CLIPPER+: A Fast Maximal Clique Algorithm for Robust Global Registration

Kaveh Fathian and Tyler Summers

***Abstract*—We present CLIPPER+, an algorithm for finding maximal cliques in unweighted graphs for outlier-robust global registration. The registration problem can be formulated as a graph and solved by finding its maximum clique. This formulation leads to extreme robustness to outliers; however, finding the maximum clique is an NP-hard problem, and therefore approximation is required in practice for large-size problems. The performance of an approximation algorithm is evaluated by its computational complexity (the lower the runtime, the better) and solution accuracy (how close the solution is to the maximum clique). Accordingly, the main contribution of CLIPPER+ is outperforming the state-of-the-art in accuracy while maintaining a relatively low runtime. CLIPPER+ builds on prior work (CLIPPER Lusk et al. (2021) and PMC Rossi et al. (2015)) and prunes the graph by removing vertices that have a small core number and cannot be a part of the maximum clique. This will result in a smaller graph, on which the maximum clique can be estimated considerably faster. We evaluate the performance of CLIPPER+ on standard graph benchmarks, as well as synthetic and real-world point cloud registration problems. These evaluations demonstrate that CLIPPER+ has the highest accuracy and can register point clouds in scenarios where over 99% of associations are outliers. Our code and evaluation benchmarks will be released at https://github.com/ariarobotics/clipperp.**

***Index Terms*—Localization, SLAM, RGB-D perception.**

I. Introduction

Data association is broadly defined as the correspondence of identical/similar elements across sets of data, and is a key component of many robotics and computer vision applications, such as localization and mapping [3], [4], [5], point cloud registration [6], [7], shape alignment [8], object detection [9], data fusion [10], [11], and multi-object tracking [12]. In these applications, it is crucial that data association is solved correctly and fast.

In point cloud registration, for example, we seek to find the rigid transformation (rotation/translation) that aligns two sets of 3D points. This requires associating points in one set with their corresponding points in the other set. Local registration techniques such as the Iterative Closest Point (ICP) algorithm [13] associate points based on their nearest neighbor. These associations are generally wrong if the point clouds are not aligned well initially, leading to wrong registration. Better associations can be established by matching descriptors that are computed around each point in the point cloud and describe the local geometry and appearance of a point (e.g., classical FPFH [14] or modern learning-based 3DMatch [15]). However, due to noise, repetitive patterns, small overlap between the point clouds, etc., these putative associations can have extreme outlier ratios (e.g., FPFH associations in Section V-C are 99% outliers/wrong). In these high-outlier regimes, existing outlier rejection techniques (such as the general RANSAC framework [16] or specific frameworks for point cloud registration [17]) either return wrong results or have impractical runtime (e.g., RANSAC's runtime grows exponentially in outlier ratio [6]).

To address these issues, we present the CLIPPER+ algorithm. CLIPPER+ formulates the data association problem as a graph, in which the inlier/correct associations are the maximum clique. This formulation is robust to high outlier ratios and applicable to any problem that admits *invariants* (see Section III) such as point, line, and plane could registration [1], [18], [19]. To address the high computational complexity of finding the maximum clique (NP-hardness), CLIPPER+ finds an approximate solution instead, which is obtained from combining an improved version of our prior work, CLIPPER [1], and the greedy maximal clique algorithm in [2]. CLIPPER+ runs in polynomial time and outperforms state-of-the-art algorithms in maximum clique estimation accuracy (Section V-A). Further, CLIPPER+ solutions are shown to be exact (i.e., the maximum clique) in over 99% of the point cloud registration trials (Section V-C).

Contributions: In summary, this work's contributions are:

- An improved solver (Algorithm 2) leading to higher accuracy over our prior work CLIPPER [1].
- The new CLIPPER+ algorithm as the combination of the greedy algorithm in [2] and CLIPPER [1], further improving both runtime and accuracy.
- Evaluations demonstrating superior accuracy of CLIPPER+ over state-of-the-art on maximum clique problems, and correct registration of real-world point clouds in regimes with over 99% outliers.
- Efficient C++ implementation of all algorithms (open-source code will be released).

II. Related Work

Robust registration: In robotic applications of data association, such as registration, formulations that are robust to uncertainties (such as extreme noise and outliers) are based on globally

Manuscript received 29 September 2023; accepted 15 January 2024. Date of publication 21 February 2024; date of current version 5 March 2024. This letter was recommended for publication by Associate Editor M. Camurri and Editor J. Civera upon evaluation of the reviewers' comments. This work was supported in part by the United States Air Force Office of Scientific Research under Grant FA9550-23-1-0424, in part by National Science Foundation under Grant ECCS-2047040, and in part by ARL SARA under Grant W911NF2420029. *(Corresponding author: Kaveh Fathian.)*

Kaveh Fathian is with the Department of Computer Science, Colorado School of Mines, Golden, CO 80401 USA (e-mail: fathian@ariarobotics.com).

Tyler Summers is with the Department of Mechanical Engineering, University of Texas at Dallas, Richardson, TX 75080 USA (e-mail: tyler.summers@utdallas.edu).

Digital Object Identifier 10.1109/LRA.2024.3368233

optimal solvers, such as Branch and Bound [6], [20], [21], and combinatorial approaches for maximum consensus [22]. These methods can tolerate extreme outlier ratios, but have worst-case exponential complexity and are slow in practice. To address this issue, fast heuristics such as RANSAC [16] and its variants [23], graduated non-convexity [17], [24], [25], or iterative local solvers for M-estimation [26], [27] are used, however, they are prone to failures in high-outlier regimes [6], [7], [28].

Graph-theoretic formulation: The dichotomy between robustness and runtime can be resolved in some data association settings that admit "invariants" (see Section III, and [1], [18], [19]) by formulation in a graph-theoretic framework. Recent algorithms that leverage this framework are CLIPPER [1], TEASER [7], and ROBIN [19] for global point cloud registration, robust to more than 99% outliers and suitable for real-time applications, and PCM [4] for pairwise-consistent loop closures in SLAM. This graph-theoretic formulation can be traced back to Bailey et al. [29] for 2D LiDAR registration as a maximum common subgraph problem, for which the maximum clique indicated the correct data association. Leordeanu and Hebert [30] extended this graph-theoretic framework to weighted consistency graphs. Enqvist et al. [31] noted the suboptimality of [30] and proposed a vertex covering formulation, essentially an alternative to the maximum clique formulation of [29]. Parra et al. [32] proposed a practical maximum clique algorithm for geometric consistency based on branch and bound and graph coloring. Recently, Zhang et al. [33] have shown that the registration problem can be solved by relaxing the maximum clique constraint and using the maximal cliques instead.

