Scalable 3D Registration via Truncated Entry-wise Absolute Residuals (Appendix)

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

Overview of the Appendix

This appendix is organized as follows:
- In Appendix A, we review the general branch-and-bound algorithm framework.
- In Appendices B and C, we introduce the proposed bound computation methods for TEAR-1 and TEAR-2, respectively.
- In Appendices D and E, we introduce the developed bound computation methods for CM-1 and TLS-1, respectively.
- In Appendix F, we prove Proposition 2 (the statement of Proposition 2 is presented in Appendix B.2).
- In Appendix G, we present extra experimental details.

A. The General Branch-and-bound Algorithm

In this section, we review the branch-and-bound framework. Given an objective function $\mathcal{T}(\cdot)$ and the corresponding decision variable $\mathbf{v}$, branch-and-bound computes a global minimizer of $\mathcal{T}(\cdot)$ up to a prescribed error $\epsilon > 0$. It does so by recursively dividing the parameter space $\mathbb{B}_0$ of the decision variable into smaller branches and computing upper and lower bounds of the objective over each branch. By upper bounds we mean upper bounds of the optimal value, and by lower bounds over a given branch we mean lower bounds of the optimal value on this branch.

The algorithmic listing of this branch-and-bound recursion is shown in Algorithm 1. The algorithm maintains the smallest upper bound U^* found so far and the corresponding point $\mathbf{v}^*$ that achieves that bound; U^* and $\mathbf{v}^*$ are initialized at Line 5, by a function that computes upper bounds. In the meantime, the algorithm starts with the initial branch $\mathbb{B}_0$ (Line 3), computes a lower bound (Line 6), and puts the branch $\mathbb{B}_0$ into a priority queue (Line 7). Next, branch-and-bound enters a while loop (Lines 8-24), at each iteration keeping examining the branch $\mathbb{B}$ in the queue with the highest priority (Line 9). If the smallest upper bound U^* found so far is smaller than the lower bound $L(\mathbb{B})$ up to a small tolerance ϵ, then the current point $\mathbf{v}^*$ already obtains the minimum value up to error ϵ, therefore the algorithm terminates (Lines 10-12). Otherwise, the algorithm proceeds by dividing the current branch $\mathbb{B}$ into smaller branches $\mathbb{B}_i$. Similarly, if the lower bound $L(\mathbb{B}_i)$ is larger than U^*, then $\mathbb{B}_i$ can never contain any optimal solutions, therefore we can prune it (Lines 15 and 16). Otherwise, the algorithm computes an upper bound for $\mathbb{B}_i$ (Line 18), updates the current solutions $(U^*, \mathbf{v}^*)$ if needed (Lines 19-21), and stores the branch $\mathbb{B}_i$ into the queue $\mathcal{Q}$ with priority $L(\mathbb{B}_i)$.

As Algorithm 1 shows, the crucial step in the branch-and-bound method is computing the upper and lower bound of the objective on a given branch $\mathbb{B}$. Hence, to solve TEAR-1, CM-1, and TLS-1, we propose methods for computing the desired bounds in Appendices B, D and E, respectively. Moreover, in Appendix C we describe the bound computation for TEAR-2.

B. Bounds for TEAR-1

B.1. Upper Bound

Given a branch of $\boldsymbol{r}$, *e.g.*, $[\alpha_1, \alpha_2] \times [\beta_1, \beta_1]$, we choose the center $\dot{\boldsymbol{r}}_1$ to compute an upper bound $\overline{U}$. Let $a_i := y_{i1} - \dot{\boldsymbol{r}}_1^\top \boldsymbol{x}_i$, and an upper bound can be computed as

$$\overline{U} = \min_{t_1 \in \mathbb{R}} \sum_{i=1}^{N} \min\left\{|a_i - t_1|,\ \xi_{i1}\right\} = \min_{t_1 \in \mathbb{R}} U(t_1), \tag{1}$$

[*] Equal contribution

Algorithm 1: The general branch-and-bound template to minimize a given objective function globally optimally.

Input: Objective function $\mathcal{T}(\cdot)$, threshold ϵ;
Output: A global minimizer $\mathbf{v}^*$;
Initial branch $\mathbb{B}_0 \leftarrow$ parameter space of $\mathbf{v}$;
$\mathcal{Q} \leftarrow$ An empty priority queue;
$U^*,\ \mathbf{v}^* \leftarrow$ getUpperBound($\mathcal{T}(\cdot),\ \mathbb{B}_0$); // U^* denotes the smallest upper bound so far, and $U^* = \mathcal{T}(\mathbf{v}^*)$.
$L(\mathbb{B}_0) \leftarrow$ getLowerBound($\mathcal{T}(\cdot),\ \mathbb{B}_0$);
Insert $\mathbb{B}_0$ into $\mathcal{Q}$ with priority $L(\mathbb{B}_0)$;
while $\mathcal{Q}$ is not empty **do**
  Pop the branch $\mathbb{B}$ from Q with the lowest lower bound $L(\mathbb{B})$;
  if $U^* - L(\mathbb{B}) < \epsilon$ **then** // Terminate the algorithm: The current $\mathbf{v}^*$ is already optimal up to tolerance ϵ.
    end while
  end if
  for $\mathbb{B}_i$ **in** divideBranch($\mathbb{B}$) **do** // Divide $\mathbb{B}$ into smaller branches $\mathbb{B}_i$'s and compute upper and lower bounds for each $\mathbb{B}_i$.
    $L(\mathbb{B}_i) \leftarrow$ getLowerBound($\mathcal{T}(\cdot),\ \mathbb{B}_i$);
    if $L(\mathbb{B}_i) \geq U^*$ **then** // The branch $\mathbb{B}_i$ does not contain any optimal solutions, so prune it.
      continue
    else // The branch $\mathbb{B}_i$ could potentially contain better solutions, so compute an upper bound and store it into the priority queue.
      $U(\mathbb{B}_i),\ \mathbf{v}_i \leftarrow$ getUpperBound($\mathcal{T}(\cdot),\ \mathbb{B}_i$);
      if $U(\mathbb{B}) < U^*$ **then**
        $\mathbf{v}^* \leftarrow \mathbf{v}$; $U^* \leftarrow U(\mathbb{B})$;
      end if
      Insert $\mathbb{B}_i$ into $\mathcal{Q}$ with priority $L(\mathbb{B}_i)$;
    end if
  end for
end while

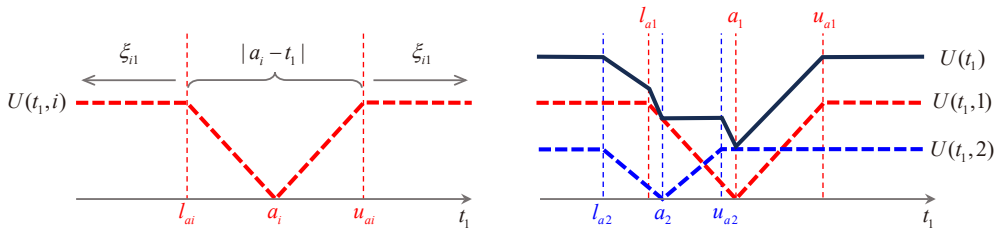


Figure 1. (a) Illustration of $U(t_1, i) = \min\{|a_i - t_1|,\ \xi_{i1}\}$; (b) Illustration of $U(t_1) = \sum_{i=1}^{N} U(t_1, i)$ when $N = 2$. (*cf*. Appendix B.1).

where $U(t_1)$ is defined to be the sum of $U(t_1, i) := \min\{|a_i - t_1|,\ \xi_{i1}\}$, that is $U(t_1) := \sum_{i=1}^{N} U(t_1, i)$. Define $l_{ai} := a_1 - \xi_{i1}$ and $u_{ai} := a_1 + \xi_{i1}$. Note that $U(t_1, i)$ equals to $|a_i - t_1|$ if $t_1 \in [l_{ai}, u_{ai}]$, or otherwise ξ_{i1} (see Fig. 1a). Furthermore, $U(t_1, i)$ is differentiable at any point t_1 except when t_1 is equal to l_{ai}, a_i, or u_{ai}. Finally, at differentiable points, the derivative of $U(t_1, i)$ is either 1 or -1 or 0, therefore the derivative of $U(t_1)$ lies in $\{-N, \ldots, -1, 0, 1, \ldots, N\}$.

We need more notations. Let $\{\lambda_k\}_{k=1}^{3N}$ be a sorted version of the $3N$ numbers $\{a_i\}_{i=1}^{N} \cup \{l_{ai}\}_{i=1}^{N} \cup \{u_{ai}\}_{i=1}^{N}$. For each $k = 1, \cdots, 3N - 1$, define the open interval $\mathcal{I}_k := (\lambda_k, \lambda_{k+1})$. Without loss of generality, we assume $\lambda_k \neq \lambda_{k+1}$ and therefore $\mathcal{I}_k$ is not empty. Note that the derivative of $U(t_1, i)$ is equal to 1 or -1 depending on the sign of $a_i - t_1$. Since the sign of $a_i - t_1$ in the definition of $U(t_1, i)$ is constant on every interval $\mathcal{I}_k$, the derivative of $U(t_1, i)$ is constant on $\mathcal{I}_k$ for every i and every k. As a consequence, the derivative of $U(t_1)$ is constant on every interval $\mathcal{I}_k$ (see Fig. 1b).

