Available online at www.sciencedirect.com

ScienceDirect

Procedia CIRP 00 (2024) 000–000

www.elsevier.com/locate/procedia

18th CIRP Conference on Intelligent Computation in Manufacturing Engineering

An approach for precise CAD reconstruction based on TEASER++, iterative ICP methods for robotics applications

Xiaomei Xu[a*], Attique Bashir[a], Rainer Müller[a]

[a]Center for Mechatronics and Automation gGmbH (ZeMA), Eschberger Weg 46, Saarbrücken 66121, Germany
* Corresponding author. Tel.:+49(0)681-85787-523; fax: +49(0)681-85787-11. *E-mail address:* xiaomei.xu@zema.de

Abstract

This paper outlines a systematic approach to improve the precision of 3D reconstructions from blurry point cloud images taken by a 3D camera. The process involves thorough 3D scanning, point cloud image alignment and precise 3D robotic pose estimation. Global registration utilizes TEASER++, and local registration employs the Iterative Closest Point algorithm for rotation and translation estimation. The alignment results are utilized to generate an enhanced CAD model. An accurate 3D robotic pose is determined from the CAD model data. Validation is done using LEGO components, benchmarked against an industrial tool for performance evaluation.

Peer review under the responsibility of the scientific committee of the 18th CIRP Conference on Intelligent Computation in Manufacturing Engineering, 10-12 July, Gulf of Naples, Italy

Keywords: Robotics; CAD Reconstruction; Point Cloud Registration; 3D Pose Estimation; Iterative Closest Point Method and Truncated least square Estimation And Semidefinite Relaxation Method

1. Introduction and the State of the Art

Computer-**a**ided **d**esign (CAD) model reconstruction from point clouds represents a significant challenge at the intersection of computer vision, graphics, and machine learning [1]. In addition, CAD reconstruction technology plays a crucial role in various industrial applications, including grinding [2], additive manufacturing [3], and other reverse engineering technologies [4]. In CAD reconstruction, registration is a crucial step. It involves aligning multiple 3D point cloud views to create a complete model. Registration methods can be broadly categorized into two types: local registration and global registration. Local registration methods are well-suited for aligning partial 3D point-cloud scans at shorter distances. The **I**terative **C**losest **P**oint (ICP) algorithm, a local registration technique, has three variants: point-to-point ICP [5], point-to-plane ICP [6, 7], and coloured point-cloud registration [8]. However, their outcome heavily relies on the initial transformation matrix configuration. In contrast to local registration, the global registration method does not need an initial transformation matrix for aligning point clouds over longer distances. Global registration methods such as **Ran**dom **sa**mple **c**onsensus (RANSAC) [9], Faster global registration [10], and **T**runcated **L**east **S**quares **E**stimation **A**nd **SE**midefinite **R**elaxation (TEASER++) [11] utilize **F**ast **P**oint **F**eature **H**istograms (FPFH) to address the initial transformation setting challenge encountered with the ICP method, resulting in computational time savings. In this paper, the TEASER++ for global registration is improved and afterwards, the iterative ICP method for precise alignment is employed. This gradual synthesis combines two point cloud images with varying deflection angles into a complete point cloud image. It aims to utilize this comprehensive point cloud data for intelligent analysis and further research and development in automated assembly processes.

Sections 2.2 to 2.7 provide a step-by-step introduction to the TEASER++ alignment method. Section 2.8 introduces iterative ICP for fine-tuning the alignment effect, and Section 2.9 covers

Peer-review under responsibility of the scientific committee of the 18th CIRP Conference on Intelligent Computation in Manufacturing Engineering, 10-12 July, Gulf of Naples, Italy

