Javier Laserna\textsuperscript{1}, Pablo San Segundo\textsuperscript{1}, and David Álvarez\textsuperscript{1}

\textsuperscript{1}Affiliation not available

January 26, 2024
CliReg: Clique-based robust Point Cloud Registration

Javier Laserna, Pablo San Segundo, David Álvarez

Abstract—We propose a branch-and-bound algorithm for robust rigid registration of two point clouds in the presence of a large number of outlier correspondences. For this purpose, we consider a maximum consensus formulation of the registration problem and reformulate it as a (large) maximal clique search in a correspondence graph, where a clique represents a complete rigid transformation. Specifically, we use a maximum clique algorithm to enumerate large maximal cliques and a fitness procedure that evaluates each clique by solving a least-squares optimization problem. The main advantages of our approach are i) it is possible to exploit the cutting-edge optimization techniques employed by current exact maximum clique algorithms, such as partial maximum satisfiability-based bounds, branching by partitioning or the use of bitstrings, etc.; ii) the correspondence graphs are expected to be sparse in real problems (confirmed empirically in our tests), and, consequently, the maximum clique problem is expected to be easy; iii) it is possible to have a good control of suboptimality with a k-nearest neighbour analysis that determines the size of the correspondence graph as a function of k. The new algorithm is called CliReg and has been implemented in C++. To evaluate CliReg, we have carried out extensive tests on a dataset of 540 instances generated from scan-matching models in the public Standford 3D Scanning repository. The results show that CliReg clearly dominates the state-of-the-art (e.g., RANSAC, FGR, and TEASER++) in terms of robustness, with a running time comparable to TEASER++ and RANSAC. In addition, we have implemented a fast variant called CliRegMutual that performs similarly to the fastest heuristic FGR.

Index Terms—Mobile robotics, scan matching, discrete optimization, maximum clique, point cloud 3D registration.

I. INTRODUCTION

POINT set representations commonly appear in a wide variety of applications that span different fields of science, including computer vision tasks such as object recognition and detection, localization, motion estimation, medical image analysis, pattern recognition, and computer graphics to generate realistic 3D models, see, e.g., [6]. Many problems in these fields can be effectively addressed by algorithms operating on point sets. This paper focuses on (3D) point cloud registration (also called point-set registration or scan matching in certain fields), which calls for determining the transformation, rotation, translation, and (potentially) scale that best aligns two point clouds. Applications to point cloud registration are encountered in a wide variety of contexts, such as, e.g., 3D model reconstruction [12, 35], object recognition and localization [37, 7], robot navigation [53, 54] or medical imaging [4, 68].

Point cloud registration has been the subject of extensive research, such as, e.g., [8, 10, 23, 38, 41, 66, 78, 6]. In an ideal setting where the ground-truth correspondences between the two clouds are known in the presence of Gaussian isotropic noise, the registration problem has an elegant closed-form solution [34, 3]. In real problems, however, the set of correspondences contains a large number of outliers (or is simply unknown) and the problem becomes exponential in the worst-case.

In the real-world scenario, one successful local approach relies on a (good) initial transformation that is refined at each iteration by finding an improved set of correspondences and a subsequent new transformation. A well-known algorithm of this type is ICP [10]. Nondeterministic local methods, such as, e.g., RANSAC [27], consider at each iteration a small sample of correspondences. In general, these approaches can be fast but brittle in the presence of a large number of outliers, and may return transformations which are far from the ground truth. The alternative to local methods are global branch-and-bound (BnB) approaches, such as, e.g., Go-ICP [80], which are capable of determining an optimal solution and can solve the problem without needing an a priori initial transformation. Although they are guaranteed to be robust, they run in exponential time in the worst case.

Motivated by the above considerations, this work presents a new BnB global algorithm CliReg for (rigid) point cloud registration. CliReg is based on a Maximum Consensus formulation, which is reformulated as a clique search on a graph. In our extensive tests, the algorithm is shown to be accurate and robust in the presence of a large number of outliers. Furthermore, it shows running times comparable to those of the fastest approaches (such as, e.g., the heuristic ransac-based procedure FGR [83]).

II. RELATED WORK

Related to this work are correspondence-based registration methods, where a set of corresponding pairs of keypoints is given or determined during a preprocessing step (the matching hypotheses). Corresponding pairs are typically established according to a local 3D feature extraction algorithm, such as, e.g., Fast Point Feature Histograms FPFH [50], 3match [81] or deep learning-based PCGF [23], and registration is restricted to keypoint correspondences with similar features (Figure 1 provides an example). It should be mentioned that 3D features are expected to be less accurate compared to their 2D counterparts, e.g., SIFT [47], and tend to produce much higher outlier rates [14]. Once outliers are present, the
problem becomes exponentially hard in the worst case and robust approaches are required.

Fig. 1: (a) An example of two point clouds with outliers, (b) Overlapping clouds resulting from the registration. Inlier keypoint correspondences appear in blue.

One of the most common algorithms used for robust registration is Random Sample Consensus (RANSAC) [27]. RANSAC is a nondeterministic method that at each iteration samples the correspondence space and uses the closed-form solution in [34] to efficiently solve a least-squares optimization problem and estimate a transformation. RANSAC is known to perform well in a low noise and low outlier regime (e.g. below 50%), since its running time grows exponentially with the number of outliers [14]. In the presence of high outlier ratios, such nondeterministic approaches can (potentially) take a long time due to countless matching proposals and evaluation attempts.

To improve this behavior, [83] introduced Fast Global Registration (FGR), a faster RANSAC-based approach that uses a robust least-squares cost function and reformulates it with Black-Rangarajan duality [11] to solve it efficiently using the Gauss-Newton method. Despite its efficiency, in our experiments FGR produced overall a less robust registration than the other algorithms reported, including RANSAC (see Section VII).

Among the numerous local methods proposed in the literature, the Iterative Closest Point (ICP) algorithm [10] [19] [82] is allegedly considered a landmark in point cloud registration. In later years, multiple improvements have been proposed that use robust cost functions to improve convergence, such as, e.g., [29] [48] [20]. Essentially, these approaches alternate between two procedures: establishing closest-point correspondences based on the current transformation and estimating a new transformation using these correspondences. The main drawbacks are that they are liable to converge to local minima and require a good initial transformation (a good guess). Phase Registration (PHASER) [9] is an alternative local registration approach that uses spherical Fourier analysis. By examining the probability distribution of its properties, this method circumvents the need for point correspondences. This approach favors decoupling the registration problem into distinct rotation and translation components.

Global methods solve the registration problem to proven optimality without requiring an initial guess. Typically, these methods are of the BnB type and are based on a maximum consensus formulation (see Section III), such as, e.g., [75] [15] [40] [17]. An interesting survey on this topic is [21].

The literature reports interesting bounds that involve geometric techniques, such as, e.g., [32] [13] or those proposed in the algorithm Go-ICP [80]. Specifically, Go-ICP integrates the local ICP algorithm in its BnB scheme, achieving good speedups while preserving global optimality. In general, these approaches run in exponential time on the size of the point clouds, which is aggravated by the strong non-convexity of the search spaces resulting from high outlier ratios.

Another recent method for robust registration, closely related to this work, is the fast and certifiable algorithm TEASER [78]. TEASER operates according to the following steps. In a first step, the registration problem is reformulated using a Truncated Least Squares (TLS) cost function, insensitive to many spurious correspondences. In a second step, the scale, rotation, and translation subproblems are decoupled and solved sequentially by using adaptive voting, semidefinite programming, and a graph-theoretic framework that drastically prunes outliers by finding a maximum clique. It should be noted that an improved variant TEASER++ [79] uses graduated non-convexity to solve the rotation subproblem efficiently.

We end this survey by mentioning the recent upsurge of Deep Learning approaches to point cloud registration, such as, e.g., [74] [2] [73] [5] [24]. According to [79], these approaches are currently still struggling to produce acceptable robust registrations in real problems. It should be mentioned that our proposed method can also be used with deep-learned features, such as those obtained by FCGF [23].

A. Our contribution

The main contributions of this work are summarized in the following points.

- A clique-based methodology for robust rigid correspondence-based point cloud registration is proposed. Similarly to TEASER [78], a correspondence graph is built based on a predetermined set of feature-based correspondences (potentially with many outliers). However, the rotation and transformation subproblems are not decoupled, and a clique in our correspondence graph represents a complete rigid transformation. We empirically show that our approach determines robust and accurate solutions even in the presence of a large number of outliers. Another advantage is its ability to control suboptimality by fixing the number of matching hypotheses in the correspondence set.
- An algorithm CliReg that efficiently implements the methodology mentioned above is described. At the core of CliReg lies a very efficient BnB clique algorithm that incrementally enumerates maximal cliques in our correspondence graph and builds on cutting-edge optimization techniques employed by recent efficient algorithms for the Maximum Clique Problem [58] [59] [57] [60] [62] [61] [44] [65]. It should be mentioned that prior clique-based approaches (e.g., TEASER [78] and TEASER++ [79]) consider the less efficient maximum clique algorithm PMC [55], which lacks many components of CliReg (e.g., branching by
partitioning, partial maximum satisfiability-based bounds etc.; see Section VII-E for a detailed description of the clique-enumeration procedure of CliReg.

