Point cloud to point cloud rigid transformations

Russell Taylor
601.455-655

Minimizing Rigid Registration Errors

Typically, given a set of points \( \{a_i\} \) in one coordinate system and another set of points \( \{b_i\} \) in a second coordinate system, the goal is to find \([R,p]\) that minimizes

\[
\eta = \sum_i e_i \cdot e_i
\]

where

\[
e_i = (R \cdot a_i + p) - b_i
\]

This is tricky, because of \( R \).
Point cloud to point cloud registration

\[ \mathbf{R} \mathbf{a}_k + \mathbf{p} = \mathbf{b}_k \]

Minimizing Rigid Registration Errors

Step 1: Compute
\[
\bar{\mathbf{a}} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{a}_i \\
\bar{\mathbf{b}} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{b}_i \\
\tilde{\mathbf{a}}_i = \mathbf{a}_i - \bar{\mathbf{a}} \\
\tilde{\mathbf{b}}_i = \mathbf{b}_i - \bar{\mathbf{b}}
\]

Step 2: Find \( \mathbf{R} \) that minimizes
\[
\sum_i (\mathbf{R} \cdot \tilde{\mathbf{a}}_i - \tilde{\mathbf{b}}_i)^2
\]

Step 3: Find \( \mathbf{p} \)
\[
\tilde{\mathbf{p}} = \tilde{\mathbf{b}} - \mathbf{R} \cdot \bar{\mathbf{a}}
\]

Step 4: Desired transformation is
\[
\mathbf{F} = \text{Frame}(\mathbf{R}, \tilde{\mathbf{p}})
\]
Point cloud to point cloud registration

\[ \mathbf{R} \mathbf{a}_k + \mathbf{p} = \mathbf{b}_k \]

Point cloud to point cloud registration

\[ \mathbf{p} = \mathbf{b} - \mathbf{R} \mathbf{a} \]
Rotation Estimation
Point cloud to point cloud registration

Solving for $R$: iteration method

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

Step 0: Make an initial guess $R_0$

Step 1: Given $R_k$, compute $\tilde{b}_i = R_k^{-1}\tilde{b}_i$

Step 2: Compute $\Delta R$ that minimizes

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

Step 3: Set $R_{k+1} = R_k \Delta R$

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 $R_0$ to

$$ R_0 \cdot [\ldots \hat{a}_j \ldots] \approx [\ldots \hat{b}_j \ldots] $$

One way to do this is as follows. Form matrices

$$ A= [\ldots \hat{a}_i \ldots ] \quad B= [\ldots \hat{b}_i \ldots ] $$

Solve least-squares problem $M_{3 \times 3} A_{3 \times N} = B_{3 \times N}$

**Note**: You may find it easier to solve $A_{3 \times N}^T M_{3 \times 3}^T \approx B_{3 \times N}^T$

Set $R_0 = orthogonalize(M_{3 \times 3})$. Verify that $R$ is a rotation

Our problem is now to solve $R_0 \Delta R A = B$. I.e., $\Delta RA \approx R_0^{-1} B$

**Iterative method: Solving for $\Delta R$**

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

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

for any vector $v$. Then, our least squares problem becomes

$$ \min_{\Delta R} \sum_i (\Delta R \cdot \hat{a}_j - \hat{b}_j)^2 \approx \min_{\vec{\alpha}} \sum_i (\hat{a}_j - \hat{b}_j + \vec{\alpha} \times \hat{a}_j)^2 $$

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

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

**Note**: Use trigonometric formulas to compute this
Direct Iterative approach for Rigid Frame

Given \( \{ \ldots, (\mathbf{a}_i, \mathbf{b}_i), \ldots \} \), want to find \( \mathbf{F} = \arg\min \sum_i \| \mathbf{F}\mathbf{a}_i - \mathbf{b}_i \|^2 \)

Step 0: Make an initial guess \( \mathbf{F}_0 \)
Step 1: Given \( \mathbf{F}_k \), compute \( \mathbf{a}_i^k = \mathbf{F}_k \mathbf{a}_i \)
Step 2: Compute \( \Delta \mathbf{F} \) that minimizes
\[
\sum_i \| \Delta \mathbf{F}\mathbf{a}_i^k - \mathbf{b}_i \|^2
\]
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)

\[
\Delta \mathbf{F}\mathbf{a}_i^k - \mathbf{b}_i \approx \mathbf{\tilde{a}}_i^k + \mathbf{\tilde{e}} + \mathbf{\tilde{a}}_i^k - \mathbf{b}_i
\]
\[
\mathbf{\tilde{a}}_i^k + \mathbf{\tilde{e}} \approx \mathbf{b}_i - \mathbf{\tilde{a}}_i^k
\]
\[
\mathbf{s}(\mathbf{\tilde{a}}_i^k) \mathbf{\tilde{a}}_i^k + \mathbf{\tilde{e}} \approx \mathbf{b}_i - \mathbf{\tilde{a}}_i^k
\]

