IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 15, 2020, accepted August 11, 2020, date of publication August 24, 2020, date of current version September 8, 2020.

*Digital Object Identifier 10.1109/ACCESS.2020.3019209*

# An Accelerated and Robust Partial Registration Algorithm for Point Clouds

**XIN WANG[1], XIAOHUANG ZHU[1], SHIHUI YING[1], (Member, IEEE), AND CHAOMIN SHEN[2]**

[1]Department of Mathematics, School of Science, Shanghai University, Shanghai 200444, China
[2]Shanghai Key Laboratory of Multidimensional Information Processing, School of Computer Science and Technology, East China Normal University, Shanghai 200062, China

Corresponding author: Chaomin Shen (cmshen@cs.ecnu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61573274, Grant 11771276, Grant 11971296, and Grant 11871327; in part by the Science and Technology Commission of Shanghai Municipality under Grant 14DZ2260800; and in part by the Capacity Construction Project of Local Universities in Shanghai under Grant 18010500600.

**ABSTRACT** Partial registration for point clouds plays an important role in various fields such as 3D mapping reconstruction, remote sensing, unmanned driving, and cultural heritage protection. Unfortunately, partial registration is challenging due to difficulties such as the low overlap ratio of two point clouds and the perturbation in the orderless and sparse 3D point clouds. Thus, a variety of the 3D shape context descriptors are introduced for finding the optimal matching. However, extracting geometric features and descriptors are time consuming and easily degenerated by noise. To overcome these problems, we introduce a parallel coarse-to-fine partial registration method. Our contributions can be summarized as: Firstly, a robust coarse trimmed method is proposed to estimate the coarse overlap area and the initial transformation via fast bilateral denoising and parallel point feature histogram (PPFH) descriptor aligning. Secondly, an accelerated fine registration procedure is conducted by a parallel trimmed iterative closest point (PTrICP) method. Moreover, most parts of our coarse-to-fine workflow are accelerated under the Graphics Processing Unit (GPU) parallel execution mode for efficiency. Thirdly, we extend our method from the rigid registration to the isotropic scaling registration, which improves its applicability. Experiments have demonstrated that our method is feasible and robust in various situations, including the low overlap ratio, outlier, noise and scaling.

**INDEX TERMS** Point clouds, GPU parallel, partial registration, coarse-to-fine, trimmed strategy.

## I. INTRODUCTION

Point cloud data processing is a hot topic in recent years [1], [2]. Point cloud registration, in particular, is a fundamental problem in computer vision and remote sensing [3]–[10]. It is the basis of various works, including surface alignment [11], [12], pose estimation [13], [14], 3D reconstruction [15]–[17], object recognition [18], and Simultaneous Localization And Mapping (SLAM) [19]. The aim of point cloud registration is to obtain an optimal matching that transforms point sets from various views into one global coordinate.

Iterative Closest Point (ICP), a widely used method, cast the point cloud registration as an iterative transformation process [20]. The original ICP algorithm took every point to point distance into consideration for estimating rigid transformations, which was suitable for aligning two closely positioned and similarly shaped point clouds. Chen and Medioni gave another version of ICP algorithm, which utilized a nonlinear process by calculating the point to plane distances in order to accelerate the convergence [21]. Besides, the generalized ICP (GICP) [22] considered the registration model into a probabilistic distribution framework, and used nonlinear optimization instead of the closed form solutions in transformation estimation. As a result, this plane-to-plane ICP method is more stable and robust for measurement noise. In general, as mentioned in [23], ICP and its variants perform well in ideal cases [24]–[26].

Unfortunately, noise is inevitable due to acquisition instruments, lighting, reflection, outliers or occlusion in the scene. Thus, point cloud processing usually includes the noise smoothing and trimming procedures [27], [28]. In general, the noise in point clouds can be classified as perturbation

The associate editor coordinating the review of this manuscript and approving it for publication was Daniel Grosu.

noise, missing points, and outliers. These noise types bring the uncertainty for registration, and also contribute to the low overlap ratio of the point clouds.

To reduce noise, some widely used filtering algorithms have been proposed for point clouds, including statistical-based filtering techniques [28]–[31]. These techniques adjust the location of each point via different projection strategies to filter point cloud, and determine the filtered position using similarity measures between a point and its neighbors. However, these kinds of noise smoothing algorithms cannot solve the low overlap ratio problem caused by missing points and outliers.

