CliReg: Clique-Based Robust Point Cloud Registration

Javier Laserna, Pablo San Segundo, and 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 1) 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.; 2) 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; 3) it is possible to have a *good* control of suboptimality with a k-nearest neighbor 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 both on synthetic and real public datasets. 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*—Discrete optimization, maximum clique, mobile robotics, point cloud 3-D registration, scan matching.**

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 3-D models, see, e.g., [6]. Many problems in these fields can be effectively addressed by algorithms operating on point sets. This article focuses on (3-D) 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., 3-D model reconstruction [12], [32], object recognition and localization [7], [34], robot navigation [50], [51] or medical imaging [4], [65].

Point cloud registration has been the subject of extensive research, such as, e.g., [6], [8], [10], [25], [33], [35], [38], [63], [74]. 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 elegant closed-form solutions [3], [31]. 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, a 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` [24], 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 that are far from the ground truth.

An alternative to local methods are global branch-and-bound (BnB) approaches, such as, e.g., `Go-ICP` [76], 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 (see Section III) that is reformulated as a clique search on a graph. In the extensive synthetic and real world tests reported (see Section VII), `CliReg` is shown to be accurate and robust even in the presence of a large number of outliers. Furthermore, we have also implemented a slightly less accurate variant that shows running times comparable to those of the fastest approaches, such as, e.g., the heuristic `FGR` [79].

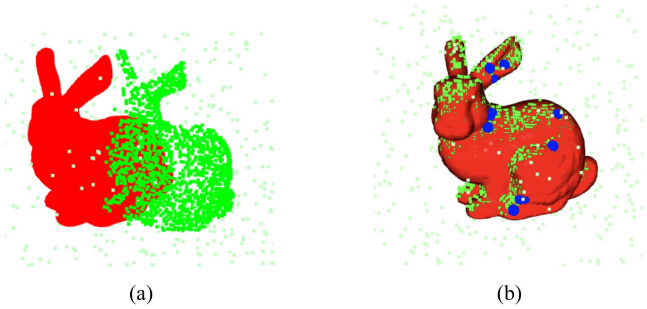
II. Related Work

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

Received 13 September 2024; revised 25 November 2024; accepted 7 December 2024. Date of publication 19 February 2025; date of current version 14 March 2025. This work was supported by the Madrid Government (Comunidad de Madrid-Spain) under the Multiannual Agreement 2023–2026 with Universidad Politécnica de Madrid in the Line A, Emerging Ph.D. researchers. Funding for the open access charge was provided by Universidad Politécnica de Madrid/Consorcio Madroño. This article was recommended for publication by Associate Editor A. Nuechter and Editor J. Civera upon evaluation of the reviewers' comments. *(Corresponding author: Javier Laserna Moratalla.)*

The authors are with the Centre for Automation and Robotics (CAR) ETSIDI, UPMCSIC, Universidad Politécnica de Madrid, 28012 Madrid, Spain (e-mail: j.laserna@upm.es).

Digital Object Identifier 10.1109/TRO.2025.3542954



(a) (b)

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

hypotheses). Corresponding pairs are typically established according to a local 3-D feature extraction algorithm, such as, e.g., fast point feature histograms FPFH [53], 3dmatch [77], or deep learning-based FCGF [22], and registration is restricted to keypoint correspondences with similar features (Fig. 1 provides an example). It should be mentioned that 3-D features are expected to be less accurate compared to their 2-D counterparts, e.g., SIFT [44], 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.

One of the most common algorithms used for robust registration is random sample consensus (RANSAC) [24]. RANSAC is a nondeterministic method that at each iteration samples the correspondence space and uses a closed-form solution [31] 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, Zhou et al. [79] introduced fast global registration (FGR), an efficient approach that uses the Geman–McClure robust least-squares cost function, reformulates it with Black–Rangarajan duality [11], and uses the Gauss–Newton method to solve the model efficiently.

Among the numerous local methods proposed in the literature, the iterative closest point (ICP) algorithm [10], [18], [78] 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., [19], [27], [45]. 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., [15], [16], [37], [72]. An interesting survey on this topic is presented by Chin et al. [20]. The literature reports interesting bounds that involve geometric techniques, such as, e.g., [13], [30], or those proposed in the algorithm Go-ICP [76]. 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 nonconvexity 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 [74]. TEASER operates according to the following steps. In a first step, the registration problem is reformulated using a truncated least squares 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 named TEASER++ [75] uses graduated nonconvexity 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., [2], [5], [23], [70], [71]. According to Yang et al. [75], 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 [22].

A. Our Contribution

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

1) A clique-based methodology for robust rigid correspondence-based point cloud registration is proposed. Similarly to TEASER [74], 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 the ability to easily control suboptimality by fixing the number of matching hypotheses in the correspondence set;
2) 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 (MCP) [41], [54], [55],

[56], [57], [58], [59], [62]. It should be mentioned that prior clique-based approaches (e.g., TEASER [74] and TEASER++ [75]) consider the less efficient maximum clique algorithm PMC [52], which lacks many components of CliReg (e.g., branching by partitioning, partial maximum satisfiability-based bounds etc.); see Section VI 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 VI-C). In previous clique-based approaches (e.g., TEASER [74] and TEASER++ [75]) only a maximum clique was considered;

3) Extensive experiments on standard scan-matching benchmarks, both with real and synthetic data, have been carried out to validate our approach (see Section VII). In addition, and to the best of the authors' knowledge, we are the first to report data concerning the structure of the correspondence graphs.

III. Maximum Consensus Formulation

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

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

$$\boldsymbol{q}_i = \ell \boldsymbol{R}\boldsymbol{m}_i + \boldsymbol{t} + \boldsymbol{\epsilon}_i, \quad i \in \{1, \ldots, |\mathcal{C}|\} \tag{1}$$

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

An important result is that if the Gaussian noise $\boldsymbol{\epsilon}_i$ is assumed to be a zero mean isotropic Gaussian distribution, i.e., $\boldsymbol{\epsilon}_i \sim \mathcal{N}(0, \sigma_i^2 \times \mathbf{I}_3)$, then model (1) can be formulated as the following least-squares optimization problem:

$$\boldsymbol{R}^*, \ \boldsymbol{t}^* = \arg\min_{\boldsymbol{R} \in SO(3), \, \boldsymbol{t} \in \mathbb{R}^3} \ \sum_{i=1}^{|\mathcal{C}|} \frac{1}{\sigma_i^2} ||\boldsymbol{q}_i - \boldsymbol{R}\boldsymbol{m}_i - \boldsymbol{t}||^2 \tag{2}$$

for which a closed-form solution is known, see [3], [31], [69].

When outliers are present, i.e., some of the correspondence pairs given in $\mathcal{C}$ cannot be determined using (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 $\mathcal{I} \subseteq \mathcal{C}$ that is consistent with model (1), i.e.,

$$\max_{\boldsymbol{R} \in SO(3), \boldsymbol{t} \in \mathbb{R}^3} \quad |\mathcal{I}| \tag{3}$$

$$\text{s.t.} \quad \frac{1}{\sigma_i^2} ||\boldsymbol{q}_i - \boldsymbol{R}\boldsymbol{m}_i - \boldsymbol{t}||^2 < \xi \quad \forall (\boldsymbol{m}_i, \boldsymbol{q}_i) \in \mathcal{I}. \tag{4}$$

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

$$\gamma(\mathcal{P}) = \frac{\sum_{\boldsymbol{p} \in \mathcal{P}} d(\boldsymbol{p}, \mathcal{P} \backslash \{\boldsymbol{p}\})}{|\mathcal{P}|} \tag{5}$$

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

$$\xi(\mathcal{M}, \mathcal{Q}) = \max \{\gamma(\mathcal{M}), \gamma(\mathcal{Q})\}. \tag{6}$$

MC is known to be $\mathcal{NP}$-hard [21] 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., [15], [20], [30], [72]. 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 $n = |V|$ vertices and $m = |E|$ edges. Notation $V(G) = V$ and $E(G) = E$ will be used for the vertex and edge sets, respectively, when the graph G is not clear from the context. Two vertices $u, v \in V$ are said to be *adjacent*, also *neighbors*, if there exists an edge $\{u, v\} \in E$. $N(v)$ denotes the *neighbor set* of the vertex v, that is, the subset of vertices in V adjacent to v. The degree of v, denoted $\deg(v)$, is the cardinality of its neighbor 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 MCP consists in finding a maximum clique in G, and the size of any solution is called the *clique number* of the graph $\omega(G)$.

A vertex α-coloring $C_\alpha(G)$ is an assignment of α numbers (colors) to the vertices in G such that no two adjacent vertices have the same color. The minimum number of (different) colors

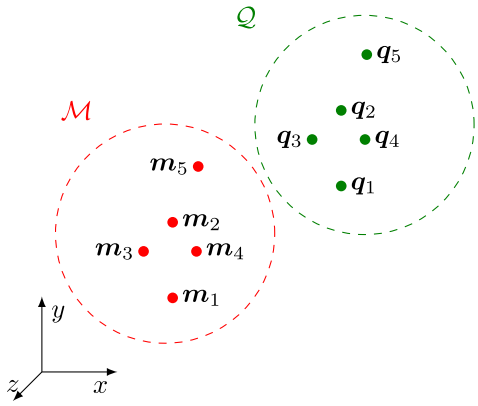


Fig. 2. Example of two point clouds $\mathcal{M}$ and $\mathcal{Q}$.

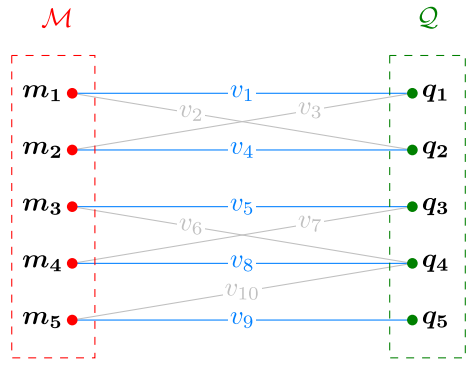


Fig. 3. Set of correspondences $\mathcal{C}$ (also vertices of the correspondence graph) for the point clouds in Fig. 2. In blue, the matches corresponding to the optimal solution to the registration problem.

required to color G is known as the *chromatic number* of the graph $\chi(G)$. Of interest to this work is the fact that the chromatic number is an upper bound on the clique number of the graph, i.e.,

$$\omega(G) \leq \chi(G) \leq \alpha. \tag{7}$$

B. Correspondence Graph

Given a moving and a target 3-D point cloud $\mathcal{M}$ and $\mathcal{Q}$, and a correspondence set $\mathcal{C}$ containing a subset $\mathcal{I} \subseteq \mathcal{C}$ 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 $\mathcal{G} = (V, E)$ as follows. Each vertex of the graph represents a hypothesis, i.e., a candidate (keypoint) pair $(\boldsymbol{m}, \boldsymbol{q})$, $\boldsymbol{m} \in \mathcal{M}$, $\boldsymbol{q} \in \mathcal{Q}$, which are similar according to their features provided by some local 3-D feature extraction algorithm, such as, e.g., `FPFH` [53]. 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 ξ (6). Formally speaking:

$$E(\mathcal{G}) = \{v, w \in V : \; ||\boldsymbol{m}_v - \boldsymbol{m}_w||^2 - ||\boldsymbol{q}_v - \boldsymbol{q}_w||^2 < \xi\} \tag{8}$$

where $(\boldsymbol{m}_v, \boldsymbol{q}_v)$ and $(\boldsymbol{m}_w, \boldsymbol{q}_w)$ 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 (8) is valid only for rigid registration. If the distances between points can change in the transformation between the two point clouds, e.g., there exists a uniform scaling factor ℓ 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 Fig. 2. The figure represents two 3-D point clouds $\mathcal{M}$ and $\mathcal{Q}$ in which the points of interest are colored in red ($|\mathcal{M}| = 5$) and green ($|\mathcal{Q}| = 5$), respectively. In addition, we consider the correspondence set $\mathcal{C}$ which comprises the following ten matching hypotheses:

$$\mathcal{C} = \{(\boldsymbol{m}_1, \boldsymbol{q}_1),\ (\boldsymbol{m}_1,\ \boldsymbol{q}_2),\ (\boldsymbol{m}_2,\ \boldsymbol{q}_1),\ (\boldsymbol{m}_2,\ \boldsymbol{q}_2),\ (\boldsymbol{m}_3,\ \boldsymbol{q}_3),$$
$$(\boldsymbol{m}_3,\ \boldsymbol{q}_4),\ (\boldsymbol{m}_4,\ \boldsymbol{q}_3),\ (\boldsymbol{m}_4,\ \boldsymbol{q}_4),\ (\boldsymbol{m}_5,\ \boldsymbol{q}_4),\ (\boldsymbol{m}_5,\ \boldsymbol{q}_5)\}$$

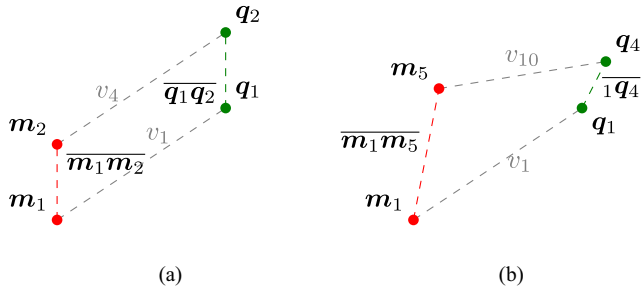


Fig. 4. (a) Example of an edge ($\{v_1, v_4\} \in E(\mathcal{G})$), and (b) a nonedge ($\{v_1, v_{10}\} \notin E(\mathcal{G})$) of the correspondence graph $\mathcal{G}$ associated to the point cloud registration problem defined in Figs. 2 and 3. $\mathcal{G}$ is shown in Fig. 5.

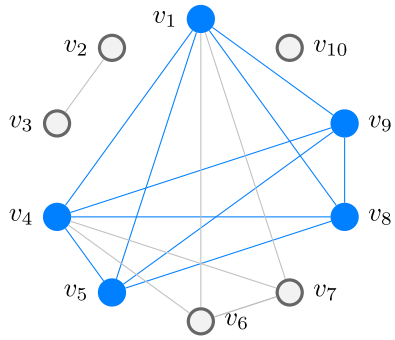


Fig. 5. Correspondence graph $\mathcal{G}$ for the point cloud registration problem defined in Figs. 2 and 3. In blue, the subgraph induced by the five-clique solution to the registration problem.

which is shown in Fig. 3, together with the associated vertices in $\mathcal{G}$. 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 $\mathcal{I} \subseteq \mathcal{C}$, $\mathcal{I} = \{(\boldsymbol{m}_1, \boldsymbol{q}_1), (\boldsymbol{m}_2, \boldsymbol{q}_2), (\boldsymbol{m}_3, \boldsymbol{q}_3), (\boldsymbol{m}_4, \boldsymbol{q}_4), (\boldsymbol{m}_5, \boldsymbol{q}_5)\}$. The (rigid) transformation $(\boldsymbol{R}^*, \boldsymbol{t}^*)$ for $\mathcal{I}$ is obtained by solving the least-squares optimization problem (2).

