Scalable 3D Registration via Truncated Entry-wise Absolute Residuals

Tianyu Huang[1*] Liangzu Peng[2*] René Vidal[2] Yun-Hui Liu[1]
[1]The Chinese University of Hong Kong [2]University of Pennsylvania
[1]tyhuang, yhliu@mae.cuhk.edu.hk [2]lpenn, vidalr@seas.upenn.edu

Abstract

Given an input set of 3D point pairs, the goal of outlier-robust 3D registration is to compute some rotation and translation that align as many point pairs as possible. This is an important problem in computer vision, for which many highly accurate approaches have been recently proposed. Despite their impressive performance, these approaches lack scalability, often overflowing the 16GB of memory of a standard laptop to handle roughly $30,000$ point pairs. In this paper, we propose a 3D registration approach that can process more than ten million (10^7) point pairs with over 99% random outliers. Moreover, our method is efficient, entails low memory costs, and maintains high accuracy at the same time. We call our method `TEAR` [1], as it involves minimizing an outlier-robust loss that computes Truncated Entry-wise Absolute Residuals. To minimize this loss, we decompose the original 6-dimensional problem into two subproblems of dimensions 3 and 2, respectively, solved in succession to global optimality via a customized branch-and-bound method. While branch-and-bound is often slow and unscalable, this does not apply to `TEAR` as we propose novel bounding functions that are tight and computationally efficient. Experiments on various datasets are conducted to validate the scalability and efficiency of our method.

1. Introduction

The 3*D registration* problem aims to find a rotation and translation that best align an input set of 3D point pairs. Ideally, the alignment errors for all point pairs are small, and we call them *inliers*. In practice, the inliers are contaminated by other point pairs, called *outliers*, that induce significant alignment errors. Given a set of inlier and outlier point pairs, *outlier-robust 3D registration* aims to align the 3D inlier point pairs via some rotation and translation.

In this paper, we tackle the 3D registration problem with extremely many outliers, via the proposed method that we call *Truncated Entry-wise Absolute Residuals* (`TEAR`). Numerically, `TEAR` can handle more than 10^7 point pairs with 99.8% random outliers, a setting in which no existing methods have been shown to succeed: They are either unscalable, inefficient, or inaccurate. In Sec. 1.1, we review prior 3D registration methods. In Sec. 1.2, we further highlight the `TEAR` approach and overview our contributions.

1.1. Prior Art

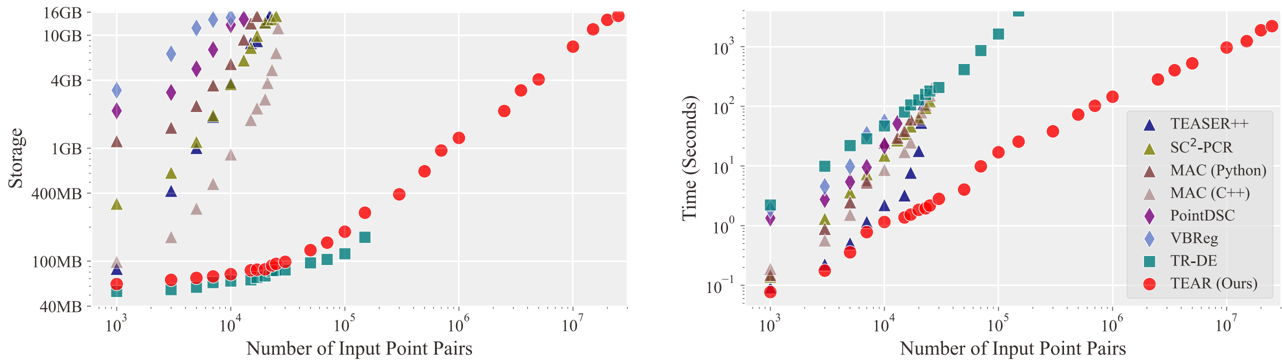
In this section, we briefly review several families of 3D robust registration methods that are relevant to our work.

First, observe that the outlier-robust 3D registration problem can be decoupled into two subproblems: (1) if the outliers are known, one can easily estimate a rotation and translation by SVD [2, 25, 26], and (2) if the true rotation and translation are given, one can easily remove the outliers. These observations turn immediately into an efficient alternating minimization method, which lies at the heart of classical approaches including *iterative closest point* [7, 55], *iteratively reweighted least-squares* [1, 6, 18, 43, 48, 50, 52], and *graduated non-convexity* [8, 29, 58, 67, 73, 78]. Such an alternating minimization scheme is shown to be convergent in [1, 52] under very general conditions, but it might not converge to a desired solution if the outlier ratio is high.

The RANSAC method [21] proceeds as follows: randomly sample a minimum number of point pairs (typically 3 pairs), compute a rotation and translation that aligns them, measure the alignment error for all pairs, repeat these steps until a termination criterion is reached, and output the rotation and translation that give the smallest alignment error. RANSAC could terminate with a correct estimation as soon as all pairs sampled in an iteration are inliers, but the probability of achieving so diminishes as the outlier ratio grows, suggesting that RANSAC, or its variants [4, 5, 54, 62], can be inefficient in the presence of extremely many outliers.

Alternatively, one could formulate some non-convex objective (*e.g.*, *consensus maximization*) for outlier-robust registration, and solve it via the *branch-and-bound* algorithm [57]. Branch-and-bound guarantees global optimality by design, so it has served well in recent years as a validation tool [11–13, 33–35, 37, 39, 40, 44, 47, 63, 70, 75].

[*] Equal contribution
[1] https://github.com/tyhuang98/TEAR-release



(a) (b)

Figure 1. Comparisons of TEAR (ours) to prior methods on random, synthetic, noisy data (20 trials). The outlier ratio is set to 95% and all presented methods find accurate solutions. Fig. 1a: TEAR is 10^3 times more scalable than methods based on consistency graphs including TEASER++ [68], SC^2-PCR [14], MAC [76] and than deep learning methods including PointDSC [3] and VBReg [28]. Fig. 1b: TEAR is 100 times faster than TR-DE [13], a recent branch-and-bound method.

Its efficiency hinges on two critical aspects: the dimension of the search space and bounds on the objective. Care is needed to reduce the search space dimension or tighten the bounds, otherwise branch-and-bound could quickly be intractable as the problem scales up. The recent well-designed branch-and-bound method of [13], called TR-DE, takes more than 10^3 seconds to handle 10^5 point pairs (Fig. 1).

Instead of tackling the non-convex robust registration problem directly, one could consider relaxing it into a convex semidefinite program, which is typically solvable in polynomial time [9, 27, 49, 66, 69]. While semidefinite relaxations might recover the solution of the original non-convex objective, this recovery property could come at the cost of working with quadratically many optimization variables and constraints [66], and thus at the expense of vast computation: State-of-the-art solvers need more than 7 hours to solve such semidefinite programs and align 1,000 point pairs, even if the translation is given [69, Table 3].

The *outlier removal* method [10, 46] has a for loop: For each point pair, assume it is an inlier, then one can reason which point pair conflicts with this assumption. Doing so allows removing some point pairs and—this is a virtue—the point pairs to be removed are guaranteed to be outliers. While outlier removal often serves as a preprocessing step that facilitates subsequent alignment, it does so by charging a large amount of time (say, exponential in the variable dimension [16, 46] or at least quadratic in the number of points); *e.g.*, on our laptop, the method of [46] takes more than 2 hours for 10^5 point pairs with 95% outliers.

The next several methods we review rely on the so-called *consistency graph*, a graph where a vertex denotes a point pair and an edge indicates two *consistent* point pairs that can both be inliers. The consistency graph finds its early use in [20, 32] and is the cornerstone of many recent methods for outlier-robust 3D registration [15, 41, 59, 65, 68, 76].

