Rigid 3D Point Registration via RANSAC Initialization and Trimmed Point-to-Point ICP

1. Inputs

Point sets. You start with two unordered sets of 3D points. The moving set is the source $\{\mathbf{p}_i\}_{i=1}^{N_s}$ and the fixed set is the target $\{\mathbf{q}_j\}_{j=1}^{N_t}$. Each point is a 3-vector, so $\mathbf{p}_i \in \mathbb{R}^3$ and $\mathbf{q}_j \in \mathbb{R}^3$. No ordering is assumed. A point can appear anywhere in the list. The method relies on geometry, not index alignment, unless you explicitly provide correspondence candidates for the robust stage.

Initial pose (optional). Some runs begin with a guess for the rigid pose. That guess is a rotation ($\mathbf{R}_0$) and a translation ($\mathbf{t}_0$). This guess matters because the local refinement step behaves like a hill-climber. If you start far away, nearest-neighbor matches can be wrong, which pushes the update in the wrong direction. The robust stage can generate a usable start when such a guess is not available.

Tuning parameters. The local refinement uses a maximum iteration count (K), a stopping tolerance (ε), and a trimming ratio (ρ). The robust stage uses an inlier threshold (τ), an iteration count (M), and a random seed. A correspondence cap ($N_{\max}$) is sometimes used for the nearest-neighbor route so you do not carry every pair forward.

Correspondence candidates (optional). The robust stage can accept a list of candidate matched pairs $\{(\mathbf{a}_k, \mathbf{b}_k)\}_{k=1}^{N_c}$. Here $\mathbf{a}_k$ is a source-side point and $\mathbf{b}_k$ is a target-side point that you believe should match. These can come from descriptors, tracking, or any upstream matching method. They can be noisy and can include many wrong matches. The robust stage is designed to survive that.

2. Outputs

Estimated rigid transform. The primary output is a rigid transform $(\widehat{\mathbf{R}}, \widehat{\mathbf{t}})$. The intent is that when you apply it to source points, they land on the target geometry. In formulas, the predicted aligned point is $\widehat{\mathbf{R}}\mathbf{p} + \widehat{\mathbf{t}}$. If alignment is good, that predicted point lies near a corresponding target point.

Fit diagnostics (common). The local refinement naturally produces an error score, which is reported as an RMSE. The robust stage naturally produces an inlier count and an inlier mask. These numbers matter because rigid registration is often used inside larger pipelines. You want to know whether the output is trustworthy, and these diagnostics are simple evidence.

3. Algorithm Pseudocode

Step 1 — Start state. Input: source points, target points, optional initial pose. Output: current pose $(\mathbf{R}, \mathbf{t})$ used by Step 2.

Step 2 — Apply current pose to the moving set. Input: current pose from (Step 1), source points. Output: transformed source points used by Step 3.

Step 3 — Propose matches by proximity. Input: transformed source points from (Step 2), target points. Output: nearest-neighbor indices and distances used by Step 4 and Step 6.

Step 4 — Keep only stable pairs. Input: distances from (Step 3), trimming ratio. Output: selected indices used by Step 5 and Step 6.

Step 5 — Solve a best rigid update from selected pairs. Input: paired points built from (Step 3) and (Step 4). Output: incremental pose update used by Step 6.

Step 6 — Update pose and decide whether to stop. Input: old pose from (Step 1), update from (Step 5), distances from (Step 3) restricted by (Step 4). Output: new pose fed back to Step 2, plus final pose when stopping.

Step 7 — Robust initializer from candidate matches (optional route). Input: candidate pairs, threshold, iteration cap. Output: robust initial pose used by Step 1.

Step 8 — Pose comparison (optional reporting). Input: estimated pose, reference pose if available. Output: rotation-angle and translation error.

Step 1 — Start state

Rigid motion model $(\mathbf{R}, \mathbf{t})$

$$\mathbf{x}' = \mathbf{R}\mathbf{x} + \mathbf{t} \tag{1}$$

This is the core geometric assumption. The rotation $\mathbf{R}$ belongs to the special orthogonal group $SO(3)$, meaning it preserves lengths and angles and has determinant $+1$. The translation $\mathbf{t}$ shifts every point by the same amount. The combination expresses a shared shape under a changed pose. This is appropriate when the two point sets come from the same rigid object or two scans of a static scene under sensor motion. It becomes unreliable for deformation, rolling-shutter distortions, or mixtures of different moving objects.

