Point cloud to point cloud rigid transformations

Russell Taylor
601.455-655

Engineering Research Center for Computer Integrated Surgical Systems and Technology

---

Minimizing Rigid Registration Errors

Typically, given a set of points $\{\mathbf{a}_i\}$ in one coordinate system and another set of points $\{\mathbf{b}_i\}$ in a second coordinate system

Goal is to find $[\mathbf{R},\mathbf{p}]$ that minimizes

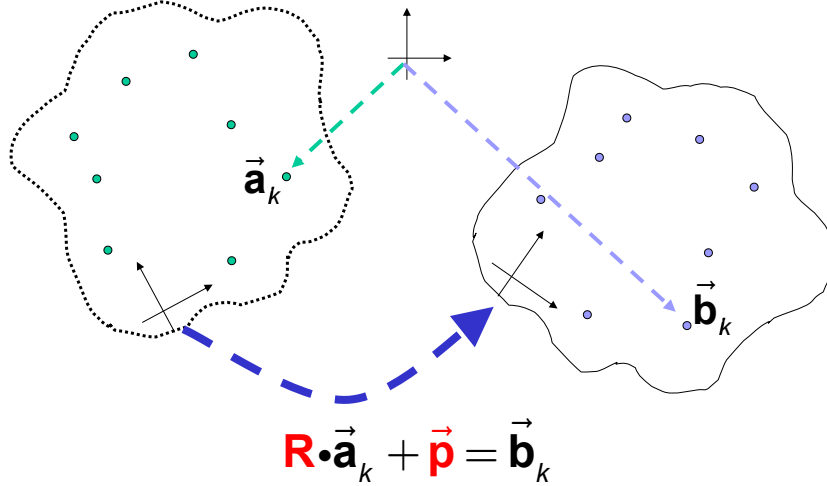$$\eta = \sum_i \mathbf{e}_i \bullet \mathbf{e}_i$$

where

$$\mathbf{e}_i = (\mathbf{R} \bullet \mathbf{a}_i + \mathbf{p}) - \mathbf{b}_i$$

This is tricky, because of $\mathbf{R}$.

Engineering Research Center for Computer Integrated Surgical Systems and Technology

Point cloud to point cloud registration

$$\vec{a}_k$$

$$\vec{b}_k$$

$$\mathbf{R} \cdot \vec{a}_k + \vec{p} = \vec{b}_k$$

Engineering Research Center for Computer Integrated Surgical Systems and Technology

Minimizing Rigid Registration Errors

Step 1: Compute

$$\bar{\mathbf{a}} = \frac{1}{N} \sum_{i=1}^{N} \vec{\mathbf{a}}_i \qquad\qquad \bar{\mathbf{b}} = \frac{1}{N} \sum_{i=1}^{N} \vec{\mathbf{b}}_i$$

$$\tilde{\mathbf{a}}_i = \vec{\mathbf{a}}_i - \bar{\mathbf{a}} \qquad\qquad \tilde{\mathbf{b}}_i = \vec{\mathbf{b}}_i - \bar{\mathbf{b}}$$

Step 2: Find **R** that minimizes

$$\sum_i (\mathbf{R} \cdot \tilde{\mathbf{a}}_i - \tilde{\mathbf{b}}_i)^2$$

Step 3: Find $\vec{\mathbf{p}}$

$$\vec{\mathbf{p}} = \bar{\mathbf{b}} - \mathbf{R} \cdot \bar{\mathbf{a}}$$

Step 4: Desired transformation is
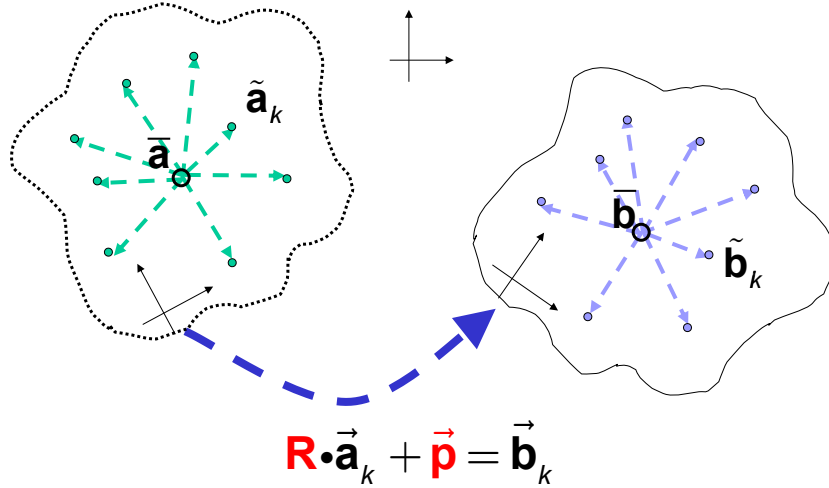
$$\mathbf{F} = Frame(\mathbf{R}, \vec{\mathbf{p}})$$

Engineering Research Center for Computer Integrated Surgical Systems and Technology

Point cloud to point cloud registration

$$\mathbf{R} \cdot \vec{\mathbf{a}}_k + \vec{\mathbf{p}} = \vec{\mathbf{b}}_k$$

5

Point cloud to point cloud registration

$$\vec{\mathbf{p}} = \overline{\mathbf{b}} - \mathbf{R} \cdot \overline{\mathbf{a}}$$
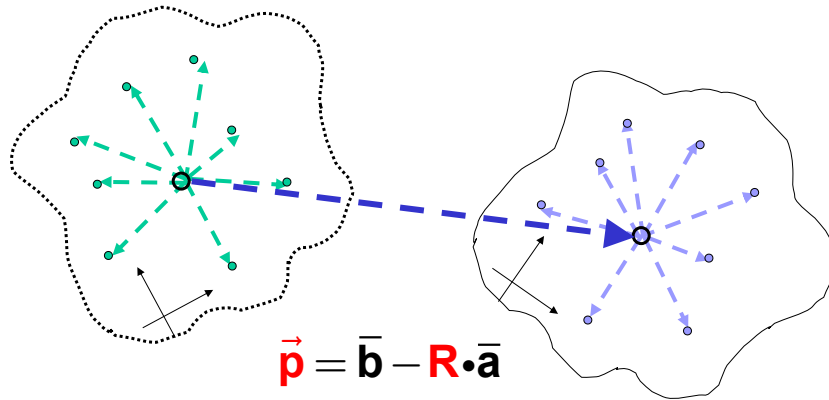
6

