The Trimmed Iterative Closest Point Algorithm

Dmitry Chetverikov and Dmitry Stepanov

http://visual.ipan.sztaki.hu
Contents

- Alignment of two roughly pre-registered 3D point sets
- Iterative Closest Point algorithm (ICP)
- Existing variants of ICP
  - Trimmed Iterative Closest Point algorithm (TrICP)
    - implementation
    - convergence
    - robustness
- Tests
- Future work
Euclidean alignment of two 3D point sets

• Given two roughly pre-registered 3D point sets, $\mathcal{P}$ (data) and $\mathcal{M}$ (model), find shift & rotation that bring $\mathcal{P}$ into best possible alignment with $\mathcal{M}$. ⬛ปลายความ

• Applications:
  ◦ 3D model acquisition – reverse engineering, scene reconstruction
  ◦ motion analysis – model-based tracking

• Problems:
  ◦ partially overlapping point sets – incomplete measurements
  ◦ noisy measurements
  ◦ erroneous measurements – outliers
  ◦ shape defects

Note: We consider Euclidean alignment. Other alignments, e.g., affine, are also studied.
Iterative Closest Point (ICP) algorithm


**Algorithm 1: Iterative Closest Point**

1. Pair each point of $\mathcal{P}$ to closest point in $\mathcal{M}$.
2. Compute motion that minimises mean square error (MSE) between paired points.
3. Apply motion to $\mathcal{P}$ and update MSE.
4. Iterate until convergence.

Chen and Medioni (1992): A similar iterative scheme using different pairing procedure based on surface normal vector. 

We use Besl’s formulation: applicable to volumetric measurements.
Properties of ICP

- **Pre-registration** required:
  - manual
  - known sensor motion between two measurements

- **Point pairing**:
  - computationally demanding
  - special data structures used to speed up (k-D trees, spatial bins)

- **Optimal motion**: closed-form solutions available

- Proved to **converge** to a local minimum

- Applicable to **surface** as well as **volumetric** measurements

- **Drawbacks**:
  - **not robust**: assumes outlier-free data and \( \mathcal{P} \subset \mathcal{M} \)
  - converges quite slowly
Closed-form solutions for optimal rigid motion

- **Unit Quaternions** (Horn 1987): used by original ICP and our TrICP
- **Singular Value Decomposition** (Arun 1987)
- **Orthogonal Matrices** (Horn 1988)
- **Dual Quaternions** (Walker 1991)

### Properties of the methods (Eggert 1997)

<table>
<thead>
<tr>
<th>Method</th>
<th>Accuracy</th>
<th>2D Stability*</th>
<th>Speed, small $N_p$</th>
<th>Speed, large $N_p$</th>
</tr>
</thead>
<tbody>
<tr>
<td>UQ</td>
<td>good</td>
<td>good</td>
<td>fair</td>
<td>fair</td>
</tr>
<tr>
<td>SVD</td>
<td>good</td>
<td>good</td>
<td>fair</td>
<td>fair</td>
</tr>
<tr>
<td>OM</td>
<td>fair</td>
<td>poor</td>
<td>good</td>
<td>poor</td>
</tr>
<tr>
<td>DQ</td>
<td>fair</td>
<td>fair</td>
<td>poor</td>
<td>good</td>
</tr>
</tbody>
</table>

* Stability in presence of degenerate (2D) data.
Existing variants of ICP

Goal: Improve robustness and convergence (speed).

Categorisation criteria (Rusinkiewicz 2001): How variants

1. Select subsets of $\mathcal{P}$ and $\mathcal{M}$
   - random sampling for a Monte-Carlo technique

2. Match (pair) selected points
   - closest point
   - in direction of normal vector: faster convergence when normals are precise

3. Weight and reject pairs
   - distribution of distances between paired points
   - geometric constraints (e.g., compatibility of normal vectors)

4. Assign error metric and minimise it
   - iterative: original ICP
   - direct: Levenberg-Marquardt algorithm
Robustness and convergence: critical issues

ICP assumes that each point of $\mathcal{P}$ has valid correspondence in $\mathcal{M}$. Not applicable to partially overlapping sets or sets containing outliers.