Composition of poses

$$(\mathbf{R}_A, \mathbf{t}_A) \circ (\mathbf{R}_B, \mathbf{t}_B) = (\mathbf{R}_A\mathbf{R}_B,\ \mathbf{R}_A\mathbf{t}_B + \mathbf{t}_A) \tag{2}$$

This equation explains how incremental updates accumulate. If you apply $(\mathbf{R}_B, \mathbf{t}_B)$ first and then $(\mathbf{R}_A, \mathbf{t}_A)$, the net rotation is the matrix product. The net translation is the rotated inner translation plus the outer translation. This mechanism is how iterative refinement walks from an initial guess to the final estimate. Each iteration produces a small update, and composition keeps those updates consistent under the correct frame ordering.

Inverse of a pose

$$(\mathbf{R}, \mathbf{t})^{-1} = \left(\mathbf{R}^\top,\ -\mathbf{R}^\top \mathbf{t}\right) \tag{3}$$

The inverse exists because $\mathbf{R}$ is orthonormal. Transpose equals inverse for rotations, which makes the rotation part straightforward. The translation part is subtler. Reversing a translation requires subtracting it, and because the inverse lives in the opposite frame, that subtraction must be expressed in the inverse-rotated basis. This identity is central for error reporting because the meaningful pose discrepancy is a relative transform.

Step 2 — Apply current pose to the moving set

Transform each source point

$$\mathbf{p}_i^{(t)} = \mathbf{R}\mathbf{p}_i + \mathbf{t} \tag{4}$$

This is the rigid model applied point-by-point. The transformed set $\{\mathbf{p}_i^{(t)}\}$ is the version of the source compared to the target at the current iteration. If the current pose is close to correct, most transformed source points land near target points, which makes correspondence selection stable. If the pose is far off, distances are large, nearest-neighbor assignments become unreliable, and the update can drift toward an incorrect local optimum.

Step 3 — Propose matches by proximity

Nearest-neighbor assignment

$$j^*(i) = \arg\min_{j \in \{1,\dots,N_t\}} \left\| \mathbf{p}_i^{(t)} - \mathbf{q}_j \right\|_2 \tag{5}$$

This step converts two unordered sets into provisional pairs. For each transformed source point, you choose the closest target point in Euclidean distance. This assumes that, under a reasonable pose, the true counterpart tends to be closest. When overlap is strong and the sampling density is adequate, this assumption holds well. When overlap is partial or geometry repeats, the nearest target can be a different physical point, which injects wrong constraints into the next pose update.

Distance of each proposed match

$$d_i = \left\| \mathbf{p}_i^{(t)} - \mathbf{q}_{j^*(i)} \right\|_2 \tag{6}$$

The distances are not only a score. They become the raw evidence used for robustness. Large values often correspond to outliers, missing overlap, or clutter points. Small values are more likely to reflect plausible matches. Later stages use these distances to decide which pairs should influence the rigid update and which should be ignored.

Step 4 — Keep only stable pairs

Trimming count

$$K_{\text{keep}} = \lceil (1 - \rho) N_s \rceil \tag{7}$$

The trimming ratio ρ provides a simple robustness mechanism inside the refinement loop. If $\rho = 0.3$, the worst 30% of matches, as measured by distance, are discarded for the next fit. This is

useful when the source includes outliers or when overlap is incomplete. It is also useful early in the loop when nearest-neighbor assignments are still settling. Trimming limits the influence of the noisiest constraints on the rigid update.

Selection of indices

$$\mathcal{I} = \text{indices of the } K_{\text{keep}} \text{ smallest values in } \{d_i\} \tag{8}$$

This is a ranking and selection operation. You sort by distance and keep the smallest set. The kept index set $\mathcal{I}$ defines the active constraints that will shape the next rigid update. Trimming is heuristic, so it can still fail when the current pose is extremely wrong. In that case even the smallest distances can correspond to wrong structure, especially for repetitive shapes or sparse targets.

Step 5 — Solve a best rigid update from selected pairs

Centroids

$$\bar{\mathbf{a}} = \frac{1}{K}\sum_{k=1}^{K}\mathbf{a}_k, \qquad \bar{\mathbf{b}} = \frac{1}{K}\sum_{k=1}^{K}\mathbf{b}_k \tag{9}$$

Centroids separate translation from rotation. Subtracting centroids moves both point sets to be centered at the origin. After centering, the alignment task becomes primarily about matching directions and relative geometry. This improves numerical stability and gives the rotation estimation step a clean objective that does not mix in a constant offset.

Cross-covariance

$$\mathbf{H} = \sum_{k=1}^{K}(\mathbf{a}_k - \bar{\mathbf{a}})(\mathbf{b}_k - \bar{\mathbf{b}})^\top \tag{10}$$

The matrix $\mathbf{H}$ accumulates how source directions correlate with target directions under the current correspondences. Each term is an outer product that contributes a rank-1 vote for how a centered source vector should map toward a centered target vector. Summation aggregates those votes. When correspondences are correct, $\mathbf{H}$ encodes a strong rotational relationship. When correspondences contain many errors, $\mathbf{H}$ becomes noisy and the resulting rotation can drift.