3

Rotation Estimation

7 601.455-655 Fall 2000-2021
Copyright © R. H. Taylor

Engineering Research Center for Computer Integrated Surgical Systems and Technology

Rotation Estimation

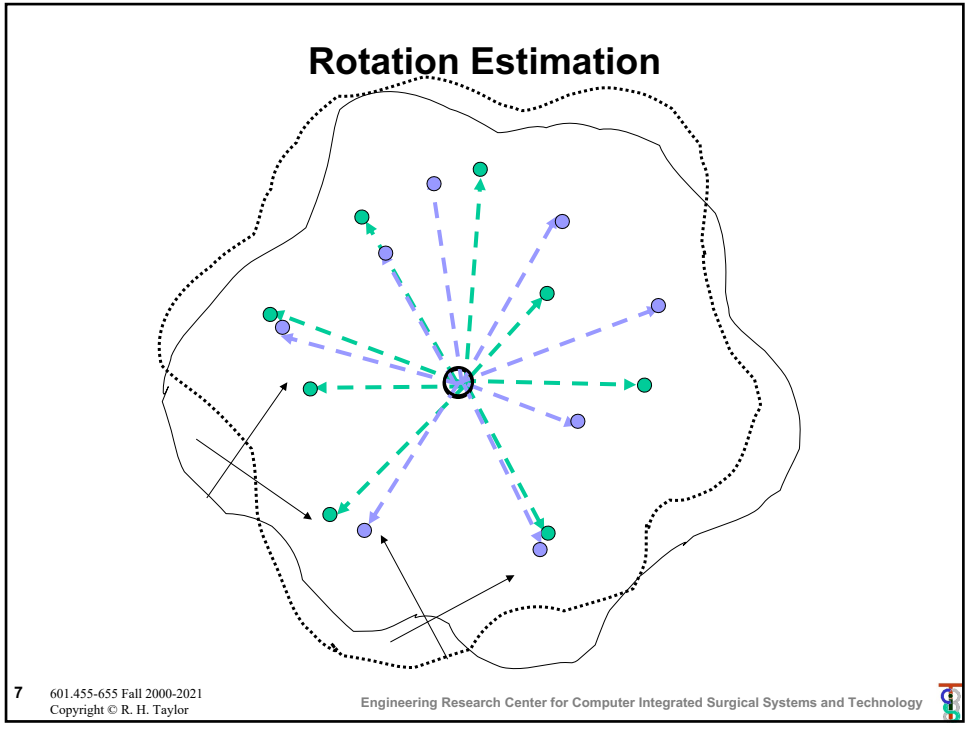**8** 601.455-655 Fall 2000-2021
Copyright © R. H. Taylor
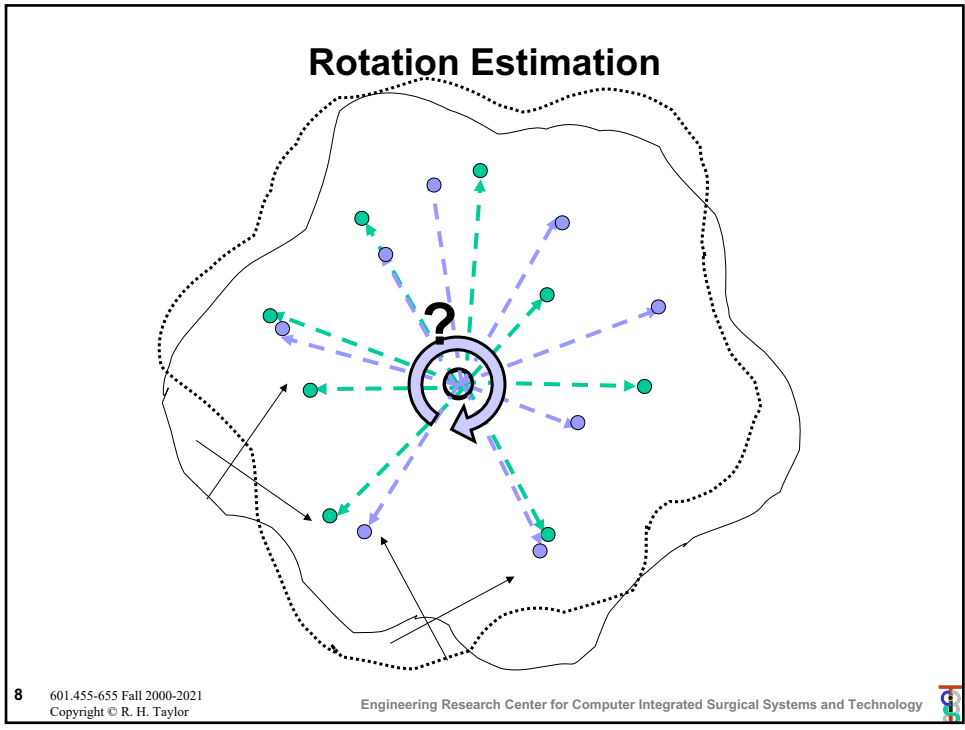
Engineering Research Center for Computer Integrated Surgical Systems and Technology
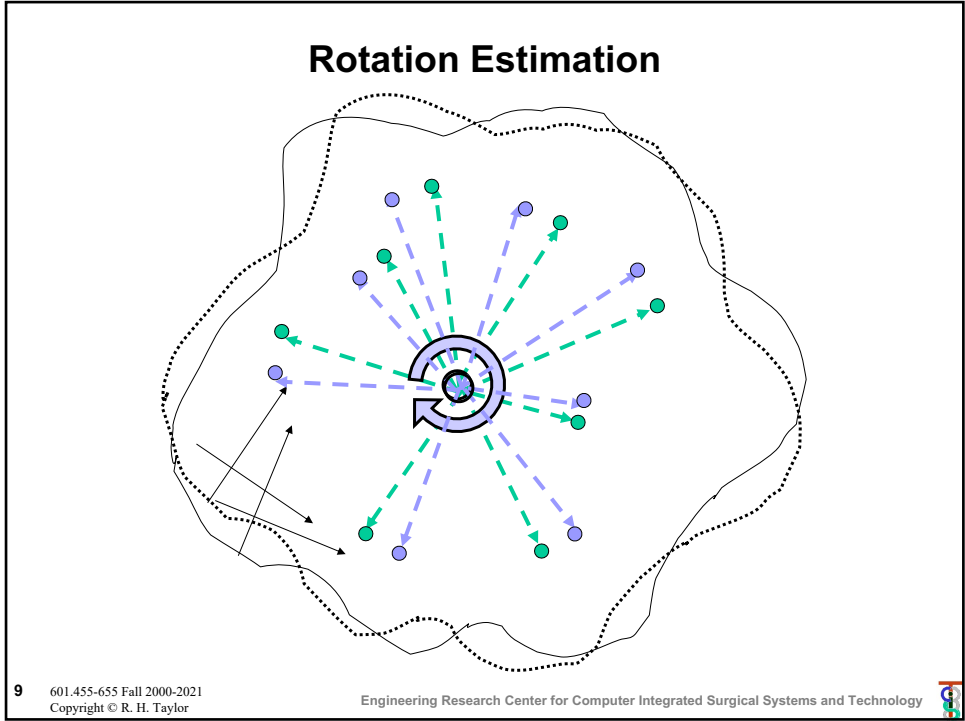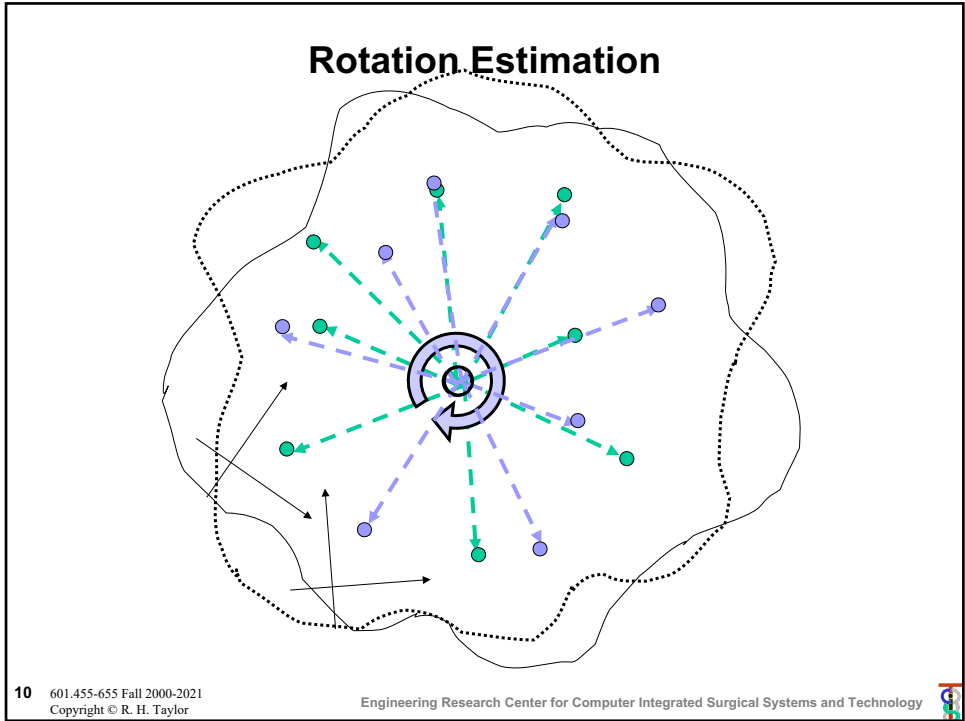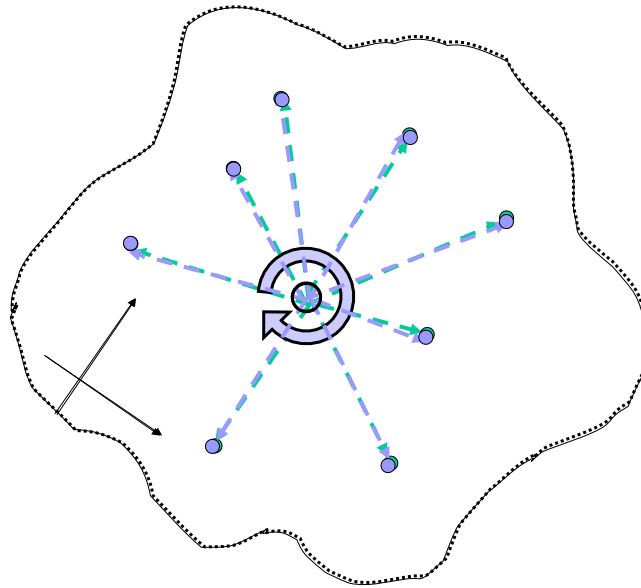
Rotation Estimation

Rotation Estimation

Point cloud to point cloud registration

Solving for R: iteration method

Given $\left\{\cdots,\left(\tilde{\mathbf{a}}_i,\tilde{\mathbf{b}}_i\right),\cdots\right\}$, want to find $\mathbf{R} = \arg\min\sum_i\left\|\mathbf{R}\tilde{\mathbf{a}}_i - \tilde{\mathbf{b}}_i\right\|^2$

Step 0: Make an initial guess $\mathbf{R}_0$

Step 1: Given $\mathbf{R}_k$, compute $\breve{\mathbf{b}}_i = \mathbf{R}_k^{-1}\tilde{\mathbf{b}}_i$

