.

Derivative

- ✓ If map/correspondences are known, we can derive rotation and translation that aligns the 1st shape with the 2nd.

- ✓ Do energy functional minimization using matrix algebra.
  - ✓ For non-zero derivatives we have ascent directions.



  - ✓ Sign of the derivative at *a* is positive (positive slope), meaning that ascend direction is → (go right to make E(x) increase).
  - ✓ Similarly, derivative here is negative, so ascend direction is ←.
  - ✓ Note that, on local maxima/minima, there is no ascent direction (slope 0).
  - ✓ Magnitude of the derivative states how eagerly E(x) wants to go up.

Rigid Alignment Transformations

✓ Translation.

Computing the translation: t

Assume R is fixed and $E(t) = \sum_{i=1}^{n} w_i \|(Rp_i + t) - q_i\|^2$

$\begin{bmatrix}2\\3\\4\end{bmatrix}, \begin{bmatrix}10\\20\\30\end{bmatrix}$ $\|\cdot\|^2 = \sqrt{(2-10)^2 + (3-20)^2 + (4-30)^2}^2$

$= [(2-10)\ (3-20)\ (4-30)] \begin{bmatrix}2-10\\3-20\\4-30\end{bmatrix}$

$\frac{\partial E}{\partial t} = 0 \Rightarrow E = \sum_{i=1}^{n} w_i \left((Rp_i + t) - q_i\right)^T \left((Rp_i + t) - q_i\right)$ // $\|a-b\|^2 = (a-b)^T(a-b)$

same 'cos

$= \sum_{i=1}^{n} w_i \left((Rp_i + t)^T (Rp_i + t) - (Rp_i + t)^T q_i - q_i^T (Rp_i + t) + q_i^T q_i\right)$ // $a^T b = b^T a$ as $[2\ 3\ 4]\begin{bmatrix}10\\20\\30\end{bmatrix} = [10\ 20\ 30]\begin{bmatrix}2\\3\\4\end{bmatrix}$

$= \sum_{i=1}^{n} w_i \left((Rp_i + t)^T (Rp_i + t) - 2(Rp_i + t)^T q_i + q_i^T q_i\right)$

terms having t $\rightarrow (Rp_i)^T t + t^T (Rp_i) = 2t^T(Rp_i) \rightarrow t^T t = \|t\|^2 \Rightarrow$

$\frac{\partial \|a\|^2}{\partial a} = 2a$ 'cos $\frac{\partial (a_0^2 + a_1^2 + a_2^2)}{\partial a} = \begin{pmatrix}\partial/\partial a_0\\ \partial/\partial a_1\\ \partial/\partial a_2\end{pmatrix} = \begin{pmatrix}2a_0\\ 2a_1\\ 2a_2\end{pmatrix} = 2\cdot a$

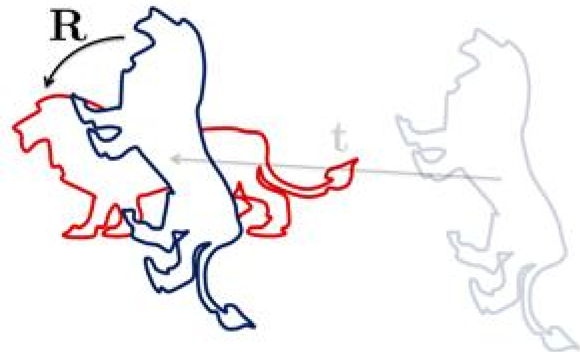
$\frac{\partial E}{\partial t} = \sum_{i=1}^{n} w_i (2\cdot Rp_i + 2t - 2q_i)$

$0 = \frac{\partial E}{\partial t} = 2R \sum_{i=1}^{n} w_i \cdot p_i + 2t \sum_{i=1}^{n} w_i - 2 \sum_{i=1}^{n} w_i q_i$ // Denote $\bar{p} = \frac{\sum_{i=1}^{n} w_i p_i}{\sum_{i=1}^{n} w_i}$, $\bar{q} = \frac{\sum_{i=1}^{n} w_i q_i}{\sum_{i=1}^{n} w_i} \Rightarrow$ weighted centroid of P and Q

$\Rightarrow 2t \sum_{i=1}^{n} w_i = 2 \sum_{i=1}^{n} w_i q_i - 2R \sum_{i=1}^{n} w_i p_i$ // Divide by $2 \cdot \sum_{i=1}^{n} w_i$

$t = \bar{q} - R\cdot\bar{p}$ ✓✓ (vector from Rotated center of P to center of Q)

Rigid Alignment



✓ Rotation.

Computing the rotation: R

Plug the optimal t in $E \Rightarrow \sum_{i=1}^{n} w_i \|(Rp_i + t) - q_i\|^2 = \sum_{i=1}^{n} w_i \|Rp_i + \bar{q} - R\bar{p} - q_i\|^2 = \sum_{i=1}^{n} w_i \|\underbrace{R(p_i - \bar{p})}_{x_i} - \underbrace{(q_i - \bar{q})}_{y_i}\|^2$

Look for $R = \underset{R}{\arg\min} \sum_{i=1}^{n} w_i \underbrace{\|Rx_i - y_i\|^2}_{\text{rearrange this}}$

$(Rx_i)^T = x_i^T R^T$ $\quad R^{-1} = R^T$

$= (Rx_i - y_i)^T (Rx_i - y_i) = (x_i^T R^T - y_i^T)(Rx_i - y_i) = x_i^T \overbrace{R^T R}^{I} x_i - \underbrace{x_i^T R^T y_i}_{\text{scalar}} - y_i^T (Rx_i) + y_i^T y_i$

scalar $\Rightarrow a^T = a \Rightarrow (x_i^T R^T y_i)^T = y_i^T R x_i$

$= x_i^T x_i - 2 y_i^T R x_i + y_i^T y_i$

$R = \underset{R}{\arg\min} \sum_{i=1}^{n} w_i (\quad " \quad) = \underset{R}{\arg\min} \left(\sum_{i=1}^{n} w_i x_i^T x_i - 2 \sum_{i=1}^{n} w_i y_i^T R x_i + \sum_{i=1}^{n} y_i^T y_i \right)$

not depend on R ⟸ so excluding them not affect the minimizer

$= \underset{R}{\arg\min} -2 \sum_{i=1}^{n} w_i y_i^T R x_i = \underset{R}{\arg\max} \boxed{\sum_{i=1}^{n} w_i y_i^T R x_i} \to = tr(WY^T RX)$

$\hookrightarrow \sum_{i=1}^{n} a_{ii}$

$w_1 \ \ w_n \quad y_i^T \quad R \quad x_i$

$tr(AB) = tr(BA)$

Look for R that maximizes $tr(WY^T RX) = tr(RXWY^T)$

Denote $d \times d$ covariance matrix $S = XWY^T = U \Sigma V^T$ // SVD of S

Look for R that maximizes $tr(RU\Sigma V^T) = tr(\Sigma \underbrace{V^T R U}_{M}) = tr(\Sigma M)$

M is orthogonal $\Rightarrow$ columns of M are orthonormal $\Rightarrow$ all entries ≤ 1

('cos V, R, U are.)