Fig. 4 shows an example of an edge (a) and a nonedge (b) in the correspondence graph according to definition (8). In the example, $\{v_1, v_4\} \in E(\mathcal{G})$ because the Euclidean distance between $\boldsymbol{m}_1$ and $\boldsymbol{m}_2$ is *similar* to the distance between $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$, i.e., falls below the threshold ξ in (8). As shown, this is not the case for $\{v_1, v_{10}\}$. The correspondence graph $\mathcal{G}$ is displayed in Fig. 5. In blue, the subgraph induced in $\mathcal{G}$ 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., $(\boldsymbol{m}_5, \boldsymbol{q}_4)$, does not find a match in any other pair according (8).

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 $\mathcal{I} \subseteq \mathcal{C}$ in the correspondence set $\mathcal{C}$ forms a clique in the correspondence graph. This observation is also valid for any subset of inliers. We recall from the example in Figs. 2 and 3, that the solution to the registration problem is the set of inliers $\mathcal{I} = \{(\boldsymbol{m}_1, \boldsymbol{q}_1), (\boldsymbol{m}_2, \boldsymbol{q}_2), (\boldsymbol{m}_3, \boldsymbol{q}_3), (\boldsymbol{m}_4, \boldsymbol{q}_4), (\boldsymbol{m}_5, \boldsymbol{q}_5)\}$, which corresponds to the 5-clique $\{v_1, v_4, v_5, v_8, v_9\}$ in $\mathcal{G}$, see Fig. 5. Essentially, an edge between two vertices in $\mathcal{G}$ refers to a rigid transformation between the associated pairs of points at both endpoints (8). Therefore, a set of s pairwise adjacent vertices, i.e., an s-clique in $\mathcal{G}$, extends the validity of the rigid transformation to every point in the s-clique.

However, not every clique in $\mathcal{G}$ 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 considered hypotheses, i.e., the vertices of $\mathcal{G}$, could be outliers and not agree with constraints (4) of the MC formulation. Furthermore, symmetries can also be a reason for flawed cliques. Consider, for example, the subset $\{(\boldsymbol{m}_1, \boldsymbol{q}_1), (\boldsymbol{m}_2, \boldsymbol{q}_2)\}$ from the previous example, associated with the clique $\{v_1, v_4\}$ in $\mathcal{G}$. It is straightforward to see that the subset $\{(\boldsymbol{m}_1, \boldsymbol{q}_2), (\boldsymbol{m}_2, \boldsymbol{q}_1)\}$ 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 $\mathcal{G}$, 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 Figs. 2 and 3, the set of points $\{\boldsymbol{m}_1, \boldsymbol{m}_2, \boldsymbol{m}_3, \boldsymbol{m}_4\}$ in the point cloud $\mathcal{M}$ are symmetric with respect to the line $\overleftrightarrow{\boldsymbol{m}_1\boldsymbol{m}_2}$, and so are the corresponding points in the point cloud $\mathcal{Q}$. Therefore, the set of correspondences $\{(\boldsymbol{m}_1, \boldsymbol{q}_1), (\boldsymbol{m}_2, \boldsymbol{q}_2), (\boldsymbol{m}_3, \boldsymbol{q}_4), (\boldsymbol{m}_4, \boldsymbol{q}_3)\}$ also forms a clique in $\mathcal{G}$, i.e., a locally valid rigid transformation. However, this transformation is incompatible with the inlier $v_9 = (\boldsymbol{m}_5, \boldsymbol{q}_5)$.

We summarize the above considerations in the following three observations.

Observation 1: Any subset of inliers $\mathcal{I} \subseteq \mathcal{C}$, i.e., subset of pairs of points $(\boldsymbol{m}, \boldsymbol{q}), \boldsymbol{m} \in \mathcal{M}, \boldsymbol{q} \in \mathcal{Q}$ that agree with constraints (4), is associated with a clique in $\mathcal{G}$. However, a clique in $\mathcal{G}$ does not necessarily correspond to a subset of inliers.

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

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

Observation 4: A maximum clique in the correspondence graph does not necessarily correspond to a maximum consensus solution of the registration problem in the presence of outliers.

To illustrate Observation 4, let us consider the correspondence graph in Fig. 5 where the vertex v_9 is removed to simulate, for example, flawed local 3-D features associated with m_5 and q_5.

Algorithm 1: Algorithm `CliReg` for Correspondence-Based Rigid Point Cloud Registration.

Input: A pair of point clouds $(\mathcal{M}, \mathcal{Q})$.
Output: A rotation $\boldsymbol{R}$ and a translation $\boldsymbol{t}$.
1 $\mathcal{C} \leftarrow \texttt{genCS}(\mathcal{M}, \mathcal{Q})$
2 $\xi \leftarrow$ determine the resolution threshold ▷ see (6)
3 $\mathcal{G} \leftarrow \texttt{genG}(\mathcal{C}, \xi)$
4 $K \leftarrow \texttt{MaxClqReg}(\mathcal{G}, \mathcal{M}, \mathcal{Q})$
5 **return** $\texttt{LSQ}(K)$ ▷ least-squares (2)

In this reduced graph, the maximum clique $\{v_1, v_4, v_6, v_7\}$ includes the subset of flawed matched pairs $\{(\boldsymbol{m}_3, \boldsymbol{q}_4), (\boldsymbol{m}_4, \boldsymbol{q}_3)\}$ and contains only two inliers. In contrast, the smaller clique $\{v_1, v_4, v_5\}$ is a better registration with three inliers.

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. New Algorithm `CliReg`

As argued at the end of the previous Section IV-C, each clique in the correspondence graph $\mathcal{G}$ represents a rigid transformation in $SE(3)$. In addition, large cliques in $\mathcal{G}$ have a high probability of being inliers (according to Observations 1 and 2), but may contain some outliers (according to Observations 3 and 4). 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 $\mathcal{G}$ 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 [67]; moreover, the bound is tight since there are graphs with precisely that amount of maximal cliques, i.e., the Moon–Moser graphs [49]. However, MCP has worst-case complexity $O(2^{n/5})$ when using bounds based on vertex coloring (according to Lavnikevich [36]), 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 (2) to evaluate every maximal clique is impractical.

The three main components of `CliReg` are the following: 1) Initialization: For the two given point clouds $(\mathcal{M}, \mathcal{Q})$, a correspondence set $\mathcal{C}$ is determined based on local features (Step 3), and a correspondence graph $\mathcal{G}$ is built (Step 5). 2) Enumeration: Maximal cliques in $\mathcal{G}$ are enumerated by the maximum clique algorithm `MaxClqReg`. `MaxClqReg` uses the latest optimization techniques (see, e.g., [41], [54], [55], [56], [57], [58], [59], [62]) and has been specifically designed for correspondence graphs. 3) Consensus: Each maximal k-clique enumerated in 2) is evaluated under 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 (4) are checked to determine the real number of inliers in $(\mathcal{M}, \mathcal{Q})$. The algorithm `CliReg` returns the maximal clique enumerated in 2) with the maximum consensus.

TABLE I
CliReg COMPONENT RESULTS ON OUR 135 SUBSET OF BUNNY INSTANCES

	n		d		$\omega(\mathcal{G})$		$\lvert K_{\text{reg}}\rvert$		$\lvert\mathcal{I}_{\mathcal{C}}\rvert$		$\lvert\mathcal{I}_{\mathcal{M}}(K_{\text{reg}})\rvert$		$\lvert\mathcal{I}_{\mathcal{M}}(\omega)\rvert$		$\overline{\lvert\mathcal{I}_{\mathcal{M}}\rvert}$
Outliers	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ
-	5862.9	641.7	0.049	0.003	83.3	19.4	28.4	17.2	173.1	57.3	2672.5	275.2	2666.3	271.1	2672.5
10 %	6109.7	628.8	0.036	0.002	61.9	17.7	29.9	16.7	112.2	38.7	2405.5	247.9	2401.3	244.4	2405.7
20 %	6371.3	663.9	0.026	0.001	43.7	14.2	23.5	7.6	74.6	27.6	2134.7	217.3	2128.5	223.8	2138.4
30 %	6634.2	680.3	0.019	0.001	27.9	8.4	18.2	6.1	41.7	15.7	1849.5	190.7	1826.7	210.2	1871.1
40 %	6918.0	696.4	0.015	0.001	19.1	7.5	14.7	3.6	24.7	12.2	1519.2	211.2	1477.2	199.7	1604.0
50 %	7189.9	716.1	0.011	0.001	12.9	3.9	11.7	4.5	12.7	9.8	828.1	476.5	773.9	501.3	1336.5
60 %	7483.3	757.5	0.008	0.001	10.1	2.0	8.7	2.6	8.8	5.5	420.9	354.0	303.5	296.5	1069.3
70 %	7750.6	774.2	0.006	0.001	8.7	0.8	8.1	1.0	7.3	3.0	153.9	130.8	127.7	137.8	802.4
80 %	7907.9	795.9	0.004	0.001	7.6	0.7	6.0	1.7	4.1	3.1	75.0	20.6	57.9	27.3	534.9

`CliReg` is outlined in Algorithm 1. Its enumeration and consensus components are the core parts of the new maximum clique procedure `MaxClqReg`, which is executed in Line 6 of `CliReg`. Both components are explained in detail in the next Section VI. Alternatively, `CliReg`'s initialization component comprises its first three steps and is described in what follows.

A. *Initialization Of* `CliReg`

As mentioned previously, the initialization component of `CliReg` covers the first three steps of Algorithm 1. In the `genCS` procedure (Line 3), the set $\mathcal{C}$ of correspondences (matching hypotheses) is determined based on the local 3-D features provided by `FPFH` [53] using a k-nearest neighbors (KNN) analysis [29], where $k \leq |\mathcal{Q}|$. As a result, the size of the correspondence graph is $n = |\mathcal{M}| \times k$ (the largest possible number $|\mathcal{M}| \times |\mathcal{Q}|$ are all the possible hypotheses). KNN analysis provides a good control over suboptimality, since, as argued at the beginning of the section, the running time of `CliReg` is bounded by $O(2^{n/5})$. Precisely, using KNN the worst-case complexity of `CliReg` is $O(2^{(|\mathcal{M}|\times k)/5})$. According to our experiments, a reasonable value of k is 3 (see Section VII-B for further details).

In Line 4, the resolution parameter ξ (6) is computed. We recall that ξ is required to determine the edges of the correspondence graph (8), as well as to determine the consensus between cliques (constraints (4) in the MC formulation). Finally, in the `genG` procedure (Line 5) the correspondence graph $\mathcal{G}$ is built according to Section IV-B.

VI. ENUMERATION OF MAXIMAL CLIQUES IN THE CORRESPONDENCE GRAPH

Clique enumeration is implemented in the procedure `MaxClqReg`, which is executed in Line 6 of `CliReg` immediately after its initialization phase. We recall from Sections II and III that finding a global optimum for the MC formulation (3) and (4) is $\mathcal{NP}$-hard, and that BnB approaches are known, such as, e.g., [15], [20], [30], [72].

Reformulating the MC problem as a tailored maximum clique search has the following advantages: 1) 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., [54], [55], [56], [57], [58], [59] and especially [41], [62]. 2) As argued in Section V, solving the least-squares optimization problem (2) for every node in the branching tree is inefficient. Consequently, our MCP algorithm enumerates a subset of (large) maximal cliques. 3) The correspondence graph $\mathcal{G}$ is expected to be reasonably sparse in many problems. This is consistent with the experiments reported in Section VII, e.g., the low average density of the graphs found in Tables I and IV, as well as Tables V, VI, and VII in the Appendix A. As a result, the MCP is expected to be easy in practice as long as the number of vertices (matching hypotheses) in $\mathcal{G}$, i.e., $|\mathcal{M}| \times k$, $k \leq |\mathcal{Q}|$ (see Section V-A), stays within reasonable bounds.

TABLE II
TRANSLATION AND ROTATION ERRORS OF ALL THE ALGORITHMS WITH UP TO (AND INCLUDING) A 50% OF OUTLIERS, FOR OUR 135 BUNNY INSTANCES

	Translation (m)		Rotation (rad)	
Algorithms	μ	σ	μ	σ
`CliReg`	**0.004**	0.014	**0.030**	0.162
`RANSAC`	0.005	0.011	0.031	0.165
`TEASER++`	0.013	0.025	0.094	0.270
`CliRegMutual`	0.013	0.024	0.123	0.336
`FGR`	0.018	0.017	0.114	0.290

The pseudocode for `MaxClqReg` is provided in Algorithm 2. In what follows, we present an overview of its main operations.

`MaxClqReg` is a constructive recursive BnB algorithm that enumerates maximal cliques in the leaf nodes of its branching tree. After an initial preprocessing phase where the vertices of the correspondence graph are sorted, i.e., $V(\mathcal{G}) = (v_1, \ldots, v_n)$, and a (large) initial clique K_0 is computed (Line 3) and assigned to $K_{\max}$, the largest clique found at any moment during the search, the algorithm determines the branching subproblems at the root node in Lines 4–5 as follows. A new branching subproblem $\widehat{\mathcal{G}}$ is associated with each candidate vertex $v_i \in V(\mathcal{G})$ selected in reverse order, i.e., $i \in \{n, \ldots, |K_0| + 1\}$. Precisely, $\widehat{\mathcal{G}}$ is the subgraph induced by the vertices adjacent to v_i that precede it in the ordering, i.e., $\widehat{\mathcal{G}} = \mathcal{G}[V_i(\mathcal{G}) \cap N(v_i)]$, where $V_i(\mathcal{G})$ is the set of vertices that precede v_i in the initial order, together with v_i. It should be noted that the subproblems derived from the candidate vertices v_i, $i \leq |K_0|$, are not considered by `MaxClqReg`, as they cannot contain a clique greater than K_0.

Enumeration is implemented as a recursive call to the procedure `EnumCLQ` in Line 6. In what follows, we provide a brief summary of `EnumCLQ`'s main operations (Lines 7–21).