With the above notations, we derive the following proposition that localizes a global minimizer of (1):

Proposition 1. *In the N points $\{a_i\}_{i=1}^{N}$, there is a globally optimal solution of* (1).

Proof. First we will show, in the $3N$ points $\{\lambda_k\}_{k=1}^{3N}$, there is a globally optimal solution of (1). To prove this, let us assume there is a global minimizer different from the $3N$ points $\{\lambda_k\}_{k=1}^{3N}$. Note that $U(t_1)$ in (1) is differentiable except at the $3N$

Algorithm 2: Globally Optimal 1D TEAR Solver for Upper Bound Computation (Proposition 1).

Input: $\{a_i\}_{i=1}^N$, $\{l_{ai}\}_{i=1}^N$, and $\{u_{ai}\}_{i=1}^N$;
Output: A global minimizer $t_1{}'$ and the minimum value $\overline{U}$ of (1);
$\{\lambda_k\}_{k=1}^{3N} \leftarrow$ Sort $\{a_i\}_{i=1}^N \cup \{l_{ai}\}_{i=1}^N \cup \{u_{ai}\}_{i=1}^N$;
$U(\lambda_1) \leftarrow \sum_{i=1}^N \xi_{i1}$; $\overline{U} \leftarrow U(\lambda_1)$; // Use $\overline{U}$ to store the smallest upper bound so far.
$m_0^u \leftarrow 0$; $m_0^d \leftarrow 0$;
for $k \leftarrow 1$ **to** $3N-1$ **do**:
  if $\lambda_k \in \{a_i\}_{i=1}^N$ **then**
    $m_k^d \leftarrow m_{k-1}^d - 1$;
    $m_k^u \leftarrow m_{k-1}^u + 1$;
    if $U(\lambda_k) < \overline{U}$ **then** // Update $\overline{U}$ and the minimizer $t_1{}'$ once $U(\lambda_k)$ is smaller than $\overline{U}$.
      $t_1{}' \leftarrow \lambda_k$; $\overline{U} \leftarrow U(\lambda_k)$;
    end if
  else if $\lambda_k \in \{l_{ai}\}_{i=1}^N$ **then**
    $m_k^d \leftarrow m_{k-1}^d + 1$;
  else if $\lambda_k \in \{u_{ai}\}_{i=1}^N$ **then**
    $m_k^u \leftarrow m_{k-1}^u - 1$;
  end if
  $U(\lambda_{k+1}) \leftarrow U(\lambda_k) + (m_k^u - m_k^d)(\lambda_{k+1} - \lambda_k)$; // Compute $U(\lambda_{k+1})$ incrementally with the updated m_k^u and m_k^d based on (4).
end for

points $\{\lambda_k\}_{k=1}^{3N}$, therefore the derivative of $U(t_1)$ at this global minimizer is 0. Moreover, this derivative is 0 on the entire interval $\mathcal{I}_k$ and the objective value $U(t_1)$ is constant on $\mathcal{I}_k$. Thus, every point on $\mathcal{I}_k = (\lambda_k, \lambda_{k+1})$ is a global minimizer. Since $U(t_1)$ is continuous on $[\lambda_k, \lambda_{k+1}]$, has derivative 0 on $(\lambda_k, \lambda_{k+1})$, and every point on $(\lambda_k, \lambda_{k+1})$ is a global minimizer, we conclude that λ_k and λ_{k+1} are global minimizers of $U(t_1)$ as well. We have just shown that in the $3N$ points $\{\lambda_k\}_{k=1}^{3N}$, there is a globally optimal solution of (1).

Next, we take one step further and prove, in the N points $\{a_i\}_{i=1}^N$, there is a globally optimal solution of (1). Let us assume λ_{k+1} is a global minimizer of $U(t_1)$. We need to show $\lambda_{k+1} \in \{a_i\}_{i=1}^N$. For the sake of contradiction, assume $\lambda_{k+1} \in \{l_{ai}\}_{i=1}^N \cup \{u_{ai}\}_{i=1}^N$. Let us analyze the two intervals associated with λ_{k+1}, namely $\mathcal{I}_k = (\lambda_k, \lambda_{k+1})$ and $\mathcal{I}_{k+1} = (\lambda_{k+1}, \lambda_{k+2})$, and let us also analyze the derivatives of $U(t_1)$ on them. Since the derivatives of $U(t_1, i)$ on them are constant, let us denote by $U'(\mathcal{I}_{k+1}, i)$ and $U'(\mathcal{I}_k, i)$, respectively, the derivatives of $U(t_1, i)$ at some point of $\mathcal{I}_{k+1}$ and $\mathcal{I}_k$. Define $U'(\mathcal{I}_{k+1})$ and $U'(\mathcal{I}_k)$ similarly. Then we have the following observations:

- If $\lambda_{k+1} = l_{ai}$ for some i, then $U'(\mathcal{I}_k, i) = 0$ and $U'(\mathcal{I}_{k+1}, i) = -1$, *i.e.*, by going from $\mathcal{I}_k$ to $\mathcal{I}_{k+1}$, the i-th component function $U(t_1, i)$ enters the linear decrease regime from the constant regime. At the same time, we have $U'(\mathcal{I}_k, j) = U'(\mathcal{I}_{k+1}, j)$ for any $j \neq i$; in other words, $U(\mathcal{I}_k, j)$ keeps the same trend (linearly decrease, linearly increase, or constant). As a result, we have $U'(\mathcal{I}_k) = U'(\mathcal{I}_{k+1}) + 1$.
- If $\lambda_{k+1} = u_{ai}$ for some i, then $U'(\mathcal{I}_k, i) = 1$ and $U'(\mathcal{I}_{k+1}, i) = 0$. And we have $U'(\mathcal{I}_k, j) = U'(\mathcal{I}_{k+1}, j)$ for any $j \neq i$. Similarly we have $U'(\mathcal{I}_k) = U'(\mathcal{I}_{k+1}) + 1$.

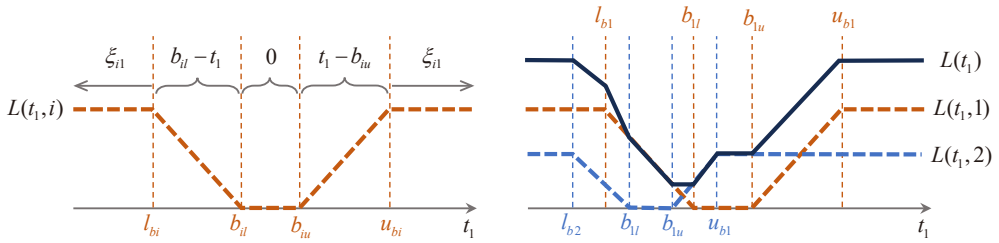
Since $U'(\mathcal{I}_k)$ and $U'(\mathcal{I}_{k+1})$ lie in $\{-N, \ldots, -1, 0, 1, \ldots, N\}$, the assumption $\lambda_{k+1} \in \{l_{ai}\}_{i=1}^N \cup \{u_{ai}\}_{i=1}^N$ leads us to the following cases:

- If $U'(\mathcal{I}_k) \geq 1$, then we have $U(\lambda_{k+1}) > U(\lambda_k)$.
- If $U'(\mathcal{I}_k) \leq 0$, then $U'(\mathcal{I}_{k+1}) \leq -1$ and we have $U(\lambda_{k+1}) > U(\lambda_{k+2})$.

We have shown $U(\lambda_{k+1}) > \min\{U(\lambda_k), U(\lambda_{k+2})\}$. This is a contradiction to the assumption $\lambda_{k+1} \in \{l_{ai}\}_{i=1}^N \cup \{u_{ai}\}_{i=1}^N$. Therefore, any global minimizer of $U(t_1)$ in $\{\lambda_k\}_{k=1}^{3N}$ must be a point of $\{a_i\}_{i=1}^N$. The proof is complete. □

In light of Proposition 1, an intuitive algorithm to solve (1) would be computing $U(a_i)$ for each $i \in \{1, \ldots, N\}$ separately; the solution is some a_i that gives the smallest objective value. This algorithm has $O(N^2)$ time complexity.

Let us show how to solve (1) in $O(N \log N)$ time. First recall that, on each interval $[\lambda_k, \lambda_{k+1}]$, $U(t_1, i)$ is monotonic or constant. Let us define $\mathcal{M}_k^u$ (*resp.* $\mathcal{M}_k^d$) to be the set of indices i for which $U(t_1, i)$ is increasing (*resp.* decreasing). Denote



(a) (b)

Figure 2. (a) Illustration of $L(t_1,i) = \min_{b_i\in[b_{il},b_{iu}]} L(t_1,b_i,i)$, where $L(t_1,b_i,i) = \min\{|b_i - t_1|,\ \xi_{i1}\}$; (b) Illustration of $L(t_1) = \sum_{i=1}^{N} L(t_1,i)$ when $N=2$. (*cf*. Appendix B.2).

by m_k^u and m_k^d the respective sizes of $\mathcal{M}_k^u$ and $\mathcal{M}_k^d$. Then we can compute the value of $U(\lambda_k)$ as follows:

$$U(\lambda_k) = \sum_{i\in\mathcal{M}_k^u} (\lambda_k - a_i) + \sum_{i\in\mathcal{M}_k^d} (a_i - \lambda_k) + \sum_{i\in\{1,\cdots,N\}\setminus\{\mathcal{M}_k^u\cup\mathcal{M}_k^d\}} \xi_{i1}. \tag{2}$$

An important observation is that, for every k, the value $U(\lambda_{k+1})$ can be computed incrementally from $U(\lambda_k)$:

$$\begin{aligned} U(\lambda_{k+1}) &= \sum_{i\in\mathcal{M}_k^u} (\lambda_{k+1} - a_i) + \sum_{i\in\mathcal{M}_k^d} (a_i - \lambda_{k+1}) + \sum_{i\in\{1,\cdots,N\}\setminus\{\mathcal{M}_k^u\cup\mathcal{M}_k^d\}} \xi_{i1} &(3)\\ &= U(\lambda_k) + (m_k^u - m_k^d)(\lambda_{k+1} - \lambda_k). &(4)\end{aligned}$$

Algorithm 2 implements the recurrence (4) and solves (1) to global optimality. Interestingly, this recurrent formulation does not depend on the index sets $\mathcal{M}_k^u$ and $\mathcal{M}_k^d$, so we only need to update m_k^u and m_k^d in Algorithm 2. Finally, since Algorithm 2 only consists of a sorting and scanning operation, its time complexity is $O(N\log N)$.