Step 2: Compute $\Delta\mathbf{R}$ that minimizes

$$\sum_i(\Delta\mathbf{R}\ \tilde{\mathbf{a}}_i - \breve{\mathbf{b}}_i)^2$$

Step 3: Set $\mathbf{R}_{k+1} = \mathbf{R}_k\Delta\mathbf{R}$

Step 4: Iterate Steps 1-3 until residual error is sufficiently small
    (or other termination condition)

Iterative method: Getting Initial Guess

We want to find an approximate solution $\mathbf{R}_0$ to

$$\mathbf{R}_0 \bullet \begin{bmatrix} \cdots \tilde{\mathbf{a}}_i \cdots \end{bmatrix} \approx \begin{bmatrix} \cdots \tilde{\mathbf{b}}_i \cdots \end{bmatrix}$$

One way to do this is as follows. Form matrices

$$\mathbf{A} = \begin{bmatrix} \cdots \tilde{\mathbf{a}}_i \cdots \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} \cdots \tilde{\mathbf{b}}_i \cdots \end{bmatrix}$$

Solve least-squares problem $\mathbf{M}_{3x3}\mathbf{A}_{3xN} \approx \mathbf{B}_{3xN}$

Note : You may find it easier to solve $\mathbf{A}_{3xN}^T \mathbf{M}_{3x3}^T \approx \mathbf{B}_{3xN}^T$

Set $\mathbf{R}_0 = orthogonalize(\mathbf{M}_{3x3})$. Verify that $\mathbf{R}$ is a rotation

Our problem is now to solve $\mathbf{R}_0 \Delta \mathbf{R} \mathbf{A} \approx \mathbf{B}$. I.e., $\Delta \mathbf{R} \mathbf{A} \approx \mathbf{R}_0^{-1}\mathbf{B}$

13

Iterative method: Solving for ΔR

Approximate $\Delta \mathbf{R}$ as $(\mathbf{I} + skew(\overline{\alpha}))$. I.e.,

$$\Delta \mathbf{R} \bullet \mathbf{v} \approx \mathbf{v} + \overline{\alpha} \times \mathbf{v}$$

for any vector $\mathbf{v}$. Then, our least squares problem becomes

$$\min_{\Delta \mathbf{R}} \ \sum_i (\Delta \mathbf{R} \bullet \tilde{\mathbf{a}}_i - \breve{\mathbf{b}}_i)^2 \approx \min_{\overline{\alpha}} \sum_i (\tilde{\mathbf{a}}_i - \breve{\mathbf{b}}_i + \overline{\alpha} \times \tilde{\mathbf{a}}_i)^2$$

This is linear least squares problem in $\overline{\alpha}$.

Then compute $\Delta \mathbf{R}(\overline{\alpha})$.

Note: Use trigonometric formulas to compute this

14

Direct Iterative approach for Rigid Frame

Given $\left\{\cdots,\left(\vec{\mathbf{a}}_i,\vec{\mathbf{b}}_i\right),\cdots\right\}$, want to find $\mathbf{F} = \arg\min\sum_i\left\|\mathbf{F}\vec{\mathbf{a}}-\vec{\mathbf{b}}\right\|^2$

Step 0: Make an initial guess $\mathbf{F}_0$

Step 1: Given $\mathbf{F}_k$, compute $\vec{\mathbf{a}}_i^k = \mathbf{F}_k\vec{\mathbf{a}}_i$

Step 2: Compute $\Delta\mathbf{F}$ that minimizes

$$\sum_i\left\|\Delta\mathbf{F}\vec{\mathbf{a}}_i^k - \vec{\mathbf{b}}_i\right\|^2$$

Step 3: Set $\mathbf{F}_{k+1} = \Delta\mathbf{F}\mathbf{F}_k$

Step 4: Iterate Steps 1-3 until residual error is sufficiently small
(or other termination condition)

15

Direct Iterative approach for Rigid Frame

To solve for $\Delta\mathbf{F} = \arg\min\sum_i\left\|\Delta\mathbf{F}\vec{\mathbf{a}}_i^k - \vec{\mathbf{b}}_i\right\|^2$

$$\Delta\mathbf{F}\vec{\mathbf{a}}_i^k - \vec{\mathbf{b}}_i \approx \vec{\alpha}\times\vec{\mathbf{a}}_i^k + \vec{\varepsilon} + \vec{\mathbf{a}}_i^k - \vec{\mathbf{b}}_i$$
$$\vec{\alpha}\times\vec{\mathbf{a}}_i^k + \vec{\varepsilon} \approx \vec{\mathbf{b}}_i - \vec{\mathbf{a}}_i^k$$
$$sk(-\vec{\mathbf{a}}_i^k)\vec{\alpha} + \vec{\varepsilon} \approx \vec{\mathbf{b}}_i - \vec{\mathbf{a}}_i^k$$

Solve the least-squares problem

$$\begin{bmatrix} \vdots & \vdots \\ sk(-\vec{\mathbf{a}}_i^k) & \mathbf{I} \\ \vdots & \vdots \end{bmatrix}\begin{bmatrix} \vec{\alpha} \\ \vec{\varepsilon} \end{bmatrix} \approx \begin{bmatrix} \vdots \\ \vec{\mathbf{b}}_i - \vec{\mathbf{a}}_i^k \\ \vdots \end{bmatrix}$$

Now set $\Delta\mathbf{F}=[\Delta\mathbf{R}(\vec{\alpha}),\vec{\varepsilon}]$

16

Direct Techniques to solve for R

- Method due to K. Arun, et. al., <u>IEEE PAMI</u>, Vol 9, no 5, pp 698-700, Sept 1987

  Step 1: Compute

  $$\mathbf{H} = \sum_i \begin{bmatrix} \tilde{a}_{i,x}\tilde{b}_{i,x} & \tilde{a}_{i,x}\tilde{b}_{i,y} & \tilde{a}_{i,x}\tilde{b}_{i,z} \\ \tilde{a}_{i,y}\tilde{b}_{i,x} & \tilde{a}_{i,y}\tilde{b}_{i,y} & \tilde{a}_{i,y}\tilde{b}_{i,z} \\ \tilde{a}_{i,z}\tilde{b}_{i,x} & \tilde{a}_{i,z}\tilde{b}_{i,y} & \tilde{a}_{i,z}\tilde{b}_{i,z} \end{bmatrix}$$

  NOTE well

  Step 2: Compute the SVD of $\mathbf{H} = \mathbf{U}\mathbf{S}\mathbf{V}^t$

  Step 3: $\mathbf{R} = \mathbf{V}\mathbf{U}^t$

  Step 4: Verify $Det(\mathbf{R}) = 1$. If not, then algorithm may fail.