For example, using the consistency graph, the TEASER++ method computes a *maximum clique*[2] often containing most inliers and few outliers [68]; the SC^2-PCR method generalizes the consistency graph into a second-order version more discriminant between inliers and outliers [14, 15]; the MAC method generalizes the maximum clique formulation into computing *maximal cliques*[2] [76]. Even though these methods have established state-of-the-art performance, they have also brought an *elephant* into the room: Computing a consistency graph uses memory quadratic in the number of point pairs, *e.g.*, doing so for 30,000 point pairs would occupy the total 16GB memory of a standard laptop, which limits the applicability of all these methods to larger-scale robust 3D registration problems (Fig. 1).

There have recently been many deep learning approaches developed to extract and match features of the input point clouds [17, 24, 30, 45, 53, 64, 71, 72], the most relevant to ours are methods that perform robust 3D registration, such as PointDSC [3] and VBReg [28]. PointDSC builds upon the consistency graph, and needs extra storage for a large network and (temporarily, during forward passes) the high-dimensional feature of each 3D point. VGReg builds upon PointDSC and uses a recurrent network that admits a variational interpretation, but it inherits the drawback of PointDSC of being not scalable (Fig. 1).

Summary. Since scalability has been *a fly in the ointment* compromising the recent success of 3D registration methods, why not simply downsample huge-scale point clouds and perform registration from there? The answer to this sticking point is that downsampling ignores some input information that one could otherwise leverage, so it ultimately impinges upon performance; for numerical evidence, see,

[2]A maximum clique of a graph is a complete subgraph containing the largest possible number of vertices, whereas a maximal clique is a complete subgraph not contained in any (other) maximum clique.

e.g., [28, Table 2], [15, Fig. 9], and Fig. 7. This thus renders downsampling into a stopgap that eventually necessitates developing scalable and efficient registration methods.

1.2. Our Contribution: TEAR

Which methodology, reviewed above, can be utilized to design a scalable approach for registrating ten million point pairs with extremely many outliers? Alternating minimization and RANSAC are known to be brittle at high outlier ratios. Semidefinite programs are costly to solve. Outlier removal typically needs quadratic time, constructing consistency graphs further consumes quadratic storage, and deep learning demands even more memory.

To design a scalable method, we advocate the branch-and-bound method. This might be surprising (if not doubtful): Conventional wisdom has it that branch-and-bound is slow and induces exponential running times! Contrary to the common wisdom, Fig. 1 indicates the running time of our proposal (`TEAR`) grows almost linearly. We achieve this by revising the problem-solving pipeline, from the problem formulation to mathematical derivations (of upper and lower bounds), and furthermore to implementation details. More explicitly, we make the following contributions:

- (*Problem Formulation*, Secs. 2.1 and 2.2) We formulate the 3D registration problem using the robust loss that we call TEAR, a shorthand for *Truncated Entry-wise Absolute Residuals* (Sec. 2.1). TEAR is similar in spirit to commonly seen robust losses (*e.g.*, consensus maximization, truncated least-squares), but it has subtle differences that enable faster branch-and-bound algorithms to be derived. Moreover, in Sec. 2.2 we decompose TEAR into two subproblems of dimensions 3 and 2 respectively, which further facilitates developing branch-and-bound implementations. In fact, at a high level, our approach is very simple: We solve the two subproblems, one after another, by a basic branch-and-bound template.
- (*Upper and Lower Bounds*, Secs. 2.3 and 2.4) The non-trivial part of our approach lies in deriving tight lower and upper bounds for the branch-and-bound method, and our key idea for achieving so is as follows (Sec. 2.3). To solve the 3-dimensional subproblem, for example, our implementation searches a 2-dimensional space (rather than 3). In this implementation, we derive upper and lower bounds that can be computed via solving a specific 1-dimensional problem in $O(N \log N)$ time, where N is the total number of point pairs. We follow a similar route to solve the other 2-dimensional subproblem to global optimality, and for simplicity, we call the final algorithm `TEAR`. Via numerical comparisons (Sec. 2.4), we will show that using TEAR as the robust loss ensures the bounds are tighter than using the commonly used consensus maximization loss, and it also ensures the bounds are more efficient to compute than using the truncated least-squares loss.
- (*Experiments*, Secs. 3 and 4) In Sec. 3 we perform standard experiments on synthetic and real data, showing that `TEAR` reaches state-of-the-art accuracy while being more efficient in most cases. In Sec. 4 we perform experiments on large-scale point clouds, presenting `TEAR` as a unique method that can handle ten million (10^7) point pairs in the presence of extremely many random outliers (99.8%).

2. The Design of TEAR

This section introduces the design of `TEAR`. In Sec. 2.1, we revisit commonly used formulations for 3D registration and their drawbacks, thus motivating our proposal of a novel formulation called *Truncated Entry-wise Absolute Residuals* (TEAR). In Sec. 2.2, we decompose TEAR into easier subproblems. In Sec. 2.3, we describe how to solve the subproblems using branch-and-bound. In Sec. 2.4, we provide numerical validation that TEAR overcomes the drawbacks of other formulations and can be solved more efficiently.

2.1. Problem Formulation: TEAR

Rethink Existing Formulations. Recall that our goal is to find some 3D rotation $\boldsymbol{R}^*$ and translation $\boldsymbol{t}^*$ that best aligns an input set of 3D point pairs $\{(\boldsymbol{y}_i, \boldsymbol{x}_i)\}_{i=1}^N$ containing a large fraction of (random) outliers. The first step of designing `TEAR` is to choose, if not to propose, an outlier-robust problem formulation that admits scalable algorithms. To this end, we recall two highly robust losses often used in geometric vision, namely *Consensus Maximization* (CM) and *Truncated Least-Squares* (TLS):

$$\max_{(\boldsymbol{R},\boldsymbol{t})\in \mathrm{SE}(3)} \sum_{i=1}^N \mathbf{1}\left(\|\boldsymbol{y}_i - \boldsymbol{R}\boldsymbol{x}_i - \boldsymbol{t}\|_2 \le \xi_i\right), \tag{CM}$$

$$\min_{(\boldsymbol{R},\boldsymbol{t})\in \mathrm{SE}(3)} \sum_{i=1}^N \min\left\{\|\boldsymbol{y}_i - \boldsymbol{R}\boldsymbol{x}_i - \boldsymbol{t}\|_2^2,\ \xi_i^2\right\}. \tag{TLS}$$

Here, $\mathbf{1}(\cdot)$ denotes the indicator function, and $\xi_i \ge 0$ is a threshold hyper-parameter, such that, if the residual $\|\boldsymbol{y}_i - \boldsymbol{R}\boldsymbol{x}_i - \boldsymbol{t}\|_2$ is larger than ξ_i, then $(\boldsymbol{y}_i, \boldsymbol{x}_i)$ is regarded as an outlier (with respect to $\boldsymbol{R}$, $\boldsymbol{t}$). Therefore, CM aims to minimize the number of outliers, whereas TLS furthermore minimizes the residuals of inliers using least-squares.

In the context of branch-and-bound, CM has been popular, as its upper and lower bounds are relatively easy to derive. However, using branch-and-bound for CM has a subtle yet crucial drawback. Indeed, note that in many cases two different rotations and translations could correspond to the same number of outliers (as determined by ξ_i), but the corresponding residuals are hardly the same. Note then that CM only counts the number of outliers with respect to the current rotation and translation, and it never calculates the sum of residuals. These imply the upper and lower bounds of CM are usually loose, which would jeopardize, consequentially, the efficiency of branch-and-bound.

Is TLS suitable for branch-and-bound? Here are some concerns, though. First, to our knowledge, no prior work applied branch-and-bound to TLS: The technical challenge is deriving the corresponding upper and lower bounds; the conceptual challenge is the impression that branch-and-bound would be slow anyway. Our second, major, concern has a deeper cause, which we will illustrate later in Sec. 2.4.