→ To maximize $tr(\Sigma M)$, M has to be I (Σ entries nonnegative)

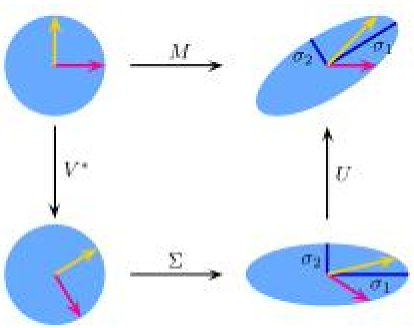
$M = I = V^T R U$

$VI = \underbrace{VV^T}_{I \text{ 'cos } V \text{ is a rotation}} RU \Rightarrow V = RU \Rightarrow R = V \cdot U^T$ ✓✓

Once R is found, $t = \bar{q} - R \cdot \bar{p}$ ✓✓

First rotate, then translate.

Recall SVD:



$M = U \cdot \Sigma \cdot V^*$

$Q^T Q = I$

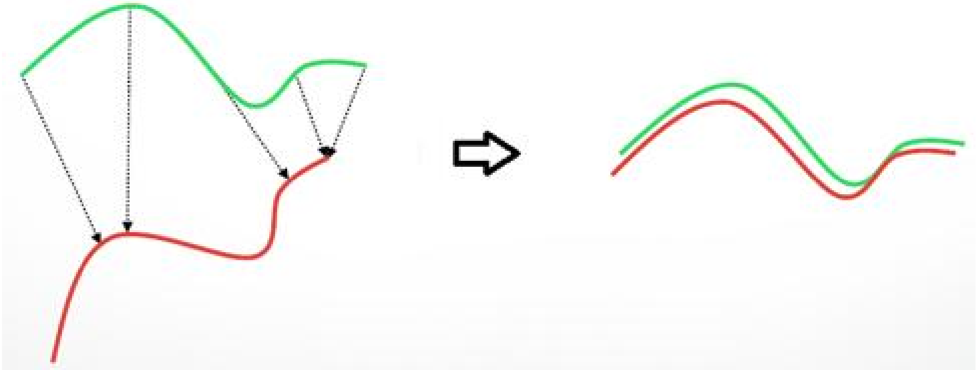
$R^T R = I,$

then

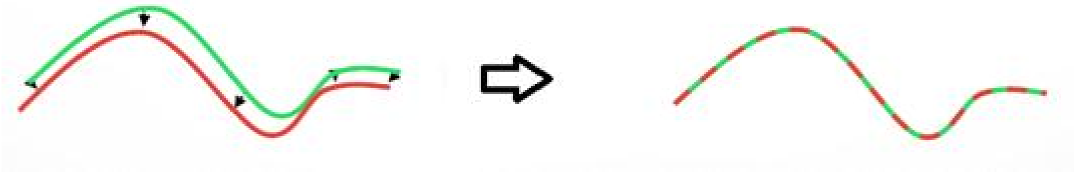
$(QR)^T (QR) = R^T (Q^T Q) R = R^T R = I$

Rigid Alignment via ICP

- ✓ Iterative Closest Point algorithm (when map unknown or imperfect).
  - ✓ Assume closest points correspond.
  - ✓ Compute the rotation and translation based on this correspondence/map.



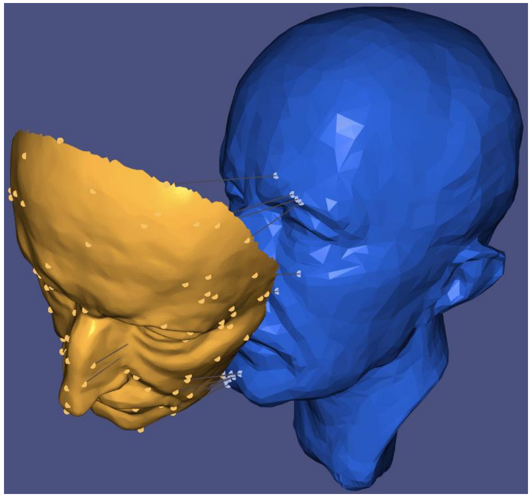
  - ✓ Update coordinates and re-compute closest-point correspondences.
  - ✓ Re-compute rotation and translation based on this map. Repeat.



Rigid Alignment via ICP

- ✓ Iterative Closest Point algorithm (when map b/w 2 shapes unknown).
- ✓ Original ICP paper *(Besl & McKay, A Method for Registration of 3-D Shapes, PAMI, 1992)* uses closed-form solution to compute the rotations (unlike our SVD-based solution before – same effect).
- ✓ Converges if starting point is close enough.

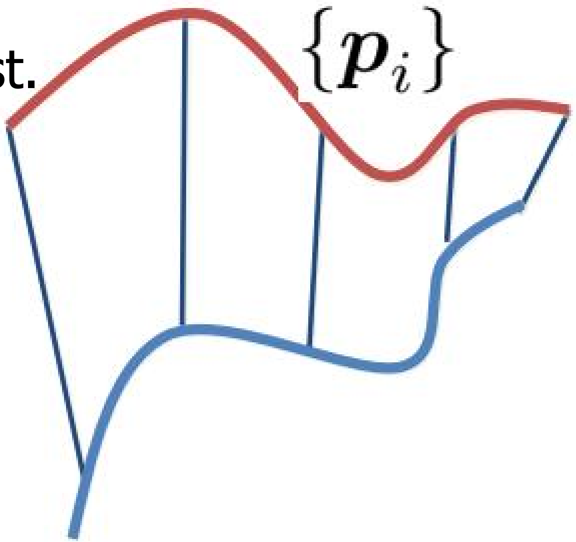
GIF from A. Jacobson's Geo. Processing course:



- ✓ For efficiency, you can do sampling on point sets. Also, use k-d trees.
- ✓ For accuracy, you can replace your matching criteria (point to plane).

Rigid Alignment via ICP

- ✓ ICP variants
  - ✓ Different error metrics exist.