CliReg also presents some unique features specifically adapted for this problem. One of the new features is that its BnB search is driven by a fitness function that selects the best maximal clique enumerated (see Section V-B). In previous clique-based approaches (e.g., TEASER [78] and TEASER++ [79]) only a maximum clique was considered.

- Extensive experiments on a standard scan-matching benchmark have been carried out. According to the results reported in Section VII-C, CliReg achieves a much more accurate registration than other state-of-the-art approaches (TEASER++ [79], RANSAC [27], FGR [83], etc.) with up to (and including) a 50% outlier rate. Furthermore, we have implemented a fast variant of CliReg that performs comparable to the fastest algorithms, e.g., FGR [83]. To the best of our knowledge, we are also the first to report data concerning the structure of correspondence graphs in registration; see Section VII-B.

III. Maximum Consensus Formulation

Let $Q$ be a target 3D point cloud modeling an object, and let $M$ be a moving (or partial) 3D point cloud of the same object to be registered against the target cloud $Q$. Let $|Q|$ and $|M|$ denote the number of points in $Q$ and $M$ respectively. In what follows, it is assumed that $|M| \leq |Q|$.

As argued in Section II in correspondence-based registration a set $C$ of corresponding pairs of points, $C \subseteq M \times Q$, each pair representing a matching hypothesis, is given (or computed in a preprocessing step). If all the correspondences in $C$ are valid, i.e., there are no outliers, then each pair of correspondences $(m_i, q_i) \in C$, $m_i \in M$, $q_i \in Q$, can be modeled by:

$$q_i = \ell R m_i + t + \epsilon_i, \quad i \in \{1, \ldots, |C|\},$$

where $\ell > 0$ is a uniform scale factor, $R \in SO(3)$ is a proper 3D rotation matrix ($SO(3)$ is the special orthogonal group of degree 3), $t \in \mathbb{R}^3$ is a 3D translation vector and $\epsilon_i \in \mathbb{R}^3$ is any unknown additive noise (e.g. Gaussian noise). This work is concerned with rigid transformations only, so the value of $\ell$ is fixed to 1 in the remainder.

An important result is that if the Gaussian noise $\epsilon_i$ is assumed to be a zero mean isotropic Gaussian distribution, i.e., $\epsilon_i \sim \mathcal{N}(0, \sigma^2 \times I_3)$, then model (1) can be formulated as the following least-squares optimization problem:

$$R^*, t^* = \arg \min_{R \in SO(3), t \in \mathbb{R}^3} \frac{1}{N} \sum_{i=1}^{N} \frac{1}{\sigma_i^2} ||q_i - R m_i - t||^2,$$

for which a closed-form solution is known, see [34, 3, 72].

When outliers are present, i.e., some of the correspondence pairs given in $C$ cannot be determined by (1), formulation (2) is not suitable and the problem becomes exponential in the worst-case. Related to this work is the Maximum Consensus (MC) formulation for robust registration (in the presence of outliers). Essentially, the method seeks to determine the largest subset of inliers $I \subseteq C$ that is consistent with model (1), i.e.:

$$\max_{I \subseteq C} \left| I \right|$$

$$s.t. \quad \frac{1}{\sigma_i^2} ||q_i - R m_i - t||^2 < \xi, \quad \forall (m_i, q_i) \in I.$$  

The set of Constraints (4) enforce that every pair of correspondences in the inlier set $I$ has residuals $\frac{1}{\sigma_i^2} ||q_i - R m_i - t||^2$ below a predetermined threshold $\xi$, which is a function of the resolution of the two input point clouds. To determine $\xi$, we first introduce the notion of resolution $\gamma$ of a point cloud $P$ as the following average distance:

$$\gamma(P) = \frac{\sum_{p \in P} d(p, P \setminus \{p\})}{|P|},$$

where $d(p, P \setminus \{p\})$ is the minimum distance from the point $p \in P$ to any other point in $P$. Having introduced the notion of cloud resolution $\gamma$, we define $\xi$ as the maximum resolution of any of the two clouds of the registration problem:

$$\xi(M, Q) = \max \{ \gamma(M), \gamma(Q) \}$$

MC is known to be NP-hard [22] and we recall from Section II that specific BnB techniques have been proposed in the literature to determine a global optimum, such as, e.g., [33, 16, 76, 21]. Our proposed global robust registration BnB method is based on the MC formulation and is described in detail in the following Section IV.

IV. Clique-Based Point Cloud Registration

This section describes our novel clique-based robust method for rigid correspondence-based point cloud registration in the presence of outliers.

A. Preliminaries

Let $G = (V, E)$ be a simple undirected graph with $n = |V(G)|$ vertices and $m = |E(G)|$ edges. Two vertices $u, v \in V(G)$ are said to be adjacent, also neighbours, if there exists an edge $u, v \in E(G)$. $N(v)$ denotes the neighbour set of the vertex $v$, that is, the subset of vertices in $V(G)$ adjacent to $v$. The degree of $v$, denoted $\deg(v)$, is the cardinality of its neighbour set $|N(v)|$. Given the subset of vertices $W \subseteq V(G)$, notation $G[W]$ denotes the induced graph by $W$, the subgraph of $G$ with the vertex set equal to $W$ and the edge set that contains those edges of $E(G)$ with both endpoints in $W$ i.e., $G[W] = (W, (W \times W) \cap E(G))$.

A clique in $G$ is a subset of pairwise adjacent vertices; also a subset of vertices that induces a complete subgraph in $G$. A clique is denoted maximal if it cannot be extended by adding one or more vertices, i.e., it cannot be a subset of a larger clique. A maximum clique is a largest maximal clique. The maximum clique problem (MCP) consists on finding a maximum clique in $G$, and the size of any solution is called the clique number of the graph $\omega(G)$.
B. The correspondence graph

Given a moving and a target 3D point cloud \( M \) and \( Q \), and a correspondence set \( C \) containing a subset \( I \subseteq C \) of inliers (correspondence pairs that can be modeled according to formulation (1)) and (possibly) outliers (correspondence pairs that cannot be modeled according to formulation (1)), we define a correspondence graph \( G = (V,E) \) as follows. Each vertex of the graph represents a hypothesis, i.e., a candidate (keypoint) pair \((m,q)\), \(m \in M, q \in Q\), based on the similarity of their point features provided by some local 3D feature extraction algorithm, such as, e.g., FPFH [56]. An edge in the graph represents that the difference, in terms of Euclidean distance, between the two pairs of points of the same point cloud at both endpoints is strictly less than the value of \( \xi \) [6]. Formally speaking:

\[
E(G) = \{v,w \in V : ||m_v - m_w||^2 - ||q_v - q_w||^2 < \xi \} \quad (7)
\]

where \((m_v,q_v)\) and \((m_w,q_w)\) are the pairs of points associated to the vertices \(v\) and \(w\) in the correspondence graph, respectively. It should be mentioned that the edge definition (7) is valid only for rigid registration. If the distances between points change in the transformation between the two point clouds, e.g., there exists a uniform scaling factor \( \ell \) different from 1, an alternative metric between points should be used.

To illustrate how the correspondence graph is built, we consider the example shown in Figure 2. The figure represents two 3D point clouds \( M \) and \( Q \) in which the points of interest are coloured red (\(|M| = 5\)) and green (\(|Q| = 5\)), respectively. In addition, we consider the correspondence set \( C \) which comprises the following 10 matching hypotheses:

\[
C = \{(m_1,q_1),(m_1,q_2),(m_2,q_1),(m_2,q_2),(m_3,q_3),
(m_3,q_4),(m_4,q_3),(m_4,q_4),(m_5,q_4),(m_5,q_5)\},
\]

which is shown in Figure 3 together with the associated vertices in \( G \). In blue, the hypotheses \( \{v_1,v_4,v_5,v_8,v_9\} \) that solve the registration problem, i.e. the set of inliers \( I \subseteq C \), \( I = \{(m_1,q_1),(m_2,q_2),(m_3,q_3),(m_4,q_4),(m_5,q_5)\} \). The (rigid) transformation \( (R^*,t^*) \) for \( I \) is obtained by solving the least-squares optimization problem (2).

![Fig. 2: Example of two point clouds M and Q.](image)

Figure 2 shows an example of an edge (a) and a non-edge (b) in the correspondence graph according to definition (7). In the example, \((v_1,v_4) \in E(G)\) because the Euclidean distance between \( m_1 \) and \( m_2 \) is similar to the distance between \( q_1 \) and \( q_2 \), i.e., falls below the threshold \( \xi \) in (7). As shown, this is not the case for \((v_1,v_{10})\). The correspondence graph \( G \) is displayed in Figure 5 in blue, the subgraph induced in \( G \) by the 5-clique \( \{v_1,v_4,v_5,v_8,v_9\} \) solution to the registration problem. The graph will be used to filter outliers. For example, the island vertex \( v_{10} \) is a spurious correspondence, since its associated pair of points, i.e., \((m_5,q_4)\), does not find a match in any other pair according (7).