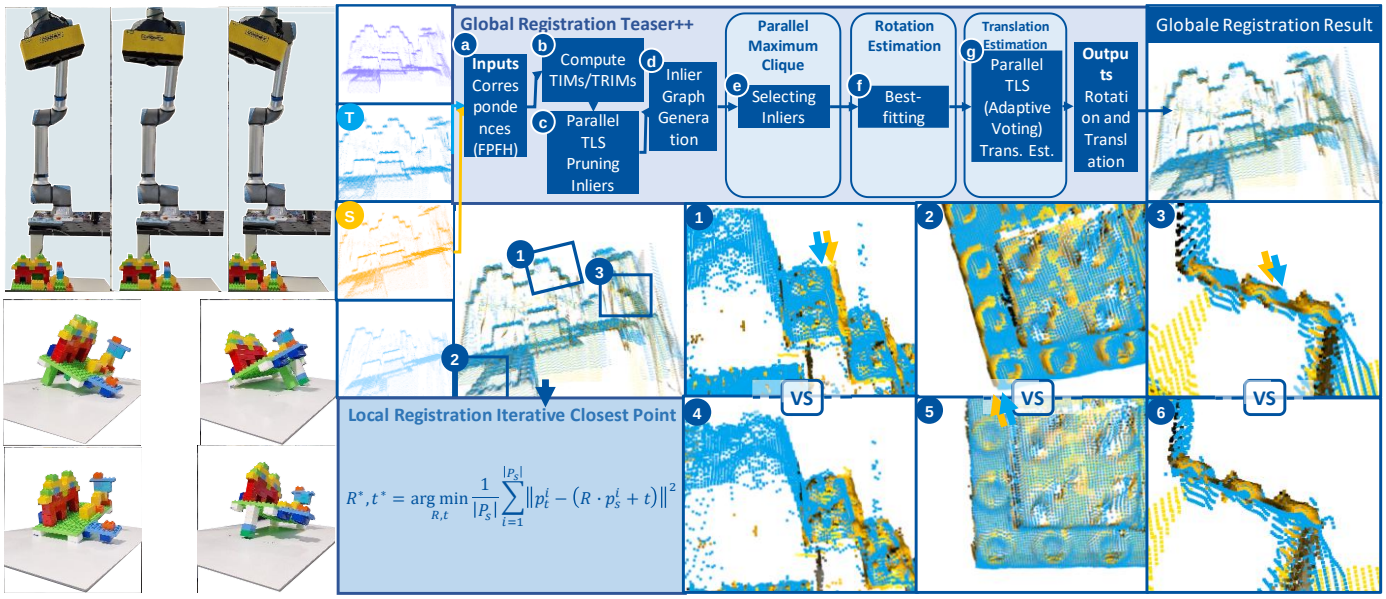

Fig. 1 **T**runcated Least Squares **E**stimation **A**nd **SE**midefinite **R**elaxation (TEASER++) and **I**terative **C**losest **P**oint (ICP) Working Flow

the conversion of camera coordinates to the robot base coordinates system. Section 3 presents the validation results. Section 4 provides a summary covering the content, advantages, limitations, and future work.

2. Solving the CAD Reconstruction Problem

For reconstructing the CAD model a 3D camera mounted on a robot arm captures multiple point clouds from various angles (see Fig. 1Left Side). Two closed point cloud images in terms of angles are fed into the TEASER++ for global registration. TEASER++ identifies initial correspondence pairs between the source and target point clouds primarily using FPFH feature descriptions (see Fig. 1a). The **t**ranslation and **r**otation **i**nvariant **m**easurement**s** (TRIMs) based on the initial correspondence pairs are calculated and limited to the noise bound to initially filter out the inliers, i.e., the correct correspondence pairs (see Fig. 1b). In the context of identifying correct correspondence points, the goal is to select the point that has the most connections to other points (see Fig. 1c). This selected point will then form a set of maximum clique inliers. To achieve this efficiently during the alignment process, the maximum clique algorithm will utilized (see Fig. 1d and Fig. 1e). Registration relies on maximum clique inliers, estimating the rotation matrix using the best-fitting method (see Fig. 1f), and determining translation vectors through an adaptive voting approach (see Fig. 1g). Following the TEASER++ global registration, iterative ICP was employed to enhance alignment accuracy (see Fig. 1①②③), particularly at edges and corners (see Fig. 1④⑤⑥). Finally, the camera coordinates are transformed into the robot base coordinates system to facilitate subsequent assembly operations.

2.1. FPFH Feature Description and Correspondences Estimation

The **F**ast **P**oint **F**eature Histograms is an improvement of the **P**oint-**F**eature **H**istogram (PFH) in the literature [12]. The PFH describes the points by calculating the Euclidean distances and normal vectors of pairs of points between each point of the target point-cloud with a radius of r in the target point cloud and statistically encoding the values of geometric relationships (Darboux frame $\underline{u},\underline{v},\underline{w}$ and angular feature $\alpha,\emptyset,\theta$) in the region of r of the points, to form a multidimensional vector to describe each point (see Fig. 2). [13] is mainly to compute the geometric relationship between each point in the source point-

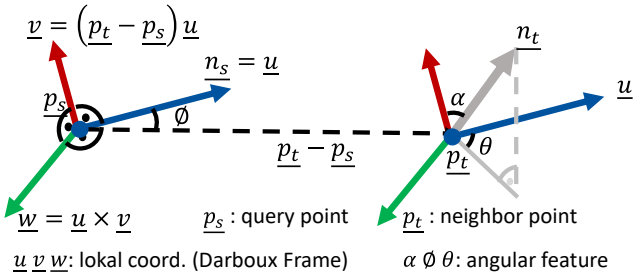

Fig. 2 Fast Point Feature Histogram_Geometric Relationships[14]

cloud and each point in the neighbourhood of the source point-cloud within the target point-cloud with r as the radius and find the points in the r-neighbourhood of the source point-cloud.

A surface's natural moving frame is constructed, known as the Darboux frame. Geometric relationships are defined based on this Darboux frame. Darboux frame origin [14] refers to the local coordinate system p_s:

$$\begin{cases} \underline{u} = \underline{n_s} \\ \underline{v} = \underline{u} \times \frac{(\underline{p_t} - \underline{p_s})}{\|\underline{p_t} - \underline{p_s}\|_2} \\ \underline{w} = \underline{u} \times \underline{v} \end{cases} \tag{1}$$

Using the Darboux frame $\underline{u},\underline{v},\underline{w}$ defines the angular feature at $\underline{p_t}$[14], which is understood as the difference between the two normal $\underline{n_s}$ and $\underline{n_t}$. In [14], a quadruplet denoted as $\langle \alpha, \emptyset, \theta, d \rangle$ is calculated for each pair of two pairs within the $\mathcal{P}^{k_2}$neighborhood:

$$\begin{aligned} \alpha &= \underline{v} \cdot \underline{n_t} \\ \emptyset &= \underline{u} \cdot \frac{(\underline{p_t} - \underline{p_s})}{d} \\ \theta &= arctan\left(\underline{w} \cdot \underline{n_t}, \underline{u} \cdot \underline{n_t}\right) \\ d &= \left\|\underline{p_t} - \underline{p_s}\right\|_2 \end{aligned} \tag{2}$$