Maximum clique estimation: Finding the maximum clique is a well-known **NP-hard** problem in its full generality [34], meaning that the complexity of finding the solution using exact algorithms (e.g., [2]) grows exponentially in graph size, which is impractical for data association applications. An approximate (maximal clique) solution can however be found by polynomial-time algorithms. Notable algorithms in this setting are Pelillo [35] and Ding et al. [36], based on the Motzkin-Straus formulation [37], Belachew and Gillis [38] based on symmetric rank-one nonnegative Matrix approximation, and our prior CLIPPER algorithm [1] based on a continuous relaxation.

III. Graph-Theoretic Robust Registration

The key idea for creating robustness to extreme outlier ratios is to find the *largest set of jointly consistent* data and/or associations. This problem can be formulated as a graph, in which this set is represented by the maximum clique. In what follows, we introduce this graph-theoretic framework and its use case for point cloud registration.

Point cloud registration: The objective of point cloud registration is to find the rigid transformation that aligns a set of points to their corresponding points in another set. The main challenge is finding the correct correspondences as usually only a subset of the points match, and, the points do not align perfectly due to noise. An *outlier* can be a point in one set that does not have a counterpart in the other set, or an association that matches wrong points across the sets. Our focus here is associations, and henceforth **outliers** imply *outlier associations*. Fig. 1 illustrates a point cloud registration example where we seek to find the blue bunny in the cluttered point cloud.

Maximum likelihood solution: In the absence of prior knowledge and when outliers are random, unbiased, and unstructured, *the largest set of jointly consistent associations* are inliers for the maximum likelihood solution. For point cloud registration, two associations are **consistent** if their endpoints are equidistant, and therefore can be aligned by a rigid transformation. In Fig 1, the green associations align 3 points while the red associations align 2 points. Thus, the green associations are the inliers and can be used to compute the correct maximum likelihood solution.

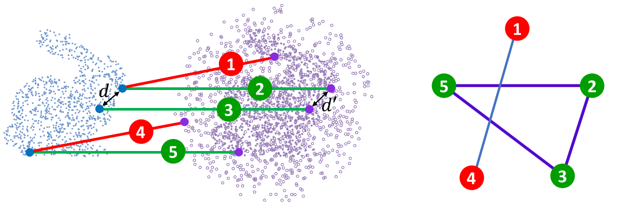

Fig. 1. Example of maximum clique formulation for robust global registration. (Left) Putative associations (red: outliers, green: inliers). (Right) Graph formulation with maximum clique indicating inlier associations.

Graph formulation: Finding the largest set of jointly consistent associations can be formulated as a maximum clique problem. Given n associations, the **consistency graph** is a graph of n vertices where each vertex represents an association. An edge between two vertices indicates that the associations are consistent. In Fig. 1, an edge between two vertices of the consistency graph (shown on the right) indicates consistency of corresponding associations. For example, there is an edge between vertices/associations 2 and 3 as their endpoints are equidistant ($d = d'$), while there is no edge between vertices/associations 1 and 3. Given two associations with endpoint distances d and d', due to noise, often a **consistency threshold** ϵ is used where if $|d - d'| < \epsilon$, the associations are deemed consistent.

A **clique** is a subset of vertices where every pair of vertices within that subset is connected by an edge. The **maximum clique** is the clique with the largest number of vertices. A **maximal clique** is a clique that is not contained in a larger clique. The maximum clique in Fig. 1 consists of vertices 2, 3, and 5. Vertices 1 and 4 form a maximal clique.

Invariants: The graph-theoretic framework can be applied to a broad array of data association problems in robotics. An **invariant** is a quantity that remains unchanged across the sets. The invariant used for the point cloud registration example above was the Euclidean distance between the endpoints. Invariants can be defined to register lines [1], planes [18], 2D–3D visual features [19], etc. Examples in this work, however, focus on point cloud registration.

IV. Maximal Clique Algorithms

We present the main algorithmic contributions of this work in Sections IV-B and IV-C, where we discuss the maximal clique algorithms based on a continuous relaxation and CLIPPER+. Section IV-A is a review of the approach in [2], and is not an algorithmic contribution. However, our C++ implementation of this algorithm improved the performance and accuracy compared to its original implementation.

A. Degeneracy-Ordered Greedy Maximal Clique Algorithm

Algorithm 1 presents a greedy approach for finding a maximal clique. Starting from an empty clique (line 3), we grow the clique one vertex at a time by looping through the vertices (line 6).

Algorithm 1: Degeneracy-Ordered Greedy Maximal Clique.

```
1: Input $A$: adjacency matrix; $K$: core numbers
2: Output $C$: vertices that form maximal clique
3: $C \leftarrow \{\}$; $c_{\max} \leftarrow 0$
4: % Sort vertices by core number in descending order:
5: $(v_1, \ldots, v_n) \leftarrow$ sort vertices $v_i$ s.t. $K(v_i) \geq K(v_j)$ if $i < j$
6: for $i = 1 : n$ do
7:   if $K(v_i) \geq c_{\max}$ then
8:     % Neighbors of $v_i$ with core numbers $\geq c_{\max}$:
9:     $S \leftarrow \{v_j : A(v_i, v_j) = 1, K(v_j) \geq c_{\max}\}$
10:    $(s_1, \ldots, s_k) \leftarrow$ sort $s_i \in S$ descending by core number
11:    $C' \leftarrow \{\}$
12:    for $j = 1 : k$ do % For each vertex in sorted $S$
13:      if $A(c_i, s_j) = 1 \, \forall_{c_i \in C'}$ then % $C' \cup \{s_j\}$ is clique
14:        $C' \leftarrow \{C', s_j\}$ % Add $s_j$ to $C'$
15:      if $|C'| > c_{\max}$ then % Found a larger clique
16:        $C \leftarrow C'$; $c_{\max} \leftarrow |C'|$ % Update the output
```

For each vertex v that this loop examines, the algorithm adds v to the current clique if it is connected to every vertex that is already in the clique, and discards v otherwise (lines 13-16). This algorithm has the overall runtime of $O(\delta|E|)$, where δ is the maximum vertex degree and $|E|$ is the number of edges [2].

The maximal clique returned by the greedy approach depends on the initial vertex chosen to grow the clique, and the ordering of vertices (as they are sequentially examined to be added to the current clique or discarded). A descending ordering of vertices by their core number greatly improves the odds of finding a large maximal clique (and possibly the maximum clique), as leveraged in Algorithm 1 (lines 5 and 10). Mathematically, the **core number** or *degeneracy* of a vertex v is the largest integer k such that the degree of v remains non-zero when all vertices of degree less than k are recursively removed from the graph. The core number of vertices can be computed efficiently in $O(|E|)$ by Batagelj and Zaversnik's algorithm [39].

