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

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

Minimizing Rigid Registration Errors

Step 1: Compute
\[
\vec{\bar{a}} = \frac{1}{N} \sum_{i=1}^{N} \vec{a}_i, \quad \vec{\bar{b}} = \frac{1}{N} \sum_{i=1}^{N} \vec{b}_i
\]
\[
\vec{\tilde{a}}_i = \vec{a}_i - \vec{\bar{a}}, \quad \vec{\tilde{b}}_i = \vec{b}_i - \vec{\bar{b}}
\]
Step 2: Find \( R \) that minimizes
\[
\sum_{i} (R \cdot \vec{\tilde{a}}_i - \vec{\tilde{b}}_i)^2
\]
Step 3: Find \( \vec{\bar{p}} \)
\[
\vec{\bar{p}} = \vec{\bar{b}} - R \cdot \vec{\bar{a}}
\]
Step 4: Desired transformation is
\[ F = Frame(R, \vec{\bar{p}}) \]
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

Rotation Estimation
Rotation Estimation
Point cloud to point cloud registration

Solving for $R$: iteration method

Given $\{\cdots (\mathbf{a}_i, \mathbf{b}_i), \cdots \}$, want to find

$$R = \arg \min \sum_i \| \mathbf{R}\mathbf{a}_i - \mathbf{b}_i \|^2$$

Step 0: Make an initial guess $R_0$

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

Step 2: Compute $\Delta R$ that minimizes

$$\sum_j (\Delta R \cdot \mathbf{a}_j - \mathbf{b}_j)^2$$

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 \( \mathbf{R}_0 \) to

\[
\mathbf{R}_0 \cdot [\cdots \mathbf{a}_j \cdots] \approx [\cdots \tilde{\mathbf{b}}_j \cdots]
\]

One way to do this is as follows. Form matrices

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

Solve least-squares problem \( \mathbf{M}_{3 \times 3} \mathbf{A}_{3 \times N} \approx \mathbf{B}_{3 \times N} \)

**Note**: You may find it easier to solve \( \mathbf{A}^{T}_{3 \times N} \mathbf{M}^{T}_{3 \times 3} \approx \mathbf{B}^{T}_{3 \times N} \)

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

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

Iterative method: Solving for \( \Delta \mathbf{R} \)

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

\[
\Delta \mathbf{R} \cdot \mathbf{v} \approx \mathbf{v} + \vec{\alpha} \times \mathbf{v}
\]

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

\[
\min_{\Delta \mathbf{R}} \sum_j (\Delta \mathbf{R} \cdot \tilde{\mathbf{a}}_j - \tilde{\mathbf{b}}_j)^2 \approx \min_{\vec{\alpha}} \sum_j (\tilde{\mathbf{a}}_j - \tilde{\mathbf{b}}_j + \vec{\alpha} \times \tilde{\mathbf{a}}_j)^2
\]

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 \( \{..., (\mathbf{a}_i, \mathbf{b}_i), ...\} \), 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 \( \bar{\mathbf{a}}^k \mathbf{a}_i \)

Step 2: Compute \( \Delta \mathbf{F} \) that minimizes
\[
\sum_i \| \Delta \mathbf{F}\bar{\mathbf{a}}^k_i - \mathbf{b}_i \|^2
\]

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

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 \( \Delta \mathbf{F} = \arg\min \sum_i \| \Delta \mathbf{F}\bar{\mathbf{a}}^k_i - \mathbf{b}_i \|^2 \)

\[
\Delta \mathbf{F}\bar{\mathbf{a}}^k_i - \mathbf{b}_i \approx \bar{\alpha}_i \bar{\mathbf{a}}^k_i + \bar{\varepsilon}_i + \bar{\mathbf{a}}^k_i - \mathbf{b}_i
\]

\[
\bar{\alpha}_i \bar{\mathbf{a}}^k_i + \bar{\varepsilon}_i \approx \mathbf{b}_i - \bar{\mathbf{a}}^k_i
\]

\[
sk(-\bar{\mathbf{a}}^k) \bar{\alpha}_i + \bar{\varepsilon}_i \approx \mathbf{b}_i - \bar{\mathbf{a}}^k_i
\]