![Fig. 3: Set of correspondences C (also vertices of the correspondence graph) for the point clouds in Figure 2. In blue, the matches corresponding to the optimal solution to the registration problem.](image)

C. Searching for large cliques in the correspondence graph

Our method for robust rigid point cloud registration is based on the observation that the set of inliers \( I \subseteq C \) in the correspondence set \( C \) forms a clique in the correspondence graph. This observation is also valid for any subset of inliers. We recall from the example in Figures 2 and 3 that the solution to the registration problem is the set of inliers \( I = \{(m_1,q_1),(m_2,q_2),(m_3,q_3),(m_4,q_4),(m_5,q_5)\} \), which corresponds to the 5-clique \( \{v_1,v_4,v_5,v_8,v_9\} \) in \( G \), see Figure 5. Essentially, an edge between two vertices in \( G \) refers to a rigid transformation between the associated pairs of points at both endpoints (7). Therefore, a set of \( k \) pairwise adjacent
vertices, i.e., a $k$-clique in $G$, extends the validity of the rigid transformation to every point in the $k$-clique.

However, not every clique in $G$ corresponds to a subset of inliers. As argued in Section IV-B, no transformation estimation is required to build the correspondence graph, so some of the hypotheses considered, i.e., the vertices of $G$, could be outliers and not agree with Constraints of the MC formulation. Furthermore, symmetries can also be a reason for flawed cliques. Consider, for example, the subset of inliers $\{(m_1, q_1), (m_2, q_2)\}$ from the previous example, associated with the clique $\{v_1, v_4\}$ in $G$. It is straightforward to see that the set of inliers $\{(m_1, q_2), (m_2, q_1)\}$ also represents a rigid transformation associated to the clique $\{v_2, v_3\}$ that is locally valid, but globally flawed.

It should be noted that the number of flawed rigid transformations, i.e., bad cliques in $G$, increases with the existence of symmetries in the set of keypoints from each point cloud in the given correspondence set. Taking again into account the example shown in Figures 2 and 3, the set of points $\{m_1, m_2, m_3, m_4\}$ in the point cloud $M$ are symmetric with respect to the line $m_1m_2$, and so are the corresponding points in the point cloud $Q$. Therefore, the set of correspondences $\{(m_1, q_1), (m_2, q_2), (m_3, q_4), (m_4, q_3)\}$ also forms a clique in $G$, i.e., a locally valid rigid transformation. However, this transformation is incompatible with the inlier $v_9 = (m_5, q_5)$.

We summarize the above considerations in the following three observations.

**Observation 1:** Any subset of inliers $I \subseteq C$, i.e., subset of pairs of points $(m, q), m \in M, q \in Q$ that agree with Constraint, is associated with a clique in $G$. However, a clique in $G$ does not necessarily correspond to a subset of inliers.

**Observation 2:** The probability that a clique in $G$ contains a large number of inliers increases with the size of the clique.

**Observation 3:** The probability that a clique in $G$ contains a large number of inliers decreases with the number of outliers and the symmetries present in the given correspondence set $C$.

In light of these observations, we propose a new clique-based method CliReg for robust rigid registration. The method is discussed in Section V that follows.

V. THE NEW ALGORITHM CliReg

As argued at the end of the previous Section IV-C, each clique in the correspondence graph $G$ represents a rigid transformation in $SE(3)$. In addition, large cliques in $G$ have a high probability of being inliers (according to Observations 1 and 2), but may contain some outliers (according to Observation 3). Consequently, we are interested in finding large maximal cliques in the correspondence graph, but not necessarily a maximum clique.

To enumerate (large) maximal cliques in $G$ we have designed CliReg as a maximum clique algorithm based on the following considerations. The problem of enumerating all maximal cliques runs in $O(3^{n/3})$ on the size $n$ of the graph $G$, moreover, the bound is tight since there are graphs with precisely that amount of maximal cliques, i.e., the Moon-Moser graphs [52]. However, MCP has worst-case complexity $O(2^{n/2})$ when using bounds based on vertex colouring (according to [59]), clearly better than maximal clique enumeration. Furthermore, this can be improved in practice by using bounds based on partial maximum satisfiability (see Section VI). As a final consideration, solving the least squares optimization problem to evaluate every maximal clique is impractical.

**Algorithm 1: The algorithm CliReg for correspondence-based rigid point cloud registration.**

**Input:** A pair of point clouds $(M, Q)$.

**Output:** A rotation $R$ and a translation $t$.

1. $C \leftarrow$ genCS$(M, Q)$ \hspace{1cm} \triangleright \text{see Section IV-B}
2. $\xi \leftarrow$ determine the resolution threshold \hspace{1cm} \triangleright \text{see Section IV-A}
3. $G \leftarrow$ genG$(C, \xi)$ \hspace{1cm} \triangleright \text{see Section IV-B}
4. $K \leftarrow$ MaxClqReg$(G)$
5. return $\text{LSQ}(K)$ \hspace{1cm} \triangleright \text{least-squares (1)}

The 3 main components of CliReg are the following:

i) **Initialization:** For the two given point clouds $M, Q$, a correspondence set $C$ is determined based on local features, and a correspondence graph $G$ is built according to Section IV-B.

ii) **Enumeration:** Maximal cliques in $G$ are enumerated with a maximum clique algorithm that uses the latest optimization techniques (see, e.g., [58, 59, 57, 60, 62, 61, 44, 65]) and is specifically designed for this problem.

iii) **Consensus:** Each maximal $k$-clique enumerated in ii) is evaluated using the assumption that it refers to a (valid) set of $k$ inliers as follows: a) rotation and translation are estimated by solving the least-squares formulation (2), and b) for the resulting transformation, Constraints are checked to determine the real number of inliers in $C$. The algorithm CliReg returns the maximal clique enumerated in ii) with the maximum consensus.

**CliReg** is outlined in Algorithm 1. In what follows, each component of CliReg is described. The initialization and consensus components are introduced below. We postpone the explanation of the more complex enumeration component to Section VI.
A. Initialization

The initialization component covers the first 3 steps of CliReg (Algorithm 1). In Step 1 the set $C$ of correspondences (matching hypotheses) is determined (as defined in Section IV-B) based on the local 3D features provided by FPFH [50], complemented by a $k$-nearest neighbours (KNN) analysis [31], where $k \leq |Q|$. As a result, the size $n$ of the correspondence graph is $n = |M| \times k$ (the largest possible number $|M| \times |Q|$ are all the possible hypotheses). KNN analysis provides a good control of suboptimality, since, as argued at the beginning of the section, the running time of CliReg is bounded by $O(2^{n/5})$. Using KNN analysis, the (suboptimal) bound is now $O(2^{(|M| \times k)/5})$. According to our experiments, a reasonable value of $k$ is 3 (see Section VII-B for further details).

In Step 2 the parameter $\xi$ is computed. We recall that $\xi$ is required to determine the edges of the correspondence graph $G$, as well as to determine the consensus between cliques (Constraints 4 in the MC formulation). In Step 3 the correspondence graph $\mathcal{G}$ is built according to Section IV-B.

B. Consensus

This Section covers the function CliEval, the consensus component of CliReg. Pseudocode for CliEval is shown in Algorithm 2. CliEval evaluates each maximal $k$-clique in $G$ enumerated during the search in the following two phases. In its first phase, the function $LSQ(K)$ determines the transformation $(R, t)$ solution to the least-squares optimization problem (2) under the assumption that the correspondences in $K$ are all inliers. Specifically, $LSQ(K)$ implements the dual quaternion (DQ) algorithm [72]. In its second phase (Step 3), the real number of inliers $\phi$ in the set of correspondences $C$ are counted in the function fitnessCLQ. Precisely, these are the subset of correspondences in $C$ that agree with Constraints 4 for the given transformation $(R, t)$.

Algorithm 2: The evaluation function CliEval.

<table>
<thead>
<tr>
<th>Input:</th>
<th>A clique $K$ in the correspondence graph.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output:</td>
<td>An integer $\phi \in {0, \ldots,</td>
</tr>
</tbody>
</table>

1. $R, t \leftarrow LSQ(K)$ \> least-squares (2)  
2. $\phi \leftarrow \text{fitnessCLQ}(R, t)$ \> #inliers (4)  
3. return $\phi$

VI. Enumeration of maximal cliques in the correspondence graph

Clique enumeration in CliReg is performed by the procedure MaxClqReg. It is executed by CliReg immediately after its initialization component has ended, i.e., in Step 3 of Algorithm 1. We recall from Sections II and III that finding a global optimum for the MC formulation (3)-(4) is $NP$-hard, and that BnB approaches are known, such as, e.g., [33, 16, 76, 21].

Reformulating the MC problem as a tailored maximum clique search has the following advantages: i) it is able to exploit most of the optimization techniques of today’s efficient MCP algorithms, which have improved greatly in the last two decades, see, e.g., [58, 59, 57, 60, 62, 61] and especially [44, 65]. ii) as argued in Section V solving the least-squares optimization problem (2) for every node in the branching tree is inefficient. Our MCP algorithm enumerates a subset of (large) maximal cliques. iii) the correspondence graph $\mathcal{G}$ is expected to be sparse in real-world problems, which is consistent with the reported experiments, see, e.g., Table I. Consequently, the MCP is expected to be easy in practice, as long as the number of vertices (matching hypotheses) in $\mathcal{G}$, i.e., $|M| \times k$, $k \leq |Q|$ (see Section VII-A), remains within reasonable limits.

In what follows, the different components of MaxClqReg are explained in detail. Section VI-A covers the general branching scheme of CliReg and Section VI-B covers its specific branching scheme, tailored for the registration problem. Section VI-C explains the general bounding scheme, and preprocessing is described in Section VI-D. Finally, pseudocode for MaxClqReg is provided in Section VI-E.