B. Continuous-Relaxation Maximal Clique Algorithm

Optimization formulation: The maximum clique problem in an undirected and unweighted graph of n vertices can be formulated as

$$\begin{aligned} \underset{u \in \{0,1\}^n}{\text{maximize}} \quad & \sum_{i=1}^{n} u_i \\ \text{subject to} \quad & u_i \, u_j = 0 \quad \text{if } A(i,j) = 0, \ \forall_{i,j} \end{aligned} \tag{1}$$

where $A \in \{0,1\}^{n \times n}$ is the **adjacency matrix** with $A(i,j) = 1$ if and only if vertices u_i and u_j are connected. The optimization variable u is a binary vector of n elements, where 1 entries indicate vertices that form a clique. As u is binary, the constraint $u_i \, u_j = 0$ if $A(i,j) = 0$ implies that if vertices u_i or u_j are disconnected, then at most one of them can be selected. For example, consider the graph in Fig. 1 with the adjacency matrix A and solution candidates $u, \bar{u}$ as

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{bmatrix}, \quad u = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 1 \end{bmatrix}, \quad \bar{u} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}. \tag{2}$$

Algorithm 2: Continuous-Relaxation Maximal Clique.

```
Input $A$: adjacency matrix; $\bar{u}$: initial guess
Output $C$: vertices that form maximal clique
Parameters $\sigma \leftarrow 0.01$; $\beta \leftarrow 0.5$; $tol \leftarrow 10^{-8}$
$M \leftarrow A + I$ % Add identity matrix $I$
$\bar{M} \leftarrow \mathbf{1} - M$ % Binary complement of $M$
$u \leftarrow \max(\bar{u}/\|\bar{u}\|, 0)$;  $d \leftarrow d_0$;  $\alpha \leftarrow 1$
while $u$ not in binary state do
  $M_d \leftarrow M - d\,\bar{M}$
  $F \leftarrow u^\top M_d u$
  % Gradient projected on $S^n$ tangent bundle:
  $\nabla F_\perp(u) = 2(I - uu^\top) M_d u$
  $\Delta u \leftarrow tol$;  $\Delta F \leftarrow tol$
  while $\Delta u \not< tol$ or $\Delta F \not< tol$ do
    $Armijo \leftarrow$ False
    while $Armijo =$ False do % Backtracking line search
      $u_+ \leftarrow u + \alpha \nabla F_\perp(u)$ % Solution update
      $u_+ \leftarrow \max(u_+/\|u_+\|, 0)$ % Retract to $\mathbb{R}^n_+ \cap S^n$
      $F_+ \leftarrow u_+^\top M_d u_+$
      $\nabla F_\perp(u_+) = 2(I - u_+ u_+^\top) M_d u_+$
      $\Delta F \leftarrow F_+ - F$;  $\Delta u \leftarrow u_+ - u$
      $Armijo \leftarrow (\Delta F \geq \sigma \nabla F_\perp(u)^\top \Delta u)$
      if $Armijo =$ False then
        $\alpha \leftarrow \alpha\,\beta$ % Reduce $\alpha$
      else  $\alpha \leftarrow \alpha/\sqrt{\beta}$ % Increase $\alpha$
    $u \leftarrow u_+$;  $\nabla F_\perp(u) \leftarrow \nabla F_\perp(u_+)$;  $F \leftarrow F_+$
  $d \leftarrow d + \Delta d$ % Increase $d$
$C \leftarrow \{i : u_i > 0\}$ % Vertices that form maximal clique
```

Both u and $\bar{u}$ satisfy the constraints in (1). Solution u is the global optimum with the objective value of 3 (the size of the maximum clique, and the number of 1 entries in u), while $\bar{u}$ gives the objective value of 2.

By defining $M \overset{\text{def}}{=} A + I$, where I is the identity matrix of appropriate size, it is straightforward to show that problem (1) is equivalent to

$$\begin{aligned} \underset{u \in \{0,1\}^n}{\text{maximize}} \quad & \frac{u^\top M \, u}{u^\top u} \\ \text{subject to} \quad & u_i \, u_j = 0 \quad \text{if } M(i,j) = 0, \ \forall_{i,j}. \end{aligned} \tag{3}$$

This follows by observing that in (1) and (3) the constraints are identical in the sense that disconnected vertices cannot be selected jointly in the solution. Further, if u^* is an optimal solution of (3) and it has m one entries, then the objective values of (3) and (1) will both be identical and equal to m (e.g., u and $\bar{u}$ in (2) give objective values of 3 and 2 in (3)).

Continuous relaxation: To overcome the NP-hardness of problem (3), an approximate solution can be obtained by a continuous relaxation, where the binary domain $u \in \{0,1\}^n$ is relaxed to the set of non-negative real numbers $u \in \mathbb{R}^n_+$. Gradient-based optimization routines with polynomial time complexity can solve the relaxed problem, and the relaxed solution can be projected/rounded to the nearest binary. The issue of this approach is that there is no guarantee that the binarized solution is a feasible solution that satisfies the constraints of the original problem (3).

Our proposed relaxation of (3) that addresses this issue is

$$\underset{u \in \mathbb{R}^n_+}{\text{maximize}} \quad \frac{u^\top M_d \, u}{u^\top u}, \tag{4}$$

$$M_d(i,j) \overset{\text{def}}{=} \begin{cases} M(i,j) & \text{if } M(i,j) \neq 0 \\ -d & \text{if } M(i,j) = 0 \end{cases} \tag{5}$$

where $d > 0$ is a positive scalar. Problem (4) can be written equivalently as

$$\begin{aligned} \underset{u \in \mathbb{R}^n_+}{\text{maximize}} \quad & F(u) \overset{\text{def}}{=} u^\top M_d\, u \\ \text{subject to} \quad & \|u\| = 1 \end{aligned} \tag{6}$$

where $\|\cdot\|$ is the ℓ_2 vector norm.[1]

Our relaxation (6) is inspired by Belachew and Gillis [38], which integrates the constraints in (3) into (6) using the matrix M_d. The difference of this work with [38] is that in [38] the relaxation $\min_{u \in \mathbb{R}^n_+} \|M_d - uu^\top\|_F^2$ is used. Intuitively, any entry $M_d(i,j) = -d$ penalizes joint selection of disconnected vertices u_i and u_j by the amount $-2\,d$ in the objective. Hence, as d increases, the entries of u that violate clique constraints converge to zero.

Optimality guarantees: When $d \geq n$ (see (5)), the optima of the proposed relaxation (6) are theoretically guaranteed to correspond to the optima of the original maximum clique problem (3). That is, a local optimum of (6) corresponds to a maximal clique, and a global optimum of (6) corresponds to a maximum clique. It is by no means trivial to prove this statement, and the interested reader should refer to Theorems 3–5 in [38] that prove this statement for the relaxation $\min_{u \in \mathbb{R}^n_+} \|M_d - uu^\top\|_F^2$, and Theorem 2 in [38] that establishes connections to our proposed relaxation (6).