This computation effectively reduces the 12 values associated with two points. Equation 2 represents θ the angle of $\underline{n_t}$ in the $\underline{w} \times \underline{v}$ plane concerning $\underline{u}$. The value of $\arctan(\cdot)$ in equation 2 is equal to the angle between the projection of the $\underline{n_t}$ normal vector in the local coordinates defined by $\underline{p_s}$ and the $\underline{u}$-axis defined by $\underline{p_s}$, as well as expressing the degree of rotation of $\underline{n_t}$ in the $\underline{u} \times \underline{w}$ plane equal to the direction angle, and finding a few angles to express the relationship between $\underline{n_t}$ and $\underline{u}\ \underline{v}\ \underline{w}$ is sufficient. d stands for the Euclidean distance between p_s and p_t.

According to the FPFH algorithm, the source point-cloud and target point-cloud are featured. Next, the Euclidean distance between each point of the two featured point clouds is calculated to set the threshold d. Subsequently, the key points are initially selected, and the pairs of points $\underline{c}$ with similar featured points are maintained. Because TEASER++ decouples the estimation of rotation and translation. In the subsequent section, the translation-invariant measurements (TIMs) as an initial step for estimating rotation will be introduced.

2.2. *Translation Invariant Measurements (TIMs)*

TEASER++[11] reformulates the measurements to derive quantities that remain invariant under a subset of transformations, including scaling, rotation, and translation. The corresponding points from the source point cloud and the target point cloud serve as the inputs for independently computing TIMs. To simplify the explanation, the correspondence points with the account N are collectively denoted as a 3xN matrix $\boldsymbol{C}$, where each point is represented by, $\underline{c_2}, \cdots, \underline{c_n}$:

$$\boldsymbol{C} = \left(\underline{c_1}, \underline{c_2}, \cdots, \underline{c_n}\right) \tag{3}$$

All the points within the $\boldsymbol{C}$ matrix must be iterated. The current point is denoted as $\underline{c_i}$. The current point constructs a 3xN actual point matrix C_{actual}, represented as$\left(\underline{c_i}, \underline{c_i}, \cdots, \underline{c_i}\right)$. The 3xN relative position matrix $\boldsymbol{C_{rel}}$, which connects all the correspondence points in matrix $\boldsymbol{C}$ to their corresponding actual points, is computed as follows:

$$\boldsymbol{C_{rel}} = \boldsymbol{C} - \boldsymbol{C_{actual}} \tag{4}$$

$\boldsymbol{V_{TIMs}}$ calculates the pairwise relative measurements between all pairs of points. Specifically, each current point $\underline{c_i}$ is paired with all the other iterative points in the matrix C. The $\boldsymbol{V_{TIMs}}$matrix has dimensions 3x(N·(N-1)/2). The values of the $\boldsymbol{C_{rel}}$ matrix from the i-th item to the last item are appended into the matrix $\boldsymbol{V_{TIMs}}$. After processing the correspondence point matrix separately from both the source point cloud $\boldsymbol{C_{src}}$ and the target point cloud $\boldsymbol{C_{tgt}}$, we obtain the TIMs matrices for each. These matrices are referred to as $\boldsymbol{V_{TIMs\_src}}$ and $\boldsymbol{V_{TIMs\_tgt}}$. Due to the effect of noise, not all estimated correspondence pairs are usually correct, due to the negative impact of incorrect correspondences on the rotation and translation estimation. The process of eliminating incorrect correspondence pairs will be introduced in the next section.

2.3. *Pruning Inlier*

A special condition is needed to exclude the incorrect correspondence pairs and retain the correct ones, also referred to as inliers. A correspondence pair is considered an **inlier** if it satisfies the two criteria.

Before explaining the criteria, the TIM matrices derived from the source $\boldsymbol{V_{TIMs\_src}}$ and target point clouds $\boldsymbol{V_{TIMs\_tgt}}$ are utilized to calculate the distance vectors $\underline{a_{ij}}$ (source) and $\underline{b_{ij}}$(target), which is based on the camera coordinate system. Equation 5 uniformly denotes distance vectors as $\underline{d}$.

$$\underline{d} = \left(\sqrt{{x_1}^2 + {y_1}^2 + {z_1}^2} \quad , \ldots, \quad \sqrt{{x_N}^2 + {y_N}^2 + {z_N}^2}\right) \tag{5}$$

In [11], the generative model involves **t**ranslation and **r**otation **i**nvariant **m**easurement**s** (TRIMs) represented by the norm of TIMs vectors. These TRIMs correspond to linear scalar measurements $\underline{s_{ij\_forward}}$ and $\underline{s_{ij\_reverse}}$, which are influenced by bounded noise $\underline{\alpha_{ij}}$(source) and $\underline{\beta_{ij}}$(target).

$$\underline{\alpha_{ij}} = \frac{\delta_{ij}}{\|\underline{a_{ij}}\|} \leq \frac{\beta_i + \beta_j}{\|\underline{a_{ij}}\|}; \quad \underline{\beta_{ij}} = \frac{\delta_{ij}}{\|\underline{b_{ij}}\|} \leq \frac{\beta_i + \beta_j}{\|\underline{b_{ij}}\|} \tag{6}$$

In [11], it was proofed using a triangular inequality that the measurement noise ϵ_{ij} remains below the noise bound β_i, β_j, where $\|\varepsilon_{ij}\| \leq \beta_i + \beta_j \doteq \delta_{ij}$.