- Failure is rare, and mostly fixable. The paper has details.

17

Quarternion Technique to solve for R

- B.K.P. Horn, "Closed form solution of absolute orientation using unit quaternions", <u>*J. Opt. Soc. America*</u>, A vol. 4, no. 4, pp 629-642, Apr. 1987.
- Method described as reported in Besl and McKay, "A method for registration of 3D shapes", <u>*IEEE Trans. on Pattern Analysis and Machine*</u> Intelligence, vol. 14, no. 2, February 1992.
- Solves a 4x4 eigenvalue problem to find a unit quaternion corresponding to the rotation
- This quaternion may be converted in closed form to get a more conventional rotation matrix

18

Digression: quaternions

Invented by Hamilton in 1843. Can be thought of as

4 elements:	$\mathbf{q} = \begin{bmatrix} q_0, q_1, q_2, q_3 \end{bmatrix}$
scalar & vector:	$\mathbf{q} = s + \vec{\mathbf{v}} = \begin{bmatrix} s, \vec{\mathbf{v}} \end{bmatrix}$
Complex number:	$\mathbf{q} = q_0 + q_1 i + q_2 j + q_3 k$

$$\text{where} \quad i^2 = j^2 = k^2 = i\,j\,k = -1$$

Properties:

Linearity:	$\lambda\mathbf{q}_1 + \mu\vec{\mathbf{q}}_2 = \begin{bmatrix} \lambda s_1 + \mu s_2, \lambda\vec{\mathbf{v}}_1 + \mu\vec{\mathbf{v}}_2 \end{bmatrix}$
Conjugate:	$\mathbf{q}^* = s - \vec{\mathbf{v}} = \begin{bmatrix} s, -\vec{\mathbf{v}} \end{bmatrix}$
Product:	$\mathbf{q}_1 \circ \mathbf{q}_2 = \begin{bmatrix} s_1 s_2 - \vec{\mathbf{v}}_1 \bullet \vec{\mathbf{v}}_2, s_1\vec{\mathbf{v}}_2 + s_2\vec{\mathbf{v}}_1 + \vec{\mathbf{v}}_1 \times \vec{\mathbf{v}}_2 \end{bmatrix}$
Transform vector:	$\mathbf{q} \circ \vec{\mathbf{p}} = \mathbf{q} \circ \begin{bmatrix} 0, \vec{\mathbf{p}} \end{bmatrix} \circ \mathbf{q}^*$
Norm:	$\|\mathbf{q}\| = \sqrt{s^2 + \vec{\mathbf{v}} \bullet \vec{\mathbf{v}}} = \sqrt{q_0^{\,2} + q_1^{\,2} + q_2^{\,2} + q_3^{\,2}}$

19

Digression continued: unit quaternions

We can associate a rotation by angle θ about an axis $\vec{\mathbf{n}}$ with the unit quaternion:

$$Rot(\vec{\mathbf{n}}, \theta) \Leftrightarrow \begin{bmatrix} \cos\dfrac{\theta}{2}, & \sin\dfrac{\theta}{2}\vec{\mathbf{n}} \end{bmatrix}$$

Exercise: Demonstrate this relationship. I.e., show

$$Rot((\vec{\mathbf{n}}, \theta) \bullet \vec{\mathbf{p}}) = \begin{bmatrix} \cos\dfrac{\theta}{2}, & \sin\dfrac{\theta}{2}\vec{\mathbf{n}} \end{bmatrix} \circ \begin{bmatrix} 0, \vec{\mathbf{p}} \end{bmatrix} \circ \begin{bmatrix} \cos\dfrac{\theta}{2}, & -\sin\dfrac{\theta}{2}\vec{\mathbf{n}} \end{bmatrix}$$

Hint: Substitute and reduce to see if you get Rodrigues' formula.

20

A bit more on quaternions

Exercise: show by substitution that the various formulations for quaternions are equivalent

A few web references:

http://mathworld.wolfram.com/Quaternion.html

http://en.wikipedia.org/wiki/Quaternion

http://en.wikipedia.org/wiki/Quaternions_and_spatial_rotation

http://www.euclideanspace.com/maths/algebra/
      realNormedAlgebra/quaternions/index.htm

CAUTION: Different software packages are not always consistent in the order of elements if a quaternion is represented as a 4 element vector. Some put the scalar part first, others (including cisst libraries) put it last.

Rotation matrix from unit quaternion

$$\mathbf{q} = \begin{bmatrix} q_0, q_1, q_2, q_3 \end{bmatrix}; \quad \|\mathbf{q}\| = 1$$

$$\mathbf{R}(\mathbf{q}) = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 - q_0q_3) & 2(q_1q_3 + q_0q_2) \\ 2(q_1q_2 + q_0q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2q_3 - q_0q_1) \\ 2(q_1q_3 - q_0q_2) & 2(q_2q_3 + q_0q_1) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix}$$

Unit quaternion from rotation matrix

$$\mathbf{R}(\mathbf{q}) = \begin{bmatrix} r_{xx} & r_{yx} & r_{zx} \\ r_{xy} & r_{yy} & r_{zy} \\ r_{xz} & r_{yz} & r_{zz} \end{bmatrix};$$

$$a_0 = 1 + r_{xx} + r_{yy} + r_{zz};\ a_1 = 1 + r_{xx} - r_{yy} - r_{zz}$$
$$a_2 = 1 - r_{xx} + r_{yy} - r_{zz};\ a_3 = 1 - r_{xx} - r_{yy} + r_{zz}$$