It is interesting to point out that any solution of (6) has a binary state (this follows from the analysis in [38]). That is, the entries of a solution vector $u^* \in \mathbb{R}^n_+$ are either 0 or equal to a positive scalar $c > 0$. These positive entries are indicators of vertices that form a maximal clique. Noting that (6) can be (locally) solved in polynomial time by using a gradient-based solver, and the one-to-one correspondence between the optima of (6) and the (NP-hard) maximum clique problem (3), one may think that the maximum clique problem can be solved in polynomial time. Note, however, that (6) is a nonlinear optimization problem, potentially with many local optima, and depending on the initial guess the solver can converge to a local optimum instead of the global optimum. Finding the global optimum of (6), hence, remains an NP-hard problem.

Optimization algorithm: We present a custom solver for (6) based on a projected gradient ascent approach, as described in Algorithm 2. Our approach is similar to the algorithm in our previous work CLIPPER [1]; however, compared to [1], Algorithm 2 is significantly improved and uses the Armijo procedure for selecting the appropriate step size, which leads to more accurate results (as we will show in the comparisons).

Problem (6) is nonlinear with many local optima in general (corresponding to maximal cliques). To improve the odds of finding the global optimum (maximum clique) and escape local optima, in Algorithm 2 we use a homotopy approach where we increase d incrementally in an outerloop (lines 7-26). As the penalty parameter d increases incrementally by Δd in each iteration of the outerloop (line 26), the elements of u that violate the clique constraints are penalized further and u converges to a feasible solution. This process continues until d is large enough ($d \geq n$) and u converges to a binary state, corresponding to a maximal clique. The Δd increments can be chosen as a small constant (as done in [38]), however, in our implementation, we use a greedy scheme where we increase d until the smallest element of u that violates the clique constraint goes to zero in the next iteration. In the final step (line 27), the vertices of the maximal clique are identified as the non-zero elements of u.

Algorithm 3: CLIPPER+

```
1:  Input $A$: adjacency matrix
2:  Output $C$: vertices that form maximal clique
3:  $K \leftarrow$ compute core number of vertices % From [39]
4:  $C \leftarrow$ Algorithm1$(A, K)$ % Degeneracy-ordered greedy clique
5:  $\mathcal{I} \leftarrow \{i : K(v_i) \geq |C|\}$ % Vertices with core number ≥ greedy
    clique size
6:  $A' \leftarrow A(\mathcal{I}, \mathcal{I})$ % Prune graph (only keep vertices in $\mathcal{I}$)
7:  if $A' = \emptyset$ Terminate then % Maximum clique found
8:  % Binary complement of greedy clique in pruned graph:
9:  $\bar{u} \leftarrow (\bar{u}_i : \bar{u}_i = 0$ if $i \in C$, else, $\bar{u}_i = 1, \forall_{i \in \mathcal{I}})$
10: $C' \leftarrow$ Algorithm2$(A', \bar{u})$ % Continuous-relaxation clique
11: if |C'| > |C|   then $C \leftarrow C'$ % Return larger clique
```

The innerloop (lines 13–25) ensures that for each d increment enough iterations of the gradient ascent are performed for the solution u to reach a steady state. Noting that the constraint manifold of the optimization problem (6) is $\mathbb{R}^n_+ \cap S^n$, where S^n is the unit sphere, to speed up the convergence, we project the gradient $\nabla F(u) \overset{\text{def}}{=} 2M_d\, u$ onto the tangent bundle of S^n at u and move along the orthogonal projection $\nabla F_\perp(u) = 2(I - uu^\top)M_d u$ (line 11).

To find an appropriate step size α along the projected gradient, we use backtracking line search (lines 15–24) with the Armijo procedure (lines 21–24), which guarantees a sufficient increase in the objective at each innerloop iteration. The convergence of the algorithm to a first-order optimal point is guaranteed by the convergence property of the projected gradient with Armijo steps [41]. The solution update is computed in line 16, retracted back onto the constraint manifold (line 17), and the gradient ascent continues until convergence.

Computational Complexity: The worst-case complexity of Algorithm 2 is $\mathrm{O}(n^4)$. This is because the gradient computation (line 19) involves matrix-vector multiplications, which have $\mathrm{O}(n^2)$ complexity, and profiling shows this is where most of the time is spent. The number of backtracking iterations (line 15) and gradient ascent iterations (line 13) can vary depending on the parameters and the data matrix, but it is linear in problem size ($\mathrm{O}(n)$) for quadratic objective $F(u) = u^\top M_d u$. Lastly, the number of outerloop iterations (line 7) depends on Δd increments (line 26) required to reach $d \geq n$ (at which point the solution is guaranteed to converge to the binary state). This is also linear in problem size.

C. CLIPPER+ Maximal Clique Algorithm

Both Algorithms 1 and 2 are algorithms for finding maximal cliques. The greedy approach of Algorithm 1 runs fast but gives relatively less accurate estimates of the maximum clique size when the graph is not sparse. In contrast, the optimization approach of Algorithm 2 is relatively slower but more accurate. The main motivation of the proposed CLIPPER+ algorithm is to

[1] Vector u can be written as $u = cu'$, where $c = \|u\|$, and $u' = u/c$ is a unit-norm vector. Replacing $u = cu'$ in (4) gives $\frac{u^\top M_d u}{u^\top u} = \frac{c^2 u'^\top M_d u'}{c^2 u'^\top u'} = u'^\top M_d\, u'$, demonstrating the equivalency to (6).

combine these two algorithms and thereby combine their relative benefits.

The enable this combination, recall that the **core number** of a vertex is the largest integer k such that the degree of the vertex remains non-zero when all vertices of degree less than k are removed. If a graph contains a clique of size k, then each vertex in the clique must have a degree of $k-1$ or larger, and therefore a core number of $k-1$ or larger. For this reason, if we hope to find a larger clique, say of size $k+1$, then only vertices that have a core number k or higher can be candidates. Using this observation, CLIPPER+, detailed in Algorithm 3, first runs the greedy algorithm and obtains a maximal clique (line 4). Assuming this clique has size k, if the graph contains a larger clique, then vertices must have core numbers of k or larger. Therefore, the algorithm prunes the graph by removing vertices with core numbers strictly less than k (line 6). The pruning effectively limits the search space for the optimization (line 10) by reducing the number of vertices, which improves the runtime.

If the clique recovered by the greedy algorithm is the maximum clique, then pruning the graph removes all the vertices and therefore we can terminate early (line 7). This early termination particularly occurs when the graph is sparse, which leads to a significant speed-up over running the optimization-based algorithm on the original graph.

When the optimization-based algorithm is required, the initial guess used for the optimization can be chosen strategically to improve the chance of finding the maximum clique. This can be done by choosing an initial guess vector that is a binary complement of the clique found from the greedy approach (line 9), and thus help the algorithm to converge to a different clique solution. Lastly, the best solution is selected as the largest clique (line 11).