SVD of the cross-covariance

$$\mathbf{H} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top \tag{11}$$

SVD expresses $\mathbf{H}$ as orthonormal bases $\mathbf{U}$ and $\mathbf{V}$ with singular values in $\mathbf{\Sigma}$. Intuitively, $\mathbf{U}$ captures dominant directions of variation on the source side as they appear through the correspondence coupling, and $\mathbf{V}$ does the same for the target side. The singular values encode how strongly each paired direction is supported. This factorization enables a closed-form rotation estimate under an orthonormality constraint.

Optimal rotation

$$\Delta\mathbf{R} = \mathbf{V}\mathbf{U}^\top \tag{12}$$

This is the Kabsch solution. It produces the rotation that best aligns the centered source points to the centered target points in a least-squares sense while preserving orthonormality. One interpretation is that it maximizes $\mathrm{tr}(\Delta\mathbf{R}\mathbf{H})$ subject to $\Delta\mathbf{R} \in SO(3)$. The $\mathbf{V}\mathbf{U}^\top$ form falls out of that constrained optimization. Operationally, it means the rigid update is fast: once correspondences are chosen, the update is a few matrix operations and one SVD.

Reflection correction

$$\det(\Delta\mathbf{R}) = +1 \tag{13}$$

SVD can yield a matrix with determinant -1, which represents a reflection rather than a proper rotation. A reflection can reduce squared error in some cases, which is why it appears, but it is not a valid rigid motion in 3D. The standard correction flips the sign of one singular vector before recomputing $\Delta\mathbf{R}$. This enforces membership in $SO(3)$ and prevents a mirror solution that would break downstream geometry.

Translation after rotation

$$\Delta\mathbf{t} = \bar{\mathbf{b}} - \Delta\mathbf{R}\,\bar{\mathbf{a}} \tag{14}$$

Once the rotation is fixed, the translation that aligns centroids is determined. The term $\Delta\mathbf{R}\,\bar{\mathbf{a}}$ is the rotated source centroid. Subtracting it from the target centroid shifts the rotated source so its center matches the target center. This equation clarifies why centroid computation is performed first. It also emphasizes that translation is not independently optimized here. It is implied by the rotational alignment of centered clouds.

Step 6 — Update pose and decide whether to stop

Pose update

$$\mathbf{R} \leftarrow \Delta\mathbf{R}\mathbf{R}, \qquad \mathbf{t} \leftarrow \Delta\mathbf{R}\mathbf{t} + \Delta\mathbf{t} \tag{15}$$

This is pose composition written as an assignment. The algorithm maintains a pose that maps original source points into the target frame. Each iteration computes $(\Delta\mathbf{R}, \Delta\mathbf{t})$ using points already expressed in that frame, so the update multiplies on the left. The translation update includes a rotation of the prior translation because translations are frame-dependent. This update rule is a common source of bugs if multiplication order is reversed, so keeping the composition equation visible helps.

RMSE on selected correspondences

$$\mathrm{RMSE} = \sqrt{\frac{1}{K}\sum_{i\in\mathcal{I}} d_i^2} \tag{16}$$

This score measures geometric consistency under the current pose using only the trimmed set. Squaring distances makes large residuals dominate, which is why trimming is important. Without trimming, a small number of gross outliers can dominate the objective and pull the rotation in an incorrect direction. RMSE is also a convenient progress signal. Under healthy behavior it decreases monotonically or almost monotonically as correspondences stabilize.

Convergence test

$$|\mathrm{RMSE}_k - \mathrm{RMSE}_{k-1}| < \varepsilon \tag{17}$$

This condition stops refinement when progress becomes smaller than a chosen tolerance. The tolerance ε is in the same units as the points. If the clouds are normalized to a unit box, ε can be small. If the clouds are in meters with typical scales of tens of meters, ε must be scaled accordingly. This criterion prevents the algorithm from spending iterations chasing numerical noise or tiny correspondence flips.

Step 7 — Robust initializer from candidate matches (optional route)

Non-degeneracy of the minimal sample

$$\|(\mathbf{a}_1 - \mathbf{a}_0) \times (\mathbf{a}_2 - \mathbf{a}_0)\|_2 > \delta \tag{18}$$

A rigid transform in 3D can be estimated from three correspondence pairs, but only if the three source points form a triangle with nonzero area. Collinear points do not constrain rotation around the line, which makes the estimate unstable and sensitive to noise. The cross product magnitude equals twice the triangle area. The threshold δ is a small constant used to reject nearly collinear triples so that the SVD fit has a meaningful geometric baseline.