TABLE III
PERCENTAGE OF CORRECT REGISTRATION RESULTS AND AVERAGE RUNTIME OF THE DIFFERENT ALGORITHMS ON THE *3-DMATCH* DATASET [77]

	Scenes									
	Kitchen (%)	Home 1 (%)	Home 2 (%)	Hotel 1 (%)	Hotel 2 (%)	Hotel 3 (%)	Study (%)	MIT Lab (%)	Avg. (%)	Avg. Runtime [s]
`CliReg`	0.976	**0.915**	**0.792**	0.967	**0.910**	**0.962**	0.855	0.756	**0.892**	3.747
`CliRegMutual`	**0.978**	0.887	0.767	0.973	0.846	**0.962**	0.855	0.733	0.875	1.512
`TEASER++`	0.976	0.896	0.761	**0.978**	0.782	**0.962**	**0.872**	**0.778**	0.875	2.574
`RANSAC`	0.960	0.896	0.704	0.929	0.821	0.846	0.748	0.578	0.810	**0.875**
`FGR`	0.813	0.717	0.610	0.819	0.641	0.731	0.547	0.578	0.682	1.062

TABLE IV
`CliReg` COMPONENT RESULTS ON THE *3-DMATCH* DATASET [77]

	n		d		$\omega(\mathcal{G})$		$\lvert K_{\text{reg}}\rvert$		$\lvert\mathcal{I}_{\mathcal{C}}\rvert$		$\lvert\mathcal{I}_{\mathcal{M}}(K_{\text{reg}})\rvert$		$\lvert\mathcal{I}_{\mathcal{M}}(\omega)\rvert$	
Scenes	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
Kitchen	13763.4	620.9	0.031	0.023	60.5	15.5	35.9	19.3	316.3	312.8	1152.9	389.7	1065.5	409.8
Home 1	13705.8	653.5	0.031	0.015	64.9	18.9	37.8	21.4	330.8	256.9	1079.3	448.0	997.0	466.1
Home 2	13741.3	834.9	0.039	0.029	55.3	19.4	33.8	19.2	345.3	314.8	1134.8	443.6	1039.2	474.2
Hotel 1	13745.6	559.3	0.034	0.020	64.7	14.7	37.9	20.0	301.0	189.7	1165.5	379.1	1103.4	381.7
Hotel 2	13853.1	557.8	0.029	0.012	61.2	19.3	39.5	23.3	264.0	205.3	1106.7	408.1	1006.3	446.4
Hotel 3	13773.6	578.4	0.029	0.010	66.7	17.0	35.6	19.8	317.5	225.5	1258.9	346.0	1175.1	381.9
Study	13868.1	446.1	0.025	0.007	57.6	16.9	31.4	18.2	192.1	145.7	1158.2	392.9	1049.4	422.5
MIT Lab	13793.5	899.7	0.031	0.021	51.7	17.8	28.7	17.0	216.7	254.8	1106.1	499.8	969.7	528.9

TABLE V
`CliReg` COMPONENT RESULTS ON OUR 135 ARMADILLO INSTANCES

	n		d		$\omega(\mathcal{G})$		$\lvert K_{\text{reg}}\rvert$		$\lvert\mathcal{I}_{\mathcal{C}}\rvert$		$\lvert\mathcal{I}_{\mathcal{M}}(K_{\text{reg}})\rvert$		$\lvert\mathcal{I}_{\mathcal{M}}(\omega)\rvert$		$\overline{\lvert\mathcal{I}_{\mathcal{M}}\rvert}$
Outliers	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ
-	4301.9	415.6	0.046	0.004	48.7	12.6	23.6	15.1	80.7	22.3	1817.7	169.9	1817.3	169.9	1817.7
10 %	4426.7	415.3	0.036	0.002	42.1	8.2	24.5	10.3	67.3	14.0	1634.7	151.7	1626.8	152.7	1636.5
20 %	4584.6	428.3	0.028	0.002	31.7	7.3	19.9	6.6	47.1	12.6	1454.0	136.6	1396.3	180.0	1454.5
30 %	4726.1	424.4	0.021	0.001	22.2	6.2	17.1	6.8	28.5	11.6	1150.7	141.6	1045.2	255.3	1272.9
40 %	4916.4	456.7	0.016	0.001	15.7	3.8	12.1	3.9	19.0	8.3	1015.1	161.6	895.1	279.1	1090.9
50 %	5058.5	454.4	0.012	0.001	12.9	3.6	10.9	4.0	12.7	7.3	577.5	313.2	479.0	323.0	909.1
60 %	5215.9	474.7	0.009	0.001	9.3	1.2	8.1	1.5	6.7	4.5	299.6	232.5	258.5	250.1	727.5
70 %	5334.5	485.1	0.006	0.001	8.1	0.6	5.5	2.2	5.0	3.2	121.3	123.1	102.0	125.0	545.7
80 %	5407.4	499.7	0.005	0.001	6.9	0.5	6.4	1.0	4.5	2.0	60.0	21.7	51.3	24.4	363.9

TABLE VI
`CliReg` COMPONENT RESULTS ON OUR 135 BUDDHA INSTANCES

	n		d		$\omega(\mathcal{G})$		$\lvert K_{\text{reg}}\rvert$		$\lvert\mathcal{I}_{\mathcal{C}}\rvert$		$\lvert\mathcal{I}_{\mathcal{M}}(K_{\text{reg}})\rvert$		$\lvert\mathcal{I}_{\mathcal{M}}(\omega)\rvert$		$\overline{\lvert\mathcal{I}_{\mathcal{M}}\rvert}$
Outliers	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ
-	5076.2	952.9	0.039	0.003	35.5	12.1	26.9	13.8	55.2	27.2	2012.2	461.5	1986.9	461.9	2046.3
10 %	5217.7	971.6	0.031	0.002	33.7	10.4	22.6	8.4	50.3	22.0	1790.6	404.6	1716.3	450.4	1842.1
20 %	5332.7	978.3	0.024	0.002	25.9	6.9	19.7	7.2	35.0	16.0	1472.7	495.5	1310.0	550.4	1637.4
30 %	5471.3	1008.3	0.018	0.001	21.1	5.1	15.1	5.0	27.0	11.5	1250.1	471.3	1203.3	475.2	1432.9
40 %	5620.7	1043.4	0.014	0.001	16.7	4.1	12.9	3.5	15.5	8.3	934.7	328.6	790.1	351.9	1228.2
50 %	5757.6	1065.0	0.010	0.001	12.2	2.6	10.8	3.1	9.1	4.4	618.5	329.1	566.4	318.8	1023.3
60 %	5888.4	1086.0	0.007	0.001	9.7	1.2	8.3	1.6	5.4	4.6	280.4	163.2	184.6	110.3	818.9
70 %	6008.3	1109.9	0.005	0.001	8.1	0.6	7.1	1.7	5.8	2.7	174.1	101.2	146.9	97.8	614.3
80 %	6086.1	1119.8	0.003	0.001	7.1	0.8	5.9	1.0	4.9	2.4	86.4	24.8	74.5	24.4	409.7

`EnumCLQ` implements a so-called *branching-by-partitioning* enumeration scheme, which partitions the vertex set $V(\widehat{\mathcal{G}})$ of the subproblem graph $\widehat{\mathcal{G}}$ associated to the current node of `MaxClqReg`'s search tree into a *branching set* of vertices (denoted B) and a *pruned set* ($P = V(\widehat{\mathcal{G}}) \setminus B$), branching only on the vertices in B. Upper bounds on the clique number of $\widehat{\mathcal{G}}$ based on vertex coloring (7) and *partial maximum satisfiability* (implemented in the procedures `PartCOL` (Line 8) and `PartSAT` (Line 10), respectively, are used to reduce the branching set B as much as possible before branching.

TABLE VII
CliReg COMPONENT RESULTS ON OUR 135 DRAGON INSTANCES

	n		d		$\omega(\mathcal{G})$		$\lvert K_{\text{reg}}\rvert$		$\lvert\mathcal{I}_{\mathcal{C}}\rvert$		$\lvert\mathcal{I}_{\mathcal{M}}(K_{\text{reg}})\rvert$		$\lvert\mathcal{I}_{\mathcal{M}}(\omega)\rvert$		$\overline{\lvert\mathcal{I}_{\mathcal{M}}\rvert}$
Outliers	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ
-	7108.3	1134.3	0.036	0.002	36.1	10.4	22.7	6.1	58.5	23.6	2939.5	491.9	2908.6	491.0	2974.5
10 %	7317.2	1167.6	0.028	0.002	31.5	9.0	17.5	6.4	49.7	17.3	2677.5	439.4	2628.5	423.4	2677.5
20 %	7552.7	1239.7	0.022	0.001	25.4	5.6	17.1	4.8	36.9	11.6	2369.3	413.6	2284.5	427.7	2380.1
30 %	7774.6	1275.2	0.017	0.001	17.5	3.8	14.5	3.9	18.7	7.1	1752.1	514.1	1552.0	549.7	2082.5
40 %	8038.3	1305.2	0.013	0.001	14.6	2.9	12.2	2.4	14.7	6.1	1403.2	496.9	1228.8	509.1	1785.1
50 %	8300.0	1357.6	0.010	0.001	10.9	1.3	9.5	1.8	7.8	4.3	569.5	380.9	461.7	337.8	1487.5
60 %	8531.6	1398.5	0.007	0.001	9.1	1.4	8.1	1.8	4.9	2.9	260.1	138.3	213.5	144.8	1190.3
70 %	8721.8	1415.0	0.005	0.001	8.1	0.8	6.7	1.7	4.1	1.3	142.1	39.6	116.3	46.1	892.9
80 %	8850.5	1441.7	0.003	0.001	7.1	0.6	6.1	1.3	4.1	2.8	89.7	21.0	77.1	28.7	595.3

Algorithm 2: Clique Algorithm `MaxClqReg` Executed by `CliReg` (Algorithm 1, Step 6)

Input: A corr. graph $\mathcal{G} = (V, E)$, point clouds $\mathcal{M}, \mathcal{Q}$.
Output: A maximal clique in K_{reg}

1 $(v_1, \ldots, v_n), K_{max} \leftarrow$ `Preproc`$(\mathcal{G})$ ▷ Sec. VI-D
2 **for** $i \leftarrow n$ **to** $|K_{max}| + 1$ **do**
3 $\quad \widehat{V} \leftarrow \{v \in V_{i-1}(\mathcal{G}) : \{v, v_i\} \in E\}$
4 $\quad$ `EnumCLQ` $(\mathcal{G}[\widehat{V}], \{v_i\}, |K_{max}| - 1)$

5 `EnumCLQ` $(\widehat{\mathcal{G}}, K, q)$
6 $P_c, B_c \leftarrow$ `PartCOL` $(V(\widehat{\mathcal{G}}), q)$ ▷ Sec. VI-A
7 **if** $B_c = \emptyset$ **then return**
8 $P_s, B_s \leftarrow$ `PartSAT` (P_c, B_c, q) ▷ Sec. VI-A
9 **if** $B_s = \emptyset$ **then return**
10 **for** $\ell \leftarrow |B_s|$ **to** 1 **do**
11 $\quad \widetilde{V} \leftarrow \{N(b_\ell) \cap \{P_s \cup \{b_1, \ldots, b_{\ell-1}\}\}\}$ ▷ $b_\ell \in B_s$
12 $\quad \widetilde{K} \leftarrow K \cup \{b_\ell\}$
13 $\quad$ **if** $\widetilde{V} = \emptyset$ **then**
14 $\quad\quad$ **if** $|\widetilde{K}| > |K_{max}|$ **then**
15 $\quad\quad\quad K_{max} \leftarrow \widetilde{K}$
16 $\quad$ **if** `CliEval`$(\widetilde{K}, \mathcal{M}, \mathcal{Q}) >$ `CliEval`$(K_{reg}, \mathcal{M}, \mathcal{Q})$ **then**
17 $\quad\quad K_{reg} \leftarrow \widetilde{K}$ ▷ Sec. VI-C
18 $\quad$ **else**
19 $\quad\quad$ `EnumCLQ` $(\widehat{\mathcal{G}}[\widetilde{V}], \widetilde{K}, |K_{max}| - |\widetilde{K}|)$

The *partial maximum satisfiability problem* (PMSP), allegedly first defined by Miyazaki [48], is a boolean satisfiability optimization problem which comprises two types of clauses denoted *hard* and *soft*. The problem asks for the maximum number of soft clauses that can be satisfied while satisfying all the hard clauses. Upper bounds on the clique number based on partial maximum satisfiability rely on the fact that, given an α-coloring $C_\alpha(G)$ of a graph G, the MCP can be reduced to a PMSP $(G, C_\alpha(G))$ in polynomial time as follows (first proposed by Li and Quan [39]).

The PMSP $(G, C_\alpha(G))$ is described naturally in conjunctive normal form, i.e., the logical formula is a conjunction of clauses and every clause is a disjunction of literals, where each literal refers to a specific vertex of the graph G. Specifically, each hard clause contains a pair of negative literals that map to a pair of nonadjacent vertices in $V(G)$, capturing the fact that at most one of the two vertices can be in a clique.

In addition, each of the α soft clauses of the PMSP $(G, C_\alpha(G))$ contains only positive literals and maps to a specific *independent set* of vertices, i.e., a subset of vertices $S \in C_\alpha(G)$ with the same color number. Solving the PMSP $(G, C_\alpha(G))$ to optimality is equivalent to determining a largest subset $\mathcal{S} \subseteq C_\alpha(G)$ of independent sets (soft clauses) that contains a clique of size $|\mathcal{S}| \leq \alpha$. As shown by Li and Quan [39], $|\mathcal{S}|$ is upper bound on the clique number $\omega(G)$ that dominates the chromatic number (7) as follows:

$$\omega(G) \leq |\mathcal{S}| \leq \chi(G) \leq \alpha. \tag{9}$$

An illustrative example of the PMSP $(G, C_\alpha(G))$ reduction is provided at the end of Section VI-E.

The critical operations of the branching-by-partitioning scheme of `MaxClqReg` are the following. The procedure `PartCOL` (Line 8) computes an initial pair of pruned and branching sets, P_c and B_c, respectively, with the help of a vertex color-based bound. In addition, the vertex coloring produced by `PartCOL` is used to determine the PMSP, and the set B_c is further reduced, if possible, by the procedure `PartSAT` that uses the bound based on partial maximum satisfiability (9) (Line 10).

Finally, each enumerated maximal clique $\widetilde{K}$ is evaluated in the algorithm's consensus phase, implemented in the function `CliEval` (Line 18), and only those cliques that provide better consensus than the current incumbent according to the MC model (3) are considered as new incumbents (Line 19).

The rest of the Section is organized as follows. The enumeration component of `MaxClqReg` is described in more detail in Sections VI-A (the general branching-by-partitioning scheme) and VI-B. Specifically, Section VI-B covers a specific branching strategy of `MaxClqReg` designed to exploit the structure of the correspondence graphs, compatible with the general branching-by-partitioning scheme. It should be noted that this technique is not used by the most recent efficient maximum clique algorithms, such as, e.g., [42] and [62], and is an additional contribution of this work. The consensus component of `MaxClqReg` is examined in Section VI-C. Finally, preprocessing is considered in Section VI-D, and Section VI-E provides an illustrative example.

A. Branching (And Bounding) by Partitioning the Candidate Vertex Set

This section describes the general *branching-by-partitioning* enumeration scheme of MaxClqReg in detail. As mentioned previously, the overarching idea is to partition the candidate vertex set into two disjoint subsets and branch only on the vertices of one of the sets, thereby reducing the size of the branching tree.