TEAR. In light of the above concerns, we propose *Truncated Entry-wise Absolute Residuals* (TEAR):

$$\min_{(\boldsymbol{R},\boldsymbol{t})\in\mathrm{SE}(3)} \sum_{i=1}^{N} \min\left\{\|\boldsymbol{y}_i - \boldsymbol{R}\boldsymbol{x}_i - \boldsymbol{t}\|_1,\ \xi_i\right\}. \quad \text{(TEAR)}$$

Different from CM, TEAR evaluates the sum of (truncated) residuals. Different from TLS, TEAR evaluates the sum of the entry-wise absolute values of $\boldsymbol{y}_i - \boldsymbol{R}\boldsymbol{x}_i - \boldsymbol{t}$ (ℓ_1 loss), rather than the squared sum (ℓ_2 loss). The benefit of having these differences is computational: With TEAR we can derive a branch-and-bound algorithm with tight and efficiently computable upper and lower bounds (Secs. 2.3 and 2.4).

Remark 1 (Truncated Least Unsquared Deviations). In a recent preprint [31], the following robust loss was considered (translated to the context of robust 3D registration):

$$\min_{(\boldsymbol{R},\boldsymbol{t})\in\mathrm{SE}(3)} \sum_{i=1}^{N} \min\left\{\|\boldsymbol{y}_i - \boldsymbol{R}\boldsymbol{x}_i - \boldsymbol{t}\|_2,\ \xi_i\right\}. \quad \text{(TLUD)}$$

TLUD uses the unsquared ℓ_2 norm rather than the ℓ_1 norm of TEAR, and if $\boldsymbol{y}_i - \boldsymbol{R}\boldsymbol{x}_i - \boldsymbol{t}$ were a scalar, TLUD would be equivalent to TEAR. One potential disadvantage of TLUD is that $\|\boldsymbol{y}_i - \boldsymbol{R}\boldsymbol{x}_i - \boldsymbol{t}\|_2$ is not separable and this might bring computational difficulties (*e.g.*, see [31, Table 1]). In Secs. 2.2 and 2.3, we will simplify TEAR and improve computational efficiency by leveraging its separable residual.

2.2. Tear Off: Decomposition of TEAR

Branching over the 6-dimensional space SE(3) would be inefficient (as prior work showed), and directly applying branch-and-bound to TEAR would lead to unscalable implementations. Therefore, in order to implement a scalable branch-and-bound method, in this section we decompose TEAR into lower-dimensional problems (tears).

Denote by y_{ij}, $\boldsymbol{r}_j^\top$, and t_j the j-th row of $\boldsymbol{y}_i$, $\boldsymbol{R}$, and $\boldsymbol{t}$, respectively. The residual $\|\boldsymbol{y}_i - \boldsymbol{R}\boldsymbol{x}_i - \boldsymbol{t}\|_1$ has 3 summands:

$$\|\boldsymbol{y}_i - \boldsymbol{R}\boldsymbol{x}_i - \boldsymbol{t}\|_1 = \sum_{j=1}^{3} |y_{ij} - \boldsymbol{r}_j^\top \boldsymbol{x}_i - t_j|.$$

While $\|\boldsymbol{y}_i - \boldsymbol{R}\boldsymbol{x}_i - \boldsymbol{t}\|_1$ has 6 degrees of freedom ($\boldsymbol{R}$ and $\boldsymbol{t}$), each summand $|y_{ij} - \boldsymbol{r}_j^\top \boldsymbol{x}_i - t_j|$ has only 3 degrees of freedom ($\boldsymbol{r}_j \in \mathbb{S}^2 := \{\boldsymbol{r} \in \mathbb{R}^3 : \|\boldsymbol{r}\| = 1\}$ and $t_j \in \mathbb{R}$). This motivates us to approximate the TEAR problem as follows. First, with some threshold hyper-parameter[3] ξ_{i1}, we *tear off* the first summand from TEAR, targeting at

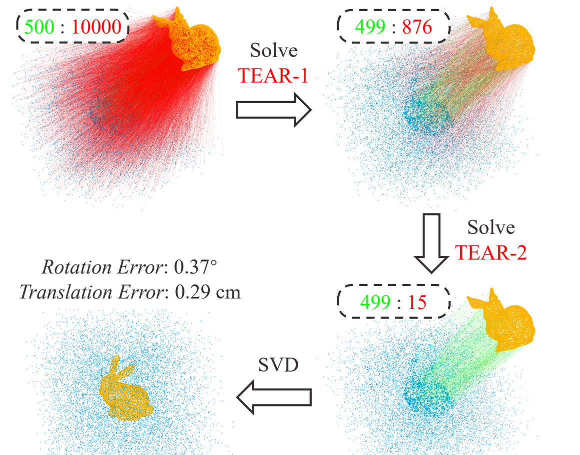


Figure 2. The pipeline of TEAR visualized (*cf*. Sec. 2.2). Green (*resp.* red) values denote the numbers of inliers (*resp.* outliers), and green (*resp.* red) lines denote inlier (*resp.* outlier) point pairs. Top left: Input point pairs; top right: point pairs indexed by $\hat{\mathcal{I}}_1$ (1) after solving TEAR-1; bottom right: point pairs indexed by $\hat{\mathcal{I}}_2$ (2) after solving TEAR-2; bottom left: the final output.

$$\min_{\boldsymbol{r}_1\in\mathbb{S}^2, t_1\in\mathbb{R}} \sum_{i=1}^{N} \min\left\{|y_{i1} - \boldsymbol{r}_1^\top \boldsymbol{x}_i - t_1|,\ \xi_{i1}\right\}. \quad \text{(TEAR-1)}$$

Solving TEAR-1 gives a solution $(\hat{\boldsymbol{r}}_1, \hat{t}_1)$ revealing the set

$$\hat{\mathcal{I}}_1 = \left\{i : |y_{i1} - \hat{\boldsymbol{r}}_1^\top \boldsymbol{x}_i - \hat{t}_1| \le \xi_{i1}\right\}, \quad (1)$$

which contains the indices of potential inliers. Then, we tear off the second summand $|y_{i2} - \boldsymbol{r}_2^\top \boldsymbol{x}_i - t_2|$, focusing on

$$\begin{aligned} \min_{\boldsymbol{r}_2\in\mathbb{S}^2, t_2\in\mathbb{R}} \ & \sum_{i\in\hat{\mathcal{I}}_1} \min\left\{|y_{i2} - \boldsymbol{r}_2^\top \boldsymbol{x}_i - t_2|,\ \xi_{i2}\right\} \quad \text{(TEAR-2)} \\ \text{s.t.} \quad & \boldsymbol{r}_2^\top \hat{\boldsymbol{r}}_1 = 0 \end{aligned}$$

where ξ_{i2} is another threshold hyper-parameter[3]. Note that the extra constraint $\boldsymbol{r}_2^\top \hat{\boldsymbol{r}}_1 = 0$ in TEAR-2 ensures the resulting rotation has orthogonal rows. Moreover, it implies TEAR-2 has 2 degrees of freedom, easier to solve than TEAR-1 after a suitable reparameterization.

Shall we proceed similarly for the third summand $|y_{i3} - \boldsymbol{r}_3^\top \boldsymbol{x}_i - t_3|$? In fact, no more tear is needed: Given some optimal $(\hat{\boldsymbol{r}}_1, \hat{t}_1)$ and $(\hat{\boldsymbol{r}}_2, \hat{t}_2)$ from TEAR-1 and TEAR-2 respectively, one needs to set $\boldsymbol{r}_3$ to be $\pm(\hat{\boldsymbol{r}}_1 \times \hat{\boldsymbol{r}}_2)$ to satisfy the rotation constraint (the sign is chosen to ensure the determinant is 1), after which one needs only to find the final

[3] The choices of hyper-parameters are discussed in the appendix.

translation component. But what we do to implement TEAR is even simpler: Solving TEAR-2 yields again a set

$$\hat{\mathcal{I}}_2 := \left\{ i \in \hat{\mathcal{I}}_1 : |y_{i2} - \hat{\boldsymbol{r}}_2^\top \boldsymbol{x}_i - \hat{t}_2| \le \xi_{i2} \right\}, \tag{2}$$

which is expected to contain very few outlier indices (*e.g.*, see Fig. 2), and next, we apply SVD [2, 25, 26] to the remaining point pairs indexed by $\hat{\mathcal{I}}_2$ and this gives a rotation and translation estimate as the final output.