B.2. Lower Bound

Here we first consider the following proposition:

Proposition 2. *Given a branch* $[\alpha_1,\alpha_2]\times[\beta_1,\beta_1]$ *of* $\boldsymbol{r}_1$*, let* $[b_{il},b_{iu}]$ *denote the range of* $b_i := y_{i1} - \boldsymbol{r}_1{}^\top \boldsymbol{x}_i$*, specifically:*

$$b_{il} = \min\left\{y_{i1} - \boldsymbol{r}_1{}^\top \boldsymbol{x}_i : \boldsymbol{r}_1 = [\sin\beta\cos\alpha, \sin\beta\sin\alpha, \cos\beta],\ \alpha\in[\alpha_1,\alpha_2],\ \beta\in[\beta_1,\beta_2]\right\}, \tag{5}$$

$$b_{iu} = \max\left\{y_{i1} - \boldsymbol{r}_1{}^\top \boldsymbol{x}_i : \boldsymbol{r}_1 = [\sin\beta\cos\alpha, \sin\beta\sin\alpha, \cos\beta],\ \alpha\in[\alpha_1,\alpha_2],\ \beta\in[\beta_1,\beta_2]\right\}. \tag{6}$$

We can compute b_{il} *and* b_{iu} *in constant time.*

With Proposition 2 proved in Appendix F, we can compute the bounds $\{b_{il},b_{iu}\}_{i=1}^N$ in $O(N)$ time. Moreover, Proposition 2 provides the idea of relaxing (4) in the main manuscript into the following problem:

$$\begin{aligned} \underline{L} &= \min_{t_1\in\mathbb{R},\ b_i\in[b_{il},b_{iu}]} \sum_{i=1}^N \min\{|b_i - t_1|,\ \xi_{i1}\} \\ &= \min_{t_1\in\mathbb{R},\ b_i\in[b_{il},b_{iu}]} \sum_{i=1}^N L(t_1,b_i,i), \end{aligned} \tag{7}$$

where $L(t_1,b_i,i) := \min\{|b_i - t_1|,\ \xi_{i1}\}$. Note that for any given $t_1\in[b_{il},b_{iu}]$, we can always set b_i to t_1 so that $L(t_1,b_i,i)$ is minimized to 0. Now let us consider

$$\begin{aligned} L(t_1,i) &:= \min_{b_i\in[b_{il},b_{iu}]} L(t_1,b_i,i) \\ &= \begin{cases} b_{il} - t_1, & t_1\in[b_{il}-\xi_{i1}, b_{il}]; \\ 0, & t_1\in[b_{il},b_{iu}]; \\ t_1 - b_{iu}, & t_1\in[b_{iu},b_{iu}+\xi_{i1}]; \\ \xi_{i1}, & \text{otherwise.} \end{cases} \end{aligned} \tag{8}$$

Algorithm 3: Globally Optimal 1D TEAR Solver for Lower Bound Computation (Proposition 3).

Input: $\{b_{il}\}_{i=1}^N$, $\{b_{iu}\}_{i=1}^N$, $\{l_{bi}\}_{i=1}^N$, and $\{u_{bi}\}_{i=1}^N$;
Output: A global minimizer $\hat{t}_1$ and the minimum value $\underline{L}$ of (9);
$\{\psi_k\}_{k=1}^{4N} \leftarrow$ Sort $\{b_{il}\}_{i=1}^N \cup \{b_{iu}\}_{i=1}^N \cup \{l_{bi}\}_{i=1}^N \cup \{u_{bi}\}_{i=1}^N$;
$L(\psi_1) \leftarrow \sum_{i=1}^N \xi_{i1}$; $\underline{L} \leftarrow L(\psi_1)$; // Use $\underline{L}$ to store the smallest lower bound so far.
$n_0^u \leftarrow 0$; $n_0^d \leftarrow 0$;
for $k \leftarrow 1$ **to** $4N-1$ **do**:
  if $\psi_k \in \{b_{il}\}_{i=1}^N$ **then**
    $n_k^d \leftarrow n_{k-1}^d - 1$;
    if $L(\psi_k) < \underline{L}$ **then** // Update $\underline{L}$ and the minimizer $\hat{t}_1$ once $L(\psi_k)$ is smaller than $\underline{L}$.
      $\hat{t}_1 \leftarrow \psi_k$; $\underline{L} \leftarrow L(\psi_k)$;
    end if
  else if $\psi_k \in \{b_{iu}\}_{i=1}^N$ **then**
    $n_k^u \leftarrow n_{k-1}^u + 1$;
    if $L(\psi_k) < \underline{L}$ **then** // Similarly to Line 9, update $\underline{L}$ and the minimizer $\hat{t}_1$ once $L(\psi_k)$ is smaller than $\underline{L}$.
      $\hat{t}_1 \leftarrow \psi_k$; $\underline{L} \leftarrow L(\psi_k)$;
    end if
  else if $\psi_k \in \{l_{bi}\}_{i=1}^N$ **then**
    $n_{k+1}^d \leftarrow n_k^d + 1$;
  else if $\psi_k \in \{u_{bi}\}_{i=1}^N$ **then**
    $n_{k+1}^u \leftarrow n_k^u - 1$;
  end if
  $L(\psi_{k+1}) = L(\psi_k) + (n_k^u - n_k^d)(\psi_{k+1} - \psi_k)$; // Compute $L(\psi_{k+1})$ incrementally with the updated n_k^u and n_k^d based on (10).
end for

Define $l_{bi} := b_{il} - \xi_{i1}$ and $u_{bi} := b_{iu} + \xi_{i1}$. Note that $L(t_1, i)$ is differentiable at any point t_1 except when t_1 is equal to l_{bi}, b_{il}, b_{iu}, or u_{bi}. Furthermore, at differentiable points, the derivative of $L(t_1, i)$ is either 1 or -1 or 0 (see Fig. 2a). Now we can convert (7) into the following equivalent problem:

$$\begin{aligned} \underline{L} &= \min_{t_1 \in \mathbb{R}} \sum_{i=1}^N \min_{b_i \in [b_{il}, b_{iu}]} L(t_1, b_i, i) \\ &= \min_{t_1 \in \mathbb{R}} \sum_{i=1}^N L(t_1, i) \\ &= \min_{t_1 \in \mathbb{R}} L(t_1), \end{aligned} \tag{9}$$

where $L(t_1)$ is defined to be the sum of $L(t_1, i)$, i.e., $L(t_1) = \sum_{i=1}^N L(t_1, i)$. According to the derivation property of $L(t_1, i)$, the derivative of $L(t_1)$ at differentiable points lies in $\{-N, \ldots, -1, 0, 1, \ldots, N\}$. Based on the above notions, we have

Proposition 3. *In the $2N$ points $\{b_{il}\}_{i=1}^N \cup \{b_{iu}\}_{i=1}^N$, there is a globally optimal solution of (9). And (9) can be solved by Algorithm 3 in $O(N \log N)$ time.*

Proof. Let $\{\psi_k\}_{k=1}^{4N}$ be a sorted version of $\{b_{il}\}_{i=1}^N \cup \{b_{iu}\}_{i=1}^N \cup \{l_{bi}\}_{i=1}^N \cup \{u_{bi}\}_{i=1}^N$. For each $k = 1, \cdots, 4N$, define the open interval $\mathcal{J}_k := (\psi_k, \psi_{k+1})$. Without loss of generality, we assume $\psi_k \neq \psi_{k+1}$ and therefore $\mathcal{J}_k$ is not empty. Similarly to $U(t_1, i)$ in Appendix B.1, the derivative of $L(t_1, i)$ on every interval $\mathcal{J}_k$ is constant (say, 0, -1, or 1); therefore the derivative of $L(t_1)$ is also constant on every interval $\mathcal{J}_k$ similarly to $U(t_1)$ in Appendix B.1 (see Fig. 2b). Since $L(t_1)$ shares similar derivation properties with $U(t_1)$, we can easily infer that in the $4N$ endpoints $\{\psi_k\}_{k=1}^{4N}$, there is a globally optimal solution of (9).