Previous attempts to robustify ICP: Reject wrong pairs based on

- **Statistical criteria.** Monte-Carlo type technique with robust statistics:
  - Least median of squares (LMedS)
  - Least trimmed squares (LTS)

- **Geometric criteria.** For example, Iterative Closest *Reciprocal* Point (Pajdla 1995) uses $\epsilon$-reciprocal correspondence:
  - if point $p \in \mathcal{P}$ has closest point $m \in \mathcal{M}$, then
  - back-project $m$ onto $\mathcal{P}$ by finding closest point $p' \in \mathcal{P}$
  - reject pair $(p, m)$ if $\|p - p'\| > \epsilon$

Heterogeneous algorithms: heuristics combined, convergence cannot be proved.
Robust statistics: LMedS and LTS

Sort distances between paired points, minimise

- **LMedS**: value in the middle of sorted sequence
  - operations **incompatible** with computation of optimal motion

- **LTS**: sum of certain number of least values (e.g., least 50%)
  - operations **compatible** with computation of optimal motion
  - better convergence rate, smoother objective function

Previous use of LMedS and LTS: Randomised robust regression

- estimate optimal motion parameters by repeatedly drawing random samples

- detect and reject outliers, find least squares solution for inliers

Robust to outliers, but breakdown point 50% ⇒ minimum overlap 50%.
Trimmed Iterative Closest Point

Assumptions:

1. 2 sets of 3D points: data set $\mathcal{P} = \{\mathbf{p}_i\}_{1}^{N_p}$ and model set $\mathcal{M} = \{\mathbf{m}_i\}_{1}^{N_m}$. ($N_p \neq N_m$.) Points may be surface as well as volumetric measurements.

2. Minimum guaranteed rate of data points that can be paired is known*: minimum overlap $\xi$. Number of data points that can be paired $N_{po} = \xi N_p$.

3. Rough pre-registration: max initial relative rotation 30°.

4. Overlapping part is characteristic enough to allow for unambiguous matching**:
   - no high symmetry
   - no ‘featureless’ data

* If $\xi$ is unknown, it is set automatically: run TrICP several times, select best result.

** Typical for most registration algorithms.
Problem statement and notation

**Informal statement:** Find Euclidean transformation that brings an $N_{po}$-point subset of $\mathcal{P}$ into best possible alignment with $\mathcal{M}$.

For rotation $\mathbf{R}$ and translation $\mathbf{t}$, transformed points of $\mathcal{P}$ are

$$p_i(\mathbf{R}, \mathbf{t}) = \mathbf{R}p_i + \mathbf{t}, \quad \mathcal{P}(\mathbf{R}, \mathbf{t}) = \{p_i(\mathbf{R}, \mathbf{t})\}_{1}^{N_p}$$

**Individual distance** from data point $p_i(\mathbf{R}, \mathbf{t})$ to $\mathcal{M}$:

$$m_{cl}(i, \mathbf{R}, \mathbf{t}) = \arg\min_{m \in \mathcal{M}} \|m - p_i(\mathbf{R}, \mathbf{t})\|$$

$$d_i(\mathbf{R}, \mathbf{t}) = \|m_{cl}(i, \mathbf{R}, \mathbf{t}) - p_i(\mathbf{R}, \mathbf{t})\|$$

**Formal statement:** Find rigid motion $(\mathbf{R}, \mathbf{t})$ that minimises sum of least $N_{po}$ square distances $d_i^2(\mathbf{R}, \mathbf{t})$.

Conventional ICP: $\xi = 1$ and $N_{po} = N_p$.

TrICP: smooth transition to ICP as $\xi \to 1$. 
Basic idea of TRiCP: Consistent use of LTS in deterministic way.

Start with previous $S'_{TS} = \text{huge}\_\text{number}$. 
Iterate until any of stopping conditions is satisfied.

Algorithm 2: Trimmed Iterative Closest Point

1. **Closest point**: For each point $p_i \in P$, find closest point in $M$ and compute $d_i^2$.

2. **Trimmed Squares**: Sort $d_i^2$, select $N_{po}$ least values and calculate their sum $S_{TS}$.

3. **Convergence test**: If any of stopping conditions is satisfied, exit; otherwise, set $S'_{TS} = S_{TS}$ and continue.

4. **Motion calculation**: For $N_{po}$ selected pairs, compute optimal motion $(R, t)$ that minimises $S_{TS}$.

5. **Data set transformation**: Transform $P$ by $(R, t)$ and go to 1.
Stopping conditions

• Maximum allowed number of iterations $N_{iter}$ has been reached, or

• Trimmed MSE is sufficiently small, or

• Change of Trimmed MSE is sufficiently small.

Trimmed MSE $e$: For sorted distances $d_{s1} \leq d_{s2} \leq \ldots \leq d_{sN_{po}} \leq \ldots \leq d_{sN_p}$,

$$S_{TS} \doteq \sum_{s_i=s1}^{sN_{po}} d_{si}^2$$

$$e \doteq \frac{S_{TS}}{N_{po}}$$

Change of Trimmed MSE: $|S_{TS} - S'_{TS}|$
Implementation details

- Finding closest point: Use **boxing structure** (Chetverikov 1991) that partitions space into uniform boxes, cubes. Update box size as $P$ approaches $M$.
  - simple
  - fast, especially at beginning of iterations
  - uses memory in inefficient way

- Sorting individual distances and calculating LTS: Use **heap sort**.