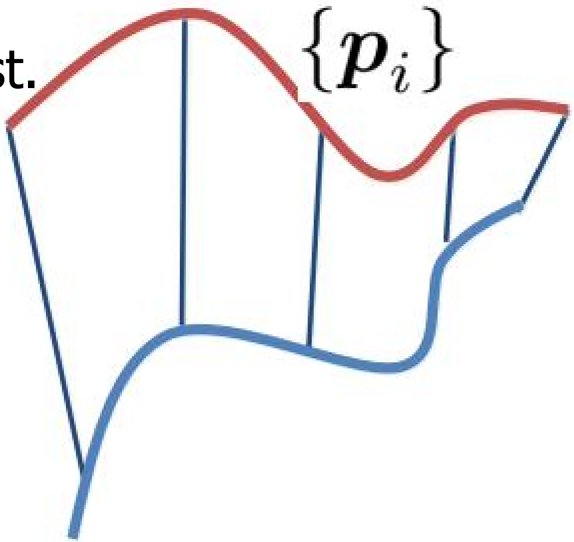


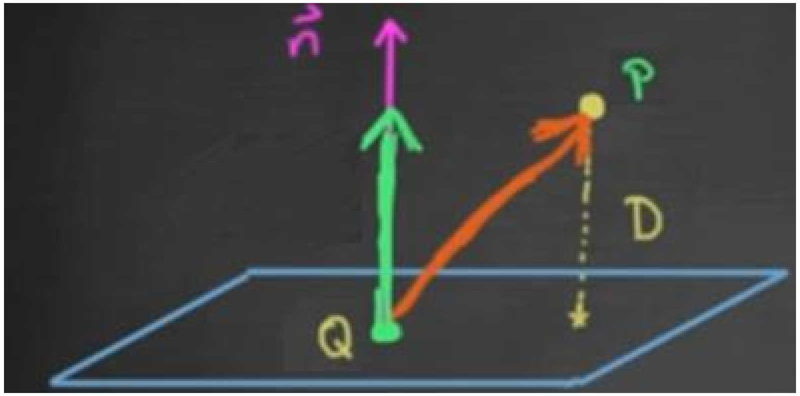
Point-to-point: $$\underset{R,\boldsymbol{t}}{\text{minimize}} \sum_{i=1}^{n} \|R(\boldsymbol{p}_i) + \boldsymbol{t} - \boldsymbol{q}_i\|^2$$

Point-to-plane: $$\underset{R,\boldsymbol{t}}{\text{minimize}} \sum_{i=1}^{n} \left((R\boldsymbol{p}_i + \boldsymbol{t} - \boldsymbol{q}_i)^T \boldsymbol{n}_i\right)^2$$

Rigid Alignment via ICP

- ✓ ICP variants
  - ✓ Different error metrics exist.





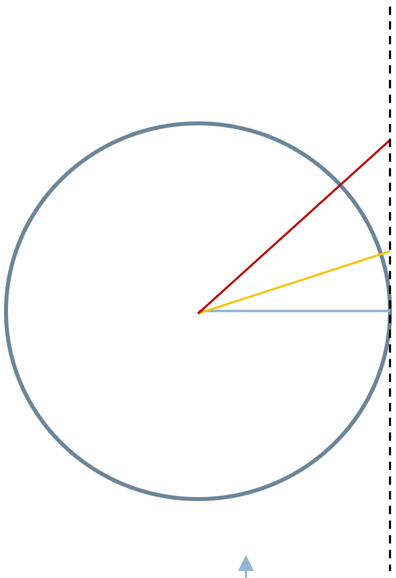
Point-to-plane: $\underset{R,\boldsymbol{t}}{\text{minimize}} \sum_{i=1}^{n} \left((R\boldsymbol{p}_i + \boldsymbol{t} - \boldsymbol{q}_i)^T \boldsymbol{n}_i \right)^2$

- ✓ Point-to-plane derivation: https://www.cs.princeton.edu/~smr/papers/icpstability.pdf
- ✓ Point-to-point vs. point-to-plane: https://youtu.be/LcghboLgTiA

Rigid Alignment via ICP

- ✓ ICP variants
  - ✓ Different error metrics exist.

$$
R_{x,\alpha} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{pmatrix} \approx \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & -\alpha \\ 0 & \alpha & 1 \end{pmatrix}.
$$

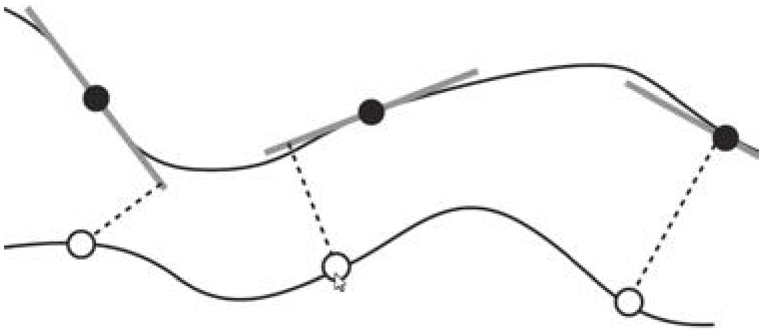


Approx err increases as we go up (blue to red).

- ✓ Point-to-plane derivation: https://www.cs.princeton.edu/~smr/papers/icpstability.pdf
- ✓ Derivation based on <u>linearization of rotations</u> (OK for small angles).

Rigid Alignment via ICP

- ✓ ICP variants
  - ✓ Different error metrics exist.
  - ✓ Point-to-plane metric shown to do a better job empirically (you lose a little bit on the theoretical convergence of the algorithm).



  - ✓ Best practice is to use a combination: $E = .1E_{p2point} + .9E_{p2plane}$.

Rigid Alignment via ICP

- ✓ Iterative Closest Point algorithm (when map b/w 2 shapes unknown).
- ✓ Resulting ICP session; pretty robust:

Rigid Alignment via ICP

- ✓ Iterative Closest Point algorithm (when map b/w 2 shapes unknown).
- ✓ Basic ICP algo; again:

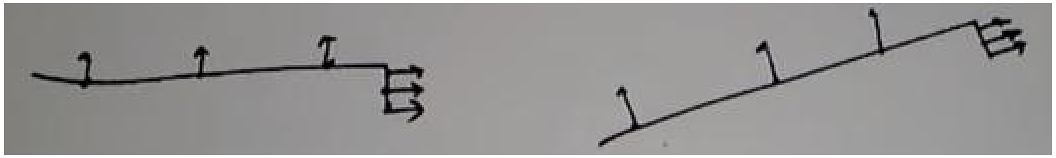
- Select e.g. 1000 random points
- Match each to closest point on other scan, using data structure such as *k*-d tree
- Reject pairs with distance > *k* times median
- Construct error function:

$$E = \sum \left| Rp_i + t - q_i \right|^2$$

- Minimize (closed form solution in [Horn 87])

Rigid Alignment via ICP

- ✓ Iterative Closest Point algorithm (when map b/w 2 shapes unknown).
- ✓ Basic ICP algo; again:
- ✓ Random sampling can be improved by getting diversity of normals.
  - ✓ Have equal number of points for each possible normal.