A tree node of MaxClqReg $(\widehat{\mathcal{G}}, K, q)$ is associated with a graph subproblem $\widehat{\mathcal{G}}$, a clique K under construction, and an integer q. Specifically, q is the number of vertices that must be added to K to have the same cardinality as the incumbent clique $K_{\max}$ (the largest clique enumerated at any moment during the search), i.e, $q = |K_{\max}| - |K|$. When branching, MaxClqReg selects a candidate vertex $v \in V(\widehat{\mathcal{G}})$ (by construction, v must be adjacent to every vertex in K) and computes the child node $(\widetilde{\mathcal{G}}, \widetilde{K}, \tilde{q})$ as follows: i) in its simplest form, $\widetilde{\mathcal{G}}$ is the subgraph induced in $\widehat{\mathcal{G}}$ by the vertices in $V(\widehat{\mathcal{G}})$ neighbors to v; ii) $\widetilde{K} = K \cup \{v\}$ and *iii)* $\tilde{q} = q - 1 = |K_{\max}| - |\widetilde{K}|$, since the original clique K has been enlarged by a single vertex. Using the above branching strategy, the leaf nodes of the MaxClqReg's search tree are, by construction, maximal cliques in $\mathcal{G}$.

Given a tree node $(\widehat{\mathcal{G}}, K, q)$, the (candidate) vertex set $V(\widehat{\mathcal{G}})$ can be partitioned into two disjoint sets of vertices called the *pruned set* P and the *branching set* $B = V(\widehat{\mathcal{G}}) \setminus P$, so that branching on the vertices of P alone cannot improve the incumbent clique. Precisely, the set P can be defined as any vertex set that induces a subgraph whose clique number is strictly less than q, i.e.,

$$\omega(\widehat{\mathcal{G}}[P]) \leq q. \tag{10}$$

Equation (10) is a valid defining property of P: q is precisely the difference between the sizes of the incumbent clique and the current clique K under construction; therefore, branching only on the vertices in P (alternatively, considering only the subgraph $\widehat{\mathcal{G}}[P]$ in the current tree node) cannot improve the size of the incumbent clique.

Furthermore, the set B should be as small as possible (alternatively P as large as possible) to reduce the size of the branching tree; see, e.g., [41], [62]. Formally, the optimal set P is the solution to the following very hard optimization problem:

$$P = \arg \max_{\widehat{P} \subseteq V(\widehat{\mathcal{G}})} \left\{ |\widehat{P}| : \omega(\widehat{\mathcal{G}}[\widehat{P}]) \leq q \right\}. \tag{11}$$

Solving the problem (11) exactly in every node is impractical, so, in practice, MaxClqReg employs the following two heuristic procedures (sometimes denoted *bounding functions*): i) PartCOL [based on the vertex color bound (7)] and ii) PartSAT [based on the partial maximum satisfiability bound (9)]. Both bounding functions are well established in the literature for the MCP, see, e.g., [42], [43], [54], [55], [56], [57], [59], [60], [61]. The implementation of both functions in MaxClqReg is similar to that of the state-of-the-art algorithm CliSAT [62], and we refer the interested reader to the latter for further details. To make the article stand-alone, the main operations of PartCOL (Line 8 of Algorithm 2) and PartSAT (Step 10) are summarized in what follows.

PartCOL uses the well-known sequential greedy vertex coloring heuristic of Matula et al. [47] to determine an α-coloring $C_\alpha(\widehat{\mathcal{G}})$ of the subproblem graph $\widehat{\mathcal{G}}$ at the current tree node. Specifically, its output is a pruned set P_c defined (typically) by the first q colors of $C_\alpha(\widehat{\mathcal{G}})$. A partial q-coloring $C_q(\widehat{\mathcal{G}})$, $q \leq \alpha$, is a valid pruned set according to its defining property (10), since the size of the coloring q is an upper bound on the clique number of any q-colored graph (7). Then, if the associated branching set $B_c = V(\widehat{\mathcal{G}}) \setminus P_c$ is not empty (otherwise the node is pruned), PartSAT is executed with the intention of reducing B_c as follows.

Given a vertex $v \in B_c$, PartSAT attempts to prove that q is also an upper bound on the clique number for the subgraph induced by the (enlarged set) $P_c \cup \{v\}$, i.e.,

$$\omega(\widehat{\mathcal{G}}[P_c \cup \{v\}]) \leq q. \tag{12}$$

It does so by encoding the MCP of $\widehat{\mathcal{G}}[P_c \cup \{v\}]$ into the PMSP $(\widehat{\mathcal{G}}[P_c \cup \{v\}], C_q(\widehat{\mathcal{G}}) \cup \{v\})$ (as explained in the introduction of Section VI) and proving, if possible, that there is no solution for its $q+1$ soft clauses, i.e., the q soft clauses associated to the q-coloring of P_c, plus the singleton soft clause associated to the vertex v. If PartSAT is successful then property (12) holds, and the new branching and pruned sets become $B_c \setminus \{v\}$ and $P_c \cup \{v\}$, respectively. The procedure continues until all the vertices in B_c have been evaluated. After the execution of PartSAT, MaxClqReg branches on the remaining vertices in the branching set (denoted B_s in the algorithm).

B. Branching in Detail: A Nonincremental Scheme

The previous section described in detail how MaxClqReg determines a (small) branching set B according to its branching-by-partitioning scheme. This section covers the specific enumeration of subproblems used by MaxClqReg when branching on the vertices in B (Lines 12 and 13 of Algorithm 2).

In recent years, efficient MCP algorithms, such as, e.g., [41], [62], employ a so-called *incremental branching* scheme to enumerate subproblems when branching on the vertices in B. Incremental branching is reminiscent of the *Russian doll* search paradigm [68], where small subproblems (the small *dolls*) are solved first and bound the search of large subproblems (the large *dolls*).

Specifically, given a tree node $(\widehat{\mathcal{G}}, K, q)$ of MaxClqReg, and the sets of vertices $B = (b_1, b_2, \ldots, b_{|B|})$ and $P = V(\widehat{\mathcal{G}}) \setminus B$ sorted according to a predetermined initial order (discussed in Section VI-D), the incremental branching scheme enumerates the following set of child subproblems $\widehat{\mathcal{G}}(b_i)$, $i \in \{1, \ldots, |B|\}$, $b_i \in B$:

$$\widehat{\mathcal{G}}(b_i) = \widehat{\mathcal{G}}\left[P \cup \{b_1, \ldots, b_i\}\right] \quad \forall i \in \{1, \ldots, |B|\}.$$

From the previous equation, $\widehat{\mathcal{G}}(b_i)$ is the subproblem graph (doll) induced by the vertices in P together with b_i and those vertices that precede b_i in B. We refer the interested reader to [41], [62] for further details on incremental branching.

Algorithm 3: Evaluation Function `CliEval`.

Input: A clique K in $\mathcal{G}$, the pair of clouds $(\mathcal{M}, \mathcal{Q})$.
Output: An integer ϕ.
1 $\boldsymbol{R}, \boldsymbol{t} \leftarrow \texttt{LSQ}(K)$ ▷ least-squares (2)
2 $\phi \leftarrow \texttt{fitnessCLQ}(\boldsymbol{R}, \boldsymbol{t}, \mathcal{M}, \mathcal{Q})$ ▷ #inliers (4)
3 **return** ϕ

The previous approach is, however, not suitable for `MaxClqReg` because, by solving the small subproblems first, it is likely that the maximal cliques enumerated early will also be small, and therefore have a small number of inliers (see Observation 2). Consequently, `MaxClqReg` enumerates the subproblems using a different strategy. Specifically, it examines subproblem dolls in order of decreasing size as follows:

$$\widehat{\mathcal{G}}(b_i) = \widehat{\mathcal{G}} \left[P \cup \{b_1, \ldots, b_i\})\right] \quad \forall i \in \{|B|, \ldots, 1\}. \tag{13}$$

Equation (13) states that the first subproblem examined by `MaxClqReg` contains all the cliques with vertex $b_{|B|}$, the second subproblem contains all the 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), and so on. The main intuition is that, by examining subproblems in this order, large maximal cliques will be enumerated early with high probability. This intuition was confirmed by extensive preliminary tests, in which *good* registration cliques were found earlier than in the case of the state-of-the-art (Russian doll) incremental scheme.

We end the section by mentioning that the definition of the vertex set $\widetilde{V} = V(\widehat{\mathcal{G}}(b_\ell))$ in Line 13 of Algorithm 2 is consistent with (13). Precisely, the pseudocode captures the fact that, since branching on b_ℓ fixes the vertex in the clique K under construction (Line 14), the vertices of the child subproblem $V(\widehat{\mathcal{G}}(b_\ell))$ should be restricted to the subset of vertices established in (13), i.e., $\{P \cup \{b_1, \ldots, b_\ell\}\}$, that are also adjacent to b_ℓ, i.e., $N(b_\ell) \cap \{P \cup \{b_1, \ldots, b_{\ell-1}\}\}$.

C. Consensus

This section covers the procedure `CliEval`, the consensus component of `CliReg`. Its pseudocode is shown in Algorithm 3. `CliEval` is executed in Line 18 of Algorithm 2 and evaluates every maximal clique (enumerated as explained in the previous Sections VI-A and VI-B) in the following two phases.

In the first phase of `CliEval` (Line 3), the function $\texttt{LSQ}(K)$ determines the transformation $(\boldsymbol{R}, \boldsymbol{t})$, solution to the least-squares optimization problem (2), under the assumption that the correspondences in K are all inliers. Specifically, $\texttt{LSQ}(K)$ implements the dual quaternion algorithm [69].

In the second phase of `CliEval` (Line 4), the actual number of inliers ϕ for the transformation $(\boldsymbol{R}, \boldsymbol{t})$ is computed by the function `fitnessCLQ`. Precisely, these are the subset of correspondence pairs of the clouds $\mathcal{M}$ and $\mathcal{Q}$ that agree with constraints (4) for the given $(\boldsymbol{R}, \boldsymbol{t})$.

D. Preprocessing

The main components of the preprocessing phase of `MaxClqReg` (Line 3 of Algorithm 2) are: i) determining an initial (large) clique K_0 and ii) determining an appropriate initial order for the input set of vertices of the correspondence graph $V(\mathcal{G})$. To compute an initial clique, `MaxClqReg` uses the adaptive multistart tabu heuristic AMTS [73], also employed in other efficient MCP algorithms such as, e.g., [62]. In what follows, we describe the initial sorting strategy of 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., [46], [58]. 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, denoted `DEG-SORT` (allegedly, first introduced by Carraghan and Pardalos [17] for the MCP) and ii) an ordering based on vertex coloring, denoted `COLOR-SORT` (allegedly, first introduced by Li et al. [40] for the MCP).

The term *degenerate* in i) indicates that the sorting criterium (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 (sorted) 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 $\mathcal{G}[V(\mathcal{G}) \setminus \{v_n, v_{n-1}\}]$, and so on. Unlike `DEG-SORT`, the basic form of `COLOR-SORT` partitions the set $V(\mathcal{G})$ into (large) independent (color) sets and sorts the vertices by nondecreasing color number.

According to the tests reported by San Segundo et al. [58] on a dataset of small and medium-size dense synthetic graphs, selecting `DEG-SORT` or `COLOR-SORT` can, in some cases, account for a performance difference of up to several orders of magnitude for the same MCP algorithm. However, we have not found significant differences between both strategies in our extensive preliminary tests, possibly because the reported correspondence graphs are sparse; see, e.g., Table I in the experiments section. Consequently, we have only considered `DEG-SORT` for `MaxClqReg` during preprocessing.

E. Illustrative Example

We illustrate the main operations of `MaxClqReg` (see Algorithm 2) with the correspondence graph $\mathcal{G}$ shown in Fig. 6 (top), borrowed from San Segundo et al. [62]. $\mathcal{G}$ has $n = 8$ vertices, $m = 22$ edges, and a clique number $\omega(\mathcal{G}) = 4$. The red vertices (and edges) refer to a maximum clique $\{v_1, v_2, v_3, v_4\}$, but there are others, e.g., $\{v_3, v_4, v_5, v_6\}$. For simplicity, we assume that the preprocessing phase of `MaxClqReg` determines an initial clique $K_0 = \{v_1, v_2\}$, so the root node is $\{\mathcal{G}, K = \emptyset, q = |K_0| - |K| = 2\}$.

According to Lines 4–6 [see also (13)] with root branching and pruned sets $B_0 = (v_3, \ldots, v_8)$ and $P_0 = \{v_1, v_2\}$, respectively, `MaxClqReg` enumerates the following root child subproblems (in order): $\mathcal{G}(v_8), \mathcal{G}(v_7), \ldots, \mathcal{G}(v_3)$.

Precisely, the bottom part of Fig. 6 shows the subproblem graph $\mathcal{G}(v_7)$, i.e., vertices $V(\mathcal{G}(v_7))$ in green and edges

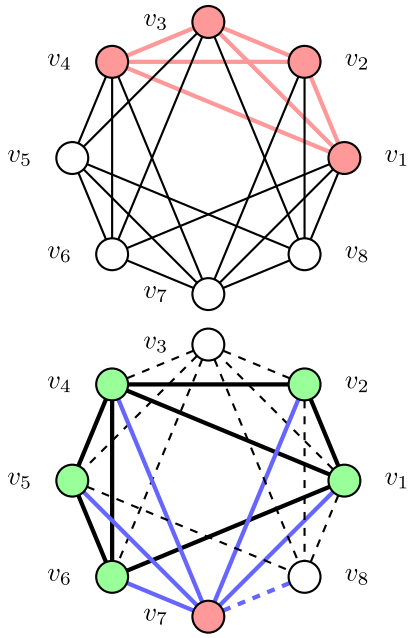


Fig. 6. Example correspondence graph $\mathcal{G}$ (top) and the subproblem graph $\mathcal{G}(v_7)$ resulting from branching on the vertex v_7 (bottom). The example is borrowed from [62].

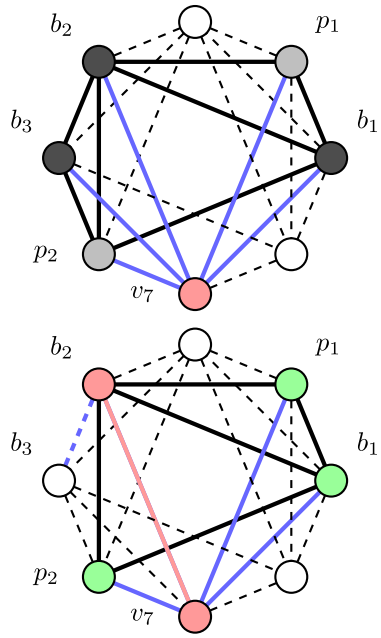


Fig. 7. Top: Pruned and branching sets P and B for the subproblem graph $\widehat{\mathcal{G}} = \mathcal{G}(v_7)$ of Fig. 6 (bottom). Bottom: the subproblem graph $\widehat{\mathcal{G}}(b_2)$.

