Point cloud to point cloud rigid transformations

Russell Taylor
601.455-655

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 [**R**,**p**] that minimizes

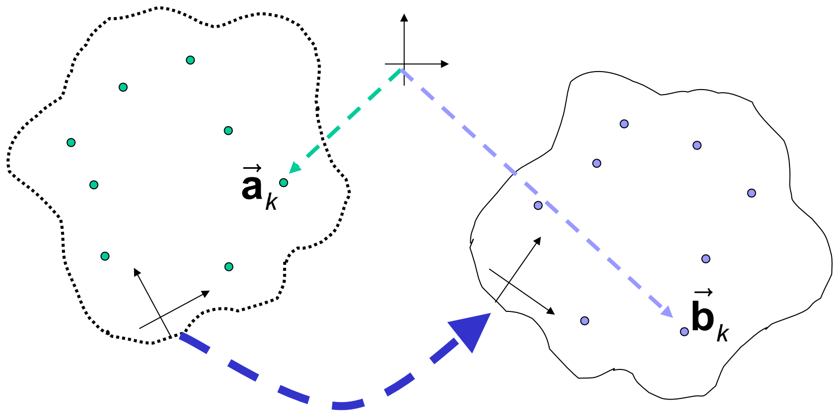
$$\eta = \sum_i \mathbf{e}_i \bullet \mathbf{e}_i$$

where

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

This is tricky, because of **R**.

Point cloud to point cloud registration



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

Minimizing Rigid Registration Errors

Step 1: Compute

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

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

Step 2: Find $\mathbf{R}$ that minimizes

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

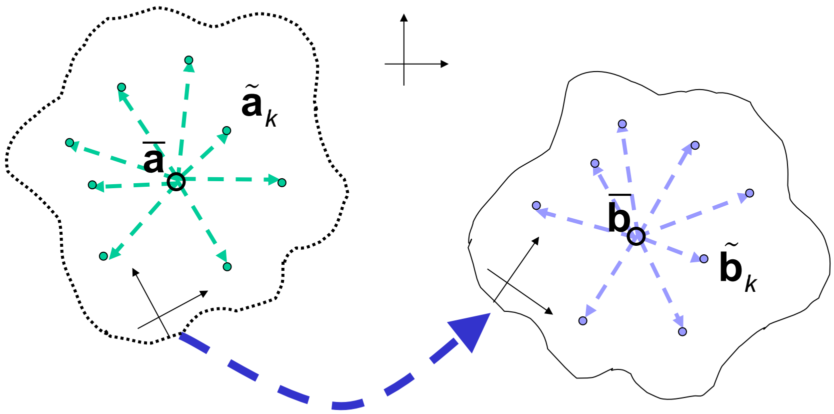
Step 3: Find $\vec{\mathbf{p}}$

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

Step 4: Desired transformation is

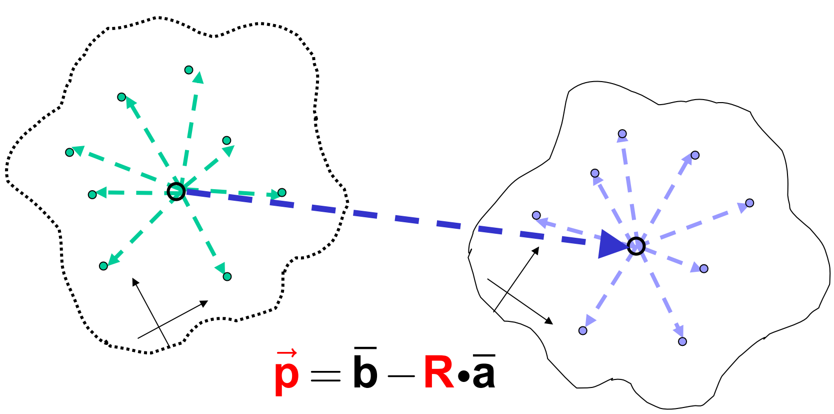
$$\mathbf{F} = \mathit{Frame}(\mathbf{R}, \vec{\mathbf{p}})$$