Solve the least-squares problem
\[
\begin{bmatrix}
\ldots & \ldots & | & \mathbf{\tilde{a}}_i^k & \mathbf{\tilde{e}} & \approx & \ldots & \mathbf{b}_i - \mathbf{\tilde{a}}_i^k
\end{bmatrix}
\]

Now set \( \Delta \mathbf{F} = [\mathbf{\Delta R(\mathbf{\tilde{a}}_i^k, \mathbf{\tilde{e}})]} \)
Direct Techniques to solve for R


Step 1: Compute
\[
H = \sum_i \begin{bmatrix}
\tilde{a}_{i,x} & \tilde{a}_{i,y} & \tilde{a}_{i,z} \\
\tilde{b}_{i,x} & \tilde{b}_{i,y} & \tilde{b}_{i,z} \\
\tilde{c}_{i,x} & \tilde{c}_{i,y} & \tilde{c}_{i,z}
\end{bmatrix}
\]

Step 2: Compute the SVD of \( H = USV^t \)
Step 3: \( R = VU^t \)
Step 4: Verify \( Det(R) = 1 \). If not, then algorithm may fail.

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

Quarternion Technique to solve for R

- 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:  \[ q = [q_0, q_1, q_2, q_3] \]

Scalar & vector:  \[ q = s + \bar{v} = [s, \bar{v}] \]

Complex number:  \[ q = q_0 + q_1i + q_2j + q_3k \]

where  \( i^2 = j^2 = k^2 = ijk = -1 \)

Properties:

Linearity:  \[ \lambda q_1 + \mu q_2 = [\lambda s_1 + \mu s_2, \lambda \bar{v}_1 + \mu \bar{v}_2] \]

Conjugate:  \[ q^* = s - \bar{v} = [s, -\bar{v}] \]

Product:  \[ q_1 \circ q_2 = [s_1 s_2 - \bar{v}_1 \cdot \bar{v}_2, s_1 \bar{v}_2 + s_2 \bar{v}_1 + \bar{v}_1 \times \bar{v}_2] \]

Transform vector:  \[ q \circ \bar{p} = q \circ [0, \bar{p}] \circ q^* \]

Norm:  \[ \|q\| = \sqrt{s^2 + \bar{v} \cdot \bar{v}} = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2} \]

Digression continued: unit quaternions

We can associate a rotation by angle \( \theta \) about an axis \( \bar{n} \) with the unit quaternion:

\[ Rot(\bar{n}, \theta) \Leftrightarrow \begin{bmatrix} \cos \frac{\theta}{2}, \sin \frac{\theta}{2} \bar{n} \end{bmatrix} \]

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

\[ Rot((\bar{n}, \theta) \circ \bar{p} = \begin{bmatrix} \cos \frac{\theta}{2}, \sin \frac{\theta}{2} \bar{n} \end{bmatrix} \circ [0, \bar{p}] \circ \begin{bmatrix} \cos \frac{\theta}{2}, -\sin \frac{\theta}{2} \bar{n} \end{bmatrix} \]

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

\[ q = [q_0, q_1, q_2, q_3]; \quad \|q\| = 1 \]

\[
R(q) = \begin{bmatrix}
q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 - q_0q_3) & 2(q_3q_0 + q_1q_2) \\
2(q_1q_2 + q_0q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2q_3 - q_0q_1) \\
2(q_3q_0 - q_1q_2) & 2(q_2q_3 + q_0q_1) & q_0^2 - q_1^2 - q_2^2 + q_3^2
\end{bmatrix}
\]
Unit quaternion from rotation matrix

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

\[
a_0 = 1 + r_{xx} + r_{yy} + r_{zz}; \quad a_1 = 1 + r_{xx} - r_{yy} - r_{zz}
\]

\[
a_2 = 1 - r_{xx} + r_{yy} - r_{zz}; \quad a_3 = 1 - r_{xx} - r_{yy} + r_{zz}
\]

\[
a_0 = \max \{a_0, a_1, a_2, a_3\}
\]

\[
q_0 = \frac{\sqrt{a_0}}{2}
\]

\[
q_1 = \frac{r_{yx} - r_{xy}}{4q_0} \quad q_2 = \frac{r_{xy} + r_{yx}}{4q_1} \quad q_3 = \frac{\sqrt{a_3}}{2}
\]

Rotation axis and angle from rotation matrix

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

\[
[n, \theta] \leftarrow \text{ExtractAxisAngle}(R)
\]