Since inliers must satisfy two conditions, the linear scalar measurement is split into $\underline{s_{ij\_forward}}$ and $\underline{s_{ij\_reverse}}$ during the determination of these two conditions:

$$\underline{s_{ij\_forward}} = \frac{\|\underline{b_{ij}}\|}{\|\underline{a_{ij}}\|}; \quad \underline{s_{ij\_reverse}} = \frac{\|\underline{a_{ij}}\|}{\|\underline{b_{ij}}\|} \tag{7}$$

Consider two vectors $\underline{v_{inlier\_forward}}$ and $\underline{v_{inlier\_reverse}}$ to assess whether each element in the $\underline{s_{ij\_forward}}$ and $\underline{s_{ij\_reverse}}$ vectors meet the criteria for being within the maximum allowable error $\underline{\alpha_{ij}}$ (source) and $\underline{\beta_{ij}}$ (target). A value of 1 indicates that the criteria are satisfied, while a value of 0 signifies that the criteria are not met. The two criteria are:

$$\begin{aligned} &criteria\ 1\text{: } \left\|\underline{s_{ij\_forward}} - 1\right\| \leq \underline{\alpha_{ij}} \\ &criteria\ 2\text{: } \left\|\underline{s_{ij\_reverse}} - 1\right\| \leq \underline{\beta_{ij}} \end{aligned} \tag{8}$$

Ultimately, $\underline{v_{inlier\_forward}}$ and $\underline{v_{inlier\_reverse}}$ are enumerated, and the correspondence pairs are classified as inliers only if both vectors have a value of 1. After pruning, the inliers form a TEASER++ graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$. This graph contains vertices denoted as $\mathcal{V}$, and each vertex is connected to a set of points, referred to as the edge set $\mathcal{E}$. The graph serves as a storage for both the vertices and their associated points.

In scenarios with high outlier rates, when the graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ is sparse, solving the maximal clique problem can be accomplished swiftly in practice [11]. The next section describes efficient approximation algorithms for finding maximal cliques of graphs that can be extended to graphs with millions of nodes [15–17].

2.4. *Using Maximum Clique Selecting Inliers*

After pruning the inliers and eliminating significant incorrect correspondence pairs, the maximum clique within the TEASER++ graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ will be calculated. The vector $\underline{v}_{vex\ all}$ contains all the points associated with the vertices in the graph. The vector $\underline{c}_{edges}$ represents the set of points connected to the points corresponding to the current item. Additionally, the vector $\underline{v}_{edges}$ appends $\underline{c}_{edges}$ corresponding to each vertex in the graph, while the vertices vector $\underline{v}_{vex}$ keeps track of the cumulative count of edges up to the first item. The k-core vector $\underline{v}_{kcore}$ represents the count of connections for each point and the number of other points it is linked to. It is

expressed as follows, where k denotes the number of vertices in the TEASR++ graph:

$$\underline{v}_{kcore} = \left(\underline{v}_{vex2} - \underline{v}_{vex1}, \ldots, \underline{v}_{vexk} - \underline{v}_{vexk-1}\right) \quad (9)$$

(a) Bucket Sorting

```
1  # include <vector>
2
3  std::vector<int> bin(k);
4  std::vector<int> pos(k+1);
5  std::vector<int> kcore(k+1);
6  std::vector<int> kcore_order(k+1);
7
8  for (v=1; v < k; v++){
9     bin[kcore[i]]++;
10 }
11
12 int start = 1;
13 for (int j=0; j < k; j++) {
14       int num = bin[j];
15       bin[j] = start;
16       start = start + num;
17 }
18
19 for (v=1; v<n; v++) {
20       pos[v] = bin[kcore[v]];
21       kcore_order[pos[v]] = v;
22       bin[kcore[v]]++;
23 }
```

(b) K-cores

```
for (d=k-1; d > 1; d--){ bin[d] = bin[d-1];}
bin[0] = 1;

for (i=1; i<k; i++) {
   v=kcore_order[i];
   for (j=vertices[v-1]; j<vertices[v]; j++) {
      u = edges[j] + 1;
      if (kcore[u] > kcore[v]) {
         du = kcore[u];   pu = pos[u];
         pw = bin[du];    w = kcore_order[pw];
         if (u != w) {
            pos[u] = pw;   kcore_order[pu] = w;
            pos[w] = pu;   kcore_order[pw] = u;
         }
         bin[du]++;   kcore[u]--;
      }
   }
}
```

(c) Max core

```
for (v = 0; v < n-1; v++) {
    kcore[v] = kcore[v+1] + 1; // K + 1
    kcore_order[v] = kcore_order[v+1]-1;}
max_core = kcore[kcore_order[num_vertices()-1]] - 1;
```

Fig. 3 Maximum Clique Core Code in C++

To illustrate the complicated logic, a concise snippet of C++ code is utilized rather than pseudocode (see Fig. 3). The bin vector $\underline{v}_{bin}$ partitions the values in the k-core vector into intervals of width 1, ranging from the minimum to the maximum value. For example, intervals include 0~1, 0~2, …, and 0~k. It then tallies the number of vertices falling within each interval that meet the interval ranges based on their $\underline{c}_{edges}$ size (see Fig. 3a line 8~17).