Point cloud to point cloud registration



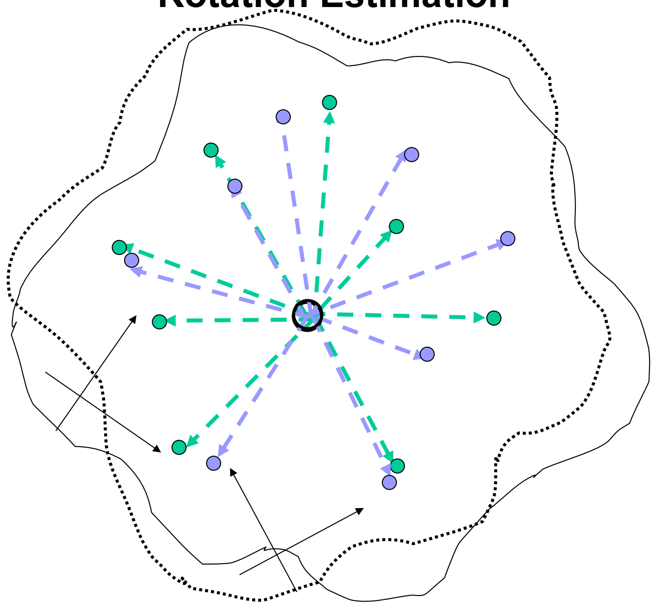
$$\mathbf{R} \bullet \vec{\mathbf{a}}_k + \vec{\mathbf{p}} = \vec{\mathbf{b}}_k$$

Point cloud to point cloud registration



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

Rotation Estimation

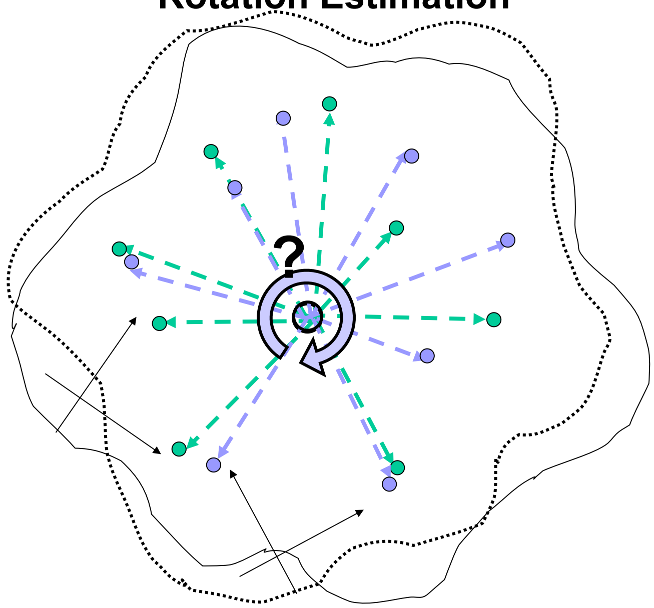


 601.455-655 Fall 2000-2022

Engineering Research Center for Computer Integrated Surgical Systems and Technology