Thus, partial registration, i.e., the registration of point clouds that are overlapped partially, is challenging caused by occlusion, missing points, outliers and measurement noise. The state-of-the-art partial registration methods are mainly based on the randomized alignment or overlap ratio estimation [7]. Widely used randomized alignment based partial registration methods include 4-point congruent sets (4PCS) and Super4PCS [32] [33]. These methods use the invariance of the ratio between the lines formed by four coplanar points, and avoid the calculation of complex geometric features. Hence, 4PCS based algorithms are relatively robust to partial registration. On the other hand, the trimmed ICP method is introduced via overlap ratio estimation [34]–[37], which can trim outliers and makes the algorithm suitable for the relatively low overlap ratio (under 50%). However, the initial transformation and additional coarse registration procedure are still necessary, when the overlap ratio is low or the noise level is high.

More specifically, the coarse registration algorithm can estimate a rough transformation under harsh conditions, thus providing a good initial transformation for the fine registration. Besides the aforementioned randomized aligning methods, the principal component analysis (PCA) [38] is often used to obtain several principal axes of two point sets. Then, these global matching methods align the centers and their principal axes for the coarse transformation, while they fail if these two point clouds are under the low overlap ratio. Furthermore, some coarse registrations are between different scanning resolutions [39]. Han *et al.* proposed a hierarchical searching scheme for multi-resolution data to improve the robustness with respect to the local minimum [40]. Recently, a coarse-to-fine GICP algorithm combines the plane-to-plane and point-to-point trimmed ICP, which balances the stability and accuracy by changing the neighborhood search range from wide-base to narrow-base in each iteration [37]. However, this adaptive strategy still fails under the low overlap ratio.

This article introduces a coarse-to-fine trimmed strategy to solve the low overlap problem. In the coarse trimmed stage, we trim the majority of non-informative points and estimate the initial overlap area via parallel point feature histogram (PFH) descriptors matching, in which the neighborhood based bilateral filter smoothes noise since the PFH feature estimation is sensitive to noise [41], [42]. Furthermore, a fine registration procedure is conducted by a parallel modified trimmed ICP based on the initial overlap ratio and corresponding transformation. Finally, most parts of our coarse-to-fine workflow are accelerated under the GPU parallel execution mode for improving efficiency.

This article is organized as follows. In section II we introduce the preliminary of our coarse trimmed stage, such as the classical PFH descriptor estimation and bilateral filter. Section III describes the details of our coarse-to-fine registration method, including the GPU parallel implementation. Section IV shows the experiments and analysis. Section V concludes the paper.

## II. PRELIMINARY

As mentioned in Introduction, neither the global matching nor the hierarchical searching schemes are suitable for partial registration, since these coarse matching methods are sensitive to the low overlap problem [32] [33], [38] [39], [40]. Thus, as the first step, some geometric descriptor based matching methods are introduced for the coarse partial registration. As we know, the choice of descriptor is the key for feature alignment. A good feature descriptor should be robust to noise and invariant with the position and orientation of point cloud [43]. For example, Rusu *et al.* designed the PFH [44] and Fast PFH in [45], based on the spatial relationship of a point with its nearest neighbor.

In this section, we give some preparation for our new coarse trim strategy. The traditional PFH is presented in section II-A, and the bilateral filter preprocessing for reducing the noise is introduced in section II-B.

### A. POINT FEATURE HISTOGRAM

The PFH descriptor is computed as a histogram of geometric description between all point pairs in the neighborhood. Firstly, PFH characterizes the relationships between the k-neighborhood points $p_s$ and $p_t$ via their estimated normal vectors $\boldsymbol{n}_s$ and $\boldsymbol{n}_t$, i.e.,

$$\begin{cases} u = \boldsymbol{n}_s \\ v = u \times \dfrac{(p_t - p_s)}{\|p_t - p_s\|_2} \\ w = u \times v \end{cases} \qquad \begin{cases} \alpha = v \cdot \boldsymbol{n}_t \\ \phi = u \cdot \dfrac{(p_t - p_s)}{\|p_t - p_s\|_2} \\ \theta = \arctan(w \cdot \boldsymbol{n}_t, u \cdot \boldsymbol{n}_t) \end{cases} \tag{1}$$

where $\alpha$, $\phi$ and $\theta$ constitute a triplet $(\alpha, \phi, \theta)$ in the k-neighborhood (Figure 1).