Using the bucket sort method [18], the vector k-core order $\underline{v}_{kcoreorder}$ is organized. Specifically, the $\underline{v}_{kcororder}$ involves arranging the index values in the TEASER++ graph based on the k-core vector $\underline{v}_{kcore}$ values, from lowest to highest (see Fig. 3a line 19~23).

Once the k-core order vector has been determined, the next step is to determine the value of the maximum core within the graph. For each item in the k-core order vector $\underline{v}_{kcoreorder}$ (see Fig. 3b line 5), extract the current point $\underline{v}_v$ connected by the vectors $\underline{v}_{vex}$ and $\underline{v}_{edges}$, and store this information as the vector $\underline{v}_u$(see Fig. 3b line 7). The k-core value corresponding to $\underline{v}_u$ compares with the k-core value corresponding to $\underline{v}_v$(see Fig. 3b line 8). If the criteria are met, designate $\underline{v}_w$ as the rank in the k-core order, and verify whether $\underline{v}_w$ corresponds to the largest k-core value (see Fig. 3b line 9~10). If the value corresponding to $\underline{v}_u$ is not the largest k-core value, then the positions of $\underline{v}_u$ and $\underline{v}_w$ in the vector k-core order $\underline{v}_{kcoreorder}$ will be exchanged (see Fig. 3b line 12~13).

The k-core vector $\underline{v}_{kcore}$ and k-core order vector $\underline{v}_{kcoreorder}$ of the steps in lines 4 to 18 of Fig. 3b need to be reorganized (see Fig. 3c line 2~3) before calculating the value of max core, to facilitate the operation of Fig. 3c to calculate the max core value (see Fig. 3c line 4). If the k-core corresponding to these vertices $\underline{v}_{vexall}$ in the TEASER++ graph exceeds the max core value, then this point is selected as an inlier filtered by the maximum clique algorithm. This maximum clique inliers $\underline{p}_i$ (from source point-cloud) and $\underline{q}_i$(from target point-cloud) will serve as the inlier set utilized for rotation estimation. In the next section introduces best-fit method to estimate the rotation.

2.5. Rotation Estimation

The best-fitting rigid transformation calculates the rotation that aligns the sets of corresponding points from the source and target point clouds. [11]integrates the rotation estimation component from the [19] as its central content. Maximum clique inliers from the source $\underline{p}_i$ and target $\underline{q}_i$ point clouds feed into the rotation estimation, the procedure for estimating the rotation matrix is as follows:

Calculate the centered vectors $\underline{x}_i$ and $\underline{y}_i$, m denotes the amount of maximum clique inliers:

$$\underline{x}_i \coloneqq \underline{p}_i - \bar{p} \quad \underline{y}_i \coloneqq \underline{q}_i - \bar{q} \quad i = 1, 2, \ldots, m \quad (10)$$

where the centroids of the weight vectors $\underline{w}_i$ for both point sets, where $\underline{w}_i$ is simplified to a 1xM vector in all terms equal to 1.

$$\bar{p} = \frac{\sum_{i=1}^{m} \underline{w}_i \cdot \underline{p}_i}{\sum_{i=1}^{m} \underline{w}_i} \qquad \bar{q} = \frac{\sum_{i=1}^{m} \underline{w}_i \cdot \underline{q}_i}{\sum_{i=1}^{m} \underline{w}_i} \quad (11)$$

The 3x3 covariance matrix $\boldsymbol{S}$, as computed by the [19] after demonstrating the derivation process, is as follows:

$$\boldsymbol{S} = \boldsymbol{XWY}^T \quad (12)$$

Let $\boldsymbol{X}$ and $\boldsymbol{Y}$ be 3xM matrices, where $\underline{x}_i$ and $\underline{y}_i$ represent their respective columns. Additionally, consider $\boldsymbol{W} = diag(w_1, w_2, \ldots, w_m)$.

Calculate the singular value decomposition $\boldsymbol{S} = \boldsymbol{U\Sigma V}^T$, the estimated rotation follows [19]:

$$\boldsymbol{R} = \boldsymbol{V}\begin{bmatrix} 1 & & \\ & 1 & \\ & & \ddots \\ & & & det(\boldsymbol{VU}^T) \end{bmatrix}\boldsymbol{U}^T \quad (13)$$

There are two possibilities for the resulting matrix $\boldsymbol{R}$ to be a rotation matrix and a reflection matrix. They are both a type of orthogonal matrix, preserving the lengths and angles of the vectors, but it is possible to distinguish whether $\boldsymbol{R}$ is a rotation matrix or a reflection matrix by the fact that $\det(\boldsymbol{VU}^{\mathbf{T}})$ is positive or negative. Given that R is a rotation matrix when $\det(\boldsymbol{VU}^{\mathbf{T}}) > 0$ and a reflection matrix when $\det(\boldsymbol{VU}^{\mathbf{T}}) < 0$, it can be concluded that the value of $\det(\boldsymbol{VU}^{\mathbf{T}})$ must be positive to have $\boldsymbol{R}$ as a rotation matrix. The estimated rotation matrix $\boldsymbol{R}$ represents the geometric change in rotational angle from the source point cloud approaching the target point cloud after global registration, based on the camera coordinate system. This matrix is denoted as ${}^{c}\boldsymbol{R}_{src\_aft\_TEA}$. The subsequent section, following the estimation of the rotation matrix, outlines the procedure for estimating translations.