7

Rotation Estimation

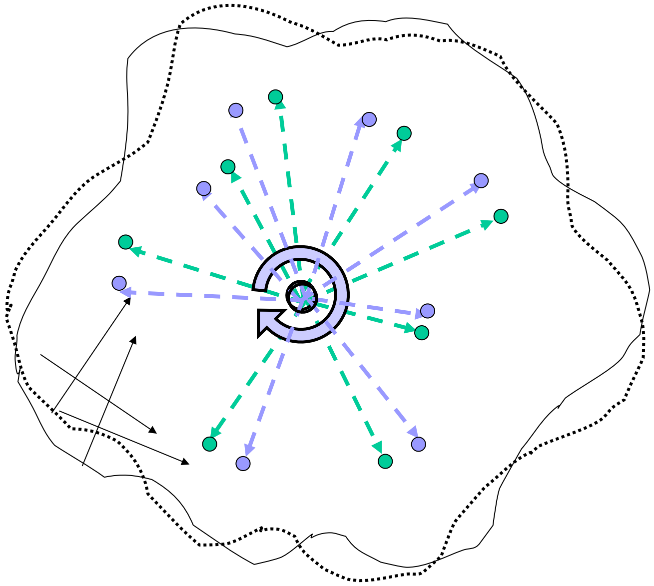


 601.455-655 Fall 2000-2022

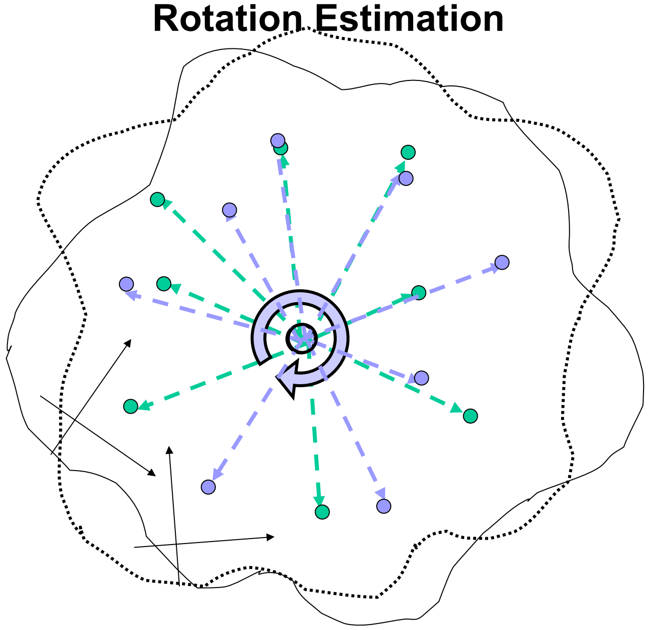
Engineering Research Center for Computer Integrated Surgical Systems and Technology

8

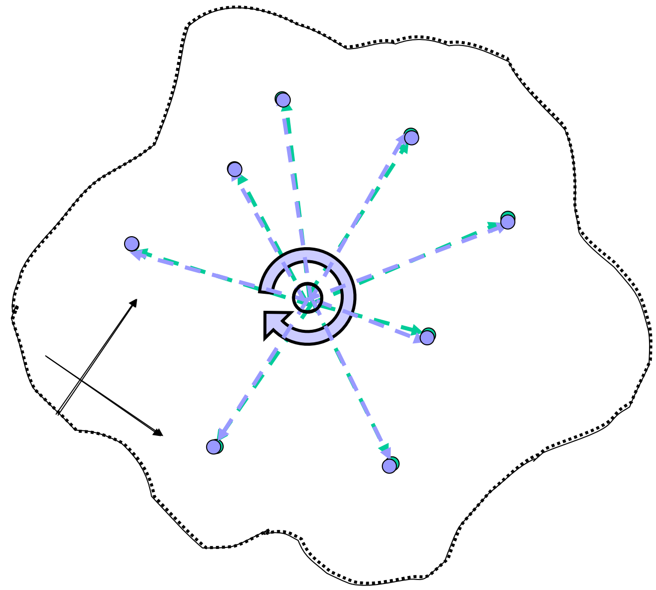
Rotation Estimation



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

$$\mathbf{R}_0 \bullet \left[\cdots \tilde{\mathbf{a}}_i \cdots\right] \approx \left[\cdots \tilde{\mathbf{b}}_i \cdots\right]$$

One way to do this is as follows. Form matrices

$$\mathbf{A} = \left[\cdots \tilde{\mathbf{a}}_i \cdots\right] \quad \mathbf{B} = \left[\cdots \tilde{\mathbf{b}}_i \cdots\right]$$

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

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

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., $\Delta\mathbf{R}\mathbf{A} \approx \mathbf{R}_0^{-1}\mathbf{B}$

Iterative method: Solving for ΔR

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

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

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

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

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

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

Note: Use trigonometric formulas to compute this

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)

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

Direct Techniques to solve for R

- Method due to K. Arun, et. al., IEEE PAMI, 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 $\mathbf{H} = \mathbf{USV}^t$

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

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

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

Quarternion Technique to solve for R

- B.K.P. Horn, "Closed form solution of absolute orientation using unit quaternions", *J. Opt. Soc. America*, 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", *IEEE Trans. on Pattern Analysis and Machine* 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

Digression: quaternions

Invented by Hamilton in 1843. Can be thought of as

4 elements: $\mathbf{q} = \left[q_0, q_1, q_2, q_3 \right]$

scalar & vector: $\mathbf{q} = s + \vec{\mathbf{v}} = \left[s, \vec{\mathbf{v}} \right]$

Complex number: $\mathbf{q} = q_0 + q_1 i + q_2 j + q_3 k$

where $i^2 = j^2 = k^2 = i\,j\,k = -1$

Properties:

Linearity: $\lambda \mathbf{q}_1 + \mu \vec{\mathbf{q}}_2 = \left[\lambda s_1 + \mu s_2, \lambda \vec{\mathbf{v}}_1 + \mu \vec{\mathbf{v}}_2 \right]$

Conjugate: $\mathbf{q}^* = s - \vec{\mathbf{v}} = \left[s, -\vec{\mathbf{v}} \right]$