In the following we take one step further and prove that in the $2N$ points $\{b_{il}\}_{i=1}^N \cup \{b_{iu}\}_{i=1}^N$, there is a globally optimal solution of (9). Assume that ψ_{k+1} is a global minimizer of $L(t_1)$. We need to show $\psi_{k+1} \notin \{l_{bi}\}_{i=1}^N \cup \{u_{bi}\}_{i=1}^N$. For

the sake of contradiction, suppose that $\psi_{k+1} \in \{l_{bi}\}_{i=1}^N \cup \{u_{bi}\}_{i=1}^N$. Consider the two intervals associated with ψ_{k+1}, namely $\mathcal{J}_k = (\psi_k, \psi_{k+1})$ and $\mathcal{J}_{k+1} = (\psi_{k+1}, \psi_{k+2})$. Since the derivatives of $L(t_1)$ on them are constant, let us denote by $L'(\mathcal{I}_{k+1})$ and $L'(\mathcal{I}_k)$, respectively, the derivatives of $L(t_1)$ at some point of $\mathcal{J}_{k+1}$ and $\mathcal{J}_k$. Then we can observe that, as long as ψ_{k+1} belongs to $\{l_{bi}\}_{i=1}^N$ or $\{u_{bi}\}_{i=1}^N$, it always holds that $L'(\mathcal{J}_k) = L'(\mathcal{J}_{k+1}) + 1$ (similarly to the equality $U'(\mathcal{I}_k) = U'(\mathcal{I}_{k+1})+1$ in Appendix B.1). Since $L'(\mathcal{I}_{k+1})$ and $L'(\mathcal{I}_k)$ also lie in $\{-N, \dots, -1, 0, 1, \dots, N\}$, the supposition $\psi_{k+1} \in \{l_{bi}\}_{i=1}^N \cup \{u_{bi}\}_{i=1}^N$ lead to the following cases:

- If $L'(\mathcal{J}_k) \geq 1$, then we have $L(\psi_{k+1}) > L(\psi_k)$.
- If $L'(\mathcal{J}_k) \leq 0$, then $L'(\mathcal{I}_{k+1}) \leq -1$ and we have $L(\psi_{k+1}) > L(\psi_{k+2})$.

We have shown $L(\psi_{k+1}) > \min\{L(\psi_k), L(\psi_{k+2})\}$. This is a contradiction to the assumption that ψ_{k+1} is a global minimizer. Therefore, any global minimizer of $L(t_1)$ in $\{\psi_k\}_{k=1}^{4N}$ must be a point of $\{b_{il}\}_{i=1}^N \cup \{b_{iu}\}_{i=1}^N$.

Now, to find a global minimizer of (9), it remains to evaluate the objective value of (9) at all points $\{b_{il}\}_{i=1}^N \cup \{b_{iu}\}_{i=1}^N$ and pick one point that yields the minimum objective. Similarly to the computation of $U(\lambda_k)$ in (4), we can compute $L(\psi_k)$ incrementally using the formula

$$L(\psi_{k+1}) = L(\psi_k) + (n_k^u - n_k^d)(\psi_{k+1} - \psi_k), \tag{10}$$

where n_k^u and n_k^d represent the numbers of increasing and decreasing terms of $\{L(t_1, i)\}_{i=1}^N$ at $\mathcal{J}_k$, respectively. The details are in Algorithm 3. In particular, Algorithm 3 iterates over $\{\psi_k\}_{k=1}^{4N}$ to update n_k^u and n_k^d, so as to compute $L(\psi_{k+1})$ based on (10). At each iteration we can update $\underline{L}$ by comparing $L(\psi_{k+1})$ and $\underline{L}$, and when the loop finishes, the minimum value $\underline{L}$ is discovered, associated with the optimal solution stored in $\hat{t}_1$. □

C. Bounds for TEAR-2

In this section, we show that we can conduct a simple reparameterization on TEAR-2, so as to solve it using similar bound computation methods that we develop for TEAR-1 (see Appendix B). Note that the extra constraint $\boldsymbol{r}_2^\top \hat{\boldsymbol{r}}_1 = 0$ in TEAR-2 restricts the $\boldsymbol{r}_2$ to be determined by only 1 degree of freedom. Then we can substitute the decision variable $\boldsymbol{r}_2$ in TEAR-2 with the variable $\alpha \in [0, 2\pi)$ in the constraint $\boldsymbol{r}_2 = [\sin\beta\cos\alpha, \sin\beta\sin\alpha, \cos\beta]^\top \in \mathbb{S}^2$, i.e.,

$$\begin{aligned} \min_{\alpha\in[0,2\pi), t_2\in\mathbb{R}} \ & \sum_{i\in\hat{\mathcal{I}}_1} \min\{|y_{i2} - \boldsymbol{r}_2^\top \boldsymbol{x}_i - t_2|,\ \xi_{i2}\} \\ \text{s.t.} \quad & \boldsymbol{r}_2^\top \hat{\boldsymbol{r}}_1 = 0 \\ & \boldsymbol{r}_2 = [\sin\beta\cos\alpha, \sin\beta\sin\alpha, \cos\beta]^\top \end{aligned} \tag{11}$$

Therefore, we can branch over the 1-dimension region $[0, 2\pi)$ to search for the globally optimal solution. Specifically, given a branch $[\alpha_1, \alpha_2]$, we consider the following upper and lower bounds:

Upper Bound. We choose the center $\dot{\alpha}$ of $[\alpha_1, \alpha_2]$ to compute an upper bound. Then the $\boldsymbol{r}_2$ can be solved by combining the constraints in (11) and we denote it by $\dot{\boldsymbol{r}}_2$. Let $a_{i2} := y_{i2} - \dot{\boldsymbol{r}}_2{}^\top \boldsymbol{x}_i$, the upper bound can be computed as follows:

$$\min_{t_2\in\mathbb{R}} \sum_{i\in\hat{\mathcal{I}}_1} \min\{|a_{i2} - t_2|,\ \xi_{i2}\}. \tag{12}$$

Obviously, (12) shares the same problem structure with (1). Therefore we can directly call Algorithm 2 to get a globally optimal solution of (12).

Lower Bound. Based on the constraints on $\boldsymbol{r}_2$ in (11), it is clear that β is restricted to an available range on the given branch $[\alpha_1, \alpha_2]$. Let $[\beta_1, \beta_2]$ denotes this range and $b_{i2} := y_{i2} - \boldsymbol{r}_2^\top \boldsymbol{x}_i$, we can compute the range $[b_{i2}^l, b_{i2}^u]$ of b_{i2} based on Proposition 2. Then it suffices to relax (11) into the following problem for a lower bound:

$$\min_{t_2\in\mathbb{R}, b_{i2}\in[b_{i2}^l, b_{i2}^u]} \sum_{i\in\hat{\mathcal{I}}_1} \min\{|b_{i2} - t_2|,\ \xi_{i2}\}. \tag{13}$$

Note that (13) and (7) share the same structure, therefore we can adopt Algorithm 3 to solve (13) with global optimality.

Algorithm 4: Interval Stabbing (Theorem 1).

Input: $\mathcal{Z} = \{[z_{pl},\ z_{pu}]\}_{p=1}^{P}$;
Output: Best stabber z^* and the corresponding number of stabbed intervals T^*;
$\hat{\mathcal{Z}} \leftarrow$ Sort all the endpoints in $\mathcal{Z}$;
$count \leftarrow 0$; $T^* \leftarrow count$;
for $p \leftarrow 1$ to $2P$ **do**:
  if $\hat{\mathcal{Z}}(p)$ is a left endpoint **then**
    $count \leftarrow count + 1$; // If the current stabber moves to a left endpoint, one more interval is stabbed.
    if $count > T^*$ **do**: // Update T^* and the best stabber z^* once the current number of stabbed intervals $count$ is larger than T^*.
      $z^* \leftarrow \hat{\mathcal{Z}}(p)$; $T^* \leftarrow count$;
    end if
  else
    $count \leftarrow count - 1$; // If the current stabber moves to a right endpoint, one less interval is stabbed.
  end if
end for

D. Bounds for CM-1

Before introducing the bounds for CM-1, we first present a relevant and widely used algorithm, called *interval stabbing*.

Theorem 1. ***(Interval Stabbing)*** *Given a set of intervals $\mathcal{Z} = \{[z_{pl}, z_{pu}]\}_{p=1}^{P}$, consider to find some stabber z that interacts with the most number of intervals, i.e.,*

$$\max_{z\in\mathbb{R}} \quad \sum_{p=1}^{P} \mathbf{1}\left(z \in [z_{pl}, z_{pu}]\right), \tag{14}$$

where $\mathbf{1}(\cdot)$ denotes the indicator function. Then Algorithm 4 solves (14) in $O(P\log P)$ time.

D.1. Upper Bound

Given a branch $[\underline{\boldsymbol{r}_1}, \overline{\boldsymbol{r}_1}]$ of $\boldsymbol{r}_1$, *e.g.*, $[\alpha_1, \alpha_2] \times [\beta_1, \beta_1]$, we define $b_i := y_{i1} - \boldsymbol{r}_1^\top \boldsymbol{x}_i$ as Appendix B.2. Based on Proposition 2, the range $[b_{il}, b_{iu}]$ of b_i can be easily computed. Moreover, we have the following proposition:

Proposition 4. *We can compute an upper bound $\overline{U}_{CM}$ of CM-1 via solving*

$$\overline{U}_{CM} = \max_{t_1\in\mathbb{R}} \sum_{i=1}^{N} \mathbf{1}\left(t_j \in [b_{il} - \xi_{i1}, b_{iu} + \xi_{i1}]\right). \tag{15}$$

Specifically, $\overline{U}_{CM}$ can be computed in $O(N\log N)$ time by Algorithm 4.