- Computing optimal motion: Use **Unit Quaternions**.
  - robust to noise
  - stable in presence of degenerate data (‘flat’ point sets)
  - relatively fast
Automatic setting of overlap parameter $\xi$

When $\xi$ is unknown, it is set automatically by minimising objective function

$$\psi(\xi) = \frac{e(\xi)}{\xi^{1+\lambda}}, \quad \lambda = 2$$

- $\psi(\xi)$ minimises trimmed MSE $e(\xi)$ and tries to use as many points as possible.
- Larger $\lambda$: avoid undesirable alignments of symmetric and/or ‘featureless’ parts.
- $\psi(\xi)$ minimised using modified Golden Section Search Algorithm.

Typical shapes of objective functions $e(\xi)$ and $\psi(\xi)$. 
Convergence

**Theorem:** *TrICP always converges monotonically to a local minimum with respect to trimmed MSE objective function.*

Sketch of proof:

- **Optimal motion** does not increase MSE: if it did, it would be inferior to identity transformation, as the latter does not change MSE.

- **Updating the closest points** does not increase MSE: no individual distance increases.

- **Updating the list of $N_{po}$ least distances** does not increase MSE: to enter the list, any new pair has to substitute a pair with larger distance.

- **Sequence of MSE values** is nonincreasing and bounded below (by zero), hence it converges to a local minimum.

**Convergence to global minimum** depends on initial guess.
Aligning two partial measurements of Frog. (≈3000 points)

Numerical results for Frog data

<table>
<thead>
<tr>
<th>Method</th>
<th>$N_{iter}$</th>
<th>MSE</th>
<th>Exec.time*</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICP</td>
<td>45</td>
<td>5.83</td>
<td>7.4 sec</td>
</tr>
<tr>
<td>TrICP 70%</td>
<td>88</td>
<td>0.10</td>
<td>2.5 sec</td>
</tr>
</tbody>
</table>

* On 1.6 GHz PC
Aligning four measurements of Skoda part. ($\approx 6000$ points)
Aligning four measurements of Fiat part. (overlap $\approx 20\%$.)
Aligning two measurements of chimpanzee Skull. (≈100000 points)
Comparing TrICP to ICRP for SQUID database

SQUID (University of Surrey, UK): \textbf{1100 shapes} of different fishes

- $\mathcal{P}$ rotated by known angle. ($1^\circ \ldots 20^\circ$)
- Different parts of $\mathcal{M}$ and $\mathcal{P}$ deleted.
- Noise added to both shapes.

Aligning deteriorated SQUID shapes. Fish: original noise-free shape.
### TrICP/ICRP errors for noisy SQUID data, degrees

<table>
<thead>
<tr>
<th></th>
<th>100%</th>
<th>90%</th>
<th>80%</th>
<th>70%</th>
<th>60%</th>
</tr>
</thead>
<tbody>
<tr>
<td>1°</td>
<td>0.05/0.05</td>
<td>0.08/0.06</td>
<td>0.07/0.08</td>
<td>0.10/0.12</td>
<td>0.19/0.23</td>
</tr>
<tr>
<td>5°</td>
<td>0.05/0.05</td>
<td>0.09/0.07</td>
<td>0.08/0.11</td>
<td>0.12/0.18</td>
<td>0.34/0.31</td>
</tr>
<tr>
<td>10°</td>
<td>0.05/0.05</td>
<td>0.09/0.07</td>
<td>0.10/0.18</td>
<td>0.19/0.60</td>
<td>0.58/1.70</td>
</tr>
<tr>
<td>15°</td>
<td>0.05/0.06</td>
<td>0.11/0.11</td>
<td>0.16/0.36</td>
<td>0.34/1.09</td>
<td>1.14/2.54</td>
</tr>
<tr>
<td>20°</td>
<td>0.05/0.11</td>
<td>0.10/0.16</td>
<td>0.20/0.49</td>
<td>0.69/1.51</td>
<td>1.79/3.03</td>
</tr>
</tbody>
</table>

- **ICRP** is efficient at small rotations and noise-free data.

- **TrICP** is more robust to rotations and incomplete, noisy data. Procedure for automatic setting of overlap available. Convergence proved.

- **Execution times** per alignment are comparable. Skoda 6000 pts: 8.3/7.2 sec, Skull 100000 pts: 38/182 sec.
Future work

- **Compare** to other methods for a large set of shapes.
- To better avoid *local minima*, perturb initial orientation of data set.
- Extension to *multiple point sets* $N > 2$.
- **Faster** operation.
- More efficient *usage of memory*.