Computational Complexity: The worst-cast complexity of Algorithm 3 is $O(n^4)$. This is because it sequentially combines Algorithms 1 and 2, and thus inherits the highest worst-case complexity of its components (all other steps in Algorithm 3 have lower complexity). This basic worst-case complexity analysis is independent of any input graph structure. The numerical results in the experimental section provide a good indication of how the complexity translates to runtimes for various graph sizes in the practical applications.

V. Experimental Evaluations

We evaluate the performance of CLIPPER+ (Algorithm 3) in terms of the maximum clique estimation accuracy and runtime. In addition, we provide ablation studies of CLIPPER+ by reporting the results of its standalone greedy (Algorithm 1) and optimization (Algorithm 2) components. We show that CLIPPER+ achieves a performance superior to these standalone components through combining them.

Algorithms: We test both classical and state-of-the-art maximum clique estimation algorithms. Classical works include Pelillo [35] and Ding et al. [36] based on relaxations of the Motzkin-Straus formulation [37]. State-of-the-art include our implementation of the greedy parallel maximum clique (PMC) algorithm [2] in Algorithm 1 (which is theoretically equivalent to ROBIN [19]), the algorithm by Belachew and Gillis [38], and our prior CLIPPER algorithm [1].

Benchmarks: Our evaluations examine finding the maximum clique on general graphs, as well as graphs that result from the graph formulation of synthetic/real-world point cloud registration problems. While general graphs can have any structure, the registration graphs have certain patterns (e.g., sparsity) that significantly affect the results. Fortunately, as we will see, finding the maximum clique is empirically easier on graphs that result from registration.

Platform and implementations: All benchmarks are run on a machine with an Intel Core i9 Processor and 32 GB RAM. The algorithms by Pelillo, Ding, and Belachew are implemented in Matlab by their original authors. Our prior work CLIPPER is also implemented in Matlab. The "Greedy" (Algorithm 1), "Optim" (Algorithm 2), and CLIPPER+ algorithms are implemented in C++, which interface to Matlab via binary MEX binders for benchmarking.

A. Maximum Clique Benchmark—DIMACS

Dataset: To evaluate the algorithms on general graphs, we use the DIMACS benchmark [40], which was introduced in 1996 and has been used widely since then to benchmark the maximum clique algorithms. The DIMACS dataset consists of graphs for which finding the maximum clique is challenging. In the interest of space, we present the result on a subset of smaller graphs in this dataset, shown in Table I. While DIMACS graphs are relatively small (100–4000 vertices), they are dense and contain more edges compared to graphs resulting from registration. Despite the dataset being around for more than 25 years, the maximum clique of some graphs is still unknown, demonstrating the difficulty of the problem. For instance, on the `C250.9` graph, the (multi-threaded) PMC exact algorithm [2], used in this work for ground truth generation and in TEASER [7] for certifiable registration, took 28 minutes on our machine to find the maximum clique.

Evaluation: Table I compares the accuracy and runtime of algorithms. The **graph sparsity** is defined as $s \overset{\text{def}}{=} 1 - \frac{|E|}{|E_{\max}|} \in [0,1]$, where $|E|$ is the number of graph edges, and $|E_{\max}|$ is the maximum possible number of edges. The small values of s show that DIMACS graphs are generally dense. The **accuracy ratio** is measured by $r \overset{\text{def}}{=} \hat{\omega}/\omega_{\text{gt}}$, where $\hat{\omega}$ is the clique size found by the algorithm, and ω_{gt} is the ground truth maximum clique size. The ratio of 1 indicates that the maximum clique is found, hence, the closer r is to 1, the more accurate an algorithm is. *CLIPPER+ outperforms all algorithms in the overall accuracy*. Unsurprisingly, the greedy algorithm has the best overall runtime due to its low computational complexity; however, it has a lower accuracy. By combining the greedy and optimization algorithms, CLIPPER+ obtains high accuracy and better overall runtime compared to the optimization-only approach (e.g., a ~10x reduction of runtime from 29.4 to 3.6 milliseconds on `brock200_2`).

B. Registration Benchmark—Stanford Bunny

Using the Stanford Bunny point cloud [44], shown in Fig. 2, we evaluate CLIPPER+ as an algorithm for global point cloud registration in various outlier regimes.

Dataset: The (downsampled) Bunny point cloud consists of 1000 points that fit in a cube of size $0.2\,\text{m}$. To generate a second point cloud, we add uniform noise in the range $[-\epsilon/2, \epsilon/2]$ to all points, where ϵ is set as the mean distance of all points to their nearest neighbors in the Bunny point cloud. Additionally,

TABLE I
COMPARISONS OF THE MAXIMUM CLIQUE ESTIMATION ACCURACY AND RUNTIME FOR CLIPPER+ ON DIMACS BENCHMARK [40] (BOLD IS BEST)

Benchmark				Pelillo		Ding		Belachew		CLIPPER		Greedy		Optim		**CLIPPER+**	
Graph	n	s	ω_{gt}	$t\downarrow$	$r\uparrow$	$t\downarrow$	$r\uparrow$	$t\downarrow$	$r\uparrow$	$t\downarrow$	$r\uparrow$	$t\downarrow$	$r\uparrow$	$t\downarrow$	$r\uparrow$	$t\downarrow$	$r\uparrow$
`C125.9`	125	0.1	34	4.1	0.97	4.5	0.56	10.9	**1**	9.1	**1**	**0.7**	0.85	0.8	**1**	1.6	**1**
`C250.9`	250	0.1	44	4.7	0.89	3.6	0.34	7.8	**0.95**	6.8	**0.95**	**2.7**	0.8	6.4	**0.95**	8.7	**0.95**
`brock200_2`	200	0.5	12	2.2	0.67	51.8	0.75	4.3	**0.83**	4	**0.83**	**1.4**	**0.83**	2.6	**0.83**	3.8	**0.83**
`brock200_4`	200	0.34	17	**1.7**	0.76	9.4	0.88	4.2	0.88	8.9	0.65	**1.7**	0.82	29.4	0.82	3.6	**0.94**
`gen200_p0.9_44`	200	0.1	44	6.1	0.75	3.9	0.2	10.4	0.84	10.9	**0.89**	**2.2**	0.73	4.7	**0.89**	6.4	**0.89**
`gen200_p0.9_55`	200	0.1	55	4.6	0.69	2.7	0.38	4.9	0.69	6.8	**1**	**2.0**	0.64	1.6	**1**	3.7	**1**
`keller4`	171	0.35	11	2.3	0.64	4.0	0.64	2.8	0.73	2.8	0.64	**1.0**	**0.82**	35.4	0.64	5.8	**0.82**
`p_hat300-1`	300	0.76	8	**1.3**	0.75	33.9	0.88	6.5	**1**	4.3	**1**	2.1	0.88	4.9	**1**	10.7	**1**
`p_hat300-2`	300	0.51	25	**2.8**	0.96	11.3	0.92	10.9	**1**	15.9	0.96	3.7	0.84	6.5	**1**	10.1	**1**

n: Number of graph vertices. s: Graph sparsity. ω_{gt}: Ground truth maximum clique size. t: Runtime (milliseconds); the lower, the better. r: Maximum clique accuracy ratio ($\hat{\omega}/\omega_{\mathrm{gt}}$); the closer to 1, the better.