A. General branching scheme

A tree node of MaxClqReg is associated with a graph subproblem $\mathcal{G}$, a clique $K$ under construction, and an integer $q$ that is the number of vertices that must be added to $K$ to have the same cardinality as the incumbent clique $K_{\text{max}}$, i.e., a largest clique found at any moment during the search. When branching, MaxClqReg selects a candidate vertex $v \in V(\mathcal{G})$ (by construction, $v$ must be adjacent to every vertex in $K$) and computes the child node $(\mathcal{G}, K, q)$ as follows: i) in the general branching scheme, $\mathcal{G}$ is the subgraph induced in $\mathcal{G}$ by the vertices in $V(\mathcal{G})$ neighbours to $v$; ii) $K = K \cup \{v\}$ and iii) $\tilde{q} = |K_{\text{max}}| - |K|$ (also $\tilde{q} = q - 1$, since the clique $K$ has been enlarged by one unit).

Using the above branching strategy, it is easy to see that the leaf nodes of MaxClqReg must be maximal cliques. Any maximal clique larger than the incumbent clique $K_{\text{max}}$ becomes the new incumbent clique. In addition, MaxClqReg keeps track of the maximal clique $K_{\text{reg}}$ with maximum consensus (according to the Consensus component of CliReg; see Section IV-B). Upon backtracking, MaxClqReg continues the search by branching on a remaining candidate vertex in $V(\mathcal{G})$.

B. Bounding by partitioning the candidate vertex set

Given a tree node $(\mathcal{G}, K, q)$, the (candidate) vertex set $V(\mathcal{G})$ can be partitioned into two disjoint sets of vertices called the pruned set $P$ and the branching set $B = V \setminus P$, so that branching on the vertices of $P$ alone cannot improve the incumbent clique, i.e.,

$$\omega(\mathcal{G}[P]) \leq q. \quad (8)$$

Equation 6 states that the clique number of the graph induced by the set $P$ in the subproblem graph $\mathcal{G}$ cannot be strictly greater than $q$. Since $q$ is precisely the difference between the sizes of the incumbent clique and the current clique $K$ under construction, it is clear that branching only on the vertices in $P$
will not improve the incumbent. Consequently, MaxClqReg branches on the vertices in the branching set \( B \).

Furthermore, and in order to reduce the size of the branching tree, the set \( B \) should be as small as possible (alternatively \( P \) as large as possible); see, e.g., [44, 65]; so the ideal set \( P \) is the solution to the following optimization problem:

\[
P = \arg \max_{P \subseteq V(\mathcal{G})} \left\{ |\hat{P}| : \omega(\hat{P}) \leq q \right\},
\]

and \( B = \hat{V} \setminus P \).

Finding the optimal set \( P \) is a very hard exponential problem so, in practice, two types of bounds (sometimes referred to as bounding functions in this context) are used to determine \( P \). In the listed order: i) a sequential greedy vertex colouring heuristic [50] and ii) a family of bounds derived from a reduction of the maximum clique problem of a vertex-coloured graph (in practice, according to the colouring determined in i)) to a partial maximum satisfiability problem (allegedly, first defined in [51]). We briefly explain both bounding functions in what follows.

It is straightforward to see that a valid set \( P \), according to its defining property [8], is any partial \( q \)-colouring of \( \mathcal{G} \), since \( q \) must be an upper bound on the clique number of \( \mathcal{G}[P] \). In addition, given a \( q \)-colouring \( C_q(P) \) of the set \( P \), and a vertex \( v \in B \), a partial maximum satisfiability-based bounding function attempts to prove that

\[
\omega(\mathcal{G}[P \cup \{v\}]) \leq q,
\]

with the aim of creating a new enlarged pruned set \( P \cup \{v\} \) if the property holds (and remove \( v \) from \( B \), thus reducing its size). Specifically, the bounding function looks for any subset of independent sets \( I \subseteq C_q \) that cannot contain a clique of size \( |I| \).

Both of these bounding functions are well established in the literature, so we will not describe them in detail here. Colour-based bounds were already first proposed in an efficient BnB MCP algorithm in [26]. Partial maximum satisfiability-based bounds were already first proposed for the MCP in [42]. Examples of efficient MCP algorithms that employ either one or both bounds as bounding functions to partition the candidate set of vertices are [45, 46, 57, 58, 59, 60, 63, 62, 64], and also today’s state-of-the-art algorithms [44, 65]. We refer the interested reader to the cited literature for further details.

### C. Branching in detail: a non-incremental scheme

As argued in the previous Section VI-B bounding by partitioning reduces the candidate vertex set \( V(\mathcal{G}) \), where \( \mathcal{G} \) is a subproblem graph, to a branching set \( B \subseteq V(\mathcal{G}) \). This section covers how MaxClqReg integrates the bounding-by-partitioning method into the general branching scheme described in Section VI-A.

In recent years, the most efficient MCP algorithms, such as, e.g., [44, 65], employ a so-called incremental branching scheme. Incremental branching is reminiscent of the Russian doll search paradigm [71], where small subproblems (the small dolls) are solved first and bound the search of bigger subproblems (the bigger dolls).

In our case, given a search node \((\mathcal{G}, K, q)\) of MaxClqReg and the sets of vertices \( P \) and \( B = \{b_1, b_2, \ldots, b_{|B|}\} \) sorted according to a predetermined initial order (discussed in Section VI-D), the incremental branching scheme examines the child subproblems \( \mathcal{G}_b \), \( b \in B \), in order of increasing size as follows:

\[
\mathcal{G}_b = \mathcal{G}[P \cup \{b\}], \forall i \in \{1, \ldots, |B|\}.
\]

According to Equation (10), \( \mathcal{G}_b \) is the subproblem graph (doll) induced by the vertices in \( P \) together with those vertices that precede \( b \) in \( B \). We refer the interested reader to [44, 65] for further details on incremental branching.

However, this approach is not suitable for MaxClqReg because, by first solving the small subproblems, there is a higher probability that the maximal cliques enumerated first will also be small and have fewer inliers (see Observation 2). Consequently, MaxClqReg implements the bounding-by-partitioning method using a different strategy. Specifically, it examines subproblem dolls in order of decreasing size as follows:

\[
\mathcal{G}_b = \mathcal{G}[P \cup \{b_1, \ldots, b_i\}], \forall i \in \{|B|, \ldots, 1\}.
\]

Equation (11) states that the first subproblem examined by MaxClqReg contains all cliques with vertex \( b_{|B|} \), the second subproblem contains all cliques with vertex \( b_{|B|-1} \) and not \( b_{|B|} \) (since all the cliques with \( b_{|B|} \) have already been examined in the previous subproblem), etc. Examining subproblems in this order enumerates large maximal cliques earlier than with the incremental branching approach, and therefore guides the search better. Further details are provided in the pseudocode of MaxClqReg in Section VI-E.

### D. Preprocessing

The main components of the preprocessing phase of MaxClqReg are i) determining an initial (large) clique \( K_{\max} \) and ii) determining an appropriate initial order for the set of vertices \( V(\mathcal{G}) \). To compute the initial clique, MaxClqReg uses the adaptive multi-start tabu heuristic AMTS [77], also used in other efficient MCP algorithms such as, e.g., [65]. In what follows, we describe the initial sorting strategy for the vertices.

It is well established in the literature that the way vertices are initially sorted is an important source of efficiency in maximum clique algorithms; see, e.g., [49, 61]. Specifically, recent efficient MCP exact algorithms employ two main types of vertex ordering strategies: i) a degenerate ordering based on the degree of the vertices, so-called DEG-SORT (allegedly, first introduced in [18] for the MCP) and ii) an ordering based on vertex colouring, so-called COLOUR-SORT (allegedly, first introduced in [43] for the MCP).

The term degenerate in i) indicates that the sorting criterion (vertex degree) is dynamic, i.e., it is recalculated on the remaining unsorted vertices each time a vertex is selected. In its basic form, the vertex set \( V(\mathcal{G}) = \{v_1, v_2, \ldots, v_n\} \) determined by DEG-SORT is such that \( v_n \) is a vertex with the smallest degree in \( \mathcal{G} \), \( v_{n-1} \) is a vertex with the smallest degree
in the induced graph \( \mathcal{G}(V(\mathcal{G}) \setminus \{v_n\}), v_{n-2} \) is a vertex with the smallest degree in the induced graph \( \mathcal{G}(V(\mathcal{G}) \setminus \{v_n, v_{n-1}\}) \), etc. Unlike \textsc{deg-sort}, the basic form of \textsc{colour-sort} partitions the set \( V(\mathcal{G}) \) into (large) independent sets (colour sets) and sorts the vertices by nondecreasing colour number.

According to the tests reported in [61] on a dataset of dense synthetic graphs of small and medium size, selecting \textsc{deg-sort} or \textsc{colour-sort} can, in some cases, make a big difference in performance. However, we have not found significant differences between both strategies in the extensive preliminary tests carried out in this work, possibly because the reported correspondence graphs are sparse; see Table I. Consequently, we have decided to use only \textsc{deg-sort} in the reported set of experiments in Section VI.