$E(\mathcal{G}(v_7))$ thick dark; the branching vertex v_7 is colored red. The edges connecting vertex v_7 to the vertices involved in the branching operation are colored blue. The edge connecting v_7 to v_8 appears dashed blue, indicating that it does not belong to $E(\mathcal{G}(v_7))$, since $\mathcal{G}(v_8)$ has already been examined at this point. Finally, all the remaining edges in $E(\mathcal{G})$ are shown as dashed black lines. The resulting child node after branching on v_7 is $(\mathcal{G}(v_7), \{v_7\}, 1)$.

The top part of Fig. 7 shows an example of pruned and branching sets $P_c = \{p_1, p_2\}$ and $B_c = \{b_1, b_2, b_3\}$, for the set of vertices $V(\widehat{\mathcal{G}}) = \{v_1, v_2, v_4, v_5, v_6\}$ of the previous subproblem graph $\widehat{\mathcal{G}} = \mathcal{G}(v_7)$ [Fig. 6 (bottom)]. It should be noted that P_c could be, in this case, any independent (color) set, since $q = 1$ in the current tree node and, as explained previously, property (10) holds for any such P_c. In the figure, the vertices have been relabeled to p_i and b_i in the order used in (13), and appear colored, gray, and black, respectively. The remaining color and line codes are the same as those in Fig. 6.

Assuming that the branching set B_c is optimal, `MaxClqReg` examines (in order) the subproblems $\widehat{\mathcal{G}}(b_3)$, $\widehat{\mathcal{G}}(b_2)$, and $\widehat{\mathcal{G}}(b_1)$ [see Lines 12 and 13 and (13)]. The bottom part of Fig. 7 illustrates $\widehat{\mathcal{G}}(b_2)$ (its vertices $\{v_1, v_2, v_6\}$ are shown in green and its edges as thick black lines). The edge connecting b_2 to b_3 appears dashed blue, indicating that it does not belong to $E(\widehat{\mathcal{G}}(b_2))$, since $\widehat{\mathcal{G}}(b_3)$ has already been examined at this point.

We end the section with a comment on the bounding function `PartSAT`, called with the goal of reducing the size of the branching set $B_c = \{b_1, b_2, b_3\}$ determined by `PartCOL`. In what follows, we describe its main operations when trying to prove that the clique number of the (colored) subgraph $\widetilde{\mathcal{G}} = \widehat{\mathcal{G}}[P_c \cup \{b_1\}]$ is less than or equal to $q = 1$ [see (12)].

We consider the associated PMSP($\widetilde{\mathcal{G}}, C_2(\widetilde{\mathcal{G}})$), where the partial 2-coloring is $C_2(\widetilde{\mathcal{G}}) = \{\{p_1, p_2\}, \{b_1\}\}$. Its singleton hard clause is $(\bar{y}_{p_1} \vee \bar{y}_{p_2})$, which corresponds to the nonadjacency relation between p_1 and p_2. Its two soft clauses are $(y_{p_1} \vee y_{p_2})$ and (y_{b_1}), which correspond to the two independent sets of the coloring. The solution to the PMSP($\widetilde{\mathcal{G}}, C_2(\widetilde{\mathcal{G}})$) is 2, since, for example, the assignment $y_{b_1} = \top$, $y_{p_1} = \bot$, and $y_{p_2} = \top$ satisfies the $q + 1 = 2$ soft clauses (as well as the hard clause). Consequently, $\omega(\widehat{\mathcal{G}}[P_c \cup \{b_1\}]) = 2$ (the set of vertices (literals) $\{p_2, b_1\}$ that satisfy the soft clauses is a 2-clique), which is strictly greater than $q = 1$, so `PartSAT` cannot reduce B_c in this case.

VII. Experimental Results

In this section, we report the results of an extensive computational campaign. The goal of this analysis is threefold: i) To provide information on the main components of `CliReg`, i.e., the correspondence graph determined by `genG`, the maximal clique enumeration and the `fitnessCLQ` clique evaluation function; ii) to compare `CliReg` against previous state-of-the-art algorithms in the literature using a synthetic dataset; iii) to evaluate `CliReg` in a practical application focused on the scan-matching problem. `CliReg` is implemented in C++ and has been compiled with gcc 11.4.0 using the optimization flag -O3. All tests were carried out on a single core of a Linux workstation with an Intel(R) CPU i9-12900 K and 16 GB of RAM.

To ensure a fair evaluation, a Python framework has been developed that executes the following three phases in order: i) Data acquisition, where our point cloud dataset is generated and preprocessed, ii) execution of the algorithms, and iii) report

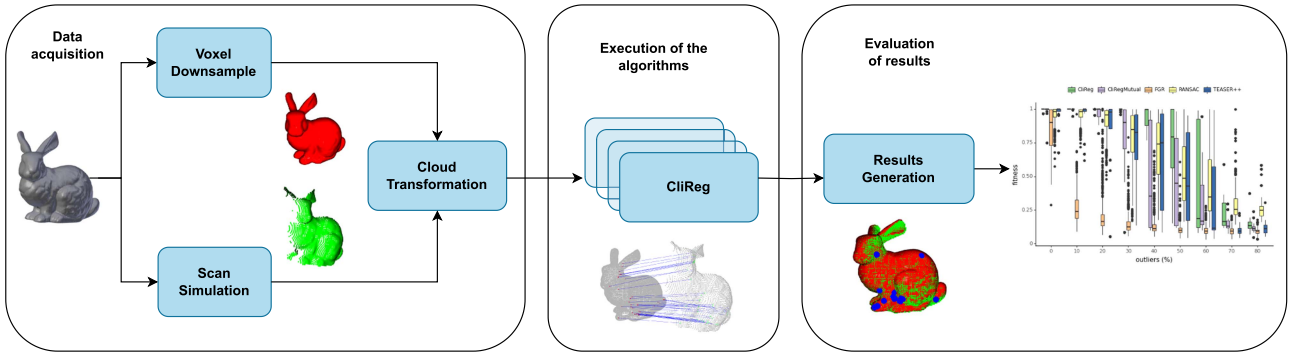


Fig. 8. Workflow of our Python framework for scan-matching tests, showing an example bunny instance. In red and green are the target $\mathcal{Q}$ and moving $\mathcal{M}$ point clouds, respectively.

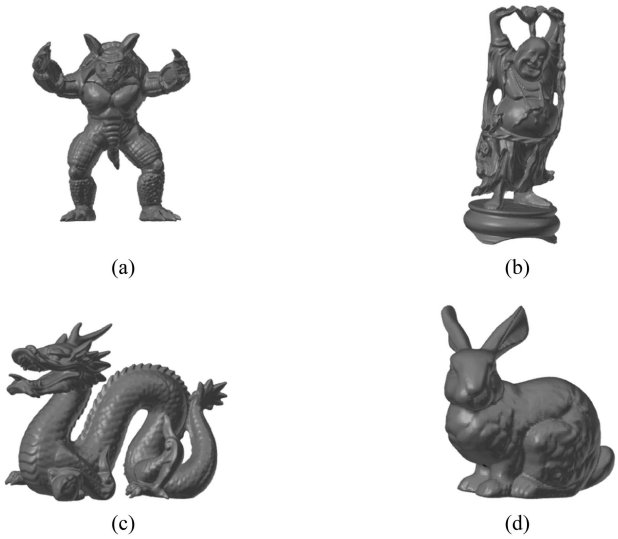


Fig. 9. Our point cloud dataset from Stanford University [1]: (a) armadillo, (b) buddha, (c) dragon, and (d) bunny.

generation and evaluation of results. This workflow is shown in Fig. 8 and is explained in the following.

A. Synthetic Dataset

Our reference triangle meshes used in the experiments comprises four 3-D models (armadillo, buddha, dragon, and bunny) from the Stanford 3-D scanning repository [1]; see Fig. 9. 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 *Voxel Downsample* procedure in Fig. 8) to derive the target point cloud $\mathcal{Q}$ representing the entire object with the same resolution as the simulated time-of-flight (ToF) camera used to obtain different views of $\mathcal{Q}$.

Specifically, the camera captures distance data with a resolution of 176×144 pixels and Gaussian noise with a mean (μ) of 0.0 m and a standard deviation (σ) of 0.004 m. Camera simulation is performed using the Blensor tool [28] in Blender.[1] With this camera, each mesh is sampled to generate the moving point clouds $\mathcal{M}$ of our dataset. Specifically, the framework produces partial views of $\mathcal{Q}$ taken from 15 different perspectives of each 3-D model (totalling $4 \times 15 = 60$ moving clouds $\mathcal{M}$). This is accomplished in the *scan simulation* procedure of Fig. 8. In the figure, the green bunny output of the camera scan procedure is an example of a cloud $\mathcal{M}$, a partial view of the red bunny target cloud $\mathcal{Q}$.

To evaluate the reported algorithms, a known rigid transformation is applied to each cloud $\mathcal{M}$ (procedure *Cloud Transformation* in Fig. 8). 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 $\mathcal{M}$, 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 $\mathcal{M}$) may not correspond to those of the target cloud $\mathcal{Q}$. 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 3-D original models.

The settings for the algorithm execution phase are as follows. As mentioned in the beginning of the section, all algorithms are executed on a single thread. The two point clouds $\mathcal{M}$ and $\mathcal{Q}$, together with a set of local geometric features for each point extracted by the `FPFH` algorithm [53], are given to the reported algorithms in the same format (two arrays with the coordinates of the points and the features associated with each point, respectively). The reported results are averaged over 40 runs.

To obtain a robust evaluation, we report the average results for 40 runs of the same instance for all algorithms tested. The 40 outputs for each instance should be exactly the same for the deterministic algorithms, except perhaps in the few cases where the running time is close to the time limit, but may differ in the nondeterministic ones, i.e., `FGR` [79] and `RANSAC` [24]. The time limit was fixed at 10 s in all runs.

The 540 instance dataset is publicly available in GitHub.[2]

[1] [Online]. Available: https://www.blender.org/

[2] [Online]. Available: https://github.com/jlaserna/CliReg-Matcher.git

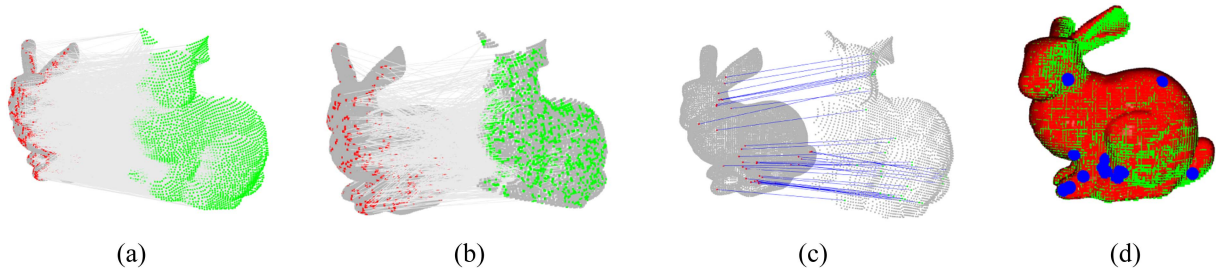


Fig. 10. Example showing the main components of CliReg, and its variant CliRegMutual; in red the bunny target point cloud $\mathcal{Q}$ and, in green, the moving point cloud $\mathcal{M}$ (a partial view of the bunny). (a) Correspondence set of CliReg; (b) Correspondence set of CliRegMutual [smaller than (a)]; (c) Matched correspondences, vertices of the clique determined by CliReg; (d) Registered overlapping clouds $\mathcal{M}$ (green) and $\mathcal{Q}$ (red). The subset of inlier correspondences $\mathcal{I}_\mathcal{C}$ from the set of correspondences C associated to the vertices of the correspondence graph are shown in blue.

B. *Analysis of the Main Components of* CliReg

In this section, we report results on our synthetic dataset that illustrate the operations of the main components of CliReg. In the report, we have included the algorithmic variant CliRegMutual of CliReg, which is described in what follows.

As argued previously, the function genCS of CliReg (executed in Line 5 of Algorithm 1) determines the correspondence set $\mathcal{C}$ of the registration problem with the help of the local geometric features determined by FPFH [53]. We also recall that the matching hypotheses in $\mathcal{C}$, i.e., the vertices of the correspondence graph, result from a k-nearest-neighbors analysis and that $k = 3$ was found to be a *good* value for CliReg in our extensive preliminary tests. Fig. 10(a) 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 $(\boldsymbol{m}, \boldsymbol{q})$, $\boldsymbol{m} \in \mathcal{M}$, $\boldsymbol{q} \in \mathcal{Q}$, in the correspondence set such that $\boldsymbol{m}$ has $\boldsymbol{q}$ as closest point in the feature space and, vice versa, $\boldsymbol{q}$ has $\boldsymbol{m}$ as its closest point. If, given a point $\boldsymbol{m} \in \mathcal{M}$, the property does not hold for any point in $\mathcal{Q}$, then no matching hypothesis for $\boldsymbol{m}$ is added to $\mathcal{C}$. As a result, the correspondence graph generated by CliRegMutual has at most $|\mathcal{M}|$ vertices (not $|\mathcal{M}| \times k$, $k = 3$, as CliReg) and is expected to have a shorter running time than CliReg at the cost of providing a less robust registration. Fig. 10(b) shows the correspondence set determined by CliRegMutual for the previous bunny instance.

In addition, Fig. 10(c) shows the matched pairs, vertices of the clique returned by CliReg (see Algorithm 1), and Fig. 10(d) shows the registered (overlapping) clouds. In the latter figure, the inliers of the correspondence set in Fig. 10(a) appear in blue.

Table I provides information on the main components of CliReg for the bunny model. The table shows aggregated results for the nine different tested outlier rates. For each outlier rate, the table reports the size n and average density d of the correspondence graph $\mathcal{G}$, its clique number $\omega(\mathcal{G})$, the size of the maximal clique $|K_{\text{reg}}|$ output of MaxClqReg and the number of inliers $|\mathcal{I}_\mathcal{C}|$ in the correspondence set $\mathcal{C}$ for the registration determined by K_{reg}, i.e., the output ϕ of our fitnessCLQ function (see Step 4 of Algorithm 3) for the maximal clique K_{reg}. Furthermore, the table also reports the total number of inliers $|\mathcal{I}_\mathcal{M}|$ in the moving cloud $\mathcal{M}$, i.e., the set of points in $\mathcal{M}$ that find a point in $\mathcal{Q}$ at a distance less than the predetermined parameter ξ (6), for a given transformation. Specifically, column $|\mathcal{I}_\mathcal{M}(K_{\text{reg}})|$ refers to the inliers in $\mathcal{M}$ associated with the registration determined by K_{reg}, while $|\mathcal{I}_\mathcal{M}(\omega)|$ refers to the inliers in $\mathcal{M}$ associated with the registration determined by a maximum clique in the correspondence graph. It is worth mentioning, that the metric $\mathcal{I}_\mathcal{M}$ is independent of the local features determined by the FPFH algorithm [53], while the set of inliers $\mathcal{I}_\mathcal{C} \subseteq C$ is feature-dependent, i.e., our correspondence set $\mathcal{C}$, by construction, contains only candidate matches compatible with the descriptors provided by the FPFH algorithm for both point clouds $\mathcal{M}$ and $\mathcal{Q}$. Finally, the table reports the original number of inliers $\overline{|\mathcal{I}_\mathcal{M}|}$, i.e., the points in the cloud $\mathcal{M}$ that were not transformed into outliers during the data acquisition phase of our test framework. $\overline{|\mathcal{I}_\mathcal{M}|}$ is an upper bound on $|\mathcal{I}_\mathcal{M}(K_{\text{reg}})|$ and $|\mathcal{I}_\mathcal{M}(\omega)|$ by construction.