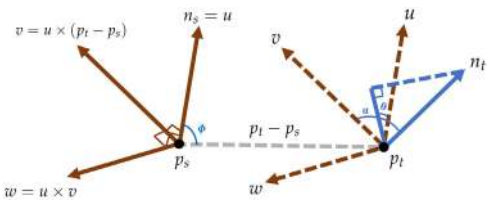



**FIGURE 1.** Local coordinate system $\{u, v, w\}$ of point $p_s$.

All triplets are then binned into a histogram, as shown in Figure 2. The binning process separates each feature

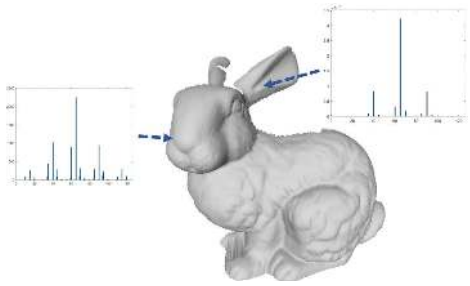


**FIGURE 2.** PFH descriptors for different points.

value range into some subdivisions (125 subdivisions in our algorithm), and counts the number of occurrences in each subdivision.

Moreover, PFH descriptor estimation has a high complexity of $\mathcal{O}(mk^2)$, where $m$ is the number of points involved in the PFH computation, and $k$ is the number of the nearest neighbors of each point. In this article, the parallel implementation methods will be proposed to reduce the complexity.

### B. BILATERAL FILTER

The bilateral filter is relatively efficient for denoising [28], although it is time-consuming for large point cloud data. [42] proposed its accelerated version using OpenMP [46].

The bilateral filter [41] allows an offset correction for each point along its normal vector. Assume that $p$ is the candidate point for denoising, and $\mathbf{n}_p$ is the corresponding normal vector. Let $p_i$ be the neighbors of $p$ for $i = 1, \cdots, N_1$, and $\omega_s(x) = e^{-x^2/2\sigma_s^2}$, $\omega_c(x) = e^{-x^2/2\sigma_c^2}$ be two Gaussians with variances $\sigma_s$ and $\sigma_c$, respectively. Then, the offset correction is

$$\delta_p = \frac{\sum_{i=1}^{N_1} \omega_s(\| p - p_i \|)\omega_c(\langle \mathbf{n}_p, p_i - p\rangle)\langle \mathbf{n}_p, p_i - p\rangle}{\sum_{i=1}^{N_1} \omega_s(\| p - p_i \|)\omega_c(\langle \mathbf{n}_p, p_i - p\rangle)}. \quad (2)$$

The new position of $p$ is $\hat{p} = p + \delta_p \mathbf{n}_p$. This is the same as projecting a point on the weighted regression plane with the neighboring weights $\omega(p_i) = \frac{1}{W}\omega_s(\|p - p_i\|)\omega_c(\langle \mathbf{n}_p, p_i - p\rangle)$, where $W$ is a normalization factor. The offset correction ensures that, to denoise a point near the edge, only points lying on the same facet will contribute. The bilateral filter for a given point $p \in \mathcal{P}$ is summarized in Algorithm 1.

**Algorithm 1** bilateral($p, N_1, \sigma_s, \sigma_c$)

**Require:** A point $p \in \mathcal{P}$, search range $N_1$, $\sigma_s$, $\sigma_c$.

$\mathcal{S}_{N_1}(p) \leftarrow$ neighbors of $p$
Compute the unit normal vector $\mathbf{n}_p$ to the regression plane from $\mathcal{S}_{N_1}(p)$
$sum = 0, normalizer = 0$
**for** $p_i \in \mathcal{S}_{N_1}(p)$ **do**
  $d_s = \|p_i - p\|$
  $d_c = \langle p_i - p, \mathbf{n}_p \rangle$
  $w' = \exp(-\frac{d_s^2}{2\sigma_s^2}) \cdot \exp(-\frac{d_c^2}{2\sigma_c^2})$
  $sum \mathrel{+}= w' \cdot d_c$
  $normalizer \mathrel{+}= w'$
**end for**
$\hat{p} \leftarrow p + \frac{sum}{normalizer}\mathbf{n}_p$

## III. THE PROPOSED METHOD

Some notations used in our algorithm are listed here. The source point cloud (red) is $\mathcal{P} = \{p_m, m = 1, \ldots, M\}$ and the target point cloud (blue) is $\mathcal{Q} = \{q_l, l = 1, \ldots, L\}$. The goal is to find the best transformation $\hat{\mathcal{T}}$ from $\mathcal{P}$ to $\mathcal{Q}$. We conduct $\hat{\mathcal{T}} = \mathcal{T}^* \cdot \mathcal{T}_0$, where $\mathcal{T}_0$ represents the transformation of coarse registration, and $\mathcal{T}^*$ represents the best transformation of fine registration. The flow chart is given in Figure 3.