Solve the least-squares problem
\[
\begin{bmatrix}
\vdots & \vdots & \vdots \\
sk(-\bar{\mathbf{a}}^k) & I & \bar{\alpha}_i \\
\vdots & \vdots & \vdots \\
\end{bmatrix}
\approx
\begin{bmatrix}
\vdots \\
\bar{\varepsilon}_i \\
\vdots \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
\mathbf{b}_i - \bar{\mathbf{a}}^k_i \\
\vdots \\
\end{bmatrix}
\]

Now set \( \Delta \mathbf{F} = [\Delta \mathbf{R}(\bar{\alpha}), \bar{\varepsilon}] \)
Direct Techniques to solve for $R$


Step 1: Compute

$$H = \sum_i \begin{bmatrix} \tilde{a}_{ix} \tilde{b}_{ix} & \tilde{a}_{iy} \tilde{b}_{iy} & \tilde{a}_{iz} \tilde{b}_{iz} \\ \tilde{a}_{ix} \tilde{b}_{ix} & \tilde{a}_{iy} \tilde{b}_{iy} & \tilde{a}_{iz} \tilde{b}_{iz} \\ \tilde{a}_{ix} \tilde{b}_{ix} & \tilde{a}_{iy} \tilde{b}_{iy} & \tilde{a}_{iz} \tilde{b}_{iz} \end{bmatrix}$$

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: \( q = s + \vec{v} = [s, \vec{v}] \)

Complex number: \( q = q_0 + q_1i + q_2j + q_3k \)

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

Properties:

- Linearity: \( \lambda q_1 + \mu q_2 = [\lambda s_1 + \mu s_2, \lambda \vec{v}_1 + \mu \vec{v}_2] \)
- Conjugate: \( q^* = s - \vec{v} = [s, -\vec{v}] \)
- Product: \( q_1 \circ q_2 = [s_1 s_2 - \vec{v}_1 \cdot \vec{v}_2, s_1 \vec{v}_2 + s_2 \vec{v}_1 + \vec{v}_1 \times \vec{v}_2] \)
- Transform vector: \( q \circ \vec{p} = q \circ [0, \vec{p}] \circ q^* \)
- Norm: \( \|q\| = \sqrt{s^2 + \vec{v} \cdot \vec{v}} = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2} \)

Digression continued: unit quaternions

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

\[ \text{Rot}(\vec{n}, \theta) \leftrightarrow \begin{bmatrix} \cos \frac{\theta}{2}, \sin \frac{\theta}{2} \vec{n} \end{bmatrix} \]

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

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

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

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

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

\[ R(q) = \begin{bmatrix}
q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 - q_0q_3) & 2(q_0q_3 + q_1q_2) \\
2(q_1q_2 + q_0q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2q_3 - q_0q_1) \\
2(q_1q_3 - q_0q_2) & 2(q_2q_3 + q_0q_1) & q_0^2 - q_1^2 - q_2^2 + q_3^2
\end{bmatrix} \]
Unit quaternion from rotation matrix

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

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

\[
a_0 = \max \{a_0\} \quad a_1 = \max \{a_1\} \quad a_2 = \max \{a_2\} \quad a_3 = \max \{a_3\}.
\]

\[
q_0 = \frac{\sqrt{a_0}}{2} \quad q_1 = \frac{r_{yx} - r_{zy}}{4q_0} \quad q_2 = \frac{r_{xy} + r_{yx}}{4q_0} \quad q_3 = \frac{r_{xy} - r_{yx}}{4q_0}
\]

Rotation axis and angle from rotation matrix

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

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

\{
    [s, \hat{v}] \leftarrow \text{ConvertToQuat(}\hat{v}\text{)}
    \text{return}([\hat{v} / \|\hat{v}\|, 2\tan(s / \|\hat{v}\|)]
\}
Quatetion method for R

Step 1: Compute

\[
H = \sum a_i b_i^T + \sum a_i b_i^T
\]

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 \(G\)

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

Step 4: The eigenvector \(Q_i = [q_0, q_1, q_2, q_3]\) corresponding to the largest eigenvalue \(\lambda_i\) is a unit quaternion corresponding to the rotation.

Another Quaternion Method for R

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

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

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

Expanding the scalar and vector parts gives

\[
-s \cdot \mathbf{a} = -\mathbf{v} \cdot \mathbf{b}
\]

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

Rearranging ...