From Table I, several conclusions can be drawn. First, the mean values of the maximal clique size $|K_{\text{reg}}|$ computed by CliReg are lower than the clique numbers $\omega(\mathcal{G})$ of the correspondence graphs $\mathcal{G}$ in all cases. The difference is greater than $3\times$ when there are no outliers present and gradually decreases as the outlier rate increases.

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 $|\mathcal{I}_\mathcal{M}(K_{\text{reg}})|$ is very close to the threshold $\overline{|\mathcal{I}_\mathcal{M}|}$. To be precise, in the case of the bunny $|\mathcal{I}_\mathcal{M}(K_{\text{reg}})|$ is optimal when no outliers are present, but this is not always the case; see Appendix A. Furthermore, the number of inliers $|\mathcal{I}_\mathcal{M}(K_{\text{reg}})|$ is always greater than $|\mathcal{I}_\mathcal{M}(\omega)|$, and the difference tends to increase with the percentage of outliers up to, and including, a 60% outliers, when the difference reaches a 38%. This empirically shows that the registration with best results (in terms of maximum consensus) is not associated to a maximum clique of 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, since, in previous clique-based approaches to the registration problem, such as, e.g., TEASER++ [75], only maximum cliques were considered.

The third result is the large difference between the number of inliers $|\mathcal{I}_\mathcal{C}|$ found in the correspondence set (as determined by our `fitnessCLQ` function in Line 4 of `CliEval`) and the actual number of inliers $|\mathcal{I}_\mathcal{M}(K_{\text{reg}})|$. 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., [66], [77]. Moreover, it empirically justifies the need for a robust methodology against outliers, as proposed in this work.

A fourth and last result is that the correspondence graphs are sparse. Precisely, the average density d of the graphs in Table I has a maximum value of 5% when no outliers are present, and decreases as the percentage of outliers increases.

Similar conclusions can be reached for the other three models tested (armadillo, buddha, and dragon), so we do not comment on them here. The results are available in Tables V, VI, and VII in Appendix A.

C. Experiments on Synthetic Data

This section covers the extensive experimental campaign carried out on our synthetic dataset to compare the performance of `CliReg` with other state-of-the-art algorithms. Specifically, we report results for `CliReg`, `CliRegMutual` (the variant of `CliReg` designed for fast execution described in the previous Section VII-B) and the following four algorithms:

1) `RANSAC` [24]: Our reference nondeterministic algorithm. Specifically, we have used its feature-based variant with 99.9% confidence configured according to the specifications defined in the Open3-D 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 ξ (6) for each instance;
2) `FGR` [79]: The efficient algorithm of Zhou et al. [79] that uses the Geman–McClure robust least-squares cost function;
3) `TEASER++` [75]: To the best of the authors' knowledge, the reference clique-based algorithm. We used as settings those in the original GitHub repository,[4] with the *noise bound* parameter fixed to ξ (6), which represents the maximum error to be expected from an inlier in our experimental setting. It should be noted that Yang et al. [75] used a sensor-dependent noise bound, but our choice was also suggested as a possibility in the same work. We believe ξ to be more robust in our dataset since it only depends on the resolution of the input point clouds;
4) `Go-ICP` [76]: 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.

[3][Online]. Available: http://www.open3d.org/docs/release/tutorial/pipelines/global_registration.html

[4][Online]. Available: https://github.com/MIT-SPARK/TEASER-plusplus/tree/master/examples/teaser_python_fpfh_icp

[5][Online]. Available: https://github.com/yangjiaolong/Go-ICP/blob/master/config_example.txt

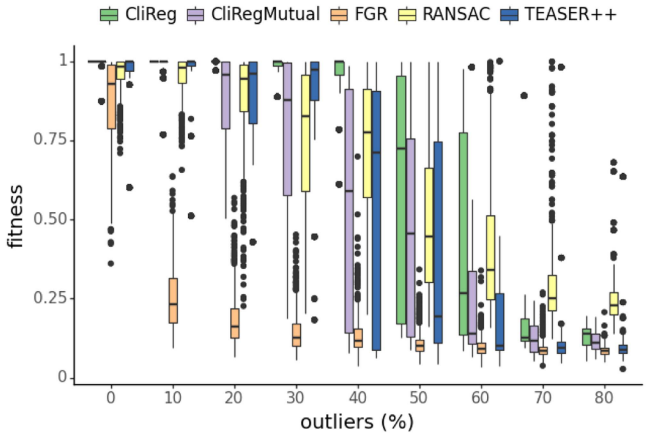


Fig. 11. Fitness of the registration provided by the different algorithms for our 135 bunny instances.

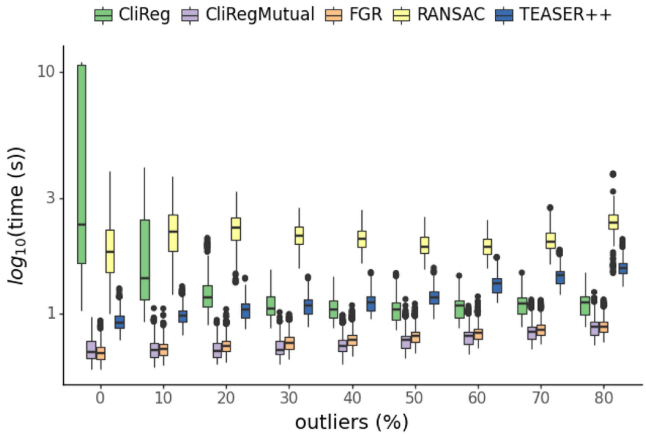


Fig. 12. Execution times (in logarithmic scale) of the different algorithms for our 135 bunny instances.

The fitness metric used to compare the results among the algorithms is the number of inliers $|\mathcal{I}_\mathcal{M}|$ in the moving cloud $\mathcal{M}$, normalized between 0 and 1 as follows:

$$\text{fitness} = |\mathcal{I}_\mathcal{M}| \;/\; \overline{|\mathcal{I}_\mathcal{M}|}, \in (0, 1] \tag{14}$$

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 phase (see Fig. 8) are considered in the fitness metric. In the case of `CliReg`, the value $|\mathcal{I}_\mathcal{M}(K_{\text{reg}})|$ is used to determine its fitness.

Figs. 11 and 12 present boxplots that evaluate the global fitness of the registration and the execution times of the five algorithms considered in this report (`CliReg`, `CliRegMutual`, `FGR`, `TEASER++`, and `RANSAC`). Specifically, both figures show a colored boxplot for each of the five 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 nine different outlier configurations. Since the behavior of the five algorithms for the remaining three models (armadillo, buddha, and dragon [1]) 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.

Fig. 11 shows the fitness (14) 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. 12 shows the execution time distribution with respect to the percentage of outliers. The execution times of CliRegMutual are comparable to those of the fastest heuristic FGR. On the other hand, its fitness clearly dominates FGR for all outlier rates and is higher than TEASER++ and RANSAC in the range 0%–20% of outliers. The figure also shows that CliReg is slower than FGR, faster than RANSAC and has similar running time as TEASER++ with exceptions. Specifically, in the no-outliers case (0% in the figure), CliReg reaches the time limit in three bunny instances. It should be noted that CliReg does not exceed the time limit in any of the other three subsets (armadillo, buddha, and dragon); see Appendix A.

It is also worth mentioning that while all methods typically need more time to compute a solution when the number of outliers increases, CliReg performs worse when the number of outliers is very small. This might be explained by the fact that, although the sizes of the correspondence graphs increase with the number of outliers, the densities of the graphs decrease (see Table I and also Tables V, VI, and VII in Appendix A). Therefore, efficient clique-based methodologies, such as the one used by MaxClqReg, can actually be faster in a setting with a high number of outliers, because they can exploit the sparsity of the graphs. Another possible explanation is that, when the number of outliers is (very) small, the probability of plateaus of maximal cliques of the same size increases. This could make MaxClqReg spend a lot of time branching without improving the incumbent solution, and be the cause of a poor performance in such a setting.

In addition, Table II reports the mean μ and standard deviation σ of the translation and rotation errors of the different algorithms (averaged up to, and including, a representative threshold of 50% of outliers) for the bunny dataset. According to the table, CliReg is the algorithm with the smallest error in both cases, which is consistent with the fitness results in Fig. 11.

Finally, we also performed analysis of variance (ANOVA) of the fitness of the registration and the execution times of the reported algorithms for the same representative range of outliers. According to the results, CliReg clearly dominates the other algorithms in terms of robustness. The full results are available in Appendix A.

D. Experiments on Real-World Data

To further evaluate the robustness of our approach, we run additional tests on the real-world *3-DMatch* dataset [77] for the same algorithms reported on synthetic data in Section VII-C, i.e., CliReg (two variants), TEASER++, RANSAC, and FGR. *3-DMatch* comprises RGB-D scans of 62 indoor scenes, divided into 54 scenes for training and eight scenes for testing. On average, there are 205 scan pairs per scene, with a maximum number of 519 pairs in the kitchen family and a minimum number of 54 in the hotel three family. Each scan contains 5000 randomly sampled keypoints. The *3-DMatch* dataset has been used for similar tests in other papers, including the TEASER++ algorithm [75].

As in Yang and Carlone [75], local features for each 3-D keypoint were determined with *3-DSmoothNet* [26], a state-of-the-art neural network, and the same set of correspondence pairs, generated using a nearest-neighbor search, was provided to all algorithms. Lastly, all algorithms were run on a single thread and with the same machine and settings as the synthetic data tests. Table III shows the percentage of correct registration results and the average running times of the different algorithms for the eight different scenes available for testing in the *3-DMatch* dataset.

In *3-DMatch*, a registration is considered successful when its associated transformation has a rotation error smaller than 10°, and a translation error below 30 cm. According to Table III, CliReg has the highest overall average of successful registrations (more than 89%), and is the best in four of the eight families reported. The second best algorithms, on average, are CliRegMutualand TEASER++. The former is slightly less accurate than CliReg, but more than twice as fast on average. The (slight) differences observed between the reported results of TEASER++ in [75] with respect to performance and registration accuracy could be explained by our single-threaded execution (while 12 cores were used in [75]) and our choice of noise parameter bound ξ (6).

Table III also reports that, on average, TEASER++ achieves slightly better registrations than CliReg in the *Hotel 1*, *Study*, and *MITLab* scenes. We propose two potential explanations for this outcome. First, CliReg is not fully deterministic. A source of variability stems from the fixed time limit imposed on its initial clique-finding heuristic (see Section VI-D). Second, and more significantly, CliReg does not enumerate all maximum cliques, prioritizing efficiency instead. To the best of the authors' knowledge, TEASER++ also does not enumerate all maximum cliques. Therefore, when multiple maximum cliques exist, the associated registrations can differ. It is plausible that in those scenes where TEASER++ outperformed CliReg, it identified a better maximum clique than CliReg.

To complement the information provided in Table III, Figs. 13 and 14 show the average translation and rotation errors, on logarithmic scale, of the reported algorithms. The dashed red line is the threshold below which the registration is considered valid in *3-DMatch*. In the case of translation error, the threshold is set at 30 cm. In the case of rotation error, it is set at 10° (≈ 0.175 rad). The best average mean translation and rotation error, considering all scenes, is provided by CliReg and RANSAC. Specifically, the translation error is slightly above 18 cm and the rotation error slightly above 0.05 rad in both cases. The worse registration is provided by FGR, both in terms of translation and rotation errors. Additional results are available in Appendix B.

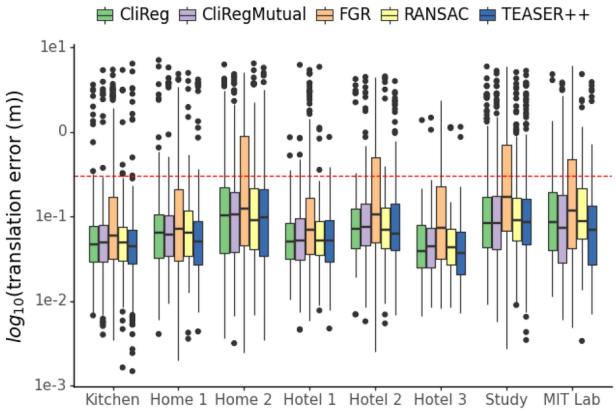


Fig. 13. Translation errors of the different algorithms on each of the eight test scenes of the *3-DMatch* dataset [77].

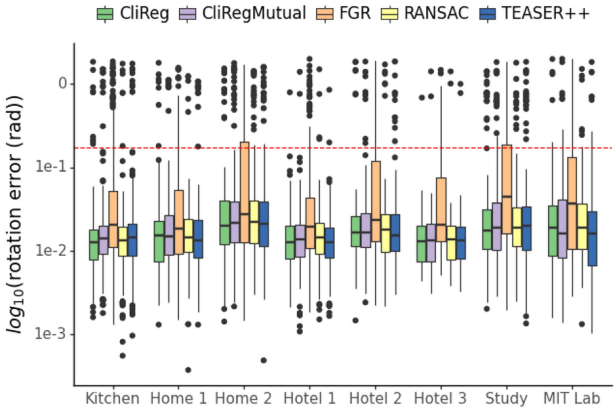


Fig. 14. Rotation errors of the different algorithms on each of the eight test scenes of the *3-DMatch* dataset [77].

Finally, the behavior of the components of `CliReg` is reported in Table IV, which shows the same data as Table I for synthetic experiments with the exception of the ground truth value of the number of inliers $|\mathcal{I}_{\mathcal{M}}|$, which is not provided (the ground truth in *3-DMatch* is described in terms of sensor movement). From the table, similar trends to the synthetic experiments can be observed: i) the correspondence graphs are sparse, with a maximum average density of 3.9% in the *Home 2* family; ii) the number of inliers of the registration reported by `CliReg`, $|\mathcal{I}_{\mathcal{M}}(K_{\text{reg}})|$, is always greater than the one corresponding to a maximum clique in the correspondence graph $|\mathcal{I}_{\mathcal{M}}(\omega)|$, although the differences are less pronounced than in the synthetic case.

VIII. Conclusion

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 MCP), efficient techniques employed by state-of-the-art maximum clique algorithms (e.g., vertex ordering, branching by partitioning, and incremental branching), and continuous optimization (e.g., solving a least-squares problem to evaluate cliques).