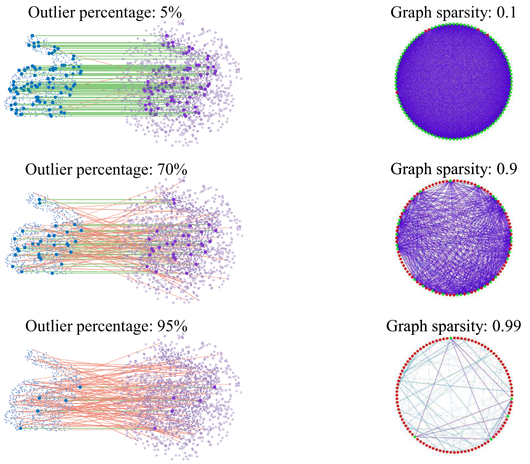

Fig. 2. Effect of outliers on the consistency graph. (Left) Putative associations (red: outliers; green: inliers). (Right) Resulting consistency graph. The higher the outlier percentage, the sparser the graph.

1000 outlier points randomly drawn from a sphere of radius 1 m, centered at the bunny point cloud, are added to simulate clutter. From the set of all possible associations between the points in the two point clouds, 200 associations are randomly selected from the set of inlier and outlier associations (we keep this number small to be able to find the ground truth maximum clique). We consider different outlier ratios (ranging from 0% to 98% in 2% increments), to test scenarios with various data association accuracy, as shown in Fig. 2. We use the noise ϵ as the threshold to generate the consistency graph (according to Section III). The ground truth maximum clique is found by running the exact PMC algorithm in [2]. Evaluations of maximum clique estimation accuracy and runtime are performed across 50 Monte Carlo runs/graphs for each increment of the outlier percentage.

Evaluation: Fig. 3 compares the accuracy ratio r of algorithms across all runs and all outlier ratios. *CLIPPER+ clearly demonstrates the highest accuracy* because the distribution of r is closest to 1. Interestingly, while the greedy and optimization algorithms returned low-accuracy solutions in some instances, by combining these algorithms CLIPPER+ obtains an accuracy beyond these standalone components.

Fig. 4 shows the mean of the accuracy ratio r (averaged across 50 Monte Carlo runs at each outlier percentage increment) versus the outlier percentage. The accuracy of the greedy approach is low in low-outlier regimes (which have dense consistency graphs, see Fig. 2). As the outlier percentage increases and the graph becomes sparser, the accuracy of the greedy method improves. Both CLIPPER+ and optimization algorithms retain a high accuracy ratio across all outlier percentages.

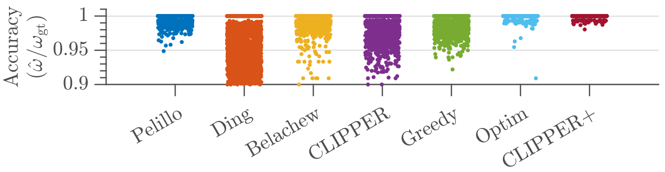

Fig. 3. Maximum clique estimation accuracy ratio on the Stanford Bunny benchmark (the closer to 1, the better). Each point corresponds to the accuracy ratio from a single trial. CLIPPER+ outperforms all algorithms.

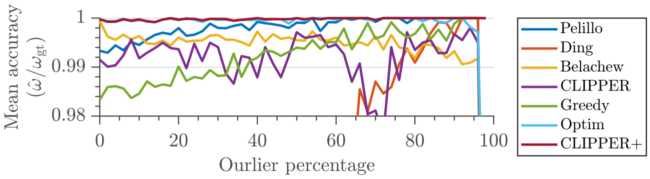

Fig. 4. Mean maximum clique estimation accuracy across different outlier percentages on the Stanford Bunny benchmark (the closer to 1, the better).

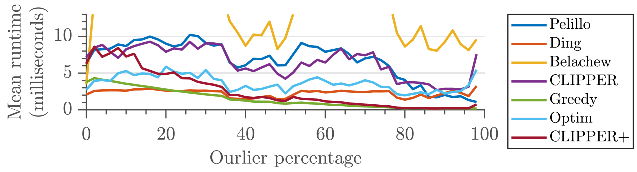

Fig. 5. Mean runtime across different outlier percentages on the Stanford Bunny benchmark (the lower, the better).

The mean runtime of the algorithms is shown in Fig. 5. The runtime of the optimization method remains roughly the same, while the runtime of the greedy method improves as the outlier ratio grows and the graph becomes sparser. The runtime of CLIPPER+ in low outlier regimes is roughly equal to the compound runtimes of its greedy and optimization components because in such regimes pruning the graph based on core numbers does not remove a significant number of vertices (if any). However, as the outlier ratio increases and the graph becomes sparser, pruning removes more vertices and its speed-up effect becomes more

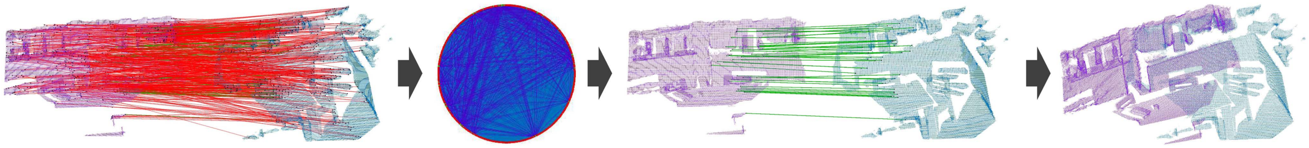

Fig. 6. Registration instance between two scans of the 7-Scenes dataset [42]. From left to right: 688 putative FPFH associations having 93.6% outliers (red lines); corresponding consistency graph used as input to CLIPPER+ (the maximum clique is highlighted); inlier associations corresponding to the maximum clique solution found by CLIPPER+; aligned point clouds using the rotation/translation computed from the Arun's method [43] using inliers.