In summary, TEAR consists of solving TEAR-1 and then TEAR-2, followed by an SVD; this is illustrated in Fig. 2 where we use the Stanford Bunny point cloud [19] as an example. We solve TEAR-1 and TEAR-2 by a tailored branch-and-bound method, and we discuss that in Sec. 2.3.

2.3. Branch and Bound with Tears

As mentioned, TEAR-2 is easier to solve, so here we only describe how to solve TEAR-1 using branch-and-bound, a global optimization technique that we assume, for conciseness, the reader is familiar with. The full recipe for solving TEAR-1 is in the appendix, and the code is available[1].

While TEAR-1 has 3 degrees of freedom, it would suffice to branch over the sphere $\mathbb{S}^2$ where $\boldsymbol{r}_1$ resides or equivalently over the rectangle $\mathcal{Q} := [0, 2\pi) \times [0, \pi]$, as each point (α, β) in this rectangle corresponds uniquely to a unit vector $\boldsymbol{r}_1 = [\sin\beta\cos\alpha, \sin\beta\sin\alpha, \cos\beta]$. To implement branch-and-bound, we need to consider two key steps:

- (*Upper Bound*) Given a point in $\mathcal{Q}$, that is given a unit vector $\boldsymbol{r}_1$, we minimize TEAR-1 in variable t_1, *i.e.*, solve

$$\min_{t_1 \in \mathbb{R}} \sum_{i=1}^{N} \min\left\{ |y_{i1} - \boldsymbol{r}_1^\top \boldsymbol{x}_i - t_1|,\ \xi_{i1} \right\}. \tag{3}$$

  By construction, the minimum value of (3) is always an upper bound of the minimum of TEAR-1.
- (*Lower Bound*) Given a subrectangle $[\alpha_1, \alpha_2] \times [\beta_1, \beta_2]$ in $\mathcal{Q}$, consider the following program:

$$\begin{aligned} \min_{\substack{\alpha \in [\alpha_1, \alpha_2] \\ \beta \in [\beta_1, \beta_2] \\ t_1 \in \mathbb{R}}} \ & \sum_{i=1}^{N} \min\left\{ |y_{i1} - \boldsymbol{r}_1^\top \boldsymbol{x}_i - t_1|,\ \xi_{i1} \right\} \\ \text{s.t.} \quad & \boldsymbol{r}_1 = [\sin\beta\cos\alpha, \sin\beta\sin\alpha, \cos\beta]^\top \end{aligned} \tag{4}$$

  Note that (4) is identical to TEAR-1 except that $\boldsymbol{r}_1$ is now constrained such that the two angles α and β are bounded in the intervals $[\alpha_1, \alpha_2]$ and $[\beta_1, \beta_2]$. While (4) is a subproblem encountered during branch-and-bound, solving (4) efficiently is not easy. The key step that we take is to relax (4) a little bit more so that a lower bound on the minimum of (4) can be efficiently calculated (see the appendix for details). By construction, this lower bound is also a lower bound of TEAR-1 subject to $\alpha \in [\alpha_1, \alpha_2]$ and $\beta \in [\beta_1, \beta_2]$. This is what we meant by computing a lower bound for the branch-and-bound method.

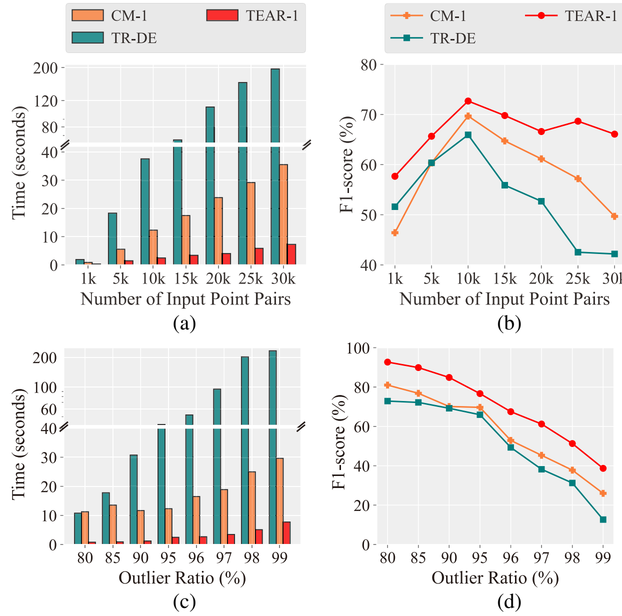


Figure 3. Solve TEAR-1 and CM-1 via branch-and-bound on random, synthetic, noisy data (Sec. 2.4). TR-DE [13], a recent branch-and-bound method, is also compared. Outlier ratio: 95% (Fig. 3a, Fig. 3b); $N = 10000$ (Fig. 3c, Fig. 3d). 30 trials.

We use the following statement to encapsulate the details of computing the desired upper and lower bounds:

Theorem 1. *We can solve* (3) *in* $O(N \log N)$ *time and compute a "tight" lower bound of* (4) *also in* $O(N \log N)$ *time.*

Behind Theorem 1 are two novel, non-trivial algorithms that we propose to compute the bounds (see the appendix for algorithmic details), and they are the game changers that enable TEAR-1 to be solved highly efficiently. The counterparts of Theorem 1 for CM and TLS are also derived, but they lead to less efficient branch-and-bound solvers than what we proposed for TEAR-1. We consolidate this claim with theoretical and numerical insights in Sec. 2.4.

2.4. TEAR Versus CM and TLS

Previously, in Sec. 2.3, we advocated performing branch-and-bound with tears. We now validate this design choice. Specifically, we follow exactly the same logic as in Secs. 2.2 and 2.3 to derive branch-and-bound methods for CM and TLS, and then compare them to TEAR. Again, at the heart of the derivation is to compute the upper and lower bounds (*cf.*, Sec. 2.3). Some details are given below.

TEAR Versus CM. Recall that TR-DE [13] is a branch-and-bound method consisting of searching a 3-dimensional space to optimize some consensus maximization objective comparable to TEAR-1, with upper and lower bounds computed in $O(N)$ time. However, similarly in Sec. 2.3, one could optimize the same objective by searching instead in a 2-dimensional space, with upper and lower bounds computable in $O(N \log N)$ time; this is in fact the core idea of the recent works [36, 75] to accelerate branch-and-bound. We contextualize this idea by implementing our version to solve a consensus maximization counterpart of TEAR-1:

$$\max_{\boldsymbol{r}_1 \in \mathbb{S}^2, t_1 \in \mathbb{R}} \sum_{i=1}^{N} \mathbf{1}\left(|y_{i1} - \boldsymbol{r}_1^\top \boldsymbol{x}_i - t_1| \leq \xi_{i1}\right). \tag{CM-1}$$

Fig. 3 visualizes the differences between these methods: TR-DE is 5 times slower than CM-1 as CM-1 searches a space of 1 dimension lower; CM-1 is more than 5 times slower than TEAR-1 as TEAR-1 better leverages numerical values of the residuals than the binary truncation of CM-1. Finally, Fig. 3 shows TEAR-1 has higher F1-scores, meaning that TEAR-1 is more robust to outliers than CM-1.

Remark 2. Since TR-DE [13] was implemented with a single thread, experiments in Sec. 2.4 (Figs. 3 and 4) all use a single thread for fair comparison. In experiments of all other sections, we use a parallel implementation of `TEAR` that is multiple times faster than the single thread version.