  - ✓ If careless, stuff on large surface will overwhelm the small surface.
    - ✓ Here shape is sliding along the large surface (horizontal) 'cos registration error there is 0. Small surface (vertical) does not warn us due to lack of samples.

Rigid Alignment via ICP

- ✓ Iterative Closest Point algorithm (when map b/w 2 shapes unknown).
- ✓ Basic ICP algo; again:
- ✓ For the rotation R in $E = \sum |Rp_i + t - q_i|^2$,

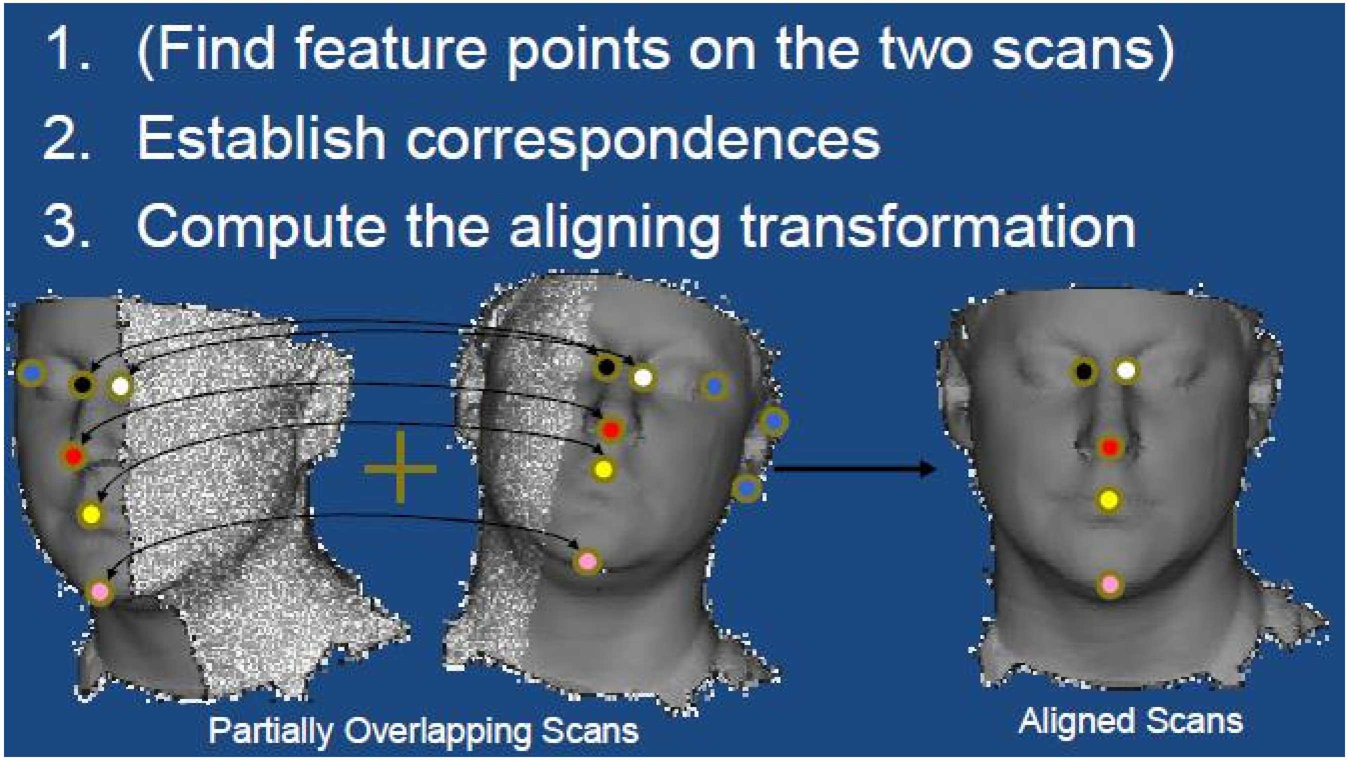
  one can use the closed formula (original Besl'92 paper):

$$R = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1 q_2 - q_0 q_3) & 2(q_1 q_3 + q_0 q_2) \\ 2(q_1 q_2 + q_0 q_3) & q_0^2 + q_2^2 - q_1^2 - q_3^2 & 2(q_2 q_3 - q_0 q_1) \\ 2(q_1 q_3 - q_0 q_2) & 2(q_2 q_3 + q_0 q_1) & q_0^2 + q_3^2 - q_1^2 - q_2^2 \end{bmatrix}$$

  or the SVD-based energy minimization described in Slide 35.

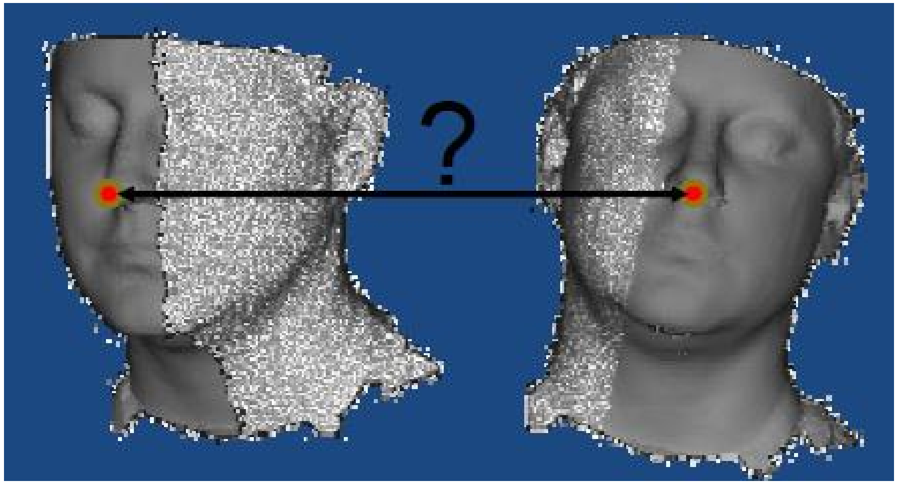
Rigid Alignment via ICP

- ✓ Iterative Closest Point algorithm (when map b/w 2 shapes unknown).
- ✓ Better correspondence estimation via feature points.