Proof. We first consider CM-1 constrained on the branch $[\underline{\boldsymbol{r}_1},\ \overline{\boldsymbol{r}_1}]$, that is,

$$\max_{t_1\in\mathbb{R}} \max_{\boldsymbol{r}_1\in[\underline{\boldsymbol{r}_1},\ \overline{\boldsymbol{r}_1}]} \sum_{i=1}^{N} \mathbf{1}\left(|y_{i1} - \boldsymbol{r}_1^\top \boldsymbol{x}_i - t_1| \le \xi_{i1}\right). \tag{16}$$

To compute an upper bound of (16), we can relax it into

$$\max_{t_1\in\mathbb{R}} \sum_{i=1}^{N} \max_{\boldsymbol{r}_1\in[\underline{\boldsymbol{r}_1},\ \overline{\boldsymbol{r}_1}]} \mathbf{1}\left(|y_{i1} - \boldsymbol{r}_1^\top \boldsymbol{x}_i - t_1| \le \xi_{i1}\right). \tag{17}$$

With the notation $b_i := y_{i1} - \boldsymbol{r}_1{}^\top \boldsymbol{x}_i$ and the bounds b_{il} and b_{iu} in Proposition 2, we have

$$\begin{aligned}
&\max_{t_1\in\mathbb{R}} \sum_{i=1}^{N} \max_{\boldsymbol{r}_1\in[\underline{\boldsymbol{r}_1},\,\overline{\boldsymbol{r}_1}]} \mathbf{1}\left(|y_{i1} - \boldsymbol{r}_1^\top \boldsymbol{x}_i - t_1| \le \xi_{i1}\right) \\
=&\max_{t_1\in\mathbb{R}} \sum_{i=1}^{N} \max_{b_i\in[b_{il},b_{iu}]} \mathbf{1}\left(b_i - \xi_{i1} \le t_1 \le b_i + \xi_{i1}\right) &\text{(18a)}\\
\le&\max_{t_1\in\mathbb{R}} \sum_{i=1}^{N} \mathbf{1}\left(b_{il} - \xi_{i1} \le t_1 \le b_{iu} + \xi_{i1}\right) &\text{(18b)}\\
=&\max_{t_1\in\mathbb{R}} \sum_{i=1}^{N} \mathbf{1}\left(t_1 \in [b_{il} - \xi_{i1},\ b_{iu} + \xi_{i1}]\right) = \overline{U}_{CM}. &\text{(18c)}
\end{aligned}$$

Thus $\overline{U}_{CM}$ is indeed an upper bound of CM-1. Note that (18c) is exactly an interval stabbing problem as (14), therefore $\overline{U}_{CM}$ can be solved in $O(N\log N)$ time by Algorithm 4. □

D.2. Lower Bound

We choose the center $\dot{\boldsymbol{r}}_1$ of the given branch $[\underline{\boldsymbol{r}_1}, \overline{\boldsymbol{r}_1}]$ to compute a lower bound $\underline{L}_{CM}$ of CM-1, *i.e.*,

$$\underline{L}_{CM} = \max_{t_1\in\mathbb{R}} \sum_{i=1}^{N} \mathbf{1}\left(|y_{i1} - \dot{\boldsymbol{r}}_1^\top \boldsymbol{x}_i - t_1| \le \xi_{i1}\right). \tag{19}$$

Define $a_i := y_{i1} - \dot{\boldsymbol{r}}_1^\top \boldsymbol{x}_i$, $l_{ai} := a_1 - \xi_{i1}$, and $u_{ai} := a_1 + \xi_{i1}$. As in Appendix D.1, let us rewrite (19) as follows:

$$\begin{aligned}
\underline{L}_{CM} &= \max_{t_1\in\mathbb{R}} \sum_{i=1}^{N} \mathbf{1}\left(a_i - \xi_{i1} \le t_1 \le a_i + \xi_{i1}\right) &\text{(20a)}\\
&= \max_{t_1\in\mathbb{R}} \sum_{i=1}^{N} \mathbf{1}\left(y_{i1} - l_{ai} \le t_1 \le u_{ai}\right) &\text{(20b)}\\
&= \max_{t_1\in\mathbb{R}} \sum_{i=1}^{N} \mathbf{1}\left(t_i \in [l_{ai},\ u_{ai}]\right). &\text{(20c)}
\end{aligned}$$

We have transformed (19) into an interval stabbing problem (20c). Therefore we can compute the lower bound $\underline{L}_{CM}$ in $O(N\log N)$ time by Algorithm 4.

E. Bounds for TLS-1

E.1. Upper Bound

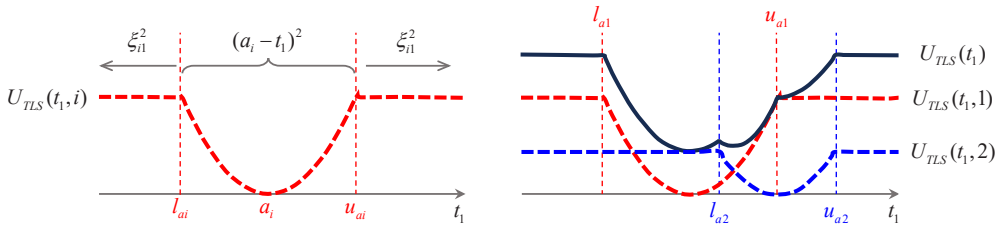
Similarly to Appendix B.1, we choose the center $\dot{\boldsymbol{r}}_1$ of the given branch $[\underline{\boldsymbol{r}_1}, \overline{\boldsymbol{r}_1}]$ to compute an upper bound $\overline{U}_{TLS}$ of this branch. Let $a_i := y_{i1} - \dot{\boldsymbol{r}}_1^\top \boldsymbol{x}_i$ as Appendix B.1 and Appendix D.2, we have

$$\begin{aligned}
\overline{U}_{TLS} &= \min_{t_1\in\mathbb{R}} \sum_{i=1}^{N} \min\{(a_i - t_1)^2,\ \xi_{i1}^2\} \\
&= \min_{t_1\in\mathbb{R}} U_{TLS}(t_1),
\end{aligned} \tag{21}$$

where $U_{TLS}(t_1)$ is defined to be the sum of $U_{TLS}(t_1, i) := \min\{(a_i - t_1)^2,\ \xi_{i1}^2\}$, that is $U_{TLS}(t_1) := \sum_{i=1}^{N} U_{TLS}(t_1, i))$. We begin by the following auxiliary result (easy to prove):

Theorem 2. *Consider the following problem:*

$$\min_{d\in[d_l,\,d_u]} \sum_{q=1}^{Q} (\gamma_q - d)^2. \tag{22}$$



(a) (b)

Figure 3. (a) Illustration of $U_{TLS}(t_1, i) = \min\{(a_i - t_1)^2,\ \xi_{i1}^2\}$; (b) Illustration of $U_{TLS}(t_1) = \sum_{i=1}^{N} U_{TLS}(t_1, i)$ when $N = 2$. (*cf.* Appendix E.1).

Let $\tilde{d} := \frac{1}{Q}\sum_{q=1}^{Q} \gamma_q$, *then the global optimizer* $\hat{d}$ *to* (22) *is given as follows:*

$$\hat{d} = \begin{cases} \tilde{d}, & \tilde{d} \in [d_l,\ d_u]; \\ d_u, & \tilde{d} > d_u; \\ d_l, & \tilde{d} < d_l. \end{cases} \tag{23}$$

Let us now consider computing $\overline{U}_{TLS}$ in (21). Define $l_{ai} := a_i - \xi_{i1}$, $u_{ai} := a_i + \xi_{i1}$. Let $\{\lambda\}_{k=1}^{2N}$ be a sorted version of the $2N$ numbers $\{l_{ai}\}_{i=1}^{N} \cup \{u_{ai}\}_{i=1}^{N}$. On each interval $\mathcal{I}_k := [\lambda_k, \lambda_{k+1}]$, it is clear that $U_{TLS}(t_1, i)$ can be only the part of *quadratic* $(a_i - t_1)^2$ or the *constant* ξ_{i1}^2 (see Fig. 3b).