TEAR Versus TLS. Similarly, we solve via branch-and-bound the TLS counterpart of TEAR-1:

$$\min_{\boldsymbol{r}_1 \in \mathbb{S}^2, t_1 \in \mathbb{R}} \sum_{i=1}^{N} \min\left\{(y_{i1} - \boldsymbol{r}_1^\top \boldsymbol{x}_i - t_1)^2,\ \xi_{i1}^2\right\}. \tag{TLS-1}$$

To our knowledge, branch-and-bound algorithms have not been applied to TLS-1, so we derive lower and upper bounds for such implementation and specify the computation complexities below (compare this to Theorem 1):

Proposition 1. *At each iteration of the branch-and-bound method for solving TLS-1, we can compute an upper bound in $O(N \log N)$ time via solving*

$$\min_{t_1 \in \mathbb{R}} \sum_{i=1}^{N} \min\left\{(y_{i1} - \boldsymbol{r}_1^\top \boldsymbol{x}_i - t_1)^2,\ \xi_{i1}^2\right\}, \tag{5}$$

and we can compute a lower bound in $O(N^2)$ time.

Remark 3. [68, Algorithm 2] is not necessarily optimal to (5) as it only finds the least-squares solution at a maximum consensus set, while an $O(N \log N)$ optimal algorithm was described in [38]. Computing lower bounds entails solving a harder problem than (5) (*e.g.*, compare (3) and (4)), and for the moment we are only able to do so in $O(N^2)$ time.

Fig. 4 compares the efficiency of solving TEAR-1 and TLS-1. Fig. 4a shows TLS-1 takes slightly fewer iterations than TEAR-1, meaning that the bounds for TLS-1 are slightly tighter. On the other hand, since computing bounds for TLS-1 is more time-consuming (*cf*. Theorem 1, Proposition 1), the overall running time of TLS-1 grows more rapidly than TEAR-1 as the number of point pairs increases. Finally, Fig. 4b shows TLS-1 and TEAR-1 have similar F1-scores, suggesting they are comparable in rejecting outliers.

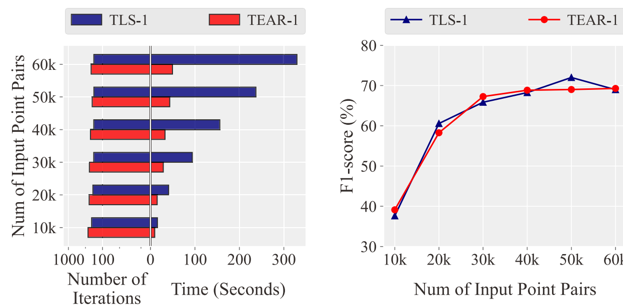


Figure 4. Solve TEAR-1 and TLS-1 via branch-and-bound on random, synthetic, noisy data (Sec. 2.4). Outlier ratio: 99%. 30 trials.

3. Standard Experiments

Here we conduct experimental comparisons that are standard as in prior works. These include experiments on synthetic data (Sec. 3.1) and real datasets (Sec. 3.2).

Setup. For a fair comparison, we evaluate all methods on a laptop equipped with an Intel Core i7-10875H@2.3 GHz and 16GB of RAM. For methods implemented in PyTorch, we slightly adjust their codes so that they run on CPUs; note that this typically only affects running times, not accuracy. We run all methods in Python or through the provided Python interfaces of the C++ codes. Since the Python and C++ codes of MAC [76] behave differently, we compare both versions, namely MAC (Python) and MAC (C++). The maximum number of iterations for RANSAC is set to 100k.

Evaluation Metrics. Following [3, 13, 14], we use 5 metrics for evaluation: Registration Recall (RR), F1-score (F1), Rotation Error (RE), Translation Error (TE), and Time. RR denotes the percentage of successful registration where the RE and TE are below specific thresholds, *e.g.*, $(15^\circ,\ 30\text{cm})$ for 3DMatch dataset, $(5^\circ,\ 60\text{cm})$ for KITTI dataset, and $(3^\circ,\ 50\text{cm})$ for ETH dataset. F1 denotes the harmonic mean of precision and recall [3]. Finally, we report the peak memory consumption of an algorithm during its execution.

3.1. Experiments on Synthetic Data

Data Generation. Following [14, 46], we generate synthetic point pairs as follows. First, we randomly sample N points $\{\boldsymbol{x}_i\}_{i=1}^N$ from $\mathcal{N}(0, \boldsymbol{I}_3)$. Then, we apply a random rotation $\boldsymbol{R}^*$ and translation $\boldsymbol{t}^*$ to each $\boldsymbol{x}_i$ and add some Gaussian noise $\boldsymbol{\epsilon}_i \sim \mathcal{N}(0, \sigma^2 \boldsymbol{I}_3)$ of variance $\sigma = 0.01$; that is $\boldsymbol{y}_i = \boldsymbol{R}^* \boldsymbol{x}_i + \boldsymbol{t}^* + \boldsymbol{\epsilon}_i$. This gives N (noisy) inlier pairs $\{\boldsymbol{x}_i, \boldsymbol{y}_i\}_i^N$. Next, to generate outliers, we replace a fraction of $\boldsymbol{y}_i$'s with random Gaussian points sampled from $\mathcal{N}(0, \tau^2 \boldsymbol{I}_3)$, where τ is set to 1.67. Almost all methods require an inlier threshold ξ_i, and we set it to 5.54σ.

Results. `TEAR` is shown to be scalable (Fig. 1), accurate (Fig. 5a, Fig. 5b) and more efficient in most cases (Fig. 5c).

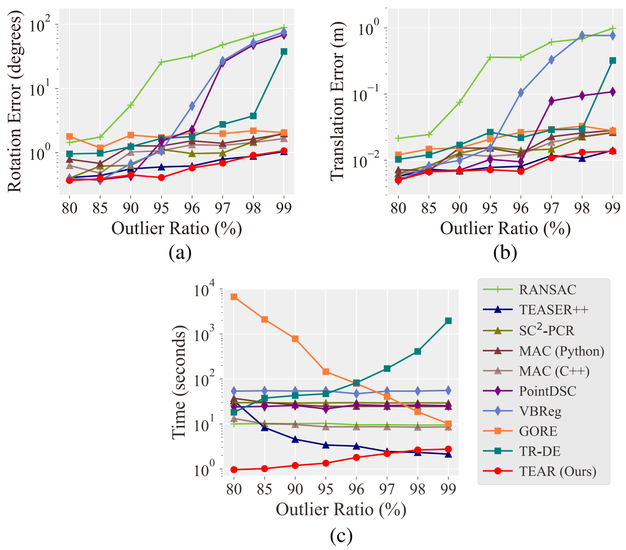


Figure 5. Synthetic Experiments (Sec. 3.1). $N = 10000$, 30 trials.

3.2. Experiments on Real Data

Here we report experimental results on three real-world datasets, namely 3DMatch (indoor scenes) [74], KITTI (outdoor scenes) [22], and ETH (outdoor, large-scale) [60].

3DMatch. We follow [14, 68] to use 3DSmoothNet [23] as the feature descriptor for the 3DMatch dataset [74]. The results are shown in Tab. 1. Note that MAC (Python) runs out of the 16GB memory, so we compared its C++ version.

KITTI. Following [3, 76], we use FPFH [56] as the feature descriptor of the KITTI dataset [22]. In Tab. 2, MAC (Python) works well, and MAC (C++) runs out of memory.

ETH. Following [36, 60], we use ISS [77] and FPFH [56] as feature descriptors for the ETH dataset. This time, in Tab. 3, more methods run out of memory including both versions of MAC and two deep learning methods, PointDSC and VBReg. This is because, after feature matching, there are still a few relatively large point clouds (*e.g.*, $N > 10$k) with high outlier ratios, making registration challenging.

Analysis of Results. That MAC often runs out of memory is not quite surprising, as it stores all maximal cliques besides the consistency graph. In theory, in the worst case, a graph can have as many as $3^{N/3}$ maximal cliques [42, 61]. In our experiments of Tab. 2, MAC (Python) typically produces more than 10^4 cliques, and in Tab. 1, MAC (C++) typically produces more than 10^5, or occasionally a few million, maximal cliques ($N = 5000$ in both tables).

That TR-DE [13] is often slower than TEAR validates our design and implementation again: Even though TR-DE is also a branch-and-bound method (similarly to TEAR), TEAR is up to 70 times faster than TR-DE on real data.