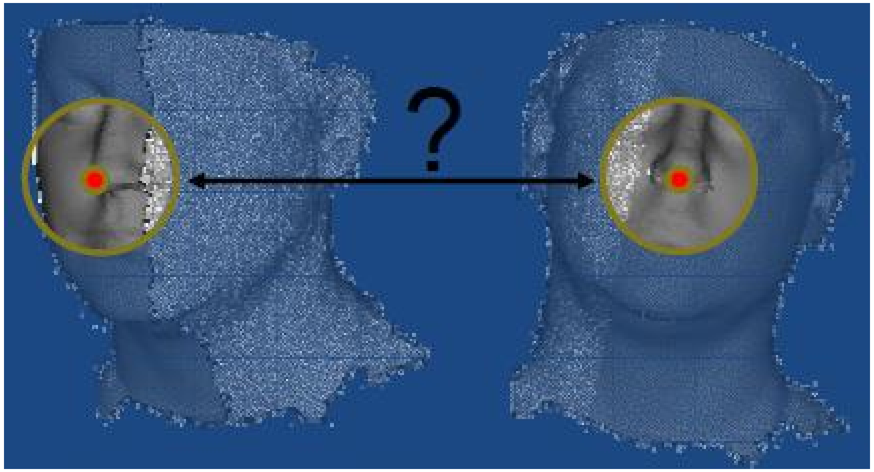
Rigid Alignment via ICP

- ✓ Iterative Closest Point algorithm (when map b/w 2 shapes unknown).
- ✓ Better correspondence estimation via feature points.
- ✓ Goal is to identify when 2 points on different scans represent the same feature.
- ✓ If this is done perfectly, you do not need to iterate (ICP). You just compute rotations and translations in one shot.



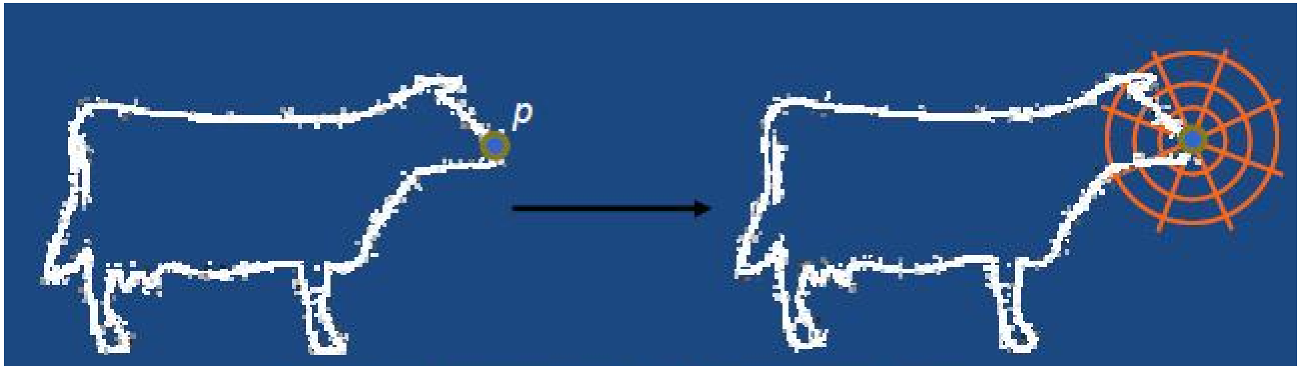
Rigid Alignment via ICP

- ✓ Iterative Closest Point algorithm (when map b/w 2 shapes unknown).
- ✓ Better correspondence estimation via feature points.
- ✓ Goal is to identify when 2 points on different scans represent the same feature.
- ✓ Hint: Are the surroundings similar?



Rigid Alignment via ICP

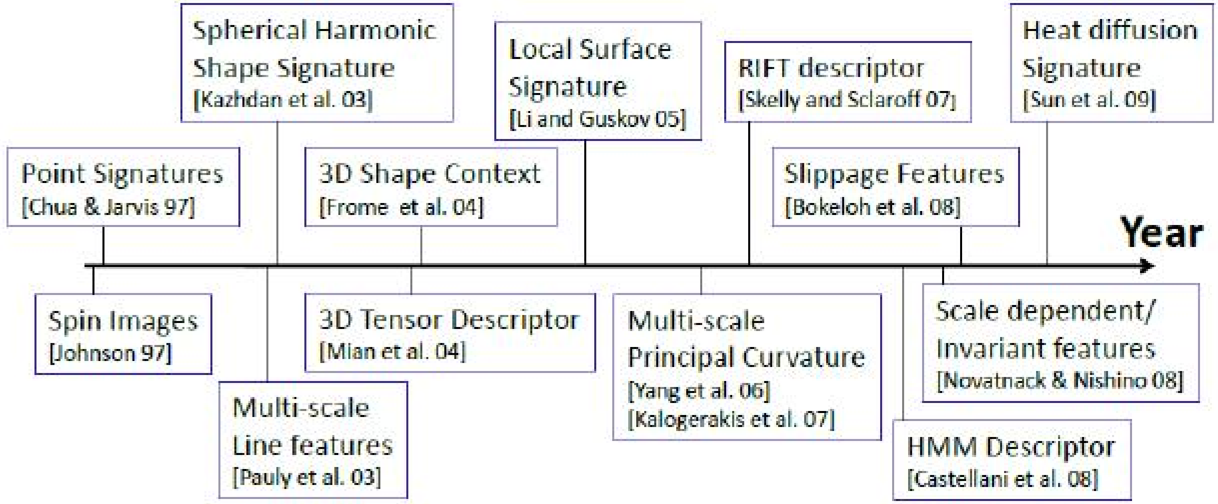
- ✓ Iterative Closest Point algorithm (when map b/w 2 shapes unknown).
- ✓ Better correspondence estimation via feature points.
- ✓ Goal is to identify when 2 points on different scans represent the same feature.
- ✓ Hint: Are the surroundings similar?
- ✓ Recall rotation-invariant shape histograms descriptor.



Shape histograms as local shape descriptors

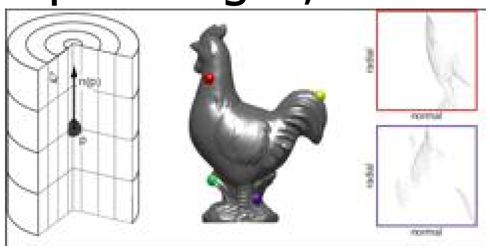
Rigid Alignment via ICP

- ✓ Iterative Closest Point algorithm (when map b/w 2 shapes unknown).
- ✓ Better correspondence estimation via feature points.
- ✓ Goal is to identify when 2 points on different scans represent the same feature.
- ✓ Hint: Are the surroundings similar?
- ✓ Lots of descriptors defined over the years.