Based on Theorem 2 and the above notation, we develop Algorithm 5 to get the globally optimal solution of (21) in $O(N \log N)$ time. Algorithm 5 iterates over $\{\lambda_k\}_{k=1}^{2N}$ and computes the minimal value U_k of $U_{TLS}(t_1)$ on each interval $\mathcal{I}_k$. Obviously the minimum U_k in $\{U_k\}_{k=1}^{2N-1}$ is exactly the minimum value $\overline{U}_{TLS}$, and next we will show that it can be computed incrementally during the iteration. Define U_k^q to be the minimal value of the sum of $U_{TLS}(t_1, i)$s being *quadratic* on each interval $\mathcal{I}_k$, as well as U_k^c to be sum of $U_{TLS}(t_1, i)$s being *constant*. It is clear that $U_k = U_k^q + U_k^c$. Based on Theorem 2, we have

$$\begin{aligned} U_k^q &= \sum_{m \in \mathcal{M}_k^q} (a_m - t_{1k})^2 \\ &= \sum_{m \in \mathcal{M}_k^q} {a_m}^2 + M_k^q \cdot {t_{1k}}^2 - 2t_{1k} \cdot M_k^q \cdot \overline{a_k}, \end{aligned} \tag{24}$$

where $\mathcal{M}_k^q$ is defined to be the set of indices of $U_{TLS}(t_1, i)$s being *quadratic* on the interval $\mathcal{I}_k$, t_{1k} is the corresponding value of t_1 leading to U_k^q in $\mathcal{I}_k$ (that is, the $\hat{d}$ given by Theorem 2), M_k^q is the set size of $\mathcal{M}_k^q$, and $\overline{a_k}$ is the average value of related a_is in $\mathcal{M}_k^q$. At each iteration, the index set $\mathcal{M}_k^q$ differs from $\mathcal{M}_{k-1}^q$ by at most one element. More precisely:

- If $\lambda_k \in \{l_{ai}\}_{i=1}^{N}$, then $\mathcal{M}_k^q$ has one more index i than $\mathcal{M}_{k-1}^q$, and for this index i, the component $U_{TLS}(t_1, i)$ is quadratic on interval $\mathcal{I}_k$. In this case, we say *we enter a quadratic regime*.
- If $\lambda_k \in \{u_{ai}\}_{i=1}^{N}$, then $\mathcal{M}_k^q$ has one less index i than $\mathcal{M}_{k-1}^q$, and for this index i, the component $U_{TLS}(t_1, i)$ is constant (or no longer quadratic) on interval $\mathcal{I}_k$. In this case, we say *we leave a quadratic regime*.

In either case, we can update U_k^q from U_{k-1}^q as follows:

$$\overline{a_k} = \begin{cases} \frac{M_{k-1}^q}{M_{k-1}^q+1} \cdot \overline{a_{k-1}} + \frac{1}{M_{k-1}^q+1} \cdot a_k, & \lambda_k \in \{l_{ai}\}_{i=1}^{N}; \\ \frac{M_{k-1}^q}{M_{k-1}^q-1} \cdot \overline{a_{k-1}} - \frac{1}{M_{k-1}^q-1} \cdot a_k, & \lambda_k \in \{u_{ai}\}_{i=1}^{N}; \end{cases} \tag{25a}$$

$$U_k^q - U_{k-1}^q = \begin{cases} {a_k}^2 + M_k^q \cdot t_{1k} \cdot (t_{1k} - 2\overline{a_k}) - M_{k-1}^q \cdot t_{1(k-1)} \cdot (t_{1(k-1)} - 2\overline{a_{k-1}}), & \lambda_k \in \{l_{ai}\}_{i=1}^{N}; \\ -{a_k}^2 + M_k^q \cdot t_{1k} \cdot (t_{1k} - 2\overline{a_k}) - M_{k-1}^q \cdot t_{1(k-1)} \cdot (t_{1(k-1)} - 2\overline{a_{k-1}}), & \lambda_k \in \{u_{ai}\}_{i=1}^{N}. \end{cases} \tag{25b}$$

Based on (25b), we can easily compute U_k^q in constant time (Lines 10-14 and 18-22) at each iteration. As to U_k^c, at each iteration it also only differ with U_{k-1}^c from one element, that is, containing one less *constant* term when λ_k belongs to $\{l_{ai}\}_{i=1}^{N}$, or containing one more *constant* term when λ_k belongs to $\{u_{ai}\}_{i=1}^{N}$. Therefore U_k^c can be also incrementally computed by adding or subtracting the threshold term at each iteration (Lines 15 and 23). Then U_k can be computed by

Algorithm 5: Globally Optimal 1D TLS Solver for Upper Bound Computation (21).

1 **Input:** $\{l_{ai}\}_{i=1}^{N}$, $\{u_{ai}\}_{i=1}^{N}$, and $\{a_i\}_{i=1}^{N}$;
2 **Output:** A global minimizer $t_1{}'$ and the minimum value $\overline{U}_{TLS}$ of (21);
3 $\{\lambda_k\}_{k=1}^{2N} \leftarrow$ Sort $\{l_{ai}\}_{i=1}^{N} \cup \{u_{ai}\}_{i=1}^{N}$;
4 $U_0 \leftarrow \sum_{i=1}^{N} \xi_{i1}^2$; $\overline{U}_{TLS} \leftarrow U_0$;
5 $U_0^q \leftarrow 0$; $M_0^q \leftarrow 0$; $\overline{a_0} \leftarrow 0$; $U_0^c \leftarrow U_0$;
6 **for** $k \leftarrow 1$ **to** $2N-1$ **do**:
7 $\mathcal{I}_k \leftarrow [\lambda_k,\ \lambda_{k+1}]$;
8 **if** $\lambda_k \in \{l_{ai}\}_{i=1}^{N}$ **then**
9 $M_k^q \leftarrow M_{k-1}^q + 1$; // Since $\lambda_k \in \{l_{ai}\}_{i=1}^{N}$, we enter a *quadratic* regime.
10 $k_{in} \leftarrow$ Index of λ_k in $\{l_{ai}\}_{i=1}^{N}$;
11 $a_k \leftarrow k_{in}^{th}$ element in $\{a_i\}_{i=1}^{N}$;
12 $\overline{a_k} \leftarrow \frac{M_{k-1}^q}{M_k^q} \cdot \overline{a_{k-1}} + \frac{1}{M_k^q} \cdot a_k$;
13 $t_{1k} \leftarrow$ Compare $\overline{a_k}$ with $\mathcal{I}_k$ based on (23); // Compute the minimizer t_{1k} on the interval $\mathcal{I}_k$ (see Theorem 2).
14 $U_k^q \leftarrow U_{k-1}^q + {a_k}^2 + M_k^q \cdot t_{1k} \cdot (t_{1k} - 2\overline{a_k}) - M_{k-1}^q \cdot t_{1(k-1)} \cdot (t_{1(k-1)} - 2\overline{a_{k-1}})$;
15 $U_k^c \leftarrow U_{k-1}^c - \xi_{k_{in}1}^2$;
16 **else if** $\lambda_k \in \{u_{ai}\}_{i=1}^{N}$ **then**
17 $M_k^q \leftarrow M_{k-1}^q - 1$; // Since $\lambda_k \{u_{ai}\}_{i=1}^{N}$, we leave a *quadratic* regime.
18 $k_{out} \leftarrow$ Index of λ_k in $\{u_{ai}\}_{i=1}^{N}$;
19 $a_k \leftarrow k_{out}^{th}$ element in $\{a_i\}_{i=1}^{N}$;
20 $\overline{a_k} \leftarrow \frac{M_{k-1}^q}{M_k^q} \cdot \overline{a_{k-1}} - \frac{1}{M_k^q} \cdot a_k$;
21 $t_{1k} \leftarrow$ Compare $\overline{a_k}$ with $\mathcal{I}_k$ based on (23); // Compute the minimizer t_{1k} on th interval $\mathcal{I}_k$ (similarly to Line 13).
22 $U_k^q \leftarrow U_{k-1}^q - {a_k}^2 + M_k^q \cdot t_{1k} \cdot (t_{1k} - 2\overline{a_k}) - M_{k-1}^q \cdot t_{1(k-1)} \cdot (t_{1(k-1)} - 2\overline{a_{k-1}})$;
23 $U_k^c \leftarrow U_{k-1}^c + \xi_{k_{out}1}^2$;
24 **end if**
25 $U_k \leftarrow U_k^q + U_k^c$;
26 **if** $U_k < \overline{U}_{TLS}$ **then** // Update $\overline{U}_{TLS}$ and the global minimizer $t_1{}'$ once U_k is smaller than $\overline{U}_{TLS}$.
27 $t_1{}' \leftarrow t_{1k}$; $\overline{U}_{TLS} \leftarrow U_k$;
28 **end if**
29 **end for**

summing U_k^q and U_k^c at each iteration. Since the sorting operation leads to $O(N \log N)$ time complexity and the iteration leads to $O(N)$ time complexity, the $\overline{U}_{TLS}$ can be computed in $O(N \log N)$ time by the proposed Algorithm 5. In addition, based on the comparison at each iteration (Lines 26-27), Algorithm 5 can solve (21) to global optimality.