That other methods perform really well—once given the memory they demand—might have been established knowledge in the literature (*e.g.*, TEASER++ [68], PointDSC [3], SC^2-PCR [14]). But the drawback of their unscalability manifests itself when memory is insufficient or point clouds encountered are huge; we re-emphasize this point in Sec. 4.

Overall, on the three standard datasets, we find TEAR to be competitive in registration accuracy, while it runs a few times faster than the second fastest method.

Table 1. Results on 3DMatch (3DSmoothNet descriptors).

Method	RR(%)↑	F1(%)↑	RE(°)↓	TE(cm)↓	Time(s)↓
RANSAC [21]	92.30	87.95	2.59	7.91	2.52
TEASER++ [68]	92.05	87.42	2.23	6.62	3.77
SC^2-PCR [14]	94.45	89.23	2.19	**6.40**	4.56
MAC (Python) [76]	out-of-memory				
MAC (C++) [76]	**94.57**	89.48	2.21	6.52	6.89
PointDSC [3]	93.65	89.07	2.17	6.75	5.28
VBReg [28]	37.09	18.07	6.15	15.65	8.07
TR-DE [13]	91.37	86.99	2.71	7.62	12.76
TEAR (Ours)	94.52	**89.65**	**2.06**	6.55	**1.26**

Table 2. Results on the KITTI dataset (FPFH descriptors).

Method	RR(%)↑	F1(%)↑	RE(°)↓	TE(cm)↓	Time(s)↓
RANSAC [21]	95.68	81.23	1.06	23.19	3.79
TEASER++ [68]	97.84	93.73	0.43	8.67	0.36
SC^2-PCR [14]	**99.64**	**94.26**	**0.39**	**8.29**	4.33
MAC (Python) [76]	94.95	89.52	0.52	10.26	4.53
MAC (C++) [76]	out-of-memory				
PointDSC [3]	98.20	92.71	0.57	8.67	6.20
VBReg [28]	98.92	92.69	0.45	8.41	8.20
TR-DE [13]	96.76	87.20	0.90	15.63	8.66
TEAR (Ours)	99.10	93.85	**0.39**	8.62	**0.25**

Table 3. Results on the ETH dataset (ISS + FPFH descriptors).

Method	RR(%)↑	F1(%)↑	RE(°)↓	TE(cm)↓	Time(s)↓
RANSAC [21]	69.05	65.17	0.44	10.31	6.12
TEASER++ [68]	**96.43**	92.23	0.29	5.84	0.85
SC^2-PCR [14]	91.67	90.34	0.32	6.25	12.93
MAC (both) [76]	out-of-memory				
PointDSC [3]	out-of-memory				
VBReg [28]	out-of-memory				
TR-DE [13]	88.09	73.40	0.62	16.49	7.57
TEAR (Ours)	**96.43**	**93.14**	**0.25**	**5.71**	**0.38**

4. Huge-Scale Experiments

Inspired by [51], here we perform *huge-scale* experiments using 5 objects (*Armadillo*, *Happy Buddha*, *Asian Dragon*, *Thai Statue*, *Lucy*) from the Stanford 3D scanning dataset [19]. The number of points in each object ranges from 10^5 to 10^7. For each object, we resize it so that all points lie in $[0, 1]^3$ and treat it as the source point cloud $\{\boldsymbol{x}_i\}_{i=1}^N$. We use the same procedure of Sec. 3.1 to generate the target point

Table 4. Experiments with huge-scale point pairs at extremely high outlier ratios on the Stanford 3D scanning dataset [19]. We report rotation errors in degrees | translation errors in centimeter | running times in seconds of various methods. Average over 20 trials.

Point Cloud Name # of Input Point Pairs (Outlier Ratio)	*Armadillo* 10^5 (99%)	*Happy Buddha* 5×10^5 (99.2%)	*Asian Dragon* 10^6 (99.4%)	*Thai Statue* 4×10^6 (99.6%)	*Lucy* 10^7 (99.8%)
Consistency Graph-based [14, 68, 76]	out-of-memory				
Deep Learning-based [3, 28]	out-of-memory				
RANSAC [21]	53.1 | 23.1 | 95.9	30.7 | 15.8 | 582	36.5 | 22.7 | 1179	37.1 | 24.6 | 6125	$\geq$ 8 hours
FGR [78]	57.1 | 39.1 | 2.48	84.1 | 23.7 | 19.3	62.1 | 19.7 | 39.8	79.7 | 15.2 | 175	88.9 | 11.5 | 449
GORE [46]	0.67 | 0.52 | 6592	$\geq$ 12 hours			
TR-DE [13]	36.5 | 16.9 | 4658	$\geq$ 9 hours			
TEAR (Ours)	0.51 | 0.25 | 12.7	0.23 | 0.13 | 119	0.14 | 0.12 | 356	0.11 | 0.08 | 1013	0.07 | 0.06 | 1972

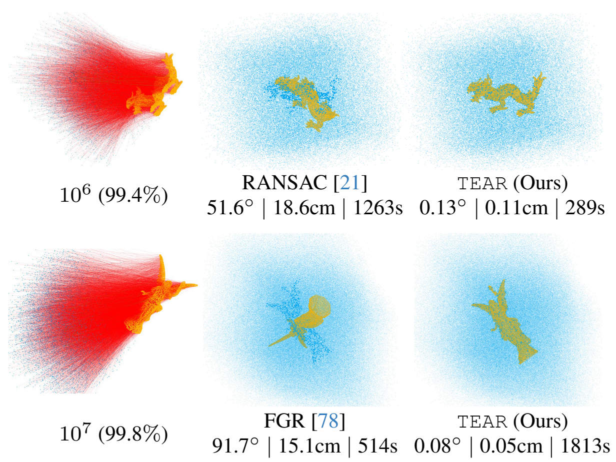


Figure 6. Column 1: input point pairs (Top: *Asian Dragon*, Bottom: *Lucy*). Column 2: Outputs of RANSAC [21] and FGR [78] (Format: rotation error | translation error | running time). Column 3: TEAR succeeds in registration for challenging scenarios.

cloud $\{\boldsymbol{y}_i\}_{i=1}^N$ with a controlled number of random outliers.

Tab. 4 shows the consistency graph methods and deep learning methods are unscalable, RANSAC [21] and FGR [78] are inaccurate at extreme outlier ratios, GORE [46] and TR-DE [13] are inefficient. Fig. 6 visualizes the results in Tab. 4 for *Asian Dragon* and *Lucy*, where TEAR is the only method that aligns the huge-scale point clouds accurately.

Inasmuch as downsampling the point clouds would enable other methods to be applied, we perform such an experiment for these methods to compare with TEAR. In particular, we take the 10^7 point pairs generated from *Lucy* and downsample them to 10^4 point pairs, which are given as inputs to other methods. In Fig. 7, the rotation errors of these methods are large (the translation errors are shown in the appendix). Indeed, downsampling would throw away inliers, making the subsequent registration problem more challenging. In fact, in experiments of Fig. 7b, we found it was not just that the total number of inliers inevitably decreased after downsampling; the outlier ratio could even grow from 95% to averagely 98.17%. In contrast, since TEAR is capable of operating the original point cloud, it delivers lower

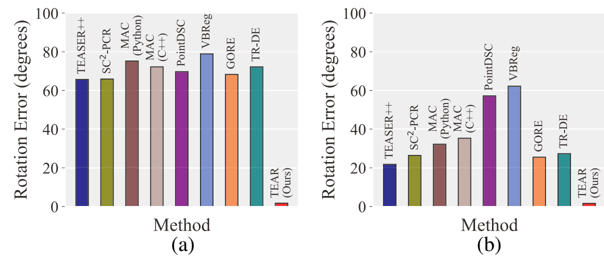


Figure 7. Average rotation errors of other methods in Tab. 4 taking as inputs the 10^4 points downsampled from *Lucy* that originally has 10^7 point pairs (Fig. 7a: 99.8% outliers; Fig. 7b: 95% outliers). TEAR runs on the original 10^7 input point pairs. 20 trials.

errors than other methods that perform downsampling.