Rigid Alignment via ICP

- ✓ Iterative Closest Point algorithm (when map b/w 2 shapes unknown).
- ✓ Better correspondence estimation via feature points.
- ✓ Goal is to identify when 2 points on different scans represent the same feature.
- ✓ Hint: Are the surroundings similar?
- ✓ Shape diameter function (mild non-rigid support), spin images, etc.



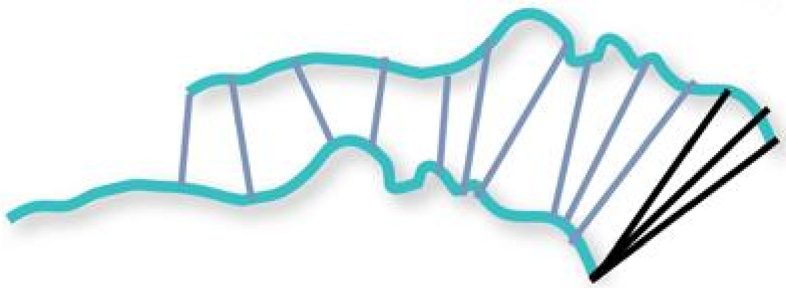
Rigid Alignment via ICP

- ✓ Iterative Closest Point algorithm (when map b/w 2 shapes unknown).
- ✓ Better correspondence estimation via feature points.
- ✓ Goal is to identify when 2 points on different scans represent the same feature.
- ✓ Hint: Are the surroundings similar?
- ✓ Do not match two close points if their normals are too different.

✓ x

Rigid Alignment via ICP

- ✓ Iterative Closest Point algorithm (when map b/w 2 shapes unknown).
- ✓ Better correspondence estimation via feature points.
- ✓ Goal is to identify when 2 points on different scans represent the same feature.
- ✓ Hint: Are the surroundings similar?
- ✓ Do not match two close points if they are on the boundaries (blacks).



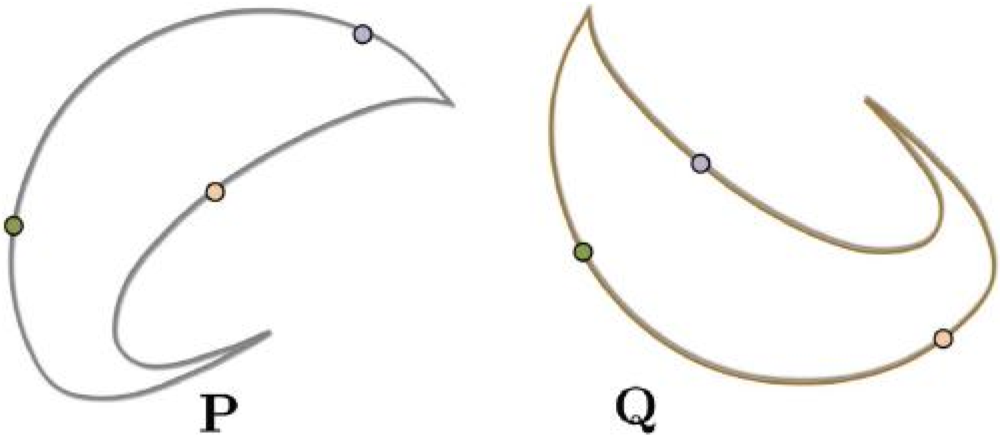
- ✓ Do not match two close points if the closeness is not satisfactory.

Rigid Alignment via RANSAC

- ✓ RANSAC (when map b/w 2 shapes unknown).
- ✓ ICP only needs 3 point pairs to define a rotation in 3D.
- ✓ Brute force. Try all triplets in one shape (C(N,3)) with a base triplet on the other one: $O(N^3)$. Repeat this for L different base triplets: $O(LN^3)$.

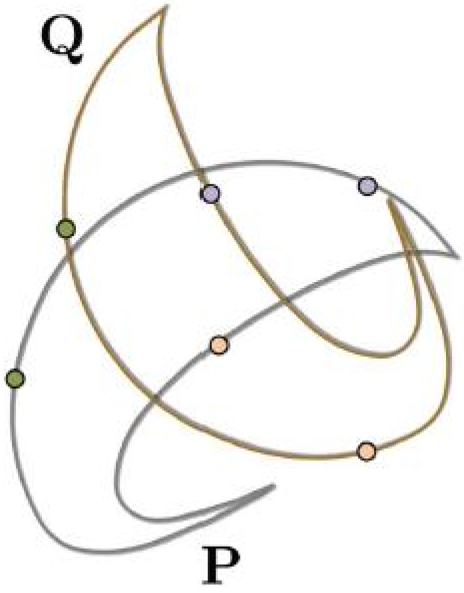
Rigid Alignment via RANSAC

- ✓ RANSAC (when map b/w 2 shapes unknown).
- ✓ ICP only needs 3 point pairs to define a rotation in 3D.
- ✓ RANSAC algo.
  - ✓ Pick a random pair of 3 points on 2 shapes.
  - ✓ Estimate alignment (compute rotation translation) in the least-squares sense.
  - ✓ Check for the error of this alignment (e.g. sum of distnces b/w matching pnts).



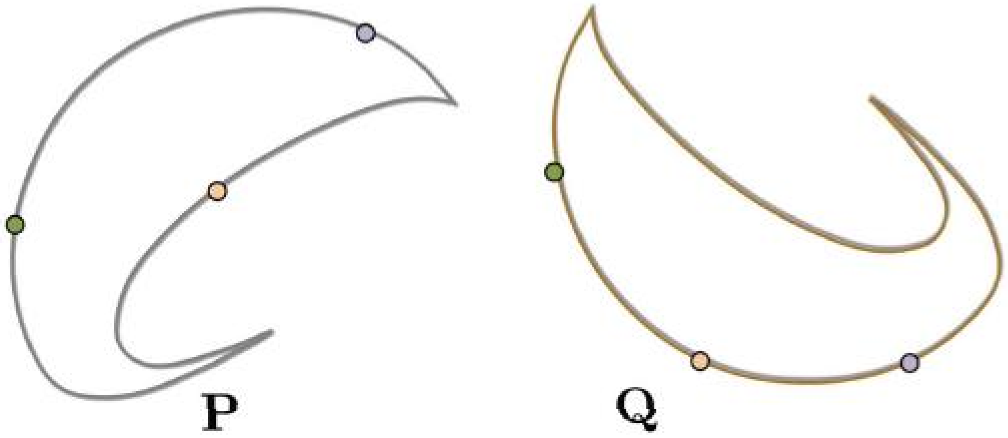
Rigid Alignment via RANSAC

- ✓ RANSAC (when map b/w 2 shapes unknown).
- ✓ ICP only needs 3 point pairs to define a rotation in 3D.
- ✓ RANSAC algo.
  - ✓ Pick a random pair of 3 points on 2 shapes.
  - ✓ Estimate alignment (compute rotation translation) in the least-squares sense.
  - ✓ Check for the error of this alignment (e.g. sum of distnces b/w matching pnts).