E. The algorithm

Algorithm 3: The clique algorithm MaxCliqueReg executed by CliReg (Algorithm 1 Step 4).

\begin{algorithm}
\begin{algorithmic}
\State \textbf{Input:} A correspondence graph \( \mathcal{G} = (V, E) \)
\State \textbf{Output:} A maximal clique in \( K_{\text{reg}} \)
\Type: \( (v_1, \ldots, v_n), K_{\text{max}} \leftarrow \text{Preproc}(\mathcal{G}) \) \Comment{Sec. VI-D}
\For {\( i \leftarrow n \) \textbf{to} \( |K_{\text{max}}| + 1 \) \textbf{do}}
\State \( \widetilde{V} \leftarrow \{v \in V_{i-1}(\mathcal{G}) : \{v, v_i\} \in E\} \)
\State \( \text{EnumCLQ}(\mathcal{G}[\widetilde{V}], \{v_i\}, |K_{\text{max}}| - 1) \)
\EndFor
\State \( \text{EnumCLQ}(\mathcal{G}, K, q) \)
\State \( P_s, B_s \leftarrow \text{PartCOL}(V(\mathcal{G}), q) \) \Comment{Sec. VI-B}
\If {\( B_s = \emptyset \) \textbf{return}}
\State \( P_s, B_s \leftarrow \text{PartSAT}(P_s, B_s, q) \) \Comment{Sec. VI-B}
\EndIf
\For {\( b \leftarrow B_s \textbf{to} 1 \) \textbf{do}}
\State \( \mathcal{V} \leftarrow \{N(b_i) \cap \{P_s \cup \{b_1, \ldots, b_{|P_s|}\}\} : \ b_i \in B_s \}
\State \( \overline{K} \leftarrow K \cup \{b_i\} \)
\If {\( \mathcal{V} = \emptyset \)}
\If {\( |\overline{K}| > |K_{\text{max}}| \) \textbf{then}}
\State \( K_{\text{max}} \leftarrow K \)
\ElseIf {\( \text{CliEval}(\overline{K} > \text{CliEval}(K_{\text{reg}}) \textbf{then}}
\State \( K_{\text{reg}} \leftarrow K \) \Comment{Sec. VI-B}
\Else
\State \( \text{EnumCLQ}(\mathcal{G}[\mathcal{V}], \overline{K}, |K_{\text{max}}| - |\overline{K}|) \)
\EndIf
\EndIf
\EndIf
\EndFor
\end{algorithm}
\end{algorithm}

The pseudocode for MaxCliqueReg is provided in Algorithm 3. Preprocessing (described in the previous Section VI-D) is executed by the procedure \textsc{Preproc} in Step 1. As a result, the vertices of the graph are sorted, and an initial large clique is computed. Next, the algorithm branches at the root node selecting one vertex at a time in reverse initial order, i.e., starting from \( v_n \). For each branched vertex \( v_i \), it determines the associated child subproblem in Step 8 (according to the general branching scheme described in Section VI-A and calls the recursive function \textsc{EnumCLQ} in Step 4. The notation \( V_s(G), s \in \{1, \ldots, |V(G)|) \) refers to the set of vertices that precede \( v_s \) in the initial order, together with \( v_s \), i.e., \( V_s(G) = \{v_1, \ldots, v_s\} \).

The recursive function \textsc{EnumCLQ} starts by executing the bounding-by-partitioning scheme described in Section VI-B. Specifically, colour-based partitioning is implemented by the function \textsc{PartCOL} (Step 6), and partial maximum satisfiability-based partitioning is implemented by the function \textsc{PartSAT} (Step 8). If, as a result, the branching set becomes empty, the node is fathomed (Steps 7 and 9).

If the node has not been pruned at this point, MaxCliqueReg branches according to the nonincremental branching strategy described in VI-C. Specifically, a vertex \( b_i \) is selected from the branching set \( B_s \), the vertex set \( V \) of the associated child subgraph \( \mathcal{G} \) is determined (Step 11) and the enlarged clique \( K \cup \{b_i\} \) is evaluated according to Section VI-B (Step 16). It is worth mentioning that the definition of the vertex set \( V \) in Step 11 is consistent with Equation (11). Precisely, \( \mathcal{G}_{b_i}, b_i \in B, vi \in \{1, \ldots, 1\} \), in Equation (11) is the same graph as \( \mathcal{G}[N(b_i) \cap \{P_s \cup \{b_1, \ldots, b_{|P_s|}\}\}] \) in Step 11 since branching on \( b_i \)&s fixes the vertex \( b_i \) in the clique, and, therefore, its associated subproblem \( \mathcal{G}_{b_i} \) can only be induced by vertices adjacent to \( b_i \).

At the end of this process, the algorithm calls the recursive function \textsc{EnumCLQ} in Step 19 to continue the search in the corresponding child node.

VII. EXPERIMENTAL RESULTS

In this section, we report the results of an extensive computational campaign. The goal of this analysis is twofold: i) to provide information on the main components of CliReg, i.e., the correspondence graph, the maximal clique enumeration and the \textsc{fitnessCLQ} clique evaluation function; ii) to compare CliReg against previous state-of-the-art algorithms in the literature. All tests have been carried out on a Linux workstation with an Intel(R) CPU i9-12900K and 16GB of RAM. CliReg is implemented in C++ and has been compiled with gcc 11.4.0 using the -O3 optimization flag.

To ensure a fair evaluation, a Python framework has been developed that executes the following 3 phases in order: i) Data acquisition, where our point cloud dataset is generated and pre-processed, ii) Execution of the algorithms, and iii) Report generation and evaluation of results. This workflow is shown in Figure 6 and is explained in the following.

A. Dataset of instances and experimental settings

Our reference triangle meshes used in the experiments comprises four 3D models (armadillo, buddha, dragon and bunny) from the Stanford 3D scanning repository [1]; see Figure 7. These models consist of a significant number of vertices, ranging from 35,947 for the smallest model (bunny) to 543,652 for the largest model (buddha). During the data acquisition phase, we first downsample the reference triangle meshes (the \textsc{Voxel Downsample} procedure in Figure 6) to derive the target point cloud \( Q \) representing the entire object with the same resolution as the simulated Time-of-Flight (ToF) camera used to obtain different views of \( Q \).
Specifically, the camera captures distance data with a resolution of $176 \times 144$ pixels and Gaussian noise with a mean ($\mu$) of 0.0 meters and a standard deviation ($\sigma$) of 0.004 meters. Camera simulation is performed using the Blensor tool [30] in Blender\[1\]. With this camera, each mesh is sampled to generate the moving point clouds $M$ of our dataset. Specifically, the framework produces partial views of $Q$ taken from 15 different perspectives of each 3D model (totalling $4 \times 15 = 60$ moving clouds $M$). This is accomplished in the Scan Simulation procedure of Figure 6. In the figure, the green bunny output of the camera scan procedure is an example of a cloud $M$, a partial view of the red bunny target cloud $Q$.

To evaluate the reported algorithms, a known rigid transformation is applied to each cloud $M$ (procedure Cloud Transformation in Figure 6). This initial transformation is unknown to all algorithms and serves as a baseline for evaluation purposes. Finally, we synthetically generate outliers randomly by moving a subset of points in the cloud $M$, producing outlier rates ranging from 10% to 80%. It should be highlighted that our dataset aims to approximate to the real-world by simulating the ToF camera, and not just isotropic Gaussian noise. Specifically, the camera samples the mesh so that the points sensed by the camera (cloud $M$) may not correspond to those of the target cloud $Q$. As described, this step is prior to the synthetic generation of outliers. Our final dataset, including the variations with outliers, comprises $60 \times 9 = 540$ instances, 135 instances for each of the four 3D original models.

Fig. 6: Workflow of our Python framework for scan-matching tests, showing an example bunny instance. In red and green are the target $Q$ and moving $M$ point clouds, respectively.

Fig. 7: Our point cloud dataset from Stanford University [1]: armadillo (a), buddha (b), dragon (c) and bunny (d).

The settings for the algorithm execution phase are as

https://www.blender.org/
follows. As mentioned in the beginning of the section, all algorithms are executed on a single thread. The two point clouds $M$ and $Q$, together with a set of local geometric features for each point extracted by the FPFH algorithm [56], are provided to the reported algorithms in the same format (two arrays with the coordinates of the points and the features associated with each point, respectively).

To obtain a robust evaluation, we report average results for 40 runs of the same instance for all tested algorithms. The 40 outputs for each instance should be exactly the same for all tested algorithms. The time limit was fixed at 10 seconds in all the runs.

The 540 instance dataset is publicly available on GitHub.

B. Analysis of the main components of CliReg

In this section, we report results that illustrate the operations of the main components of CliReg. We begin by describing the variant CliRegMutual of CliReg in what follows.

As argued previously, the function genCS of CliReg (executed at Step 1 of Algorithm 1) determines the correspondence set $C$ of the registration problem with the help of the local geometric features determined by FPFH [56]. We also recall that the matching hypotheses in $C$, i.e., the vertices of the correspondence graph, result from a $k$-nearest-neighbours analysis and that $k = 3$ was found to be a good value for CliReg in our extensive preliminary tests. Figure 8a shows the set of correspondences determined by CliReg for an example bunny instance from our dataset.

We further consider the variant CliRegMutual that only considers correspondence pairs $(m, q), m \in M, q \in Q$, in the correspondence set such that $m$ has $q$ as closest point in the feature space and, vice versa, $q$ has $m$ as its closest point. If, given a point $m \in M$, the property does not hold for any point in $Q$, then no matching hypothesis for $m$ is added to $C$. As a result, the correspondence graph generated by CliRegMutual has at most $|M| \times k$ vertices (not $|M| \times k, k = 3$, as CliReg) and is expected to have a shorter running time than CliReg at the cost of providing a less robust registration. Figure 8b shows the correspondence set determined by CliRegMutual for the previous bunny instance.