E.2. Lower Bound

Let $b_i := y_{i1} - \boldsymbol{r}_1{}^\top \boldsymbol{x}_i$ as Appendix B.2 and Appendix D.1, then the range $[b_{il}, b_{iu}]$ of b_i in the given branch can be solved by using Proposition 2. To compute a lower bound, it suffices to relax TLS-1 into the following problem:

$$\begin{aligned} \underline{L}_{TLS} &= \min_{t_1 \in \mathbb{R},\ b_i \in [b_{il}, b_{iu}]} \sum_{i=1}^{N} \min\{(b_i - t_1)^2,\ \xi_{i1}^2\} \\ &= \min_{t_1 \in \mathbb{R},\ b_i \in [b_{il}, b_{iu}]} \sum_{i=1}^{N} L_{TLS}(t_1, b_i, i), \end{aligned} \tag{26}$$

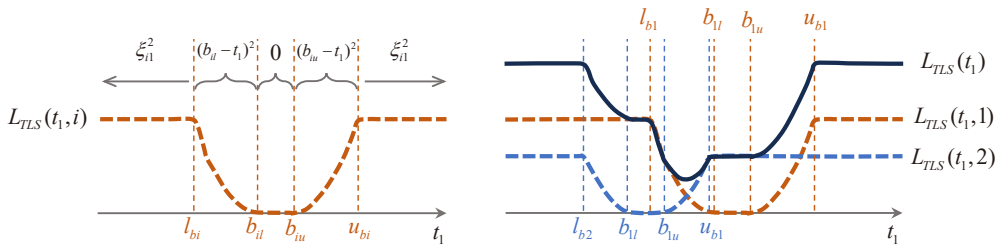


Figure 4. (a) Illustration of $L_{TLS}(t_1,i)=\min_{b_i\in[b_{il},b_{iu}]}L_{TLS}(t_1,b_i,i)$, where $L_{TLS}(t_1,b_i,i)=\min\{(b_i-t_1)^2,\ \xi_{i1}^2\}$; (b) Illustration of $L_{TLS}(t_1)=\sum_{i=1}^N L_{TLS}(t_1,i)$ when $N=2$. (*cf*. Appendix E.2).

where $L_{TLS}(t_1,b_i,i):=\min\{(b_i-t_1)^2,\ \xi_{i1}^2\}$. Note that $\forall t_1\in[b_{il},b_{iu}]$, b_i could be set by t_1 so as to minimize $L_{TLS}(t_1,b_i,i)$, accordingly we define

$$\begin{aligned}L_{TLS}(t_1,i)&=\min_{b_i\in[b_{il},b_{iu}]}L_{TLS}(t_1,b_i,i)\\&=\begin{cases}(b_{il}-t_1)^2, & t_i\in[b_{il}-\xi_{i1},\ b_{il}];\\ 0, & t_i\in[b_{il},\ b_{iu}];\\ (b_{iu}-t_1)^2, & t_i\in[b_{iu},\ b_{iu}+\xi_{i1}];\\ \xi_{i1}^2, & \text{otherwise.}\end{cases}\end{aligned}\tag{27}$$

Then (26) can be transformed equivalently to:

$$\begin{aligned}\underline{L}_{TLS}&=\min_{t_1\in\mathbb{R}}\sum_{i=1}^N\min_{b_i\in[b_{il},b_{iu}]}L_{TLS}(t_1,b_i,i)\\&=\min_{t_1\in\mathbb{R}}\sum_{i=1}^N L_{TLS}(t_1,i)\\&=\min_{t_1\in\mathbb{R}}L_{TLS}(t_1),\end{aligned}\tag{28}$$

where $L_{TLS}(t_1)$ is defined to be the sum of $L_{TLS}(t_1,i)$, that is $L_{TLS}(t_1):=\sum_{i=1}^N L_{TLS}(t_1,i)$. Define $l_{bi}:=b_{il}-\xi_{i1}$, $u_{bi}:=b_{iu}+\xi_{i1}$, and $L_{TLS}(t_1):=\sum_{i=1}^N L_{TLS}(t_1,i)$. Let $\{\psi_k\}_{k=1}^{4N}$ be a sorted version of $\{b_{il}\}_{i=1}^N\cup\{b_{iu}\}_{i=1}^N\cup\{l_{bi}\}_{i=1}^N\cup\{u_{bi}\}_{i=1}^N$. Then on each interval $\mathcal{J}_k:=[\psi_k,\psi_{k+1}]$, it is obvious that $L_{TLS}(t_1,i)$ can be only one the *two quadratics* $\{(b_{il}-t_1)^2,\ (b_{iu}-t_1)^2\}$ or one of the *two constants* $\{0,\ \xi_{i1}^2\}$ (see Fig. 4). In the following, we develop Algorithm 6 to get the globally optimal solution of (28) with $O(N^2)$ time complexity based on the above notions.

Specifically, we iterate over $\{\psi_k\}_{k=1}^{4N}$ to compute the minimal value L_k of $L_{TLS}(t_1)$ on each interval $\mathcal{J}_k$, and the minimum one in $\{L_k\}_{k=1}^{4N-1}$ is exactly the $\underline{L}_{TLS}$. On each $\mathcal{J}_k$, we define a set $\mathcal{N}_k^q$ to store the indices of $L_{TLS}(t_1,i)$s being *quadratic*, as well as a variable L_k^c to store the sum of $L_{TLS}(t_1,i)$s being *constant*. At each iteration, the index set $\mathcal{N}_k^q$ differ from $\mathcal{N}_{k-1}^q$ by at most one element and the variable L_k^c can be updated from L_{k-1}^c. Suppose k' is the index of ψ_k in one of the sets $\{l_{bi}\}_{i=1}^N/\{b_{il}\}_{i=1}^N/\{b_{iu}\}_{i=1}^N/\{u_{bi}\}_{i=1}^N$, we have

- If $\psi_k\in\{l_{bi}\}_{i=1}^N$, then $\mathcal{N}_k^q$ has one more index k' than $\mathcal{N}_{k-1}^q$, and L_k^c has one less constant term $\xi_{k'}^2$ than $L_{k'}^c$. In this case, similarly to Appendix E.1, we say *we enter a quadratic regime*.
- If $\psi_k\in\{b_{il}\}_{i=1}^N$, then $\mathcal{N}_k^q$ has one less index k' than $\mathcal{N}_{k-1}^q$, and L_k^c equals to L_{k-1}^c. In this case, similarly to Appendix E.1, we say *we leave a quadratic regime*.
- If $\psi_k\in\{b_{iu}\}_{i=1}^N$, then $\mathcal{N}_k^q$ has one more index k' than $\mathcal{N}_{k-1}^q$, and L_k^c equals to L_{k-1}^c. In this case, *we enter a quadratic regime*.
- If $\psi_k\in\{u_{bi}\}_{i=1}^N$, then $\mathcal{N}_k^q$ has one less index k' than $\mathcal{N}_{k-1}^q$, and L_k^c has one more constant term $\xi_{k'}^2$ than $L_{k'}^c$. In this case, *we leave a quadratic regime*.

In either cases, $\mathcal{N}_k^q$ and L_k^c can be updated. Then we can compute the minimal value L_k^q of the sum of $L_{TLS}(t_1,i)$s that $i\in\mathcal{N}_k^q$ (*quadratic* terms) based on Theorem 2 with at most $O(N)$ time complexity, and furthermore the minimal value

Algorithm 6: Globally Optimal 1D TLS Solver for Lower Bound Computation (28).

1 **Input:** $\{b_{il}\}_{i=1}^{N}$, $\{b_{iu}\}_{i=1}^{N}$,$\{l_{bi}\}_{i=1}^{N}$, and $\{u_{bi}\}_{i=1}^{N}$;
2 **Output:** A global minimizer $\hat{t}_1$ and the minimum value $\underline{L}_{TLS}$ of (28);
3 $\{\psi_k\}_{k=1}^{4N} \leftarrow$ Sort $\{b_{il}\}_{i=1}^{N} \cup \{b_{iu}\}_{i=1}^{N} \cup \{l_{bi}\}_{i=1}^{N} \cup \{u_{bi}\}_{i=1}^{N}$;
4 $L_0 \leftarrow \sum_{i=1}^{N} \xi_{i1}^2$; $\underline{L}_{TLS} \leftarrow L_0$;
5 $\mathcal{N}_0^q \leftarrow \emptyset$; $L_0^c \leftarrow L_0$;
6 **for** $k \leftarrow 1$ **to** $4N-1$ **do**:
7 $\mathcal{J}_k = [\lambda_k,\ \lambda_{k+1}]$;
8 **if** $\psi_k \in \{l_{bi}\}_{i=1}^{N}$ **then**
9 $k' \leftarrow$ Index of ψ_k in $\{l_{bi}\}_{i=1}^{N}$;
10 $\mathcal{N}_k^q \leftarrow \mathcal{N}_{k-1}^q \cup \{k'\}$; // Since $\psi_k \in \{l_{bi}\}_{i=1}^{N}$, we enter a *quadratic regime*.
11 $L_k^c \leftarrow L_{k-1}^c - \xi_{k_{in}1}^2$;
12 **else if** $\psi_k \in \{b_{il}\}_{i=1}^{N}$ **then**
13 $k' \leftarrow$ Index of ψ_k in $\{b_{il}\}_{i=1}^{N}$;
14 $\mathcal{N}_k^q \leftarrow \mathcal{N}_{k-1}^q \setminus \{k'\}$; // Since $\psi_k \in \{b_{il}\}_{i=1}^{N}$, we leave a *quadratic regime*.
15 $L_k^c \leftarrow L_{k-1}^c$;
16 **else if** $\psi_k \in \{b_{iu}\}_{i=1}^{N}$ **then**
17 $k' \leftarrow$ Index of ψ_k in $\{b_{iu}\}_{i=1}^{N}$;
18 $\mathcal{N}_k^q \leftarrow \mathcal{N}_{k-1}^q \cup \{k'\}$; // Since $\psi_k \in \{b_{iu}\}_{i=1}^{N}$, we enter a *quadratic regime*.
19 $L_k^c \leftarrow L_{k-1}^c$;
20 **else if** $\psi_k \in \{u_{bi}\}_{i=1}^{N}$ **then**
21 $k' \leftarrow$ Index of ψ_k in $\{u_{bi}\}_{i=1}^{N}$;
22 $\mathcal{N}_k^q \leftarrow \mathcal{N}_{k-1}^q \setminus \{k'\}$; // Since $\psi_k \in \{u_{bi}\}_{i=1}^{N}$, we leave a *quadratic regime*.
23 $L_k^c \leftarrow L_{k-1}^c + \xi_{k_{out}1}^2$;
24 **end if**
25 $\hat{t}_{1k},\ L_k^q \leftarrow \min_{t_{1k} \in \mathcal{J}_k} \sum_{n \in \mathcal{N}^q} (a_n - t_{1k})^2$ based on Theorem 2; // Compute the minimizer $\hat{t}_{1k}$ and minimal value L_k^q of the sum of *quadratic* terms on the interval $\mathcal{J}_k$ (see Theorem 2).
26 $L_k \leftarrow L_k^q + L_k^c$;
27 **if** $L_k < \underline{L}_{TLS}$ **then** // Update $\underline{L}_{TLS}$ and the global minimizer $\hat{t}_1$ once L_k is smaller than $\underline{L}_{TLS}$.
28 $\hat{t}_1 \leftarrow \hat{t}_{1k}$; $\underline{L}_{TLS} \leftarrow L_k$;
29 **end if**
30 **end for**

$L_k = L_k^q + L_k^c$ on each interval $\mathcal{J}_k$. Based on the comparison at each iteration (Lines 27-28), Algorithm 6 can solve (28) to global optimality with $O(N^2)$ time complexity.