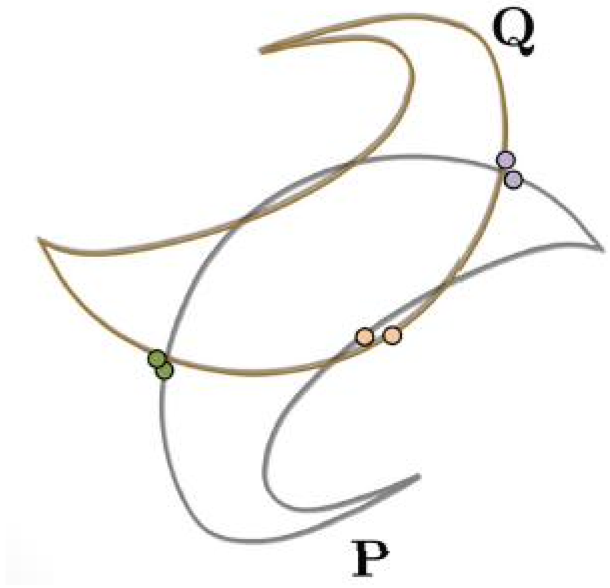
Rigid Alignment via RANSAC

- ✓ RANSAC (when map b/w 2 shapes unknown).
- ✓ ICP only needs 3 point pairs to define a rotation in 3D.
- ✓ RANSAC algo.
  - ✓ Pick a random pair of 3 points on 2 shapes.
  - ✓ Estimate alignment (compute rotation translation) in the least-squares sense.
  - ✓ Check for the error of this alignment (e.g. sum of distnces b/w matching pnts).



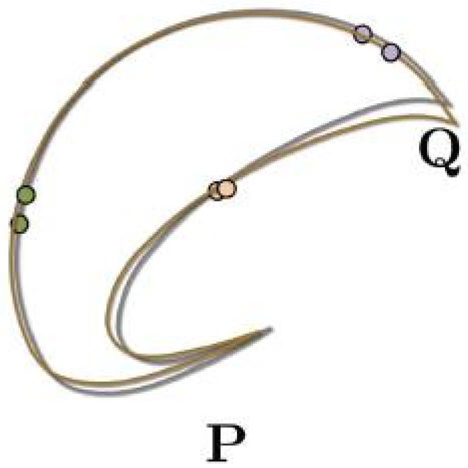
Rigid Alignment via RANSAC

- ✓ RANSAC (when map b/w 2 shapes unknown).
- ✓ ICP only needs 3 point pairs to define a rotation in 3D.
- ✓ RANSAC algo.
  - ✓ Pick a random pair of 3 points on 2 shapes.
  - ✓ Estimate alignment (compute rotation translation) in the least-squares sense.
  - ✓ Check for the error of this alignment (e.g. sum of distnces b/w matching pnts).



Rigid Alignment via RANSAC

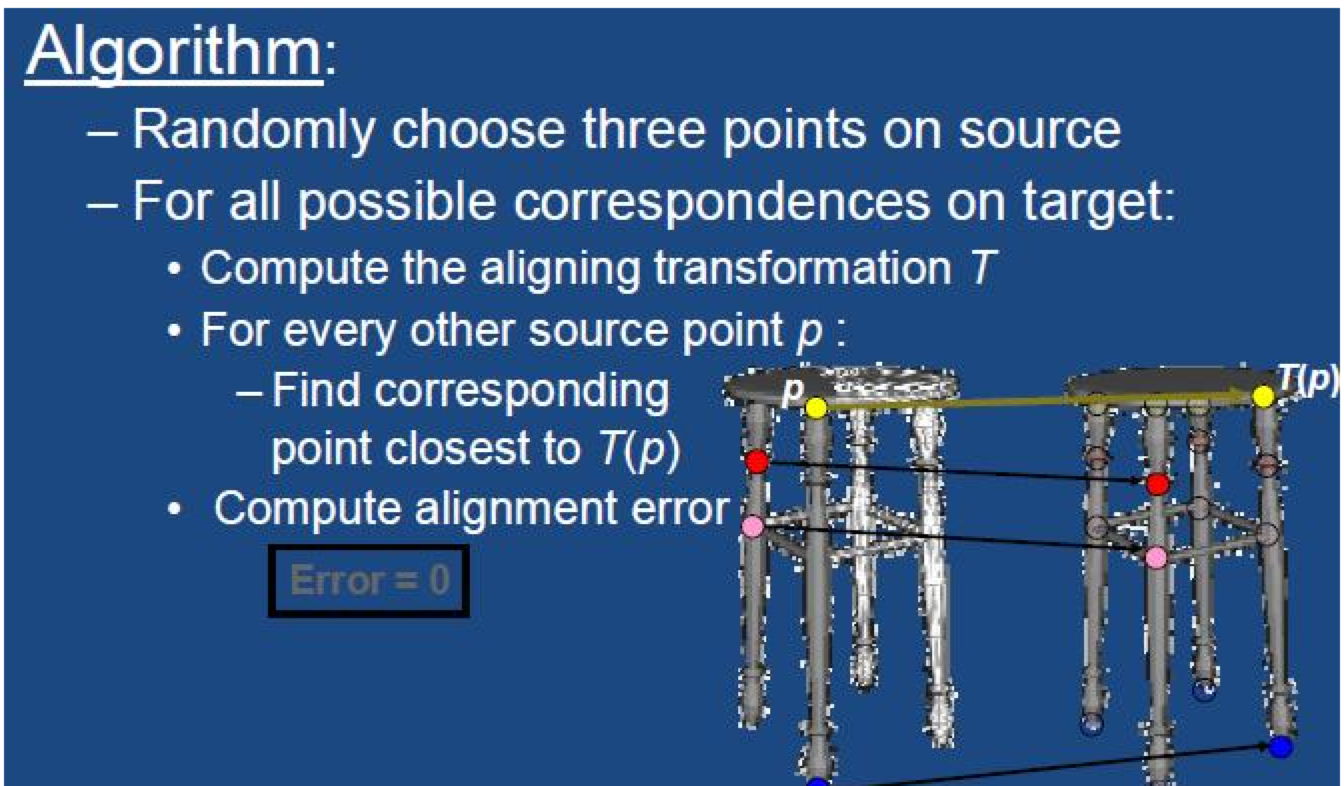
- ✓ RANSAC (when map b/w 2 shapes unknown).
- ✓ ICP only needs 3 point pairs to define a rotation in 3D.
- ✓ RANSAC algo.
  - ✓ Pick a random pair of 3 points on 2 shapes.
  - ✓ Estimate alignment (compute rotation translation) in the least-squares sense.
  - ✓ Check for the error of this alignment (e.g. sum of distnces b/w matching pnts).