Product: $\mathbf{q}_1 \circ \mathbf{q}_2 = \left[s_1 s_2 - \vec{\mathbf{v}}_1 \bullet \vec{\mathbf{v}}_2, s_1 \vec{\mathbf{v}}_2 + s_2 \vec{\mathbf{v}}_1 + \vec{\mathbf{v}}_1 \times \vec{\mathbf{v}}_2 \right]$

Transform vector: $\mathbf{q} \circ \vec{\mathbf{p}} \quad = \mathbf{q} \circ \left[0, \vec{\mathbf{p}} \right] \circ \mathbf{q}^*$

Norm: $\left\| \mathbf{q} \right\| = \sqrt{s^2 + \vec{\mathbf{v}} \bullet \vec{\mathbf{v}}} = \sqrt{q_0^{\,2} + q_1^{\,2} + q_2^{\,2} + q_3^{\,2}}$

Digression continued: unit quaternions

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

$$Rot(\vec{\mathbf{n}}, \theta) \Leftrightarrow \left[\cos\frac{\theta}{2}, \ \sin\frac{\theta}{2} \vec{\mathbf{n}} \right]$$

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

$$Rot((\vec{\mathbf{n}}, \theta) \bullet \vec{\mathbf{p}} = \left[\cos\frac{\theta}{2}, \ \sin\frac{\theta}{2} \vec{\mathbf{n}} \right] \circ [0, \vec{\mathbf{p}}] \circ \left[\cos\frac{\theta}{2}, \ -\sin\frac{\theta}{2} \vec{\mathbf{n}} \right]$$

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

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

$$\mathbf{q} = [q_0, q_1, q_2, q_3]; \quad \|\mathbf{q}\| = 1$$

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

Unit quaternion from rotation matrix

$$\mathbf{R}(\mathbf{q}) = \begin{bmatrix} r_{xx} & r_{yx} & r_{zx} \\ r_{xy} & r_{yy} & r_{zy} \\ r_{xz} & r_{yz} & r_{zz} \end{bmatrix}; \quad \begin{matrix} a_0 = 1 + r_{xx} + r_{yy} + r_{zz};\ a_1 = 1 + r_{xx} - r_{yy} - r_{zz} \\ a_2 = 1 - r_{xx} + r_{yy} - r_{zz};\ a_3 = 1 - r_{xx} - r_{yy} + r_{zz} \end{matrix}$$

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

Rotation axis and angle from rotation matrix

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

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

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 $\mathbf{Q}_k=\left[q_0,q_1,q_2,q_3\right]$ corresponding to the largest eigenvalue λ_k is a unit quaternion corresponding to the rotation.

25 601.455-655 Fall 2000-2022

Engineering Research Center for Computer Integrated Surgical Systems and Technology

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 $\tilde{\mathbf{b}}=\mathbf{R}\bullet\tilde{\mathbf{a}}$ then we have the quaternion equation

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

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

Expanding the scalar and vector parts gives

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

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

Rearranging ...

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

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

NOTE: This method works for any set of vectors $\vec{\mathbf{a}}$ and $\vec{\mathbf{b}}$. We are using the symbols $\tilde{\mathbf{a}}$ and $\tilde{\mathbf{b}}$ to maintain consistency with the discussion of the previous method.

26 601.455-655 Fall 2000-2022

Engineering Research Center for Computer Integrated Surgical Systems and Technology

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 **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

$$\mathbf{M}(\vec{\mathbf{a}},\vec{\mathbf{b}})\vec{\mathbf{q}}=\vec{\mathbf{0}}_4$$
$$\left\|\vec{\mathbf{q}}\right\|=1$$

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

$$\mathbf{M}=\begin{bmatrix} \mathbf{M}(\vec{\mathbf{a}}_1,\vec{\mathbf{b}}_1) \\ \vdots \\ \mathbf{M}(\vec{\mathbf{a}}_n,\vec{\mathbf{b}}_n) \end{bmatrix}$$ and n 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