In addition, Figure 8c shows the matched pairs, vertices of the clique returned by CliReg (Step 4 of Algorithm 1), and Figure 8d shows the registered (overlapping) clouds. In the latter figure, the inliers of the correspondence set in Figure 8a appear in blue.

Table 1 provides information on the main components of CliReg for the bunny model. The table shows aggregated results for the 9 different outlier rates tested. For each outlier rate, the table reports the size $n$ and average density $d$ of the correspondence graph $G$, its clique number $\omega(G)$, the size of the maximal clique $|K_{reg}|$ output of MaxClqReg and the number of inliers $|I_C|$ in the correspondence set $C$ for the registration determined by $K_{reg}$, i.e., the output $\phi$ of our fitnessCLQ function (Step 2 of Algorithm 2) for the maximal clique $K_{reg}$. Furthermore, the table also reports the total number of inliers $|I_M|$ in the moving cloud $M$. Precisely, $I_M$ is the set of points in $M$ that find a point in $Q$ at a distance less than the predetermined parameter $\xi (6)$. It should be mentioned that the set $I_M$ is independent of the local features determined by the FPFH algorithm [56], while the set of inliers $I_C$ is feature-dependent. Finally, the table reports the original number of inliers $|I_M|$, i.e., the points in the cloud $M$ that were not transformed into outliers during the Data Acquisition phase of our test framework. $|I_M|$ is an upper bound on $|I_M|$ by construction.

From Table 1 several conclusions can be drawn. The first result is that the mean values of the maximal clique size $|K_{reg}|$ computed by CliReg are lower than the clique numbers $\omega(G)$ of the correspondence graphs $G$ in all cases. The difference is greater than $3 \times$ when there are no outliers present and gradually decreases as the outlier rate increases. This empirically shows that optimal registration is not associated to a maximum clique in the correspondence graph in the general case. As argued in previous sections, this can be explained by the fact that the vertices of the graph (matching hypotheses) are based on the similarity of local features, not global ones. We believe this to be an important result and contribution of this work; In previous clique-based approaches to the registration problem, such as, e.g., TEASER++ [79], only maximum cliques were considered.

The second result is that CliReg performs very well in the presence of outliers. Specifically, for experiments with outlier rates ranging from 0% to 40%, the registration is close to optimal, i.e., the number of inliers $|I_M|$ is very close to the threshold $|I_M|$.

The third and last result is the large difference between the number of inliers $|I_C|$ found in the correspondence set (as determined by our fitnessCLQ function in Step 2 of CliEval) and the actual number of inliers $|I_M|$. This discrepancy indicates that the quality of candidate matching hypotheses based on local features is poor, which is consistent with the literature, such as, e.g., [81] [69]. Moreover, it empirically justifies the need for a robust methodology against outliers, as proposed in this work. Motivated by this result, we have redefined our fitnessCLQ function to return the number of inliers in $M$, i.e., matches $(m, q), m \in M, q \in Q$ at a distance less than $\xi$, for the transformation associated to $K$.

The main components of CliReg produce similar results in the other 3 models (armadillo, buddha, and dragon) so we do not comment on them here. The results are available in Appendix A.

C. Comparison of CliReg with state-of-the-art algorithms

We have carried out an extensive experimental campaign to compare the efficiency of CliReg with other state-of-the-art algorithms. In addition to CliRegMutual, our proposed variant of CliReg designed for fast execution (see Section VII-B), we have considered the following 4 reference algorithms:

https://github.com/jlaserna/CliReg-Matcher.git
Fig. 8: An example showing the main components of CliReg, and its variant CliRegMutual; in red the bunny target point cloud \( Q \) and, in green, the moving point cloud \( M \) (a partial view of the bunny). (a) the correspondence set of CliReg; (b) the correspondence set of CliRegMutual (smaller than (a)); (c) the matched correspondences, vertices of the clique determined by CliReg; (d) the registered overlapping clouds \( M \) (green) and \( Q \) (red). The inlier correspondences are shown in blue.

TABLE I: CliReg component results on our 135 subset of bunny instances.

| outliers | n \( \mu \)  | d \( \mu \)  | \( \omega(G) \mu \)  | \( |K_{reg}| \mu \)  | \( |I_{C}| \mu \)  | \( |I_{M}| \mu \)  | \( |I_{M}| \mu \)  |
|----------|----------|----------|-----------------|-----------------|-----------------|-----------------|-----------------|
| 10%      | 5860.1   | 625.1    | 0.049           | 0.002           | 84.7            | 12.6            | 168.6           | 56.2            | 2666.5          | 277.9           | 2672.5          |
| 20%      | 5613.1   | 656.9    | 0.036           | 0.002           | 64.5            | 16.4            | 23.6            | 16.2            | 118.1           | 39.2            | 2406.5          | 247.8           | 2407.6          |
| 30%      | 6370.1   | 656.9    | 0.026           | 0.001           | 40.4            | 11.0            | 23.3            | 8.4             | 66.8            | 22.2            | 2132.9          | 224.1           | 2138.4          |
| 40%      | 6622.6   | 682.7    | 0.019           | 0.001           | 27.6            | 9.4             | 17.7            | 10.1            | 42.7            | 21.5            | 1862.5          | 190.0           | 1871.1          |
| 50%      | 6898.5   | 703.6    | 0.015           | 0.001           | 19.9            | 7.5             | 14.6            | 5.3             | 28.1            | 16.5            | 1516.3          | 196.2           | 1604.0          |
| 60%      | 7197.8   | 739.9    | 0.011           | 0.001           | 14.3            | 4.4             | 11.8            | 3.7             | 15.4            | 7.2             | 990.9           | 371.6           | 1336.5          |
| 70%      | 7503.7   | 770.3    | 0.008           | 0.001           | 10.7            | 2.9             | 9.6             | 3.0             | 9.2             | 9.1             | 445.1           | 395.3           | 1069.3          |
| 80%      | 7733.0   | 779.6    | 0.006           | 0.001           | 8.7             | 0.7             | 7.3             | 1.4             | 4.7             | 3.3             | 172.1           | 93.2            | 802.4           |

- RANSAC [27]: Our reference nondeterministic algorithm. Specifically, we have used its feature-based variant with 99.9% confidence configured according to the specifications defined in the Open3D documentation\(^3\) i.e., a maximum number of 100,000 iterations and the mutual feature filter enabled. Finally, the distance threshold parameter is set to the value \( \xi \) (6) for each instance.

- FGR [83]: A fast variant of the RANSAC algorithm and with the same settings.

- TEASER++ [79]: To the best of our knowledge, the reference clique-based algorithm. We used as settings those in the original GitHub repository\(^4\) with the noise bound parameter fixed to \( \xi \) (6).

- Go-ICP [80]: A global variant of the well known ICP algorithm. In our extensive preliminary tests, the algorithm performed very poorly compared to CliReg with default parameter settings\(^5\). Since no additional information from the developers was received, we finally decided not to include Go-ICP in this report.

The fitness metric used to compare the results among the algorithms is the number of inliers \( |I_M| \) in the moving cloud \( M \), normalized between 0 and 1:

\[
\text{fitness} = |I_M| / |I_M|, \quad (0, 1],
\]

where a fitness value of 1 corresponds to an optimal registration. It should be mentioned that only the points that have not become outliers during the Cloud Transformation procedure (see Figure 6) are considered in the fitness metric.

Figures 9 and 10 present boxplots that evaluate the global fitness of the registration and the execution times of the 5 algorithms considered in this report (CliReg, CliRegMutual, FGR, TEASER++, and RANSAC). Specifically, both figures

\(^3\)http://www.open3d.org/docs/release/tutorial/pipelines/global_registration.html

\(^4\)https://github.com/MIT-SPARK/TEASER-plusplus/tree/master/examples/teaser_python_fpffeacute_icp

\(^5\)https://github.com/yangjiadong/Go-ICP/blob/master/config_example.txt
show a coloured boxplot for each of the 5 algorithms and percentage of outliers. We recall from Section VII-A that our bunny dataset comprises 15 instances, each corresponding to a different camera view of the Bunny, that for each view there are 9 different outlier configurations. The results are averaged over 40 runs. Since the behavior of the 5 algorithms for the remaining 3 models (armadillo, buddha, and dragon) shows a similar trend to that of the bunny, we have decided to restrict our analysis to the latter in what follows. The full report is available in the Appendix A.

Figure 9 shows the fitness distribution with respect to the percentage of outliers. The figure shows that CliReg clearly dominates the other algorithms in terms of robustness and accuracy (FGR, TEASER++, and RANSAC) in the range of 0%-50% of outliers, and the same applies for CliRegMutual in the range 0%-30%. In the presence of outlier rates greater than 50%, the robustness of all algorithms drops drastically to fitness values below 0.5, showing that our dataset is hard. The figure also shows that nondeterministic algorithms, i.e., FGR and RANSAC, tend to have a high fitness variance.