Inlier test for a candidate model

$$\|\mathbf{R}\mathbf{a}_k + \mathbf{t} - \mathbf{b}_k\|_2 < \tau \tag{19}$$

This inequality decides whether a candidate correspondence is consistent with a proposed pose. The threshold τ defines what deviation you are willing to attribute to noise and sampling. If τ is too tight, even correct correspondences fail the test due to noise, and the robust stage struggles to find a large inlier set. If τ is too loose, many wrong matches pass, which makes the inlier count less meaningful and can lead to a poor initialization for the refinement stage.

Refit on inliers

$$(\mathbf{R}, \mathbf{t}) = \arg \min_{\mathbf{R} \in SO(3), \mathbf{t} \in \mathbb{R}^3} \sum_{k \in \mathcal{K}_{\mathrm{in}}} \|\mathbf{R}\mathbf{a}_k + \mathbf{t} - \mathbf{b}_k\|_2^2 \tag{20}$$

This objective is the same least-squares rigid fit solved by the SVD method in Step 5, now restricted to the inlier index set $\mathcal{K}_{\mathrm{in}}$. The rationale is that the first model was fit to only three pairs, which may contain substantial noise. Once a larger inlier set is identified, refitting averages that noise out and yields a more accurate pose. That refined pose is then passed as the initial pose for the refinement loop.

Inlier RMSE (optional diagnostic)

$$\mathrm{RMSE}_{\mathrm{in}} = \sqrt{\frac{1}{n_{\mathrm{in}}} \sum_{k \in \mathcal{K}_{\mathrm{in}}} \|\mathbf{R}\mathbf{a}_k + \mathbf{t} - \mathbf{b}_k\|_2^2} \tag{21}$$

This quantity is a sanity check that complements the inlier count. A very large inlier count with a high inlier RMSE suggests that the threshold is permissive or the correspondences are weak. A moderate inlier count with a low RMSE can still be an excellent seed for refinement because the subsequent nearest-neighbor stage becomes reliable once the pose is in the right basin of attraction.

Step 8 — Pose comparison (optional reporting)

Relative transform between estimate and reference

$$(\mathbf{R}_\Delta, \mathbf{t}_\Delta) = (\widehat{\mathbf{R}}, \widehat{\mathbf{t}})^{-1} \circ (\mathbf{R}_\star, \mathbf{t}_\star) \tag{22}$$

This constructs the transform that maps the estimated pose to the reference pose. If estimate and reference coincide, the relative transform is identity. This is preferable to subtracting translations or comparing matrix entries because it respects the group structure of rigid motions. The inverse and composition operations used here are the ones defined in Step 1.

Rotation angle from the trace

$$\theta = \arccos\left(\mathrm{clip}\left(\frac{\mathrm{tr}(\mathbf{R}_\Delta) - 1}{2}, -1, 1\right)\right) \tag{23}$$

This maps a rotation matrix to a single angle. For a proper rotation, the relationship between trace and angle is exact: $(\mathrm{tr}(\mathbf{R}) - 1)/2 = \cos\theta$. The clipping accounts for floating-point drift that can push the argument slightly outside $[-1, 1]$, which would make arccos undefined. The resulting angle is an intuitive measure of rotational discrepancy that is independent of axis choice.

Translation error norm

$$e_t = \|\mathbf{t}_\Delta\|_2 \tag{24}$$

This compresses the 3D translation discrepancy into a scalar magnitude. Reporting this along with the rotation-angle error gives a compact description of how far the estimated rigid motion is from the reference. In many applications, these two scalars are the primary success metrics used to decide whether the registration is acceptable.

Coverage map (equations tied to computational stages)

Identity pose construction (identity transform constructor (identitySE3)) corresponds to $\mathbf{R} = \mathbf{I}$ and $\mathbf{t} = \mathbf{0}$. Pose composition (compose) corresponds to the composition equation in Step 1. Pose inversion (inverse) corresponds to the inverse equation in Step 1. Point transformation (applyPoint) corresponds to $\mathbf{x}' = \mathbf{R}\mathbf{x} + \mathbf{t}$. Nearest-neighbor pairing (nearestNeighborBruteforce) corresponds to the arg min assignment and distance definitions in Step 3. Trimming selection (trimSelection) corresponds to the K_{keep} and $\mathcal{I}$ definitions in Step 4. SVD rigid fitting (kabschSE3) corresponds to centroid, cross-covariance, SVD, rotation, determinant constraint, and translation equations in Step 5. The iterative refinement loop (icpPointToPoint) corresponds to the pose update, RMSE, and convergence equations in Step 6. Non-degeneracy checking (nondegenerateTriple) corresponds to the cross-product inequality in Step 7. Robust model selection and refit (ransacSE3) corresponds to the inlier test and refit objective in Step 7. Pose error reporting (poseError) corresponds to the relative transform, trace-to-angle mapping, and translation norm in Step 8.