Random picks do not have to be perfect matches.

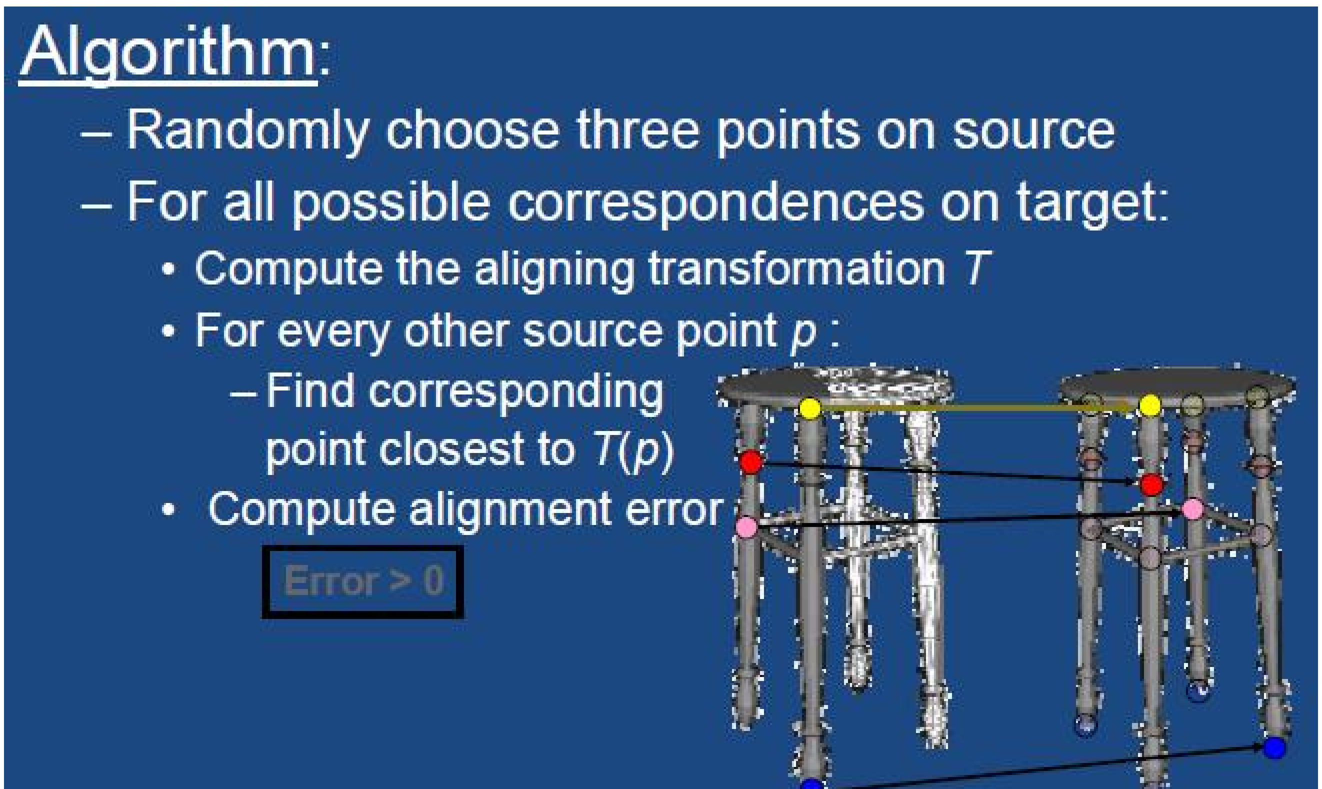
Rigid Alignment via RANSAC

- ✓ RANSAC (when map b/w 2 shapes unknown).
- ✓ ICP only needs 3 point pairs to define a rotation in 3D.
- ✓ Resulting RANSAC session:



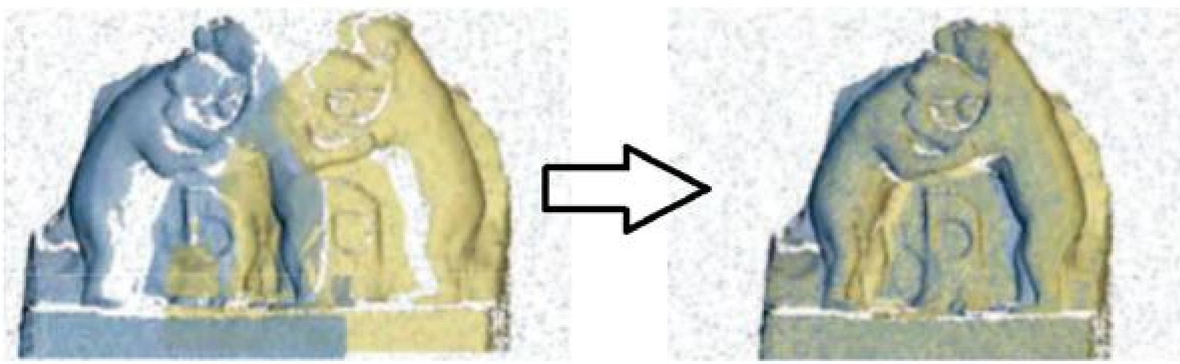
Rigid Alignment via RANSAC

- ✓ RANSAC (when map b/w 2 shapes unknown).
- ✓ ICP only needs 3 point pairs to define a rotation in 3D.
- ✓ Resulting RANSAC session:



Potential Project Topics

- ✓ ICP more resilient to noisy and missing data; paper: Sparse Iterative Closest Point (traditional way: prune or reweight correspondences).



- ✓ Non-rigid ICP to register deforming objects; paper: Robust Articulated-ICP for Real-Time Hand Tracking (Section 4.2 is cool).