![Fig. 9: Fitness of the registration provided by the different algorithms for our 135 bunny instances.](image)

In addition, we performed a variance analysis (ANOVA) of the fitness and execution times of the algorithms with up to 50% of outliers, considering the full 540 instance database. The analysis showed statistically significant differences between the algorithms in both cases, i.e., p-value < 0.005; see Table V in the Appendix A. To further elucidate these differences, we conducted Tukey post hoc tests between each pair of algorithms, with a 95% confidence level and the same outlier rates; see Tables VI and VII in the Appendix A. According to the latter tests, the 5 reported algorithms can be classified into the following 4 groups, ordered from best to worst. Related to fitness, groups (a) CliReg, (b) RANSAC, (c) CliRegMutual and TEASER++, and (d) FGR; see Figure 11. Related to execution times, groups (a) CliRegMutual and FGR, (b) TEASER++, (c) CliReg, and (d) RANSAC; see Figure 12. Both figures show boxplots for the fitness and execution times, respectively, together with the mean (marked with a red rhombus) and group. For example, FGR is classified under group (d) in Figure 11 and its fitness distribution has a mean value of 0.29.

Based on the ANOVA analysis, the following conclusions can be drawn. In terms of robustness, CliReg clearly dominates the rest, i.e., it is the only algorithm in the group (a), has a mean fitness value of 0.87 and shows reduced variability. The second-best algorithm is RANSAC, with a mean value of 0.71, closely followed by CliRegMutual and TEASER++, with a similar mean value (0.68) but greater variability. In terms of execution times, the two fastest algorithms are CliRegMutual and FGR with average running times slightly above 0.5 seconds. Moreover, CliRegMutual finds the best compromise between speed and robustness, since FGR is clearly not robust (its mean fitness value is 0.28).

### VIII. Conclusions and future work

In this research, we have proposed a methodology for fast and robust rigid registration problems in the presence of a large number of outlier correspondences. Our methodology leverages graph theory (e.g., theoretical bounds for the maximum clique problem), efficient techniques employed by state-of-the-art maximum clique algorithms (e.g., vertex ordering, branching by partitioning, incremental branching), and continuous optimization (e.g., solving a least-squares problem to evaluate cliques). These ideas have been implemented in two algorithms, CliReg is efficient and robust, but considers a larger number of hypotheses than CliRegMutual, so the latter is faster but less robust. Both algorithms have been extensively evaluated in a scan-matching dataset generated...
registration problems. Moreover, it would be interesting to consider the behavior of the algorithms in the correspondence-free case using all possible matching hypotheses. Alternative future work is the analysis of weights in the correspondence graph to capture, for example, the quality of individual features according to the feature extractor algorithm. In this setting, the enumeration of weighted maximal cliques could be used to filter out keypoints that are less representative.

ACKNOWLEDGMENTS

This publication is part of the R&D project "Cognitive Personal Assistance for Social Environments (ACOGES)", reference PID2020-113096RB-I00, funded by MCIN/AEI/10.13039/501100011033.

IX. BIBLIOGRAPHY

REFERENCES


from well-known public 3D models. The reported results show that the new algorithms dominate the state of the art in terms of robustness, in the presence of up to 50% of outliers (CliReg also in the 50% value). Regarding preformance, CliRegMutual can run in milliseconds and is comparable to the fastest heuristic FGR. Consequently, the combined two algorithms can be considered to be today’s most efficient approach for robust registration with outlier correspondences. This work opens the way for research on new clique-based ideas for robust registration, such as, e.g., tailoring the clique search procedure or improving the fitness clique evaluation function to make it less sensitive to outliers. Interesting future work includes the study of alternative definitions of $\xi$, the analysis of different point features, or the addition of a preliminary step to estimate the scale in the case of non-rigid
APPENDIX A
ADDITIONAL RESULTS

This appendix covers the results of the tests not shown in the experimental Section VII. The report includes the fitness and execution times of the different algorithms for the subset of armadillo instances (Figures 13 and 16), buddha instances (Figures 14 and 17), and dragon instances (Figures 15 and 18), respectively. The appendix also shows the results of the components of CliReg on the subset of armadillo instances (Table II), the buddha instances (Table III) and the dragon (Table IV).

Additionally, Table V presents the full results of the analysis of variance (ANOVA) for the fitness of the registration and execution times, considering up to 50% outliers. The first 3 columns report the subset of instances (type), the variable (var) and the source, i.e., algorithms or residuals. The type total refers to the whole database (not specific subsets). The next columns show, for each of the two sources, the degrees of freedom (#dof), the sum of squares (SS), the $F$-value (the ratio of the in-between algorithm variation and within algorithm variation of each variable) and the $p$-value. Values of $p$ below 0.005 indicate that the differences observed in the corresponding row are statistically significant. We refer the interested reader to, e.g., [67], for a detailed description of the ANOVA analysis.

Finally, we report the Tukey tests that compare the fitness (Table VI) and the execution times (Table VII) of every pair of algorithms considering the subset of instances with up to 50% outliers. In both tables, the column $\mu_d$ shows the difference between the means of the variable under study for the corresponding pair of algorithms, while $\mu_{ld}$ and $\mu_{ud}$ denote the lower and upper confidence intervals for $\mu_d$, respectively. The final column of the table displays the $p$-value.
TABLE II: CliReg component results on our 135 armadillo instances.

<table>
<thead>
<tr>
<th>outliers</th>
<th>n</th>
<th>d</th>
<th>ω (G)</th>
<th>∣K_{reg}∣</th>
<th>∣I_C∣</th>
<th>∣I_M∣</th>
<th>∣T_M∣</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>μ</td>
<td>σ</td>
<td>μ</td>
<td>σ</td>
<td>μ</td>
<td>σ</td>
<td>μ</td>
</tr>
<tr>
<td>-</td>
<td>4294.5</td>
<td>412.0</td>
<td>0.045</td>
<td>0.003</td>
<td>45.9</td>
<td>13.7</td>
<td>7.7</td>
</tr>
<tr>
<td>10 %</td>
<td>4439.1</td>
<td>412.1</td>
<td>0.036</td>
<td>0.002</td>
<td>39.3</td>
<td>10.9</td>
<td>4.3</td>
</tr>
<tr>
<td>20 %</td>
<td>4579.5</td>
<td>421.0</td>
<td>0.028</td>
<td>0.001</td>
<td>29.3</td>
<td>8.7</td>
<td>4.2</td>
</tr>
<tr>
<td>30 %</td>
<td>4746.1</td>
<td>433.4</td>
<td>0.021</td>
<td>0.001</td>
<td>22.5</td>
<td>4.0</td>
<td>3.1</td>
</tr>
<tr>
<td>40 %</td>
<td>4903.9</td>
<td>459.4</td>
<td>0.012</td>
<td>0.001</td>
<td>13.3</td>
<td>2.3</td>
<td>1.4</td>
</tr>
<tr>
<td>50 %</td>
<td>5067.3</td>
<td>495.1</td>
<td>0.009</td>
<td>0.001</td>
<td>9.7</td>
<td>1.1</td>
<td>0.7</td>
</tr>
<tr>
<td>60 %</td>
<td>5218.9</td>
<td>497.9</td>
<td>0.006</td>
<td>0.001</td>
<td>8.2</td>
<td>1.0</td>
<td>0.5</td>
</tr>
<tr>
<td>70 %</td>
<td>5331.2</td>
<td>502.9</td>
<td>0.005</td>
<td>0.001</td>
<td>7.4</td>
<td>0.8</td>
<td>0.4</td>
</tr>
<tr>
<td>80 %</td>
<td>5405.4</td>
<td>502.9</td>
<td>0.005</td>
<td>0.001</td>
<td>7.4</td>
<td>0.8</td>
<td>0.4</td>
</tr>
</tbody>
</table>

TABLE III: CliReg component results on our 135 buddha instances.

<table>
<thead>
<tr>
<th>outliers</th>
<th>n</th>
<th>d</th>
<th>ω (G)</th>
<th>∣K_{reg}∣</th>
<th>∣I_C∣</th>
<th>∣I_M∣</th>
<th>∣T_M∣</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>μ</td>
<td>σ</td>
<td>μ</td>
<td>σ</td>
<td>μ</td>
<td>σ</td>
<td>μ</td>
</tr>
<tr>
<td>-</td>
<td>5065.3</td>
<td>947.3</td>
<td>0.039</td>
<td>0.004</td>
<td>36.8</td>
<td>13.7</td>
<td>7.7</td>
</tr>
<tr>
<td>10 %</td>
<td>5201.8</td>
<td>966.4</td>
<td>0.031</td>
<td>0.002</td>
<td>33.7</td>
<td>10.4</td>
<td>7.0</td>
</tr>
<tr>
<td>20 %</td>
<td>5377.9</td>
<td>981.5</td>
<td>0.024</td>
<td>0.002</td>
<td>26.7</td>
<td>7.9</td>
<td>6.3</td>
</tr>
<tr>
<td>30 %</td>
<td>5477.9</td>
<td>1011.9</td>
<td>0.018</td>
<td>0.001</td>
<td>20.1</td>
<td>5.0</td>
<td>3.1</td>
</tr>
<tr>
<td>40 %</td>
<td>5621.3</td>
<td>1044.5</td>
<td>0.013</td>
<td>0.001</td>
<td>15.7</td>
<td>4.0</td>
<td>3.1</td>
</tr>
<tr>
<td>50 %</td>
<td>5752.3</td>
<td>1061.4</td>
<td>0.010</td>
<td>0.001</td>
<td>13.1</td>
<td>3.1</td>
<td>2.2</td>
</tr>
<tr>
<td>60 %</td>
<td>5897.6</td>
<td>1094.7</td>
<td>0.007</td>
<td>0.001</td>
<td>10.0</td>
<td>2.3</td>
<td>2.0</td>
</tr>
<tr>
<td>70 %</td>
<td>6007.2</td>
<td>1103.7</td>
<td>0.005</td>
<td>0.001</td>
<td>8.5</td>
<td>1.2</td>
<td>1.7</td>
</tr>
<tr>
<td>80 %</td>
<td>6084.3</td>
<td>1188.6</td>
<td>0.003</td>
<td>0.001</td>
<td>7.1</td>
<td>0.8</td>
<td>1.3</td>
</tr>
</tbody>
</table>

TABLE IV: CliReg component results on our 135 dragon instances.