\{
    [s, \vec{v}] \leftarrow \text{ConvertToQuaterrion}(R)
    return(\vec{\vec{v}} / \|\vec{v}\|, 2\tan(s / \|\vec{v}\|)
\}
**Quaternion method for R**

Step 1: Compute

\[ H = \sum_q \begin{bmatrix} \tilde{a}_q \tilde{b}_x & \tilde{a}_q \tilde{b}_y & \tilde{a}_q \tilde{b}_z \\ \tilde{a}_q \tilde{b}_y & \tilde{a}_q \tilde{b}_x & \tilde{a}_q \tilde{b}_z \\ \tilde{a}_q \tilde{b}_z & \tilde{a}_q \tilde{b}_y & \tilde{a}_q \tilde{b}_x \end{bmatrix} \]

Step 2: Compute

\[ G = \begin{bmatrix} \text{trace}(H) & \Delta^T \\ \Delta & H + H^T - \text{trace}(H)I \end{bmatrix} \]

where \( \Delta^T = \begin{bmatrix} H_{2,3} - H_{3,2} & H_{3,1} - H_{1,3} & H_{1,2} - H_{2,1} \end{bmatrix} \)

Step 3: Compute eigenvalue decomposition of \( G \)

\[ \text{diag}(\tilde{\lambda}) = Q^T GQ \]

Step 4: The eigenvector \( Q_+ = [q_0, q_1, q_2, q_3] \) corresponding to the largest eigenvalue \( \tilde{\lambda}_+ \) is a unit quaternion corresponding to the rotation.

---

**Another Quaternion Method for R**

Let \( q = s + \tilde{v} \) be the unit quaternion corresponding to \( R \). Let \( \tilde{a} \) and \( \tilde{b} \) be vectors with \( \tilde{b} = R \cdot \tilde{a} \) then we have the quaternion equation

\[(s + \tilde{v}) \cdot (0 + \tilde{a}) = 0 + \tilde{b}\]

\[(s + \tilde{v}) \cdot (0 + \tilde{a}) = (0 + \tilde{b}) \cdot (s + \tilde{v}) \quad \text{since} \quad (s - \tilde{v})(s + \tilde{v}) = 1 + \tilde{0}\]

Expanding the scalar and vector parts gives

\[-\vec{v} \cdot \tilde{a} = -\tilde{v} \cdot \tilde{b}\]

\[s \tilde{a} + \tilde{v} \times \tilde{a} = s \tilde{b} + \tilde{b} \times \tilde{v}\]

Rearranging ...

\[(\tilde{b} - \tilde{a}) \cdot \tilde{v} = 0\]

\[s(\tilde{b} - \tilde{a}) + (\tilde{b} + \tilde{a}) \times \tilde{v} = \tilde{0}_3\]

**NOTE:** This method works for any set of vectors \( \tilde{a} \) and \( \tilde{b} \). We are using the symbols \( \tilde{a} \) and \( \tilde{b} \) to maintain consistency with the discussion of the previous method.
Another Quaternion Method for R

Expressing this as a matrix equation

\[
\begin{bmatrix}
0 & (\dot{b} - \dot{a})^T \\
\frac{\dot{b} - \dot{a}}{sk(b + \dot{a})} & s \\
\end{bmatrix}
\begin{bmatrix}
s \\ v
\end{bmatrix}
= \begin{bmatrix}
0 \\
0
\end{bmatrix}
\]

If we now express the quaternion \( q \) as a 4-vector \( \bar{q} = [s, v]^T \), we can express the rotation problem as the constrained linear system

\[ M(\bar{a}, \bar{b})\bar{q} = \bar{0}_4 \]

\[ \|\bar{q}\| = 1 \]

Another Quaternion Method for R

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

\[ \min \|M\bar{q}\| \text{ subject to } \|\bar{q}\| = 1 \]

where

\[ M = \begin{bmatrix}
M(\bar{a}_1, \bar{b}_1) \\
\vdots \\
M(\bar{a}_n, \bar{b}_n)
\end{bmatrix} \]

and \( n \) is the number of observations

Taking the singular value decomposition of \( M = \Sigma \Sigma^T \) reduces this to the easier problem

\[ \min \|\Sigma \Sigma^T \bar{q} x\| = \|\Sigma \bar{y}\| \text{ subject to } \|\bar{y}\| = \|\Sigma^T \bar{q} = \|\bar{q}\| = 1 \]
Another Quaternion Method for R

This problem is just

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

where $\sigma_i$ are the singular values. Recall that SVD routines typically return the $\sigma_i \geq 0$ and sorted in decreasing magnitude. So $\sigma_4$ is the smallest singular value and the value of $\hat{y}$ with $\| \hat{y} \| = 1$ that minimizes $\| \Sigma \hat{y} \|$ is $\hat{y} = [0,0,0,1]^T$. The corresponding value of $\hat{q}$ is given by $\hat{q} = V \hat{y} = \mathbf{V}_4$. Where $\mathbf{V}_4$ is the 4th column of $\mathbf{V}$.