2.6. Translation Estimation

The maximum clique inliers from the source $\underline{p}_i$ and target $\underline{q}_i$ point clouds input to the translation estimation, the procedure for estimating the translation vector is as follows:

A raw translation vector $\underline{t}_i$ from the source point cloud to the target point cloud is determined as follows:

$$\underline{t}_i = \underline{p}_i - \underline{q}_i \quad (14)$$

The translation vectors are estimated using the adaptive voting algorithm of [11]. At each $\underline{t}_i$, construct the boundaries of the intervals$\alpha_i \bar{c}$ and sort the 1x(2M-1) vector $\underline{h}$ in ascending order, m denotes the amount of maximum clique inliers.

$$\underline{h} = (t_1 - \alpha_1\bar{c}, t_1 + \alpha_1\bar{c}, \ldots, t_m - \alpha_m\bar{c}, t_m + \alpha_m\bar{c},) \quad (15)$$

The interval centre vector $\underline{h}_{center}$ is computed from vector $\underline{h}$ as follows to evaluate the consensus set:

$$\underline{h}_{center} = \left(\frac{h_1+h_2}{2}, \ldots, \frac{h_{2k-2}+h_{2k-1}}{2}\right) \quad (16)$$

For each interval, it evaluates the consensus set $C(s)$ using the following condition [11]:

$$C(s) = \{m: \|\underline{h} - \underline{h}_{center}\| \leq \alpha_m\bar{c}\} \quad (17)$$

Truncated **L**east **S**quares (TLS) calculates a least squares solution for measurements, focusing on those with small residuals while excluding measurements with large residuals. TLS is insensitive to a large fraction of spurious correspondences when it is used to estimate the translation vector, but it involve hard combinatorial and non-convex optimization [11]. The objective function $f_i(\widehat{t_i})$ is defined as follows to solve the TLS estimator in polynomial time using simple enumeration [11]:

$$f_i(\widehat{t_i}): \widehat{t_i} = \left(\sum_{m\in C_i} \frac{1}{\alpha_m{}^2}\right)^{-1} \sum_{m\in C_i} \frac{t_m}{\alpha_m{}^2} \; i = 1,2,3 \quad (18)$$

$\widehat{t_1}$, $\widehat{t_2}$, and $\widehat{t_3}$ correspond to the values of the estimated translation vectors X, Y, Z value respectively. $\widehat{t_i}$ in equation x18 represents the value of $\underline{t}_i$ in equation 14 that qualifies for the consensus set as $\underline{t}_i^{\circ}$ in equation 17. The translation error is defined as $\left\|\underline{t}_i^{\circ} - \widehat{t_i}\right\|$, which represents the 2-norm of the difference between the raw translation in the consensus set and the estimated translation vector value $\widehat{t_i}$ [11]. Given the cost of each consensus set (i = 1,2,3) as $X_{cost} = \left(\underline{t}_i^{\circ} - \widehat{t_i}\right) \cdot \left(\underline{t}_i^{\circ} - \widehat{t_i}\right)$, the index corresponding to the minimal value of X_{cost} is determined and return that small cost value as the optimal solution. The value in $\widehat{t_i}$ associated with this index represents the estimated $\widehat{t_i}$.

The estimated translation vector $(\widehat{t_1}, \widehat{t_2}, \widehat{t_3})^T$ represents the geometric shift from the source point cloud to the target point cloud after global registration, based on the camera coordinate system. This vector is denoted as ${}^{c}\underline{r}_{src\_aft\_TEA,\ src}$.

In TEASER++, rigid transformations follow a two-step process. First, they are rotated based on the estimated rotation. Then, they are translated according to the estimated translation. Since rotation and translation are treated separately, they cannot be expressed in a single 4x4 transformation matrix. To simplify understanding, here are the formulas:

The term ${}^{c}\underline{r}_{src,c}$ refers to each point within the source point cloud, based on the camera coordinate system. The source point cloud, which has been rotated based on the estimated rotation matrix ${}^{c}\boldsymbol{R}_{src\_aft\_TEA}$, is denoted as ${}^{c}\underline{r}_{src,c}{}'$:

$${}^{c}\underline{r}_{src,c}{}' = {}^{c}\boldsymbol{R}_{src\_aft\_TEA} \cdot {}^{src\_aft\_TEA}\underline{r}_{src,c} \quad (19)$$

The '*src_aft_TEA*' coordinate system closely resembles the target point cloud after the source point cloud, based on the camera coordinate system, has undergone the estimated rotation matrix ${}^{c}\boldsymbol{R}_{src\_aft\_TEA}$. However, the translation remains unchanged. Consequently, ${}^{\text{src\_aft\_TEA}}\underline{r}_{src,c}$ is equivalent to ${}^{c}\underline{r}_{src,c}$ in the equation 19. Therefore, the formula can also be expressed as:

$${}^{c}\underline{r}_{src,c}{}' = {}^{c}\boldsymbol{R}_{src\_aft\_TEA} \cdot {}^{c}\underline{r}_{src,c} \quad (20)$$

After rotating the source point cloud, the estimated translation vector is utilized to compute the rigid transformation of the source point cloud ${}^{c}\underline{r}_{src\_aft\_TEA,c}$ following the TEASER++ method.

$${}^{c}\underline{r}_{src\_aft\_TEA,c} = {}^{c}\underline{r}_{src\_aft\_TEA,src} + {}^{c}\underline{r}_{src,c}{}' \quad (21)$$

TEASER++, while effective in global registration, still exhibits some inaccuracies in the finer details. Specifically, the edges or corners denoted as Fig. 1 ①②③ remain imprecise.