5. Conclusion and Future Work

In the paper, we showed that TEAR stands on the simple principle of branch-and-bound, stands with state-of-the-art methods at the same level of accuracy, stands in contrast to other slower branch-and-bound methods, and stands out as a scalable method for outlier-robust 3D registration.

We found it exciting to exhibit a case where branch-and-bound, a technique famously known for its global optimality guarantees and infamously known for its being slow, can actually be competitive in outlier-robust 3D registration. The key ideas of achieving so include using *Truncated Entry-wise Absolute Residuals* (TEAR) as the robust loss, deriving tight upper and lower bounds based on TEAR, and engineering an efficient implementation. We look forward to extending these ideas to other geometric vision problems, for example, absolute pose estimation (2D-3D registration).

Acknowledgements. This work is supported in part by the Shenzhen Portion of Shenzhen-Hong Kong Science and Technology Innovation Cooperation Zone under Grant HZQBKCZYB-20200089, in part by the Hong Kong RGC under T42-409/18-R and 14218322, in part by the Hong Kong Centre for Logistics Robotics, and in part by the VC Fund 4930745 of the CUHK T Stone Robotics Institute.

References

[1] Khurrum Aftab and Richard Hartley. Convergence of iteratively re-weighted least squares to robust M-estimators. In *IEEE Winter Conference on Applications of Computer Vision*, 2015. 1

[2] K Somani Arun, Thomas S Huang, and Steven D Blostein. Least-squares fitting of two 3D point sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (5):698–700, 1987. 1, 5

[3] Xuyang Bai, Zixin Luo, Lei Zhou, Hongkai Chen, Lei Li, Zeyu Hu, Hongbo Fu, and Chiew-Lan Tai. PointDSC: Robust point cloud registration using deep spatial consistency. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2021. 2, 6, 7, 8

[4] Daniel Barath and Jiří Matas. Graph-cut RANSAC. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 1

[5] Daniel Barath, Jiri Matas, and Jana Noskova. MAGSAC: marginalizing sample consensus. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019. 1

[6] Amir Beck. On the convergence of alternating minimization for convex programming with applications to iteratively reweighted least squares and decomposition schemes. *SIAM Journal on Optimization*, 25(1):185–209, 2015. 1

[7] PJ Besl and Neil D McKay. A method for registration of 3D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992. 1

[8] Andrew Blake and Andrew Zisserman. *Visual Reconstruction*. MIT Press, 1987. 1

[9] Jesus Briales and Javier Gonzalez-Jimenez. Convex global 3D registration with lagrangian duality. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 2

[10] Álvaro Parra Bustos and Tat-Jun Chin. Guaranteed outlier removal for rotation search. In *IEEE International Conference on Computer Vision*, 2015. 2

[11] Zhipeng Cai, Tat-Jun Chin, Alvaro Parra Bustos, and Konrad Schindler. Practical optimal registration of terrestrial LiDAR scan pairs. *ISPRS Journal of Photogrammetry and Remote Sensing*, 147:118–131, 2019. 1

[12] Dylan Campbell, Lars Petersson, Laurent Kneip, and Hongdong Li. Globally-optimal inlier set maximisation for camera pose and correspondence estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(2):328–342, 2018.

[13] Wen Chen, Haoang Li, Qiang Nie, and Yun-Hui Liu. Deterministic point cloud registration via novel transformation decomposition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022. 1, 2, 5, 6, 7, 8

[14] Zhi Chen, Kun Sun, Fan Yang, and Wenbing Tao. SC^2-PCR: A second order spatial compatibility for efficient and robust point cloud registration. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022. 2, 6, 7, 8

[15] Zhi Chen, Kun Sun, Fan Yang, Lin Guo, and Wenbing Tao. SC^2-PCR++: Rethinking the generation and selection for efficient and robust point cloud registration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023. 2, 3

[16] Tat-Jun Chin, Yang Heng Kee, Anders Eriksson, and Frank Neumann. Guaranteed outlier removal with mixed integer linear programs. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 2

[17] Christopher Choy, Wei Dong, and Vladlen Koltun. Deep global registration. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2020. 2

[18] David Coleman, Paul Holland, Neil Kaden, Virginia Klema, and Stephen C Peters. A system of subroutines for iteratively reweighted least squares computations. *ACM Transactions on Mathematical Software*, 6(3):327–336, 1980. 1

[19] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Annual Conference on Computer Graphics and Interactive Techniques*, 1996. 5, 7, 8

[20] Olof Enqvist, Klas Josephson, and Fredrik Kahl. Optimal correspondences from pairwise constraints. In *International Conference on Computer Vision*, 2009. 2

[21] Martin A Fischler and Robert C Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981. 1, 7, 8

[22] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research*, 2013. 7

[23] Zan Gojcic, Caifa Zhou, Jan D. Wegner, and Andreas Wieser. The perfect match: 3D point cloud matching with smoothed densities. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5540–5549, 2019. 7

[24] Shiyi Guo, Fulin Tang, Bingxi Liu, Yujie Fu, and Yihong Wu. An accurate outlier rejection network with higher generalization ability for point cloud registration. *IEEE Robotics and Automation Letters*, 8(8):4649–4656, 2023. 2

[25] Berthold KP Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America A*, 4(4):629–642, 1987. 1, 5

[26] Berthold KP Horn, Hugh M Hilden, and Shahriar Negahdaripour. Closed-form solution of absolute orientation using orthonormal matrices. *Journal of the Optical Society of America A*, 5(7):1127–1135, 1988. 1, 5

[27] Jose Pedro Iglesias, Carl Olsson, and Fredrik Kahl. Global optimality for point set registration using semidefinite programming. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2020. 2

[28] Haobo Jiang, Zheng Dang, Zhen Wei, Jin Xie, Jian Yang, and Mathieu Salzmann. Robust outlier rejection for 3d registration with variational bayes. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023. 2, 3, 7, 8

[29] Huu Le and Christopher Zach. Robust fitting with truncated least squares: A bilevel optimization approach. In *International Conference on 3D Vision*, 2021. 1

[30] Junha Lee, Seungwook Kim, Minsu Cho, and Jaesik Park. Deep hough voting for robust global registration. In *IEEE/CVF International Conference on Computer Vision*, 2021. 2

[31] Seong Hun Lee and Javier Civera. Robust single rotation averaging revisited. Technical report, arXiv:2309.05388 [cs.CV], 2023. 4

[32] Marius Leordeanu and Martial Hebert. A spectral technique for correspondence problems using pairwise constraints. In *International Conference on Computer Vision*, 2005. 2

[33] Hongdong Li. Consensus set maximization with guaranteed global optimality for robust geometry estimation. In *International Conference on Computer Vision*, 2009. 1

[34] Hongdong Li and Richard Hartley. The 3D-3D registration problem revisited. In *IEEE International Conference on Computer Vision*, 2007.