TABLE II
COMPARISONS OF MAXIMUM CLIQUE ESTIMATION ACCURACY AND RUNTIME OF CLIPPER+ ON REAL-WORLD POINT CLOUD REGISTRATION DATASETS (BOLD IS BEST)

Benchmark						Pelillo		Ding		Belachew		CLIPPER		Greedy		Optim		**CLIPPER+**	
Name	Sequence	N	$\bar{op}$	$\bar{n}$	$\bar{s}$	$\bar{t}\downarrow$	$\bar{r}\uparrow$	$\bar{t}\downarrow$	$\bar{r}\uparrow$	$\bar{t}\downarrow$	$\bar{r}\uparrow$	$\bar{t}\downarrow$	$\bar{r}\uparrow$	$\bar{t}\downarrow$	$\bar{r}\uparrow$	$\bar{t}\downarrow$	$\bar{r}\uparrow$	$\bar{t}\downarrow$	$\bar{r}\uparrow$
7-Scenes	`redkitchen`	244	92.8%	680	0.88	118.5	0.96	28.3	0.9	73.7	**0.99**	62.7	0.97	**10.1**	0.89	119.9	**0.99**	53.1	**0.99**
Sun3D	`home_at_scan1`	81	88%	670	0.83	133.1	0.97	17.5	0.87	64.0	0.98	58.2	0.98	**10.3**	0.89	76.0	**0.99**	46.4	**0.99**
	`hotel_uc_scan3`	117	93.5%	843	0.89	215.3	0.96	28.9	0.89	116.5	**0.99**	109.3	0.98	**12.3**	0.89	159.4	**0.99**	56.1	**0.99**
	`mit_76_1studyroom2`	96	94.2%	936	0.9	192.9	0.97	32.2	0.87	120.6	**0.99**	102.8	0.97	**14**	0.88	180.8	**0.99**	83.9	**0.99**
ETH	`gazebo_summer`	116	98.3%	4320	0.97	4255.2	0.95	1275.6	0.8	8467.8	0.98	7978.5	0.97	**351.5**	0.81	7595.7	**0.99**	3769	**0.99**
	`wood_autumn`	87	99.4%	5703	0.98	4866.2	0.94	2990.9	0.89	14464	0.98	12774	0.97	**564.5**	0.75	14758	**0.99**	14594	**0.99**

N: Total number of overlapping point cloud scans on which registration is performed. $\bar{op}$: Mean of outlier percentages in putative associations. $\bar{n}$: Mean graph size. $\bar{s}$: Mean graph sparsity. $\bar{t}$: Mean runtime (milliseconds); the lower, the better. $\bar{r}$: Mean maximum-clique accuracy ratio ($\hat{\omega}/\omega_{\mathrm{gt}}$); the closer to 1, the better.

apparent. This can be seen around the 20% outlier ratio, where CLIPPER+ becomes faster than the optimization method, and its speed improves further as the outlier percentage increases.

C. Registration Benchmark—Real-World Point Clouds

Datasets: We use sequences in the real-world 7-Scenes [42], Sun3D [45], and ETH [46] datasets (similar to 3DMatch [15] and 3DSmoothNet [47] evaluations). Sun3D and 7-Scenes are dense indoor RGB-D point clouds, while ETH is an outdoor LiDAR point cloud. In each sequence, we consider pairs of point clouds (or scans) that have an overlap. To increase registration speed, we downsample the point clouds by discretizing the 3D space into cubes of size $\epsilon = 0.05\,\mathrm{m}$ for 7-Scenes and Sun3D datasets, and $\epsilon = 0.1\,\mathrm{m}$ for the ETH dataset, and using the mean of the point coordinates in each cube as a single-point representative. For each downsampled point, FPFH descriptor vectors [14] are computed and associated bilaterally based on their l_2 norm distance using the k-nearest neighbors algorithm. To generate the ground truth for our evaluations, we use the exact maximum clique algorithm of [2] to find the largest set of geometrically consistent associations in these putative FPFH associations. We store results if this maximum clique solution correctly registers the point clouds according to the ground truth provided by the datasets—the maximum clique/likelihood solution may register points wrongly due to practical limitations such as repetitive patterns (perceptual aliasing), insignificant overlap between the point clouds, and lack of any inlier associations caused by downsampling and FPFH inaccuracies. For evaluations, the consistency graph is generated according to Section III from putative FPFH associations, using the downsampling ϵ as the consistency threshold. An evaluation instance is shown in Fig. 6.

Evaluation: Table II presents the evaluation results for all datasets and algorithms. *CLIPPER+ outperforms all algorithms in accuracy on all datasets/sequences.* The greedy algorithm has the smallest runtime at the expense of the lowest overall accuracy. The standalone optimization algorithm has similar accuracy to CLIPPER+. However, except on the last sequence, it is around 2x slower. This demonstrates the advantage of CLIPPER+ over its standalone greedy and optimization components.

Lastly, we point out the high outlier ratios of FPFH associations (e.g., on average, 99.4% of associations in the `wood_autumn` sequence are outliers). Due to the maximum clique solution correctly registering the point clouds in our benchmark, the accuracy ratio r shows the point cloud registration success rate. Thus, *CLIPPER+ correctly registers the point clouds in 99% of the trials despite extreme outlier percentages.* This is what distinguishes CLIPPER+ from existing robust registration frameworks (such as RANSAC [16]), that can fail in these high-outlier regimes.

VI. Conclusion and Future Directions

We presented CLIPPER+, a maximal-clique-finding algorithm for unweighted graphs that enables robust global registration in robotics and computer vision applications. Future work includes investigating alternative optimization methods such as second-order or quasi-Newton methods, and an extension to the weighted graphs based on our previous work [1] (designed for weighted graphs) and an extension of core numbers to weighted settings (studied in [48]). We also plan to integrate the algorithm in point cloud registration pipelines [7] as the outlier rejection module to improve the runtime and outlier rejection capacity.

References

[1] P. C. Lusk, K. Fathian, and J. P. How, “CLIPPER: A graph-theoretic framework for robust data association,” in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 13828–13834.

[2] R. A. Rossi, D. F. Gleich, and A. H. Gebremedhin, “Parallel maximum clique algorithms with applications to network analysis,” *SIAM J. Sci. Comput.*, vol. 37, no. 5, pp. C589–C616, 2015.

[3] P. Y. Lajoie, B. Ramtoula, Y. Chang, L. Carlone, and G. Beltrame, "DOOR-SLAM: Distributed, online, and outlier resilient slam for robotic teams," *IEEE Robot. Automat. Lett.*, vol. 5, no. 2, pp. 1656–1663, Apr. 2020.

[4] J. G. Mangelson, D. Dominic, R. M. Eustice, and R. Vasudevan, "Pairwise consistent measurement set maximization for robust multi-robot map merging," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 2916–2923.

[5] J. Ankenbauer, K. Fathian, and J. P. How, "View-invariant localization using semantic objects in changing environments," 2022, *arXiv:2209.14426*.

[6] 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.

[7] H. Yang, J. Shi, and L. Carlone, "TEASER: Fast and certifiable point cloud registration," *IEEE Trans. Robot.*, vol. 37, no. 2, pp. 314–333, Apr. 2021.