2.7. Accurate local registration using iterative ICP algorithms

To achieve more accurate alignment results, it becomes necessary to iterate the **I**terative **C**losest **P**oint (ICP) algorithm. After applying the TEASER++ rigid transformation, the source point cloud ${}^{c}\underline{r}_{src\_aft\_TEA,c}$ and target point cloud ${}^{c}\underline{r}_{tgt,c}$, both based on the camera coordinate system, along with a 4x4 Identity matrix, are then input into the iterative ICP algorithm. In the ICP algorithm, the translation and rotation estimates for each iteration are treated separately. This is unlike TEASER++, where ICP first translates and then rotates the point cloud [5]. ${}^{c}\underline{r}_{s\_near\_t,\ src\_aft\_TEA}$ represents the estimated translation vector as it approaches the target point cloud from the source point cloud.

$${}^{c}\underline{r}_{s\_near\_t,c}{}' = {}^{c}\underline{r}_{s\_near\_t,\ src\_aft\_TEA} + {}^{c}\underline{r}_{src\_aft\_TEA,c} \quad (22)$$

${}^{c}\underline{r}_{s\_near\_t,c}{}'$ refers to a source point cloud that has undergone translation but has not yet been rotated. The rotation matrix ${}^{c}\boldsymbol{R}_{s\_near\_t}$ estimated by ICP signifies geometrically that, after precise alignment, the source point cloud is brought closer to the target point cloud.

$${}^{c}\underline{r}_{s\_near\_t,c} = {}^{c}\boldsymbol{R}_{s\_near\_t} \cdot {}^{s\_near\_t}\underline{r}_{s\_near\_t,c}{}' \quad (23)$$

After rotating the source point cloud, it becomes very close to the target point cloud, while the translation remains unchanged. Consequently, ${}^{\text{s\_near\_t}}\underline{r}_{s\_near\_t,c}{}'$ is equivalent to ${}^{c}\underline{r}_{s\_near\_t,c}{}'$. Equation 23 can be expressed as follows:

$${}^{c}\underline{r}_{s\_near\_t,c} = {}^{c}\boldsymbol{R}_{s\_near\_t} \cdot {}^{c}\underline{r}_{s\_near\_t,c}{}' \quad (24)$$

During each iteration, two steps are executed: Equation 22 and Equation 24. Eventually, the process converges to the micrometre level, with the absolute error of ${}^{c}\underline{r}_{s\_near\_t,\ src\_aft\_TEA}$ from Equation 22 serving as the termination criterion. By combining the TEASER++ method with the iterative ICP method, it can align two adjacent point clouds. Since each point cloud has a different relative position concerning the 3D camera, this approach can align point cloud images from various angles. Ultimately, this enables real-time 360-degree scanning and complete alignment of the point cloud image. After scanning, the point cloud model is initially based on camera coordinates. However, for the subsequent assembly process of the robot arm, it becomes necessary to convert the point cloud image to the robot base coordinate system.

2.8. Transformation of the complete point cloud from Camera Coordinate System to Robot Base Coordinate System

The transformation of ${}^{C}\boldsymbol{T}_{LegoGroup}$ to the robot base coordinate system is achieved using the calibration matrix ${}^{F}\boldsymbol{T}_{C}$ (see Fig. 4b). The Hand-Eye Calibration Process determines the calibration matrix ${}^{F}\boldsymbol{T}_{C}$ from the camera coordinate system to the robot flange coordinate system [see

Fig. 4 a]. The transformation matrix ${}^{B}\boldsymbol{T}_{F}$ can be obtained from the teach pendant (see Fig. 4b).

$$ {}^{B}\boldsymbol{T}_{LegoGroup} = {}^{B}\boldsymbol{T}_{F} \cdot {}^{F}\boldsymbol{T}_{C} \cdot {}^{C}\boldsymbol{T}_{LegoGroup} \quad (25) $$

The robot, equipped with the measuring tip, is positioned toward the centre of the Lego group (see Fig. 4c). The ${}^{TCP}\boldsymbol{T}_{B}$ value is read from the teach pendant, enabling the determination of the relative transformation matrix from the TCP to the centre point of the Lego group.

$$ {}^{TCP}\boldsymbol{T}_{LegoGroup} = {}^{TCP}\boldsymbol{T}_{B} \cdot {}^{B}\boldsymbol{T}_{LegoGroup} \quad (26) $$

The new measurement for the other Lego group can be calculated as follows, where ${}^{B}\boldsymbol{T}_{LegoGroup2}$ detected from the 3D camera system is used in equation 26:

$$ {}^{B}\boldsymbol{T}_{TCP} = {}^{B}\boldsymbol{T}_{LegoGroup2} \cdot {}^{TCP}\boldsymbol{T}_{LegoGroup}{}^{-1} \quad (27) $$

Any tool can replace the measuring tip using equations 25 to 27 for the new object, as well as the new camera position. This allows computation of the 3D pose based on the robot base coordinate system. The new tool can be utilized with the teach pendant to move the robotic arm into four distinct positions through tool calibration, ensuring that the TCP remains in contact with the same fixed position throughout.