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

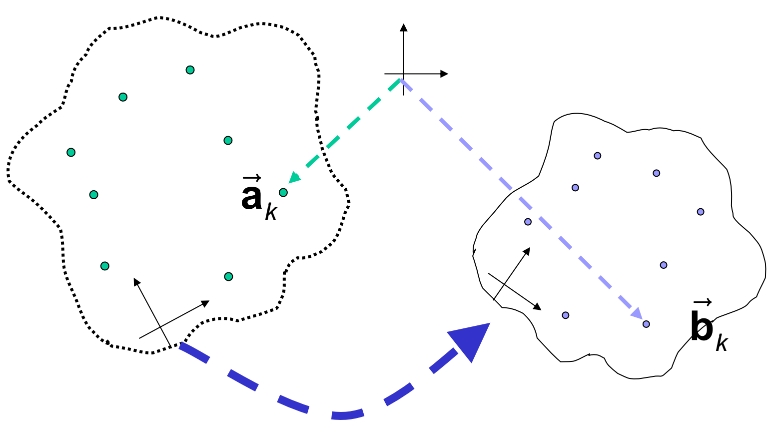
Another Quaternion Method for R

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 $\vec{\mathbf{y}} = \left[0,0,0,1\right]^T$. 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 $\mathbf{V}$.

Non-reflective spatial similarity (rigid+scale)



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

Non-reflective spatial similarity

Step 1: Compute

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

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

Step 2: Estimate scale

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

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

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

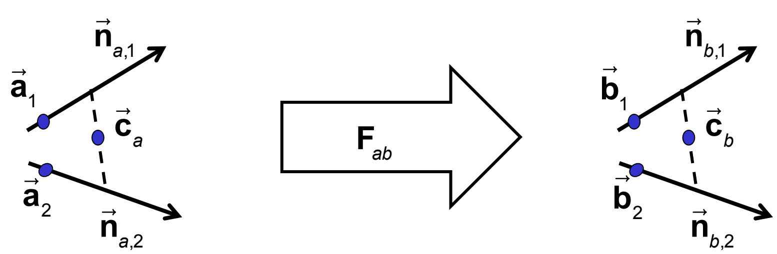
Step 4: Find $\vec{\mathbf{p}}$

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

Step 5: Desired transformation is

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

Registration from line pairs



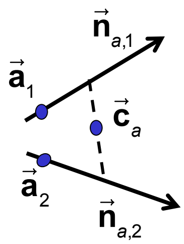
Approach 1:

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*

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

Registration from line pairs



$$\mathbf{R}_a = \left[\quad \vec{\mathbf{x}}_1 = \vec{\mathbf{n}}_{a,1} \quad \vec{\mathbf{y}}_1 = \frac{\vec{\mathbf{n}}_{a,1} \times \vec{\mathbf{n}}_{a,2}}{\left\| \vec{\mathbf{n}}_{a,1} \times \vec{\mathbf{n}}_{a,2} \right\|} \quad \vec{\mathbf{z}}_1 = \vec{\mathbf{x}}_1 \times \vec{\mathbf{y}}_1 \quad \right]$$

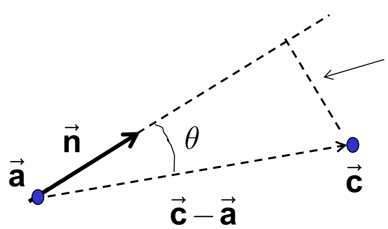
$\vec{\mathbf{c}}_a =$ midpoint between the two lines

To get the midpoint:

Solve $\left[\vec{\mathbf{n}}_{a,1} \quad -\vec{\mathbf{n}}_{a,2} \right] \begin{bmatrix} \lambda \\ \nu \end{bmatrix} \approx \left[\vec{\mathbf{a}}_2 - \vec{\mathbf{a}}_1 \right]$

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

Distance of a point from a line



$$d = \left\| \vec{\mathbf{c}} - \vec{\mathbf{a}} \right\| \sin\theta$$
$$= \left\| \vec{\mathbf{n}} \times (\vec{\mathbf{c}} - \vec{\mathbf{a}}) \right\|$$

So, to find the closest point to multiple lines

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

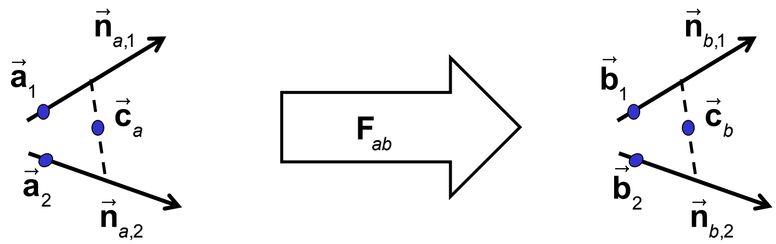
Solve this problem in a least squares sense:

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

Equivalently, solve

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

Registration from multiple corresponding lines



Approach 2:

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

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

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

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

 601.455-655 Fall 2000-2022