$a_0 = \max\{a_k\}$	$a_1 = \max\{a_k\}$	$a_2 = \max\{a_k\}$	$a_3 = \max\{a_k\}$
$q_0 = \dfrac{\sqrt{a_0}}{2}$	$q_0 = \dfrac{r_{yz} - r_{zy}}{4q_1}$	$q_0 = \dfrac{r_{zx} - r_{xz}}{4q_2}$	$q_0 = \dfrac{r_{xy} - r_{yx}}{4q_3}$
$q_1 = \dfrac{r_{xy} - r_{yx}}{4q_0}$	$q_1 = \dfrac{\sqrt{a_1}}{2}$	$q_1 = \dfrac{r_{xy} + r_{yx}}{4q_2}$	$q_1 = \dfrac{r_{xz} + r_{zx}}{4q_3}$
$q_2 = \dfrac{r_{zx} - r_{xz}}{4q_0}$	$q_2 = \dfrac{r_{xy} + r_{yx}}{4q_1}$	$q_2 = \dfrac{\sqrt{a_2}}{2}$	$q_2 = \dfrac{r_{yz} + r_{zy}}{4q_3}$
$q_3 = \dfrac{r_{yz} - r_{zy}}{4q_0}$	$q_3 = \dfrac{r_{xz} + r_{zx}}{4q_1}$	$q_3 = \dfrac{r_{yz} + r_{zy}}{4q_2}$	$q_3 = \dfrac{\sqrt{a_3}}{2}$

Engineering Research Center for Computer Integrated Surgical Systems and Technology

23

Rotation axis and angle from rotation matrix

Many options, including direct trigonemetric solution.
But this works:

$$[\vec{\mathbf{n}}, \theta] \leftarrow ExtractAxisAngle(\mathbf{R})$$
$$\{$$
$$\quad [s, \vec{\mathbf{v}}] \leftarrow ConvertToQuaternion(\mathbf{R})$$
$$\quad\quad return([\vec{\mathbf{v}} / \|\vec{\mathbf{v}}\|, 2\,\mathrm{atan}(s / \|\vec{\mathbf{v}}\|)]$$
$$\}$$

Engineering Research Center for Computer Integrated Surgical Systems and Technology

24

Quaternion method for R

Step 1: Compute

$$\mathbf{H} = \sum_i \begin{bmatrix} \tilde{a}_{i,x}\tilde{b}_{i,x} & \tilde{a}_{i,x}\tilde{b}_{i,y} & \tilde{a}_{i,x}\tilde{b}_{i,z} \\ \tilde{a}_{i,y}\tilde{b}_{i,x} & \tilde{a}_{i,y}\tilde{b}_{i,y} & \tilde{a}_{i,y}\tilde{b}_{i,z} \\ \tilde{a}_{i,z}\tilde{b}_{i,x} & \tilde{a}_{i,z}\tilde{b}_{i,y} & \tilde{a}_{i,z}\tilde{b}_{i,z} \end{bmatrix}$$

Step 2: Compute

$$\mathbf{G} = \begin{bmatrix} trace(\mathbf{H}) & \Delta^T \\ \Delta & \mathbf{H} + \mathbf{H}^T - trace(\mathbf{H})\mathbf{I} \end{bmatrix}$$

where $\Delta^T = \begin{bmatrix} \mathbf{H}_{2,3} - \mathbf{H}_{3,2} & \mathbf{H}_{3,1} - \mathbf{H}_{1,3} & \mathbf{H}_{1,2} - \mathbf{H}_{2,1} \end{bmatrix}$

Step 3: Compute eigen value decomposition of $\mathbf{G}$

$$diag(\bar{\lambda}) = \mathbf{Q}^T\mathbf{G}\mathbf{Q}$$

Step 4: The eigenvector $\mathbf{Q}_k = \begin{bmatrix} q_0, q_1, q_2, q_3 \end{bmatrix}$ corresponding to

the largest eigenvalue λ_k is a unit quaternion corresponding

to the rotation.

25

Another Quaternion Method for R

Let $\mathbf{q} = s + \vec{\mathbf{v}}$ be the unit quaternion corresponding

to $\mathbf{R}$. Let $\tilde{\mathbf{a}}$ and $\tilde{\mathbf{b}}$ be vectors with $\tilde{\mathbf{b}} = \mathbf{R} \cdot \tilde{\mathbf{a}}$ then we have

the quaternion equation

$$(s + \vec{\mathbf{v}}) \cdot (0 + \tilde{\mathbf{a}})(s - \vec{\mathbf{v}}) = 0 + \tilde{\mathbf{b}}$$

$$(s + \vec{\mathbf{v}}) \cdot (0 + \tilde{\mathbf{a}}) = (0 + \tilde{\mathbf{b}}) \cdot (s + \vec{\mathbf{v}}) \quad \text{since } (s - \vec{\mathbf{v}})(s + \vec{\mathbf{v}}) = 1 + \vec{\mathbf{0}}$$

Expanding the scalar and vector parts gives

$$-\vec{\mathbf{v}} \cdot \tilde{\mathbf{a}} = -\vec{\mathbf{v}} \cdot \tilde{\mathbf{b}}$$

$$s\vec{\mathbf{a}} + \vec{\mathbf{v}} \times \tilde{\mathbf{a}} = s\tilde{\mathbf{b}} + \tilde{\mathbf{b}} \times \vec{\mathbf{v}}$$

Rearranging ...

$$(\tilde{\mathbf{b}} - \tilde{\mathbf{a}}) \cdot \vec{\mathbf{v}} = 0$$

$$s(\tilde{\mathbf{b}} - \tilde{\mathbf{a}}) + (\tilde{\mathbf{b}} + \tilde{\mathbf{a}}) \times \vec{\mathbf{v}} = \vec{\mathbf{0}}_3$$

NOTE: This method works for any set of vectors $\vec{a}$ and $\vec{b}$. We are

using the symbols $\tilde{\mathbf{a}}$ and $\tilde{\mathbf{b}}$ to maintain consistency with the discussion of

the previous method.