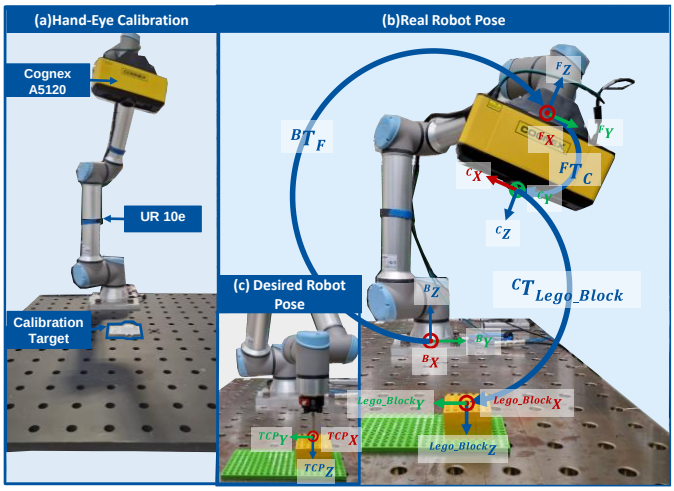

Fig. 4 Hand Eye Calibration and Training with a Lego Group to obtain more accurate robot base pose

3. Validation

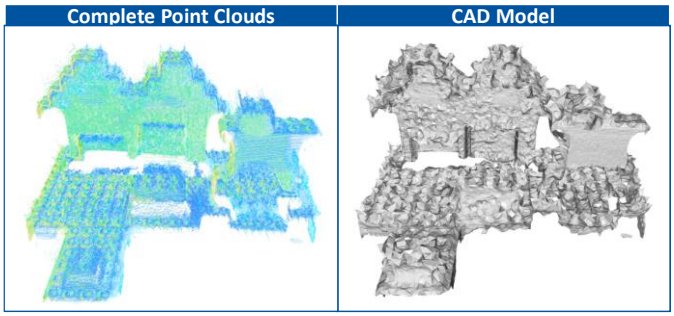

Fig 5 Validation Results with 46 point cloud images

To capture diverse point cloud images, tilt and rotate the LEGO components from various angles within 5 degrees. These point cloud images are then synthesized together. Despite high noise interference, the stability of the point cloud images remains intact. Using the entire point cloud data, the Lego Group generates the actual CAD model. (see Fig. 5)

4. Conclusion

In this research work, the TEASER++ method for global registration is employed, ensuring that the source point cloud closely aligns with the target point cloud. Subsequently, we utilize the iterative ICP method to achieve fine alignment, particularly enhancing accuracy in edge and corner details.

The advantage of the proposed method is that TEASER++ is not sensitive to the large number of incorrect correspondence pairs between source- and target point clouds due to interference noise. Even if the object deflection angle is relatively large, the alignment results of the combined iterative ICP method are still accurate and stable.

One limitation of the proposed method is that TEASER++ global registration may not achieve sufficient accuracy for objects with inconspicuous feature points, such as washbasins, toilets, and other smooth surfaces.

In future research, we will focus on enhancing the FPFH feature point method to enable stable alignment of objects even when geometric features are not readily apparent. The comprehensive point cloud images obtained in this study will be leveraged for subsequent intelligent analyses and the advancement of fully automated assembly processes. (*The research is funded by the EU-supported RICAIP project GA: 857306)

References

[1] Liu, Y. et.al. Point2CAD: Reverse Engineering CAD Models from 3D Point Clouds.
[2] Ding, X. et al., 2024. Robotic grinding based on point cloud data: developments, applications, challenges, and key technologies 131, p. 3351.
[3] Tianyun Yuan et. al. 2019. Direct 3D Printing System: from Point Cloud to Additive Manufacturing.
[4] Li, Y. et.al. Using the Point Cloud Data to Reconstructing CAD Model by 3D Geometric Modeling Method in Reverse Engineering, ICMEIM 2017
[5] Besl, P.J., McKay, N.D., 1992. A method for registration of 3-D shapes
[6] Chen, Y. et.al. Object modelling by registration of multiple range images
[7] Rusinkiewicz, S. et.al. Efficient variants of the ICP algorithm, in Third International Conference on 3-D Digital Imaging and Modeling
[8] Park, J..al. Coloured Point Cloud Registration Revisited, in 2017 IEEE International Conference on Computer Vision: ICCV
[9] Zhou, Q et.al. Fast Global Registration, in Computer Vision - ECCV 2016
[10] Open 3D. Fast Global Registration. http://www.open3d.org/.
[11] Yang, H. et.al. TEASER: Fast and Certifiable Point Cloud Registration.
[12] Rusu et. al.., 2008. Aligning point cloud views using persistent feature histograms, IEEE IROS 2008
[13] Rusu et. al.. Fast Point Feature Histograms (FPFH) for 3D registration, in 2009 IEEE ICRA
[14] Rusu. Semantic 3D Object Maps Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments.
[15] Wu, Q. et.al., 2015. A review of algorithms for maximum clique problems
[16] Pattabiraman et al., 2015. Fast Algorithms for the Maximum Clique Problem on Massive Graphs with Applications to Overlapping Community Detection
[17] Bron, C.et. al., Algorithm 457: finding all cliques of an undirected graph
[18] Burnetas et.al., 1997. An analysis and implementation of an efficient in-place bucket sort
[19] 2017. Least-squares rigid motion using SVD.