[8] Y. Li, L. Gu, and T. Kanade, "Robustly aligning a shape model and its application to car alignment of unknown pose," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 9, pp. 1860–1876, Sep. 2011.

[9] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep learning on point sets for 3 d classification and segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 652–660.

[10] K. Fathian, K. Khosoussi, Y. Tian, P. Lusk, and J. P. How, "CLEAR: A consistent lifting, embedding, and alignment rectification algorithm for multiview data association," *IEEE Trans. Robot.*, vol. 36, no. 6, pp. 1686–1703, Dec. 2020.

[11] P. C. Lusk, K. Fathian, and J. P. How, "MIXER: Multiattribute, multiway fusion of uncertain pairwise affinities," *IEEE Robot. Automat. Lett.*, vol. 8, no. 5, pp. 2462–2469, May 2023.

[12] L. Rakai, H. Song, S. Sun, W. Zhang, and Y. Yang, "Data association in multiple object tracking: A survey of recent techniques," *Expert Syst. Appl.*, vol. 192, 2022, Art. no. 116300.

[13] P. J. Besl and N. D. McKay, "Method for registration of 3-D shapes," *Proc. SPIE*, vol. 1611, pp. 586–606, 1992.

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

[15] 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.

[16] 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.

[17] H. Yang, P. Antonante, V. Tzoumas, and L. Carlone, "Graduated non-convexity for robust spatial perception: From non-minimal solvers to global outlier rejection," *IEEE Robot. Automat. Lett.*, vol. 5, no. 2, pp. 1127–1134, Apr. 2020.

[18] P. C. Lusk, D. Parikh, and J. P. How, "GraffMatch: Global matching of 3D lines and planes for wide baseline LiDAR registration," *IEEE Robot. Automat. Lett.*, vol. 8, no. 2, pp. 632–639, Feb. 2023.

[19] J. Shi, H. Yang, and L. Carlone, "ROBIN: A graph-theoretic approach to reject outliers in robust estimation using invariants," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 13820–13827.

[20] G. Izatt, H. Dai, and R. Tedrake, "Globally optimal object pose estimation in point clouds with mixed-integer programming," in *Proc. 18th Int. Symp. Robot. Res.*, Springer, 2020, pp. 695–710.

[21] 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. 38, no. 11, pp. 2241–2254, Nov. 2016.

[22] T. J. Chin and D. Suter, "The maximum consensus problem: Recent algorithmic advances," *Synth. Lectures Comput. Vis.*, vol. 7, no. 2, pp. 1–194, 2017.

[23] D. Barath and J. Matas, "Graph-cut RANSAC," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 6733–6741.

[24] 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.

[25] A. Blake and A. Zisserman, *Visual Reconstruction*. Cambridge, MA, USA: MIT Press, 1987.

[26] P. Agarwal, G. D. Tipaldi, L. Spinello, C. Stachniss, and W. Burgard, "Robust map optimization using dynamic covariance scaling," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2013, pp. 62–69.

[27] J. L. Schonberger and J.-M. Frahm, "Structure-from-motion revisited," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 4104–4113.

[28] R. Raguram, J.-M. Frahm, and M. Pollefeys, "A comparative analysis of ransac techniques leading to adaptive real-time random sample consensus," in *Proc. Proc. Eur. Conf. Comput. Vis.*, Springer, 2008, pp. 500–513.

[29] T. Bailey, E. M. Nebot, J. Rosenblatt, and H. F. Durrant-Whyte, "Data association for mobile robot navigation: A graph theoretic approach," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2000, vol. 3, pp. 2512–2517.

[30] M. Leordeanu and M. Hebert, "A spectral technique for correspondence problems using pairwise constraints," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2005, vol. 2, pp. 1482–1489.

[31] O. Enqvist, K. Josephson, and F. Kahl, "Optimal correspondences from pairwise constraints," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2009, pp. 1295–1302.

[32] A. P. Bustos, T.-J. Chin, F. Neumann, T. Friedrich, and M. Katzmann, "A practical maximum clique algorithm for matching with pairwise constraints," 2019, *arXiv:1902.01534*.

[33] X. Zhang, J. Yang, S. Zhang, and Y. Zhang, "3D registration with maximal cliques," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 17745–17754.

[34] Q. Wu and J.-K. Hao, "A review on algorithms for maximum clique problems," *Eur. J. Oper. Res.*, vol. 242, no. 3, pp. 693–709, 2015.

[35] M. Pelillo, "Relaxation labeling networks for the maximum clique problem," *J. Artif. Neural Netw.*, vol. 2, no. 4, pp. 313–328, 1995.

[36] C. Ding, T. Li, and M. I. Jordan, "Nonnegative matrix factorization for combinatorial optimization: Spectral clustering, graph matching, and clique finding," in *Proc. IEEE Int. Conf. Data Mining*, 2008, pp. 183–192.

[37] T. S. Motzkin and E. G. Straus, "Maxima for graphs and a new proof of a theorem of Turan," *Can. J. Math.*, vol. 17, pp. 533–540, 1965.

[38] M. T. Belachew and N. Gillis, "Solving the maximum clique problem with symmetric rank-one non-negative matrix approximation," *J. Optim. Theory Appl.*, vol. 173, no. 1, pp. 279–296, 2017.

[39] V. Batagelj and M. Zaversnik, "An O(m) algorithm for cores decomposition of networks," 2003, *arXiv:cs/0310049*.

[40] D. S. Johnson and M. A. Trick, *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge, Oct. 11-13, 1993*, vol. 26. Providence, RI, USA: Amer. Math. Soc., 1996.

[41] D. P. Bertsekas, "Nonlinear programming," *J. Oper. Res. Soc.*, vol. 48, no. 3, pp. 334–334, 1997.

[42] J. Shotton, B. Glocker, C. Zach, S. Izadi, A. Criminisi, and A. Fitzgibbon, "Scene coordinate regression forests for camera relocalization in RGB-D images," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2013, pp. 2930–2937.

[43] 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.

[44] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," in *Proc. 23rd Annu. Conf. Comput. Graph. Interactive Techn.*, 1996, pp. 303–312.

[45] J. Xiao, A. Owens, and A. Torralba, "SUN3D: A database of big spaces reconstructed using SFM and object labels," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2013, pp. 1625–1632.

[46] F. Pomerleau, M. Liu, F. Colas, and R. Siegwart, "Challenging data sets for point cloud registration algorithms," *Int. J. Robot. Res.*, vol. 31, no. 14, pp. 1705–1711, 2012.

[47] 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. 5545–5554.

[48] B. Liu and F. Zhang, "Incremental algorithms of the core maintenance problem on edge-weighted graphs," *IEEE Access*, vol. 8, pp. 63872–63884, 2020.