26

Another Quaternion Method for R

Expressing this as a matrix equation

$$\left[\begin{array}{c|c} 0 & \left(\tilde{\mathbf{b}}-\tilde{\mathbf{a}}\right)^{T} \\ \hline \left(\tilde{\mathbf{b}}-\tilde{\mathbf{a}}\right) & sk\left(\tilde{\mathbf{b}}+\tilde{\mathbf{a}}\right) \end{array}\right]\left[\begin{array}{c} s \\ \hline \vec{\mathbf{v}} \end{array}\right]=\left[\begin{array}{c} 0 \\ \hline \vec{\mathbf{0}}_{3} \end{array}\right]$$

If we now express the quaternion $\mathbf{q}$ as a 4-vector $\vec{\mathbf{q}}=\left[s,\vec{\mathbf{v}}\right]^{T}$,

we can express the rotation problem as the constrained linear
system

$$\mathbf{M}(\vec{\mathbf{a}},\vec{\mathbf{b}})\vec{\mathbf{q}}=\vec{\mathbf{0}}_{4}$$
$$\left\|\vec{\mathbf{q}}\right\|=1$$

27

Another Quaternion Method for R

In general, we have many observations, and we want to
solve the problem in a least squares sense:

$$\min \left\|\mathbf{M}\vec{\mathbf{q}}\right\| \text{ subject to } \left\|\vec{\mathbf{q}}\right\|=1$$

where

$$\mathbf{M}=\left[\begin{array}{c} \mathbf{M}(\vec{\mathbf{a}}_{1},\vec{\mathbf{b}}_{1}) \\ \vdots \\ \mathbf{M}(\vec{\mathbf{a}}_{n},\vec{\mathbf{b}}_{n}) \end{array}\right] \text{ and } n \text{ is the number of observations}$$

Taking the singular value decomposition of $\mathbf{M}=\mathbf{U}\Sigma\mathbf{V}^{T}$ reduces
this to the easier problem

$$\min \left\|\mathbf{U}\Sigma\mathbf{V}^{T}\vec{\mathbf{q}}_{X}\right\|=\left\|\mathbf{U}\left(\Sigma\vec{\mathbf{y}}\right)\right\|=\left\|\Sigma\vec{\mathbf{y}}\right\| \text{ subject to } \left\|\vec{\mathbf{y}}\right\|=\left\|\mathbf{V}^{T}\vec{\mathbf{q}}\right\|=\left\|\vec{\mathbf{q}}\right\|=1$$

28

## Another Quaternion Method for R

This problem is just

$$\min \left\| \Sigma \vec{\mathbf{y}} \right\| = \left\| \begin{bmatrix} \sigma_1 & 0 & 0 & 0 \\ 0 & \sigma_2 & 0 & 0 \\ 0 & 0 & \sigma_3 & 0 \\ 0 & 0 & 0 & \sigma_4 \end{bmatrix} \vec{\mathbf{y}} \right\| \quad \text{subject to} \ \left\| \vec{\mathbf{y}} \right\| = 1$$

where σ_i are the singular values. Recall that SVD routines typically return the $\sigma_i \geq 0$ and sorted in decreasing magnitude. So σ_4 is the smallest singular value and the value of $\vec{\mathbf{y}}$ with $\left\| \vec{\mathbf{y}} \right\| = 1$ that minimizes $\left\| \Sigma \vec{\mathbf{y}} \right\|$ is $\vec{\mathbf{y}} = \begin{bmatrix} 0,0,0,1 \end{bmatrix}^T$. The corresponding value of $\vec{\mathbf{q}}$ is given by $\vec{\mathbf{q}} = \mathbf{V}\vec{\mathbf{y}} = \mathbf{V}_4$. Where $\mathbf{V}_4$ is the 4th column of $\mathbf{V}$.

29

Non-reflective spatial similarity (rigid+scale)

$$\sigma \mathbf{R} \cdot \vec{\mathbf{a}}_k + \vec{\mathbf{p}} = \vec{\mathbf{b}}_k$$
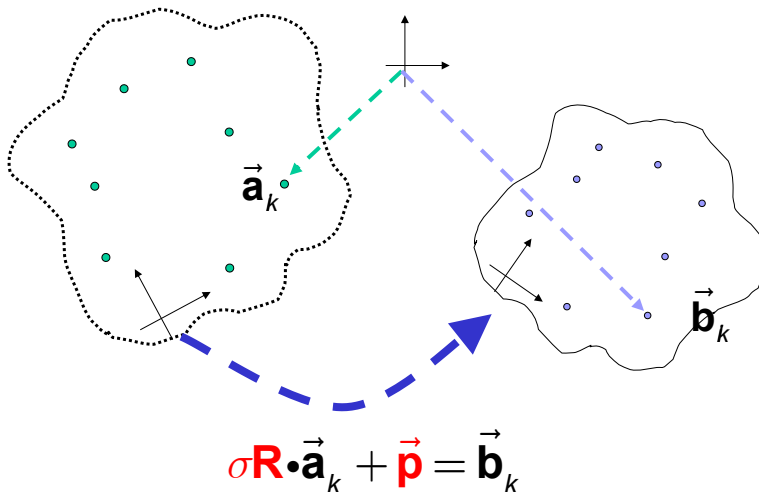
30

Non-reflective spatial similarity

Step 1: Compute

$$\bar{\mathbf{a}} = \frac{1}{N}\sum_{i=1}^{N}\vec{\mathbf{a}}_i \qquad\qquad \bar{\mathbf{b}} = \frac{1}{N}\sum_{i=1}^{N}\vec{\mathbf{b}}_i$$

$$\tilde{\mathbf{a}}_i = \vec{\mathbf{a}}_i - \bar{\mathbf{a}} \qquad\qquad \tilde{\mathbf{b}}_i = \vec{\mathbf{b}}_i - \bar{\mathbf{b}}$$