Non-reflective spatial similarity (rigid+scale)

$$\sigma \mathbf{R} \cdot \hat{a}_k + \hat{p} = \hat{b}_k$$
Non-reflective spatial similarity

Step 1: Compute

\[ \bar{a} = \frac{1}{N} \sum_{i} a_i \]
\[ \bar{b} = \frac{1}{N} \sum_{i} b_i \]
\[ \tilde{a}_i = a_i - \bar{a} \]
\[ \tilde{b}_i = b_i - \bar{b} \]

Step 2: Estimate scale

\[ \sigma = \frac{\sum ||\tilde{b}_i||}{\sum ||\tilde{a}_i||} \]

Step 3: Find \( R \) that minimizes

\[ \sum_i (R (\sigma \tilde{a}_i) - \tilde{b}_i)^2 \]

Step 4: Find \( \tilde{p} \)

\[ \tilde{p} = \bar{b} - R \cdot \bar{a} \]

Step 5: Desired transformation is

\[ F = \text{SimilarityFrame}(\sigma, R, \tilde{p}) \]

Registration from line pairs

Approach 1:

Compute \( F_a = [R_a, \tilde{c}_a] \) from line pair \( a \)

Compute \( F_b = [R_b, \tilde{c}_b] \) from line pair \( b \)

\[ F_{ab} = F_b^{-1} F_a \]
Registration from line pairs

\[ \mathbf{R}_a = \begin{bmatrix} \mathbf{x}_1 = \mathbf{n}_{a,1} \cdot \hat{\mathbf{c}}_a \\ \mathbf{y}_1 = \frac{\mathbf{n}_{a,1} \times \mathbf{n}_{a,2}}{||\mathbf{n}_{a,1} \times \mathbf{n}_{a,2}||} \\ \mathbf{z}_1 = \mathbf{x}_1 \times \mathbf{y}_1 \end{bmatrix} \]

\[ \hat{\mathbf{c}}_a = \text{midpoint between the two lines} \]

To get the midpoint:

Solve

\[ \begin{bmatrix} \mathbf{n}_{a,1} & -\mathbf{n}_{a,2} \end{bmatrix} \begin{bmatrix} \lambda \\ \nu \end{bmatrix} \approx \hat{\mathbf{a}}_2 - \hat{\mathbf{a}}_1 \]

Then

\[ \hat{\mathbf{c}}_a = \frac{(\hat{\mathbf{a}}_1 + \lambda \mathbf{n}_{a,1}) + (\hat{\mathbf{a}}_2 + \nu \mathbf{n}_{a,2})}{2} \]

Distance of a point from a line

\[ d = ||\mathbf{c} - \hat{\mathbf{a}}|| \sin \theta \]

\[ = ||\mathbf{n} \times (\mathbf{c} - \hat{\mathbf{a}})|| \]

So, to find the closest point to multiple lines

\[ \hat{\mathbf{c}} = \arg\min \sum_k d_k^2 \]

Solve this problem in a least squares sense:

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

Equivalently, solve

\[ \mathbf{n}_k \times \hat{\mathbf{c}} \approx \mathbf{n}_k \times \hat{\mathbf{a}}_k \text{ for } k = 1, \ldots, n \]
Registration from multiple corresponding lines

\[ \begin{align*}
\vec{a_1} & \to \vec{b_1} \\
\vec{a_2} & \to \vec{b_2}
\end{align*}\]

\[ \begin{align*}
\vec{n_a,1} & \to \vec{n_b,1} \\
\vec{n_a,2} & \to \vec{n_b,2}
\end{align*}\]

\[ \mathbf{F}_{ab} \]

Approach 2:

1. Solve \( \mathbf{R}_{ab} \vec{n}_{b,k} \approx \vec{n}_{a,k} \) for \( \mathbf{R}_{ab} \)
2. Solve \( \vec{n}_{a,k} \times \vec{c}_a \approx \vec{n}_{a,k} \times \vec{a}_k \) for \( \vec{c}_a \)
3. Solve \( \vec{n}_{b,k} \times \vec{c}_b \approx \vec{n}_{b,k} \times \vec{b}_k \) for \( \vec{c}_b \)
4. \[ \vec{p}_{ab} = \vec{c}_a - \mathbf{R}_{ab} \vec{c}_b \]