<table>
<thead>
<tr>
<th>outliers</th>
<th>n</th>
<th>d</th>
<th>ω (G)</th>
<th>∣K_{reg}∣</th>
<th>∣I_C∣</th>
<th>∣I_M∣</th>
<th>∣T_M∣</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>μ</td>
<td>σ</td>
<td>μ</td>
<td>σ</td>
<td>μ</td>
<td>σ</td>
<td>μ</td>
</tr>
<tr>
<td>-</td>
<td>7107.333</td>
<td>1128.690</td>
<td>0.036</td>
<td>0.002</td>
<td>38.600</td>
<td>11.141</td>
<td>9.410</td>
</tr>
<tr>
<td>10 %</td>
<td>7333.267</td>
<td>1183.741</td>
<td>0.028</td>
<td>0.001</td>
<td>31.000</td>
<td>8.281</td>
<td>7.331</td>
</tr>
<tr>
<td>20 %</td>
<td>7560.467</td>
<td>1210.392</td>
<td>0.022</td>
<td>0.001</td>
<td>24.533</td>
<td>5.805</td>
<td>6.925</td>
</tr>
<tr>
<td>30 %</td>
<td>7776.200</td>
<td>1261.845</td>
<td>0.017</td>
<td>0.001</td>
<td>19.200</td>
<td>5.809</td>
<td>5.713</td>
</tr>
<tr>
<td>40 %</td>
<td>8034.000</td>
<td>1298.262</td>
<td>0.013</td>
<td>0.001</td>
<td>14.733</td>
<td>5.918</td>
<td>5.066</td>
</tr>
<tr>
<td>50 %</td>
<td>8297.200</td>
<td>1356.043</td>
<td>0.010</td>
<td>0.001</td>
<td>10.333</td>
<td>5.995</td>
<td>4.993</td>
</tr>
<tr>
<td>60 %</td>
<td>8537.600</td>
<td>1387.834</td>
<td>0.007</td>
<td>0.001</td>
<td>7.833</td>
<td>5.799</td>
<td>4.200</td>
</tr>
<tr>
<td>70 %</td>
<td>8732.200</td>
<td>1422.208</td>
<td>0.005</td>
<td>0.001</td>
<td>7.867</td>
<td>0.743</td>
<td>4.200</td>
</tr>
<tr>
<td>80 %</td>
<td>8843.933</td>
<td>1437.052</td>
<td>0.003</td>
<td>0.001</td>
<td>7.100</td>
<td>0.676</td>
<td>4.200</td>
</tr>
</tbody>
</table>

Fig. 16: Execution times of the different algorithms for our 135 armadillo instances.

Fig. 17: Execution times of the different algorithms for our 135 buddha instances.
Fig. 18: Execution times of the different algorithms for our 135 dragon instances.

TABLE V: ANOVA fitness and execution time analysis with up to 50% of outliers.

<table>
<thead>
<tr>
<th>type</th>
<th>var</th>
<th>source</th>
<th>#dof</th>
<th>SS</th>
<th>F-value</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>armadillo</td>
<td>fitness</td>
<td>algorithms residuals</td>
<td>4</td>
<td>629.468</td>
<td>1964.434</td>
<td>0.000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1441.544</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>time (s)</td>
<td>algorithms residuals</td>
<td>4</td>
<td>1463.764</td>
<td>35194.137</td>
<td>0.000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>187.108</td>
<td></td>
<td></td>
</tr>
<tr>
<td>bunny</td>
<td>fitness</td>
<td>algorithms residuals</td>
<td>4</td>
<td>912.843</td>
<td>3284.009</td>
<td>0.000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1250.500</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>time (s)</td>
<td>algorithms residuals</td>
<td>4</td>
<td>2311.257</td>
<td>757.941</td>
<td>0.000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>13718.436</td>
<td></td>
<td></td>
</tr>
<tr>
<td>buddha</td>
<td>fitness</td>
<td>algorithms residuals</td>
<td>4</td>
<td>553.458</td>
<td>2158.177</td>
<td>0.000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1153.691</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>time (s)</td>
<td>algorithms residuals</td>
<td>4</td>
<td>2140.514</td>
<td>38378.941</td>
<td>0.000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>250.909</td>
<td></td>
<td></td>
</tr>
<tr>
<td>dragon</td>
<td>fitness</td>
<td>algorithms residuals</td>
<td>4</td>
<td>660.924</td>
<td>2064.474</td>
<td>0.000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1440.237</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>time (s)</td>
<td>algorithms residuals</td>
<td>4</td>
<td>2217.123</td>
<td>32173.394</td>
<td>0.000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>310.016</td>
<td></td>
<td></td>
</tr>
<tr>
<td>total</td>
<td>fitness</td>
<td>Residuals</td>
<td>71995</td>
<td>2673.998</td>
<td>8666.490</td>
<td>0.000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5553.416</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>time (s)</td>
<td>algorithms residuals</td>
<td>4</td>
<td>6958.314</td>
<td>6946.757</td>
<td>0.000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>18028.692</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

TABLE VI: Tukey test for the fitness variable on the subset of instances with up to 50% of outliers.

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>$\mu_d$</th>
<th>$\mu_d$</th>
<th>$\mu_d$</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CliRegMutual</td>
<td>-0.332</td>
<td>-0.348</td>
<td>-0.316</td>
<td>0.000</td>
</tr>
<tr>
<td>FGR</td>
<td>-0.326</td>
<td>-0.342</td>
<td>-0.309</td>
<td>0.000</td>
</tr>
<tr>
<td>RANSAC</td>
<td>0.519</td>
<td>0.503</td>
<td>0.535</td>
<td>0.000</td>
</tr>
<tr>
<td>TEASER++</td>
<td>-0.090</td>
<td>-0.106</td>
<td>-0.074</td>
<td>0.000</td>
</tr>
<tr>
<td>FGR</td>
<td>0.006</td>
<td>0.010</td>
<td>0.022</td>
<td>0.825</td>
</tr>
<tr>
<td>RANSAC</td>
<td>0.851</td>
<td>0.835</td>
<td>0.867</td>
<td>0.000</td>
</tr>
<tr>
<td>TEASER++</td>
<td>0.242</td>
<td>0.225</td>
<td>0.258</td>
<td>0.000</td>
</tr>
<tr>
<td>FGR</td>
<td>0.845</td>
<td>0.829</td>
<td>0.861</td>
<td>0.000</td>
</tr>
<tr>
<td>TEASER++</td>
<td>0.235</td>
<td>0.219</td>
<td>0.251</td>
<td>0.000</td>
</tr>
<tr>
<td>TEASER++</td>
<td>-0.609</td>
<td>-0.625</td>
<td>-0.593</td>
<td>0.000</td>
</tr>
</tbody>
</table>

TABLE VII: Tukey test for the execution time on the subset of instances with up to 50% of outliers.

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>$\mu_d$</th>
<th>$\mu_d$</th>
<th>$\mu_d$</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CliRegMutual</td>
<td>-0.332</td>
<td>-0.348</td>
<td>-0.316</td>
<td>0.000</td>
</tr>
<tr>
<td>FGR</td>
<td>-0.326</td>
<td>-0.342</td>
<td>-0.309</td>
<td>0.000</td>
</tr>
<tr>
<td>RANSAC</td>
<td>0.519</td>
<td>0.503</td>
<td>0.535</td>
<td>0.000</td>
</tr>
<tr>
<td>TEASER++</td>
<td>-0.090</td>
<td>-0.106</td>
<td>-0.074</td>
<td>0.000</td>
</tr>
<tr>
<td>FGR</td>
<td>0.006</td>
<td>0.010</td>
<td>0.022</td>
<td>0.825</td>
</tr>
<tr>
<td>RANSAC</td>
<td>0.851</td>
<td>0.835</td>
<td>0.867</td>
<td>0.000</td>
</tr>
<tr>
<td>TEASER++</td>
<td>0.242</td>
<td>0.225</td>
<td>0.258</td>
<td>0.000</td>
</tr>
<tr>
<td>FGR</td>
<td>0.845</td>
<td>0.829</td>
<td>0.861</td>
<td>0.000</td>
</tr>
<tr>
<td>TEASER++</td>
<td>0.235</td>
<td>0.219</td>
<td>0.251</td>
<td>0.000</td>
</tr>
<tr>
<td>TEASER++</td>
<td>-0.609</td>
<td>-0.625</td>
<td>-0.593</td>
<td>0.000</td>
</tr>
</tbody>
</table>