[35] Haoang Li, Pyojin Kim, Ji Zhao, Kyungdon Joo, Zhipeng Cai, Zhe Liu, and Yun-Hui Liu. Globally optimal and efficient vanishing point estimation in atlanta world. In *European Conference on Computer Vision*, 2020. 1

[36] Xinyi Li, Yinlong Liu, Hu Cao, Xueli Liu, Feihu Zhang, and Alois Knoll. Efficient and deterministic search strategy based on residual projections for point cloud registration. *arXiv:2305.11716 [cs.CV]*, 2023. 6, 7

[37] Wei Lian, Lei Zhang, and Ming-Hsuan Yang. An efficient globally optimal algorithm for asymmetric point matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(7):1281–1293, 2017. 1

[38] Tzu-Ying Liu and Hui Jiang. Minimizing sum of truncated convex functions and its applications. *Journal of Computational and Graphical Statistics*, 28(1):1–10, 2019. 6

[39] Yinlong Liu, Chen Wang, Zhijian Song, and Manning Wang. Efficient global point cloud registration by matching rotation invariant features through translation search. In *European Conference on Computer Vision*, 2018. 1

[40] Yinlong Liu, Yiru Wang, Manning Wang, Guang Chen, Alois Knoll, and Zhijian Song. Globally optimal linear model fitting with unit-norm constraint. *International Journal of Computer Vision*, pages 1–14, 2022. 1

[41] Parker C Lusk, Kaveh Fathian, and Jonathan P How. Clipper: A graph-theoretic framework for robust data association. In *IEEE International Conference on Robotics and Automation*, 2021. 2

[42] John W Moon and Leo Moser. On cliques in graphs. *Israel Journal of Mathematics*, 3:23–28, 1965. 7

[43] Peter Ochs, Alexey Dosovitskiy, Thomas Brox, and Thomas Pock. On iteratively reweighted algorithms for nonsmooth nonconvex optimization in computer vision. *SIAM Journal on Imaging Sciences*, 8(1):331–372, 2015. 1

[44] Carl Olsson, Fredrik Kahl, and Magnus Oskarsson. Branch-and-bound methods for Euclidean registration problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(5):783–794, 2008. 1

[45] G Dias Pais, Srikumar Ramalingam, Venu Madhav Govindu, Jacinto C Nascimento, Rama Chellappa, and Pedro Miraldo. 3DRegNet: A deep neural network for 3D point registration. In *IEEE/CVF conference on computer vision and pattern recognition*, 2020. 2

[46] Álvaro Parra Bustos and Tat-Jun Chin. Guaranteed outlier removal for point cloud registration with correspondences. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12):2868–2882, 2018. 2, 6, 8

[47] Álvaro Parra Bustos, Tat-Jun Chin, Anders Eriksson, Hongdong Li, and David Suter. Fast rotation search with stereographic projections for 3D registration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(11):2227–2240, 2016. 1

[48] Liangzu Peng and René Vidal. Block coordinate descent on smooth manifolds: Convergence theory and twenty-one examples. Technical report, arXiv:2305.14744v3 [math.OC], 2023. 1

[49] Liangzu Peng, Mahyar Fazlyab, and René Vidal. Semidefinite relaxations of truncated least-squares in robust rotation search: Tight or not. In *European Conference on Computer Vision*, 2022. 2

[50] Liangzu Peng, Christian Kümmerle, and René Vidal. Global linear and local superlinear convergence of IRLS for non-smooth robust regression. In *Advances in Neural Information Processing Systems*, 2022. 1

[51] Liangzu Peng, Manolis C. Tsakiris, and René Vidal. ARCS: Accurate rotation and correspondences search. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022. 7

[52] Liangzu Peng, Christian Kümmerle, and René Vidal. On the convergence of IRLS and its variants in outlier-robust estimation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023. 1

[53] Zheng Qin, Hao Yu, Changjian Wang, Yuxing Peng, and Kai Xu. Deep graph-based spatial consistency for robust non-rigid point cloud registration. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023. 2

[54] Rahul Raguram, Ondrej Chum, Marc Pollefeys, Jiri Matas, and Jan-Michael Frahm. USAC: A universal framework for random sample consensus. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):2022–2038, 2012. 1

[55] Szymon Rusinkiewicz and Marc Levoy. Efficient variants of the ICP algorithm. In *International Conference on 3D Digital Imaging and Modeling*, 2001. 1

[56] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (FPFH) for 3D registration. In *IEEE International Conference on Robotics and Automation*, pages 3212–3217, 2009. 7

[57] Daniel Scholz. *Deterministic Global Optimization: Geometric Branch-and-bound Methods and Their Applications*. Springer Science & Business Media, 2011. 1

[58] Chitturi Sidhartha, Lalit Manam, and Venu Madhav Govindu. Adaptive annealing for robust geometric estimation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023. 1

[59] Lei Sun and Lu Deng. TriVoC: Efficient voting-based consensus maximization for robust point cloud registration with extreme outlier ratios. *IEEE Robotics and Automation Letters*, 7(2):4654–4661, 2022. 2

[60] Pascal Willy Theiler, Jan Dirk Wegner, and Konrad Schindler. Keypoint-based 4-points congruent sets–automated marker-less registration of laser scans. *ISPRS Journal of Photogrammetry and Remote Sensing*, 96:149–163, 2014. 7

[61] Etsuji Tomita, Akira Tanaka, and Haruhisa Takahashi. The worst-case time complexity for generating all maximal cliques and computational experiments. *Theoretical Computer Science*, 363(1):28–42, 2006. 7

[62] Philip HS Torr and Andrew Zisserman. MLESAC: A new robust estimator with application to estimating image geometry. *Computer Vision and Image Understanding*, 78(1):138–156, 2000. 1

[63] Chen Wang, Yinlong Liu, Yiru Wang, Xuechen Li, and Manning Wang. Efficient and outlier-robust simultaneous pose and correspondence determination by branch-and-bound and transformation decomposition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(10):6924–6938, 2022. 1

[64] Yue Wang and Justin M Solomon. Deep closest point: Learning representations for point cloud registration. In *IEEE/CVF International Conference on Computer Vision*, 2019. 2

[65] Li Yan, Pengcheng Wei, Hong Xie, Jicheng Dai, Hao Wu, and Ming Huang. A new outlier removal strategy based on reliability of correspondence graph for fast point cloud registration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022. 2

[66] Heng Yang and Luca Carlone. A quaternion-based certifiably optimal solution to the Wahba problem with outliers. In *IEEE International Conference on Computer Vision*, 2019. 2

[67] Heng Yang, Pasquale Antonante, Vasileios Tzoumas, and Luca Carlone. Graduated non-convexity for robust spatial perception: From non-minimal solvers to global outlier rejection. *IEEE Robotics and Automation Letters*, 5(2):1127–1134, 2020. 1

[68] Heng Yang, Jingnan Shi, and Luca Carlone. TEASER: Fast and certifiable point cloud registration. *IEEE Transactions on Robotics*, 37(2):314–333, 2021. 2, 6, 7, 8

[69] Heng Yang, Ling Liang, Luca Carlone, and Kim-Chuan Toh. An inexact projected gradient method with rounding and lifting by nonlinear programming for solving rank-one semidefinite relaxation of polynomial optimization. *Mathematical Programming*, 201(1-2):409–472, 2023. 2

[70] Jiaolong Yang, Hongdong Li, Dylan Campbell, and Yunde Jia. Go-ICP: A globally optimal solution to 3D ICP point-set registration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(11):2241–2254, 2016. 1

[71] Runzhao Yao, Shaoyi Du, Wenting Cui, Aixue Ye, Feng Wen, Hongbo Zhang, Zhiqiang Tian, and Yue Gao. Hunter: Exploring high-order consistency for point cloud registration with severe outliers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023. 2

[72] Hao Yu, Zheng Qin, Ji Hou, Mahdi Saleh, Dongsheng Li, Benjamin Busam, and Slobodan Ilic. Rotation-invariant transformer for point cloud matching. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023. 2

[73] Christopher Zach and Guillaume Bourmaud. Descending, lifting or smoothing: Secrets of robust cost optimization. In *European Conference on Computer Vision*, 2018. 1

[74] Andy Zeng, Shuran Song, Matthias Nießner, Matthew Fisher, Jianxiong Xiao, and T Funkhouser. 3DMatch: Learning the matching of local 3D geometry in range scans. In *IEEE Conference on Computer Vision and Pattern Recognition*, page 4, 2017. 7

[75] Xinyue Zhang, Liangzu Peng, Wangting Xu, and Laurent Kneip. Accelerating globally optimal consensus maximization in geometric vision. Technical report, arXiv:2304.05156 [cs.CV], 2023. 1, 6

[76] Xiyu Zhang, Jiaqi Yang, Shikun Zhang, and Yanning Zhang. 3D registration with maximal cliques. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023. 2, 6, 7, 8

[77] Yu Zhong. Intrinsic shape signatures: A shape descriptor for 3D object recognition. In *International Conference on Computer Vision Workshops*, 2009. 7

[78] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Fast global registration. In *European Conference on Computer Vision*, 2016. 1, 8