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

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

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

Expressing this as a matrix equation

\[
\begin{bmatrix}
0 & (\hat{b} - \hat{a})^T \\
(\hat{b} - \hat{a}) & sk (\hat{b} + \hat{a})
\end{bmatrix}
\begin{bmatrix}
s \\
v
\end{bmatrix} =
\begin{bmatrix}
0 \\
0_3
\end{bmatrix}
\]

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

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

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

Another Quaternion Method for R

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

\[
\min ||\mathbf{M}q|| \quad \text{subject to } ||\tilde{q}|| = 1
\]

where

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

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

\[
\min ||\mathbf{U} \Sigma \mathbf{V}^T \tilde{q}_x|| \quad ||\mathbf{U} \Sigma \tilde{y}|| \quad \text{subject to } ||\tilde{y}|| = ||\mathbf{V}^T \tilde{q}|| = ||\tilde{q}|| = 1
\]
Non-reflective spatial similarity (rigid+scale)

\[ \sigma R \cdot \tilde{a}_k + \tilde{p} = \tilde{b}_k \]

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 \( \tilde{y} \) with \( ||\tilde{y}|| = 1 \) subject to \( ||\tilde{y}|| = 1 \).

The corresponding value of \( q \) is given by \( q = V \tilde{y} \). Where \( V \) is the 4th column of \( V \).
Non-reflective spatial similarity

Step 1: Compute

\[ \bar{a} = \frac{1}{N} \sum_{j=1}^{N} \bar{a}_j \]
\[ \bar{b} = \frac{1}{N} \sum_{j=1}^{N} \bar{b}_j \]
\[ \bar{a}_j = a_j - \bar{a} \]
\[ \bar{b}_j = b_j - \bar{b} \]

Step 2: Estimate scale

\[ \sigma = \frac{\sum_{j} ||\bar{b}_j||}{\sum_{j} ||\bar{a}_j||} \]

Step 3: Find \( R \) that minimizes

\[ \sum_i (R \cdot (\sigma \bar{a}_i) - \bar{b}_i)^T \]

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

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

Step 5: Desired transformation is

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

---

Registration from line pairs

Approach 1:

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

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

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

\[
\begin{align*}
R_a &= \begin{bmatrix} \mathbf{x}_1 &= \mathbf{n}_{a,1} \\
\mathbf{y}_1 &= \mathbf{n}_{a,1} \times \mathbf{n}_{a,2} \\
\mathbf{z}_1 &= \mathbf{x}_1 \times \mathbf{y}_1 \end{bmatrix} \\
\mathbf{c}_a &= \text{midpoint between the two lines}
\end{align*}
\]

To get the midpoint:

\[
\begin{align*}
\text{Solve} & \quad \begin{bmatrix} \mathbf{n}_{a,1} \\
- \mathbf{n}_{a,2} \end{bmatrix} \begin{bmatrix} \lambda \\
\nu \end{bmatrix} \approx \begin{bmatrix} \mathbf{a}_2 - \mathbf{a}_1 \end{bmatrix} \\
\text{Then} & \quad \mathbf{c}_a = \frac{\left( \mathbf{a}_1 + \lambda \mathbf{n}_{a,1} \right) + \left( \mathbf{a}_2 + \nu \mathbf{n}_{a,2} \right)}{2}
\end{align*}
\]

Distance of a point from a line

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

So, to find the closest point to multiple lines

\[
\mathbf{c} = \text{argmin} \sum_k d_k^2
\]

Solve this problem in a least squares sense:

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

Equivalently, solve

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

\[
\begin{align*}
\vec{n}_{a,1} & \quad \vec{n}_{a,2} \\
\vec{a}_1 & \quad \vec{a}_2 \\
\vec{c}_a & \quad \vec{c}_b \\
\end{align*}
\]

\[
\begin{align*}
\vec{n}_{b,1} & \quad \vec{n}_{b,2} \\
\vec{b}_1 & \quad \vec{b}_2 \\
\vec{c}_b & \quad \vec{c}_a \\
\end{align*}
\]

\[
F_{ab}
\]

Approach 2:

Solve \( R_{ab} \vec{n}_{b,k} \approx \vec{n}_{a,k} \) for \( R_{ab} \)

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

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

\[
\vec{p}_{ab} = \vec{c}_a - R_{ab} \vec{c}_b
\]