Step 2: Estimate scale

$$\sigma = \frac{\sum_i \left\| \tilde{\mathbf{b}}_i \right\|}{\sum_i \left\| \tilde{\mathbf{a}}_i \right\|}$$

Step 3: Find $\mathbf{R}$ that minimizes

$$\sum_i (\mathbf{R}\cdot(\sigma\tilde{\mathbf{a}}_i) - \tilde{\mathbf{b}}_i)^2$$

Step 4: Find $\vec{\mathbf{p}}$

$$\vec{\mathbf{p}} = \bar{\mathbf{b}} - \mathbf{R}\cdot\bar{\mathbf{a}}$$

Step 5: Desired transformation is

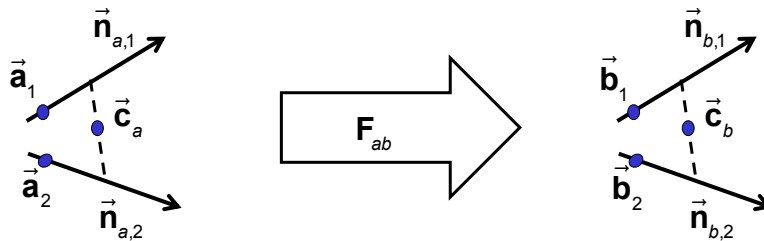$$\mathbf{F} = SimilarityFrame(\sigma, \mathbf{R}, \vec{\mathbf{p}})$$

31

Registration from line pairs

Approach 1:

Compute $\mathbf{F}_a = [\mathbf{R}_a, \vec{\mathbf{c}}_a]$ from line pair a

Compute $\mathbf{F}_b = [\mathbf{R}_b, \vec{\mathbf{c}}_b]$ from line pair b

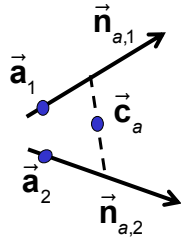$$\mathbf{F}_{ab} = \mathbf{F}_a^{-1}\mathbf{F}_b$$

32

16

Registration from line pairs

$$R_a = \begin{bmatrix} \vec{x}_1 = \vec{n}_{a,1} & \vec{y}_1 = \dfrac{\vec{n}_{a,1} \times \vec{n}_{a,2}}{\left\| \vec{n}_{a,1} \times \vec{n}_{a,2} \right\|} & \vec{z}_1 = \vec{x}_1 \times \vec{y}_1 \end{bmatrix}$$

$\vec{c}_a$ = midpoint between the two lines

To get the midpoint:

$$\text{Solve} \quad \begin{bmatrix} \vec{n}_{a,1} & -\vec{n}_{a,2} \end{bmatrix} \begin{bmatrix} \lambda \\ \nu \end{bmatrix} \approx \begin{bmatrix} \vec{a}_2 - \vec{a}_1 \end{bmatrix}$$

$$\text{Then} \quad \vec{c}_a = \frac{\left(\vec{a}_1 + \lambda \vec{n}_{a,1} \right) + \left(\vec{a}_2 + \nu \vec{n}_{a,2} \right)}{2}$$

Distance of a point from a line

$$d = \left\| \vec{c} - \vec{a} \right\| \sin\theta$$
$$= \left\| \vec{n} \times (\vec{c} - \vec{a}) \right\|$$
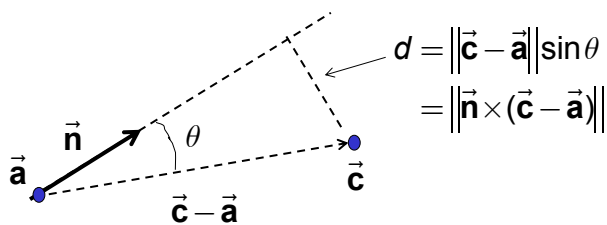
So, to find the closest point to multiple lines

$$\vec{c} = \arg\min \sum_k d_k^2$$

Solve this problem in a least squares sense:

$$\vec{n}_k \times (\vec{c} - \vec{a}_k) \approx \vec{0} \text{ for } k = 1,...,n$$

Equivalently, solve

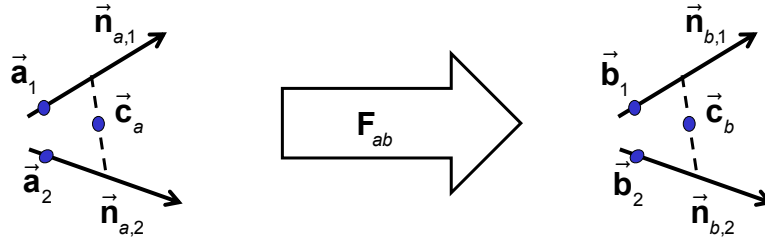$$\vec{n}_k \times \vec{c} \approx \vec{n}_k \times \vec{a}_k \text{ for } k = 1,...,n$$

Registration from multiple corresponding lines

Approach 2:

$$\text{Solve } \mathbf{R}_{ab}\vec{\mathbf{n}}_{b,k} \approx \vec{\mathbf{n}}_{a,k} \text{ for } \mathbf{R}_{ab}$$

$$\text{Solve } \vec{\mathbf{n}}_{a,k} \times \vec{\mathbf{c}}_a \approx \vec{\mathbf{n}}_{a,k} \times \vec{\mathbf{a}}_k \text{ for } \vec{\mathbf{c}}_a$$

$$\text{Solve } \vec{\mathbf{n}}_{b,k} \times \vec{\mathbf{c}}_b \approx \vec{\mathbf{n}}_{b,k} \times \vec{\mathbf{b}}_k \text{ for } \vec{\mathbf{c}}_b$$

$$\vec{\mathbf{p}}_{ab} = \vec{\mathbf{c}}_a - \mathbf{R}_{ab}\vec{\mathbf{c}}_b$$

35