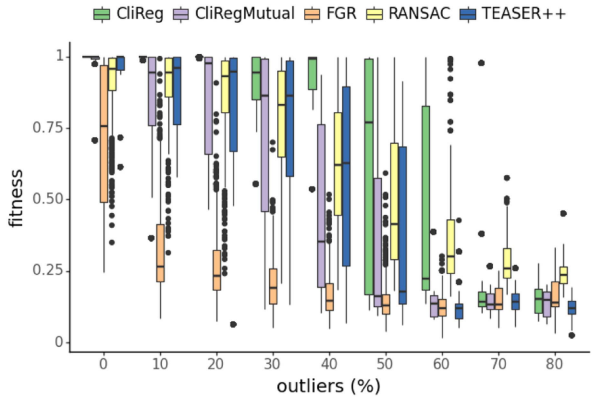


Fig. 15. Fitness of the different algorithms for our 135 subset of armadillo instances.

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 from well-known public 3-D models, and also using real-world indoor scenes. 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. Regarding performance, `CliRegMutual` can run in milliseconds and is comparable to the fastest heuristic `FGR`. Consequently, the combined two algorithms can be considered 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 by improving its fitness clique evaluation function to make it less sensitive to outliers. Interesting future work is, in our opinion, the study of alternative definitions of ξ, the analysis of different local point features, the analysis of different values of k in the KNN algorithm, and the addition of a preliminary step to estimate the scale in the case of nonrigid registration problems. Also of interest is the inclusion 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. Lastly, the behavior of the algorithm in the correspondence-free case, i.e., using all possible matching hypotheses, also deserves consideration.

Appendix A
Additional Results on Synthetic Data

This appendix covers additional results of the experiments on synthetic data reported in Section VII-C. The section includes the fitness and execution times of the different algorithms for the subset of armadillo instances (Figs. 15 and 18), buddha instances (Figs. 16 and 19), and dragon instances (Figs. 17 and 20), respectively.

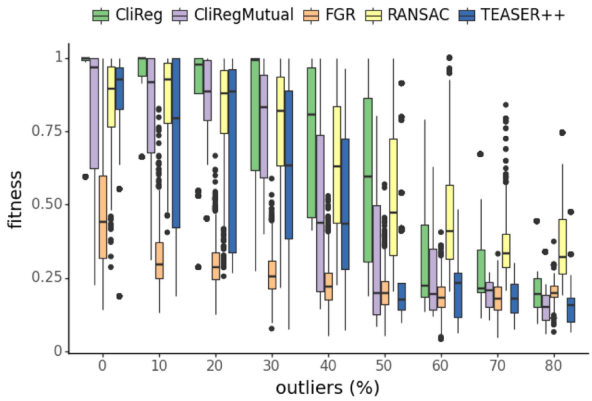


Fig. 16. Fitness of the different algorithms for our 135 subset of buddha instances.

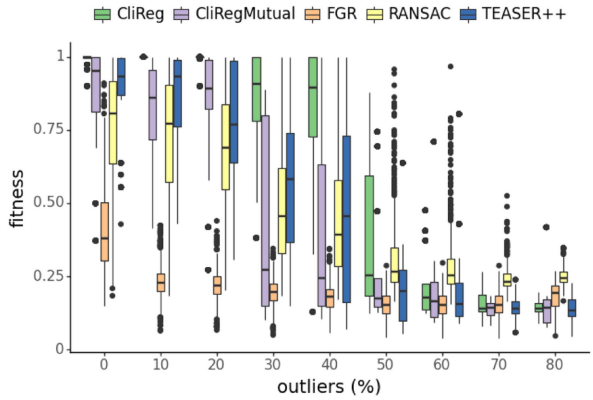


Fig. 17. Fitness of the different algorithms for our 135 subset of dragon instances.

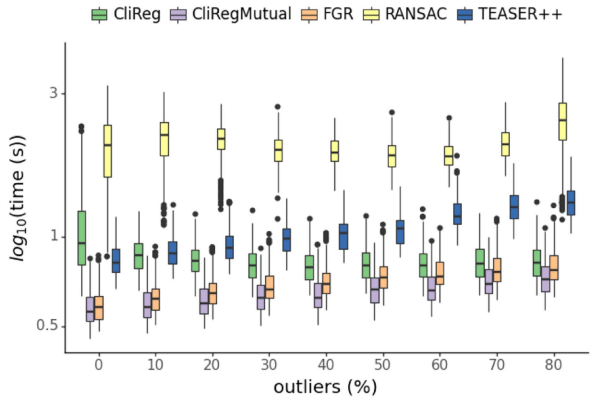


Fig. 18. Execution times (in logarithmic scale) of the different algorithms for our 135 armadillo instances.

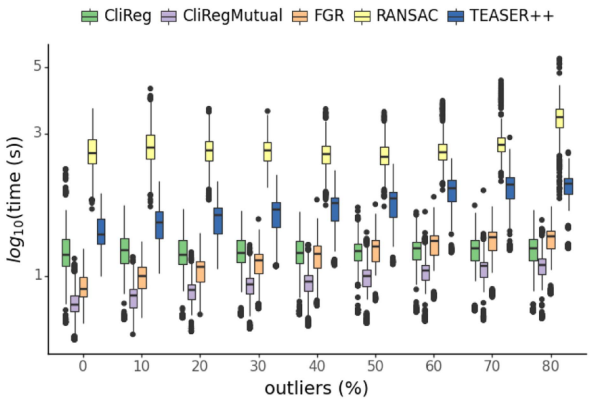


Fig. 19. Execution times (in logarithmic scale) of the different algorithms for our 135 buddha instances.

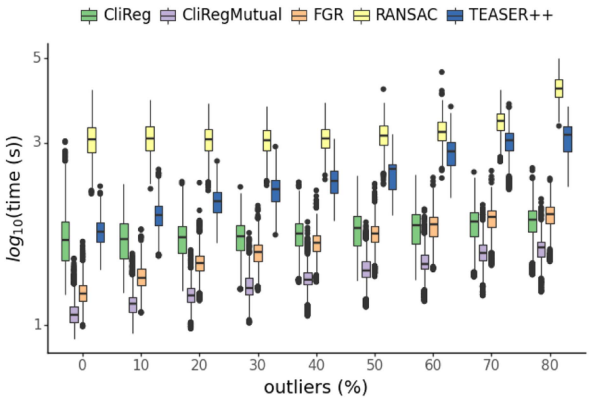


Fig. 20. Execution times (in logarithmic scale) of the different algorithms for our 135 dragon instances.

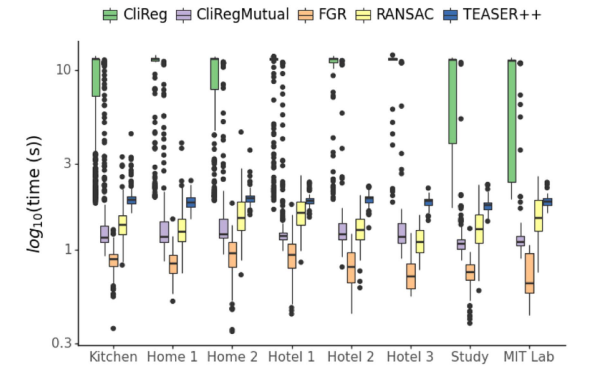


Fig. 21. Execution times (in logarithmic scale) of the different algorithms for the 8 test scenes of the 3-DMatch dataset [77].

The appendix also shows the results of the components of `CliReg` in the subset of armadillo instances (see Table V), buddha instances (see Table VI), and dragon instances (see Table VII). Each table shows aggregated results for the nine different outlier rates. The reported parameters are the same as explained in Table I of Section VII-B.

In addition, Table VIII presents the results of the ANOVA for the fitness of the registration and execution times, considering up to 50% outliers. The first three 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., [64], for a detailed description of the ANOVA analysis.

TABLE VIII
ANOVA FITNESS AND EXECUTION TIME ANALYSIS ON THE SUBSET OF SYNTHETIC INSTANCES WITH UP TO 50% OF OUTLIERS

Type	Var	Source	#dof	SS	F-value	p-value
armadillo	fitness	algorithms	4	733.210	2441.663	0.000
		residuals	17 995	1350.936		
	time (s)	algorithms	4	4557.975	31 448.170	0.000
		residuals	17 995	652.031		
bunny	fitness	algorithms	4	887.875	3042.571	0.000
		residuals	17 995	1312.813		
	time (s)	algorithms	4	5673.096	1475.628	0.000
		residuals	17 995	17 295.573		
buddha	fitness	algorithms	4	591.624	2323.762	0.000
		residuals	17 995	1145.371		
	time (s)	algorithms	4	6751.716	31 257.512	0.000
		residuals	17 995	971.744		
dragon	fitness	algorithms	4	670.901	2288.764	0.000
		residuals	17 995	1318.711		
	time (s)	algorithms	4	7501.790	27 767.833	0.000
		residuals	17 995	1215.388		
total	fitness	algorithms	4	2804.962	9328.113	0.000
		Residuals	71 995	5412.222		
	time (s)	algorithms	4	22 018.068	12 984.899	0.000
		residuals	71 995	30 519.891		

TABLE IX
TUKEY TEST FOR THE FITNESS VARIABLE ON THE SUBSET OF SYNTHETIC INSTANCES WITH UP TO 50% OF OUTLIERS

Algorithms		μ_d	$\underline{\mu_d}$	$\overline{\mu_d}$	p-value
CliRegMutual	CliReg	–0.186	–0.195	–0.177	0.000
FGR	CliReg	–0.594	–0.603	–0.586	0.000
RANSAC	CliReg	–0.156	–0.164	–0.147	0.000
TEASER++	CliReg	–0.188	–0.197	–0.179	0.000
FGR	CliRegMutual	–0.408	–0.417	–0.399	0.000
RANSAC	CliRegMutual	0.031	0.022	0.040	0.000
TEASER++	CliRegMutual	–0.002	–0.011	0.007	0.968
RANSAC	FGR	0.439	0.430	0.448	0.000
TEASER++	FGR	0.406	0.397	0.415	0.000
TEASER++	RANSAC	–0.033	–0.042	–0.024	0.000

TABLE X
TUKEY TEST FOR THE EXECUTION TIME ON THE SUBSET OF SYNTHETIC INSTANCES WITH UP TO 50% OF OUTLIERS

Algorithms		μ_d	$\underline{\mu_d}$	$\overline{\mu_d}$	p-value
CliRegMutual	CliReg	–0.526	–0.547	–0.506	0.000
FGR	CliReg	–0.405	–0.426	–0.384	0.000
RANSAC	CliReg	1.044	1.023	1.065	0.000
TEASER++	CliReg	0.035	0.014	0.056	0.000
FGR	CliRegMutual	0.121	0.100	0.142	0.000
RANSAC	CliRegMutual	1.571	1.550	1.592	0.000
TEASER++	CliRegMutual	0.562	0.541	0.583	0.000
RANSAC	FGR	1.450	1.429	1.471	0.000
TEASER++	FGR	0.441	0.420	0.462	0.000
TEASER++	RANSAC	–1.009	–1.030	–0.988	0.000

Finally, we report the Tukey tests that compare the fitness (see Table IX) and the execution times (see Table X) of every pair of algorithms considering the subset of instances with up to 50% outliers. In both tables, the column μ_d shows the difference between the means of the variable under study for the corresponding pair of algorithms, while $\underline{\mu_d}$ and $\overline{\mu_d}$ denote the lower and upper confidence intervals for μ_d, respectively. The final column of the table displays the p-value.

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 0.88, and shows reduced variability. The second-best algorithm is `RANSAC`, with a mean value of 0.72, closely followed by `CliRegMutual` and `TEASER++`, with a similar mean value (0.69) but greater variability. In terms of execution times, the fastest algorithm is `CliRegMutual` with average running time of 0.88 s.

APPENDIX B
ADDITIONAL RESULTS ON REAL DATA

This section covers additional results of the experiments on the *3-DMatch* dataset [77] reported in the experimental Section VII-D. Fig. 21 shows the execution times of the different algorithms for the eight test scenes of the dataset. A comparison of fitness values for the different algorithms is not reported since the ground-truth number of inliers $\overline{\mathcal{I}_{\mathcal{M}}}$ is unavailable for the *3-DMatch* dataset (the ground truth is described in terms of sensor movement).

ACKNOWLEDGMENT

The authors would like to thank the anonymous referees for their interesting suggestions and insights, which have contributed to the improvement of this work.

REFERENCES

[1] "The stanford 3-D scanning repository," 1996. Accessed: Mar. 14, 2024. [Online]. Available: http://graphics.stanford.edu/data/3Dscanrep/
[2] Y. Aoki, H. Goforth, R. A. Srivatsan, and S. Lucey, "PointNetLK: Robust & efficient point cloud registration using pointNet," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 7163–7172.
[3] K. S. Arun, T. S. Huang, and S. D. Blostein, "Least-squares fitting of two 3-D point sets," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-9, no. 5, pp. 698–700, Sep. 1987.
[4] M. A. Audette, F. P. Ferrie, and T. M. Peters, "An algorithmic overview of surface registration techniques for medical imaging," *Med. Image Anal.*, vol. 4, no. 3, pp. 201–217, 2000.
[5] A. Avetisyan, M. Dahnert, A. Dai, M. Savva, A. X. Chang, and M. Nießner, "Scan2CAD: Learning cad model alignment in RGB-D scans," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 2614–2623.
[6] B. Bellekens, V. Spruyt, R. Berkvens, R. Penne, and M. Weyn, "A benchmark survey of rigid 3D point cloud registration algorithms," *Int. J. Adv. Intell. Syst.*, vol. 8, no. 5, pp. 118–127, 2015.
[7] S. Belongie, J. Malik, and J. Puzicha, "Shape matching and object recognition using shape contexts," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 4, pp. 509–522, Apr. 2002.
[8] F. Bernard, C. Theobalt, and M. Moeller, "DS*: Tighter lifting-free convex relaxations for quadratic matching problems," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 4310–4319.
[9] L. Bernreiter, L. Ott, J. Nieto, R. Siegwart, and C. Cadena, "PHASER: A robust and correspondence-free global pointcloud registration," *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 855–862, Apr. 2021.
[10] P. J. Besl and N. D. McKay, "Method for registration of 3-D shapes," in *Sensor Fusion IV: Control Paradigms and Data Structures*, vol. 1611. Bellingham, WA, USA: SPIE, 1992, pp. 586–606.
[11] M. J. Black and A. Rangarajan, "On the unification of line processes, outlier rejection, and robust statistics with applications in early vision," *Int. J. Comput. Vis.*, vol. 19, no. 1, pp. 57–91, 1996.
[12] G. Blais and M. D. Levine, "Registering multiview range data to create 3D computer objects," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 17, no. 8, pp. 820–824, Aug. 1995.
[13] T. M. Breuel, "Implementation techniques for geometric branch-and-bound matching methods," *Comput. Vis. Image Understanding*, vol. 90, no. 3, pp. 258–294, 2003.