F. Proof of Proposition 2

Let $\boldsymbol{x}_i := [x_{i1},\ x_{i2},\ x_{i3}]^\top$, we have

$$\begin{aligned} b_i &= y_{i1} - \boldsymbol{r}_1{}^\top \boldsymbol{x}_i \\ &= y_{i1} - x_{i1} \sin\beta \cos\alpha - x_{i2} \sin\beta \sin\alpha - x_{i3} \cos\beta \\ &= y_{i1} - (x_{i1} \cos\alpha + x_{i2} \sin\alpha) \sin\beta - x_{i3} \cos\beta \\ &= y_{i1} - \sqrt{x_{i1}{}^2 + x_{i2}{}^2} \cos(\alpha - \alpha^*) \sin\beta - x_{i3} \cos\beta, \end{aligned} \tag{29}$$

where $\alpha^* \in [0, \pi]$ denotes the arc-tangent angle of x_{i2}/x_{i1}. Now consider the following lemma:

Lemma 1. *Given $\theta \in [\theta_1, \theta_2] \subseteq [0, \pi]$ and $\phi \in [0, \pi]$, define $f(\theta) := cos(\theta - \phi)$, there is*

$$f(\theta) \in \begin{cases} [f(\theta_1),\ f(\theta_2)], & \text{if } \phi \geq \theta_2; \\ [f(\theta_2),\ f(\theta_1)], & \text{if } \phi \leq \theta_1; \\ [\min\{f(\theta_1),\ f(\theta_2)\},\ 1], & \text{otherwise.} \end{cases} \tag{30}$$

Else if $[\theta_1, \theta_2] \subseteq (\pi, 2\pi]$, there is

$$f(\theta) \in \begin{cases} [f(\theta_2),\ f(\theta_1)], & \text{if } \phi \geq \theta_2; \\ [f(\theta_1),\ f(\theta_2)], & \text{if } \phi \leq \theta_1; \\ [-1, \max\{f(\theta_1),\ f(\theta_2)\}], & \text{otherwise.} \end{cases} \tag{31}$$

Based on Lemma 1 (easy to prove), the range of $\cos(\alpha - \alpha^*)$ in (29) can be computed and we define it by $[\Psi_l,\ \Psi_u]$. Since $\sin\beta \geq 0$, we have

$$\begin{aligned} & y_{i1} - \sqrt{{x_{i1}}^2 + {x_{i2}}^2}\Psi_u \sin\beta - x_{i3}\cos\beta \leq b_i \leq y_{i1} - \sqrt{{x_{i1}}^2 + {x_{i2}}^2}\Psi_l \sin\beta - x_{i3}\cos\beta \\ \Leftrightarrow\ & y_{i1} - \sqrt{({x_{i1}}^2 + {x_{i2}}^2){\Psi_u}^2 + {x_{i3}}^2}\cos(\beta - \beta_l^*) \leq b_i \leq y_{i1} - \sqrt{({x_{i1}}^2 + {x_{i2}}^2){\Psi_l}^2 + {x_{i3}}^2}\cos(\beta - \beta_u^*), \end{aligned} \tag{32}$$

where $\beta_l^*,\ \beta_u^* \in [0, \pi]$ denote the arc-tangent angles of $\left(\sqrt{{x_{i1}}^2 + {x_{i2}}^2}\Psi_u\right)/x_{i3}$ and $\left(\sqrt{{x_{i1}}^2 + {x_{i2}}^2}\Psi_l\right)/x_{i3}$, respectively. Note that the ranges of $\cos(\beta - \beta_l^*)$ and $\cos(\beta - \beta_u^*)$ can be solved based on Lemma 1. Denote γ_l^u as the upper bound of $\cos(\beta - \beta_l^*)$, γ_u^l as the lower bound of $\cos(\beta - \beta_u^*)$, we have

$$b_{il} = y_{i1} - \gamma_l^u\sqrt{({x_{i1}}^2 + {x_{i2}}^2){\Psi_u}^2 + {x_{i3}}^2} \leq b_i \leq y_{i1} - \gamma_u^l\sqrt{({x_{i1}}^2 + {x_{i2}}^2){\Psi_l}^2 + {x_{i3}}^2} = b_{iu}. \tag{33}$$

Since Ψ_l, Ψ_u, γ_l^u, and γ_u^l in (33) can be easily computed based on Lemma 1, we can therefore get the range $[b_{il}, b_{iu}]$ of b_i in (29) in constant time.

G. Extra Experimental Details

Hyperparameter Setup. In Sec. 2.2 of the main manuscript, we decomposed the original 6-dimensional problem, TEAR, into two subproblems, TEAR-1 and TEAR-2. TEAR has a threshold hyperparameter ξ_i, as is typical in many prior works. Moreover, TEAR-1 has its own threshold ξ_{i1} and TEAR-2 has ξ_{i2}.

However, this is not to say our method requires more hyperparameters than prior works. In fact, given the commonly used parameter ξ_i, we can choose ξ_{i1} and ξ_{i2} relatively easily, and here is how we do it. First, we simply set ξ_{i1} to be equal to ξ_i. Second, recall the optimal solution $(\hat{\boldsymbol{r}}, \hat{t}_1)$ and the associated inlier indices $\hat{\mathcal{I}}_1$ defined in (1) in the main manuscript. For each $i \in \hat{\mathcal{I}}_1$, we set ξ_{i2} to $\xi_i - |y_{i1} - \hat{\boldsymbol{r}}_1^\top \boldsymbol{x}_i - \hat{t}_1|$. Our experiments justify the choices of the hyperparameters. In addition, we set the minimal branch resolution in the BnB part of our method as 1e-3. As to TR-DE [1], we set the resolution as 5e-2 to guarantee its experimental time limited in five days (otherwise it costs averagely more than 200s for each real-world pair).

Dataset Details. For the three real-world datasets (3DMatch [7], KITTI [4], and ETH [6]), we follow [1, 2, 5] to set the inlier threshold ξ_i based on the downsampling voxel size. Specifically, ξ_i is set to 10 cm for the 3DMatch Dataset [7], 60 cm for the KITTI Dataset [4], and 30 cm for the ETH Dataset [6], respectively.

In the Stanford 3D scanning dataset [3], we used 5 objects. And in Tab. 4 of the main manuscript, we reported the number of points each object contains, namely, *Armadillo* has 10^5 points, *Happy Buddha* has 5×10^5 points, *Asian Dragon* has 10^6 points, *Thai Statue* has 4×10^6 points, and *Lucy* has 10^7 points. We emphasize that each of these objects has slightly more points, and for clarity we downsampled them a little bit. In fact, *Armadillo* has approximately 1.7×10^5 points, *Happy Buddha* has approximately 5.4×10^5 points, *Asian Dragon* has approximately 3.6×10^6 points, *Thai Statue* has approximately 4.9×10^6 points, and *Lucy* has approximately 1.4×10^7 points.

Related Translation Errors of Fig. 7. Recall that in Fig. 7 of the main manuscript, we report the average rotation errors of the unscalable methods in Tab. 4 of the manuscript evaluated on the downsampled data and our `TEAR` on the original data. As shown in Fig. 5, we additionally report the related average translation errors.

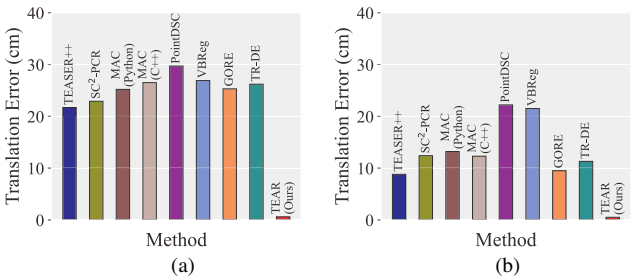


Figure 5. Average translation errors of other methods in Tab. 4 of the manuscript taking as inputs the 10^4 points downsampled from *Lucy* that originally has 10^7 point pairs (Fig. 5a: 99.8% outliers; Fig. 5b: 95% outliers). TEAR runs on the original 10^7 input point pairs. 20 trials.

References

[1] 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. 13

[2] 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. 13

[3] 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. 13

[4] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research*, 2013. 13

[5] 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. 13

[6] 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. 13

[7] 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. 13