[14] A. P. Bustos and T.-J. Chin, "Guaranteed outlier removal for point cloud registration with correspondences," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 12, pp. 2868–2882, Dec. 2018.

[15] Z. Cai, T.-J. Chin, and V. Koltun, "Consensus maximization tree search revisited," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 1637–1645.

[16] D. Campbell, L. Petersson, L. Kneip, and H. Li, "Globally-optimal inlier set maximisation for simultaneous camera pose and feature correspondence," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 1–10.

[17] R. Carraghan and P. M. Pardalos, "An exact algorithm for the maximum clique problem," *Operations Res. Lett.*, vol. 9, no. 6, pp. 375–382, 1990.

[18] Y. Chen and G. Medioni, "Object modelling by registration of multiple range images," *Image Vis. Comput.*, vol. 10, no. 3, pp. 145–155, 1992.

[19] D. Chetverikov, D. Stepanov, and P. Krsek, "Robust euclidean alignment of 3D point sets: The trimmed iterative closest point algorithm," *Image Vis. Comput.*, vol. 23, no. 3, pp. 299–309, 2005.

[20] T.-J. Chin and D. Suter, *The Maximum Consensus Problem: Recent Algorithmic Advances*. Berlin, Germany: Springer Nature, 2022.

[21] T.-J. Chin, Z. Cai, and F. Neumann, "Robust fitting in computer vision: Easy or hard," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 701–716.

[22] C. Choy, J. Park, and V. Koltun, "Fully convolutional geometric features," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 8958–8966.

[23] C. Choy, W. Dong, and V. Koltun, "Deep global registration," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 2514–2523.

[24] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[25] W. Forstner and K. Khoshelham, "Efficient and accurate registration of point clouds with plane to plane correspondences," in *Proc. IEEE Int. Conf. Comput. Vis. Workshops*, 2017, pp. 2165–2173.

[26] Z. Gojcic, C. Zhou, J. D. Wegner, and W. Andreas, "The perfect match: 3D point cloud matching with smoothed densities," in *Proc. Int. Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 5540–5549.

[27] S. Granger and X. Pennec, "Multi-scale EM-ICP: A fast and robust approach for surface registration," in *Proc. Comput. Vis. 7th Eur. Conf. Comput. Vis.*, Copenhagen, Denmark, Springer, 2002, pp. 418–432.

[28] M. Gschwandtner, R. Kwitt, A. Uhl, and W. Pree, "BlenSor: Blender sensor simulation toolbox," in *Advances in Visual Computing* (Lecture Notes in Computer Science), G. Bebis et al., Eds., Berlin, Heidelberg: Springer, 2011, pp. 199–208.

[29] G. Guo, H. Wang, D. Bell, Y. Bi, and K. Greer, "KNN model-based approach in classification," in *On The Move to Meaningful Internet Systems* (Lecture Notes in Computer Science), R. Meersman, Z. Tari, and Douglas C. Schmidt, Eds., Berlin, Heidelberg: Springer, 2003, pp. 986–996.

[30] R. I. Hartley and F. Kahl, "Global optimization through rotation space search," *Int. J. Comput. Vis.*, vol. 82, no. 1, pp. 64–79, 2009.

[31] B. K. Horn, "Closed-form solution of absolute orientation using unit quaternions," *Josa A*, vol. 4, no. 4, pp. 629–642, 1987.

[32] D. F. Huber and M. Hebert, "Fully automatic registration of multiple 3D data sets," *Image Vis. Comput.*, vol. 21, no. 7, pp. 637–650, 2003.

[33] J. P. Iglesias, C. Olsson, and F. Kahl, "Global optimality for point set registration using semidefinite programming," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 8287–8295.

[34] A. E. Johnson and M. Hebert, "Using spin images for efficient object recognition in cluttered 3D scenes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 21, no. 5, pp. 433–449, May 1999.

[35] I. Kezurer, S. Z. Kovalsky, R. Basri, and Y. Lipman, "Tight relaxation of quadratic matching," *Comput. Graph. Forum*, vol. 34, no. 5, pp. 115–128, 2015.

[36] N. Lavnikevich, "On the complexity of maximum clique algorithms: Usage of coloring heuristics leads to the $2^{\langle n \backslash 5 \rangle}$ algorithm running time lower bound," 2013, *arXiv:1303.2546*.

[37] H. Le, T.-J. Chin, A. Eriksson, T.-T. Do, and D. Suter, "Deterministic approximate methods for maximum consensus robust fitting," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 3, pp. 842–857, Mar. 2021.

[38] M. Huu, T.-T. Le, T. D. Hoang, and N.-M. Cheung, "SDRSAC: Semidefinite-based randomized approach for robust point cloud registration without correspondences," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 124–133.

[39] C.-M. Li and Z. Quan, "An efficient branch-and-bound algorithm based on maxsat for the maximum clique problem," in *Proc. 24th AAAI Conf. Artif. Intell.*, 2010, pp. 128–133.

[40] C.-M. Li, Z. Fang, and K. Xu, "Combining maxsat reasoning and incremental upper bound for the maximum clique problem," in *Proc. IEEE 25th Int. Conf. Tools Artif. Intell.*, 2013, pp. 939–946.

[41] C.-M. Li, H. Jiang, and F. Manyà, "On minimization of the number of branches in branch-and-bound algorithms for the maximum clique problem," *Comput. Operations Res.*, vol. 84, pp. 1–15, 2017.

[42] C.-M. Li, Z. Fang, H. Jiang, and K. Xu, "Incremental upper bound for the maximum clique problem," *INFORMS J. Comput.*, vol. 30, no. 1, pp. 137–153, 2018.

[43] C.-M. Li, Y. Liu, H. Jiang, F. Manyà, and Y. Li, "A new upper bound for the maximum weight clique problem," *Eur. J. Oper. Res.*, vol. 270, no. 1, pp. 66–77, 2018.

[44] G. D. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, pp. 91–110, 2004.

[45] L. Maier-Hein et al., "Convergent iterative closest-point algorithm to accomodate anisotropic and inhomogenous localization error," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 8, pp. 1520–1532, Aug. 2012.

[46] E. Maslov, M. Batsyn, and P. M. Pardalos, "Speeding up branch and bound algorithms for solving the maximum clique problem," *J. Glob. Optim.*, vol. 590, no. 1, pp. 1–21, 2014.

[47] D. W. Matula, G. Marble, and J. D. Isaacson, "Graph coloring algorithms," in *Graph Theory and Computing*. Cambridge, MA, USA: Academic Press, 1972, pp. 109–122.

[48] S. Miyazaki, "Database queries as combinatorial optimization problems," in *Proc. Int. Symp. Cooperative Database Syst. Adv. Appl.*, 1996, pp. 448–454.

[49] J. W. Moon and L. Moser, "On cliques in graphs," *Isr. J. Math.*, vol. 3, pp. 23–28, 1965.

[50] A. Nüchter, K. Lingemann, J. Hertzberg, and H. Surmann, "6D SLAM—3D mapping outdoor environments," *J. Field Robot.*, vol. 240, no. 8-9, pp. 699–722, 2007.

[51] F. Pomerleau, F. Colas, R. Siegwart, and S. Magnenat, "Comparing ICP variants on real-world data sets: Open-source library and experimental protocol," *Auton. Robots*, vol. 34, pp. 133–148, 2013.

[52] R. A. Rossi et al., "Fast maximum clique algorithms for large graphs," in *Proc. 23rd Int. Conf. World Wide Web*, 2014, pp. 365–366.

[53] B. Radu, N. R. Blodow, and M. Beetz, "Fast point feature histograms (FPFH) for 3D registration," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2009, pp. 3212–3217.

[54] P. San Segundo and C. Tapia, "Relaxed approximate coloring in exact maximum clique search," *Comput. Operations Res.*, vol. 44, pp. 185–192, 2014.

[55] P. San Segundo, D. Rodriguez-Losada, and A. Jimenez, "An exact bit-parallel algorithm for the maximum clique problem," *Comput. Operations Res.*, vol. 380, no. 2, pp. 571–581, 2011.

[56] P. San Segundo, F. Matia, D. Rodriguez-Losada, and M. Hernando, "An improved bit parallel exact maximum clique algorithm," *Optim. Lett.*, 70, no. 3, pp. 467–479, 2013.

[57] P. San Segundo, A. Nikolaev, and M. Batsyn, "Infra-chromatic bound for exact maximum clique search," *Comput. Operations Res.*, vol. 64, pp. 293–303, 2015.

[58] P. San Segundo, A. Lopez, M. Batsyn, A. Nikolaev, and P. M. Pardalos, "Improved initial vertex ordering for exact maximum clique search," *Appl. Intell.*, vol. 450, no. 3, pp. 868–880, 2016.

[59] P. San Segundo, A. Lopez, and P. M. Pardalos, "A new exact maximum clique algorithm for large and massive sparse graphs," *Comput. Operations Res.*, vol. 66, pp. 81–94, 2016.

[60] P. San Segundo, A. Nikolaev, M. Batsyn, and P. M. Pardalos, "Improved infra-chromatic bound for exact maximum clique search," *Informatica*, vol. 270, no. 2, pp. 463–487, 2016.

[61] P. San Segundo, S. Coniglio, F. Furini, and I. Ljubić, "A new branch-and-bound algorithm for the maximum edge-weighted clique problem," *Eur. J. Oper. Res.*, vol. 2780, no. 1, pp. 76–90, 2019.

[62] P. San Segundo, F. Furini, D. Álvarez, and P. M. Pardalos, "CLISAT: A new exact algorithm for hard maximum clique problems," *Eur. J. Oper. Res.*, vol. 3070, no. 3, pp. 1008–1025, 2023.

[63] A. Segal, D. Haehnel, and S. Thrun, "Generalized-ICP," *Robot. Sci. Syst.*, vol. 2, 2009, pp. 1–8.

[64] L. Sthle and S. Wold, "Analysis of variance (ANOVA)," *Chemometrics Intell. Lab. Syst.*, vol. 6, no. 4, pp. 259–272, Nov. 1989.

[65] G. K. L Tam et al., "Registration of 3D point clouds and meshes: A survey from rigid to nonrigid," *IEEE Trans. Visual. Comput. Graph.*, vol. 190, no. 7, pp. 1199–1217, Jul. 2013.

[66] F. Tombari, S. Salti, and L. D. Stefano, "Performance evaluation of 3D keypoint detectors," *Int. J. Comput. Vis.*, vol. 1020, no. 1-3, pp. 198–220, 2013.

[67] E. Tomita, A. Tanaka, and H. Takahashi, "The worst-case time complexity for generating all maximal cliques and computational experiments," *Theor. Comput. Sci.*, vol. 3630, no. 1, pp. 28–42, 2006.
[68] G. Verfaillie, M. Lemaitre, and T. Schiex, "Russian doll search for solving constraint optimization problems," in *Proc. AAAI/IAAI*, 1996, pp. 181–187.
[69] M. W. Walker, L. Shao, and R. A. Volz, "Estimating 3-D location parameters using dual number quaternions," *CVGIP: Image Understanding*, vol. 540, no. 3, pp. 358–367, 1991.
[70] Y. Wang and J. M. Solomon, "PRNet: Self-supervised learning for partial-to-partial registration," *Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, Art. no. 791.
[71] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph CNN for learning on point clouds," *ACM Trans. Graph.*, vol. 380, no. 5, pp. 1–12, 2019.
[72] F. Wen, R. Ying, Z. Gong, and P. Liu, "Efficient algorithms for maximum consensus robust fitting," *IEEE Trans. Robot.*, vol. 360, no. 1, pp. 92–106, Feb. 2020.
[73] Q. Wu and J.-K. Hao, "An adaptive multistart tabu search approach to solve the maximum clique problem," *J. Combinatorial Optim.*, vol. 26, pp. 86–108, 2013.
[74] H. Yang and L. Carlone, "A polynomial-time solution for robust registration with extreme outlier rates," in *Proc. Robot.: Sci. Syst. XV*, A. Bicchi, H. Kress-Gazit, and S. Hutchinson, Freiburg im Breisgau, Germany, 2019.
[75] H. Yang, J. Shi, and L. Carlone, "TEASER: Fast and certifiable point cloud registration," *IEEE Trans. Robot.*, vol. 370, no. 2, pp. 314–333, Apr. 2021.
[76] J. Yang, H. Li, D. Campbell, and Y. Jia, "Go-ICP: A globally optimal solution to 3D ICP point-set registration," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 380, no. 11, pp. 2241–2254, Nov. 2016.
[77] A. Zeng, S. Song, M. Nießner, M. Fisher, J. Xiao, and T. Funkhouser, "3DMatch: Learning local geometric descriptors from RGB-D reconstructions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 1802–1811.
[78] Z. Zhang, "Iterative point matching for registration of free-form curves and surfaces," *Int. J. Comput. Vis.*, vol. 130, no. 2, pp. 119–152, 1994.
[79] Q.-Y. Zhou, J. Park, and V. Koltun, "Fast global registration," in *Proc. Comput. Vis. 14th Eur. Conf.*, Amsterdam, The Netherlands, 2016, pp. 766–782.

Javier Laserna received the B.Eng. degree in industrial electronics and automation engineering in 2020 and the master's degree in industrial electronics in 2022 from the Universidad Politécnica de Madrid (UPM), Madrid, Spain, where he is working toward the Ph.D. degree with Center for Automation and Robotics (CAR).

He was a Site Reliability Engineer with IBM and Kyndryl and a postgraduate Researcher with CAR, UPM-CSIC. During doctoral studies he has been focusing on exact algorithms for NP-complete combinatorial problems and the design of social robotic platforms. His research interests include combinatorial optimization, computer vision, intelligent control, and deep learning on reconfigurable hardware.

P. San Segundo received the Ph.D. degree in robotics and artificial intelligence from the Universidad Politécnica de Madrid (UPM), Madrid, Spain, in 2003.

He is currently a Full Professor with Universidad Politécnica de Madrid (UPM), Madrid, Spain. His research interests include artificial intelligence and combinatorial optimization in graphs, with notable contributions to exact algorithms for challenging problems such as vertex coloring, maximum clique, the knapsack problem with conflicts, and applications in computer vision and robotics.

David Álvarez Sánchez received the Ph.D. degree in robotics and automation from the University Carlos III of Madrid (UC3M), Getafe, Spain, in 2016.

During 2012 and 2013, he has done research stages in Japan with Gifu University, and during 2014 and 2015 with German Aerospace Center. He is currently an Associate Professor with the Escuela Técnica Superior de Ingeniería y Diseño Industrial (ETSIDI-UPM), Madrid, Spain, and a Researcher with the Center for Automation and Robotics (CAR UPM-CSIC), Madrid. His research interests include different areas such as path planning, mobile manipulators, state estimation, 3-D environment modeling, evolutionary computation, and image processing problems.