The coarse trimmed block (the yellow block) gives a good initial transformation and coarse overlap estimation, which is used to smooth the noise and trim most of outliers. On the other hand, the fine registration algorithm (the brown block) computes the fine transformation $\mathcal{T}^*$. To make our acceleration strategies clear, we give some parallel computation procedure details in section III-C. Firstly, we give a detailed representation of our coarse trimmed strategy.

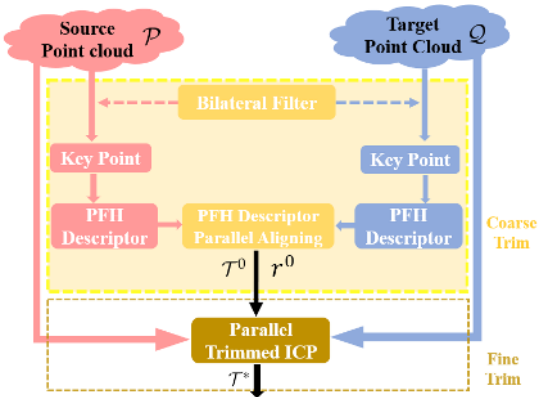



**FIGURE 3.** Flow chart of the parallel coarse-to-fine registration.

### A. COARSE TRIMMED STRATEGY FOR PARTIAL REGISTRATION

The coarse trimmed strategy, the key step of the coarse-to-fine partial registration, is used to estimate the coarse overlap area and the initial transformation. In particular, a parallel bilateral denoising method (section III-A1) and parallel point feature histogram (PPFH) aligning between key point pairs are used to eliminate the majority of non-overlap information.

#### 1) PARALLEL BILATERAL FILTER DENOISING

The key point and PFH descriptor extraction are easily perturbed by noise, as they heavily depend on the point positions and their corresponding normal vectors. Therefore, we add a bilateral filter to denoise the noisy point cloud, and refer to the parallel implementation of point cloud bilateral filter in [42]. This algorithm could be summarized in Algorithm 2, and the bilateral function is shown in section II-B.

**Algorithm 2** Parallelization of the Bilateral Filter

**Require:** Input point cloud $\mathcal{P}$ stored into an octree $\psi$, a search range $N_1$, and two variances $\sigma_s$, $\sigma_c$ of Gaussian distribution.

1: **for** each node child $C$ **in parallel do**
2: **for** $p \in C$ **do**
3: $\hat{p} \leftarrow \text{bilateral}(p, N_1, \sigma_s, \sigma_c)$ ▷ reference to Alg. 1
4: Update octree node $\hat{C}$ where $\hat{p}$ is located
5: **end for**
6: **end for**

#### 2) PARALLEL KEY POINTS SELECTION

For saving the computation cost of extracting the key points from the original data, in the beginning, we define

$$\gamma(p) = \frac{\sum_{i=1}^{N_2} |\boldsymbol{n} \cdot \boldsymbol{n}_i|}{N_2} = \frac{\sum_{i=1}^{N_2} |\cos\langle \boldsymbol{n}, \boldsymbol{n}_i \rangle|}{N_2} \tag{3}$$

as the local curvature descriptor [47], where $N_2$ is the neighborhood search range, and $i$ is the index of neighborhood search result of each point $p$.

We keep the point with the smallest $\gamma$ value among its neighbors, i.e., keeping the local region edge or corner points. The key point sets are marked as $\mathcal{P}' = \{\boldsymbol{p}' : \boldsymbol{p}' \in \mathcal{P}\}$ and $\mathcal{Q}' = \{\boldsymbol{q}' : \boldsymbol{q}' \in \mathcal{Q}\}$. Figure 4 illustrates the key point selection for point clouds $\mathcal{P}$ and $\mathcal{Q}$.

**FIGURE 4.** Key point sets (yellow points) $\mathcal{P}'$ and $\mathcal{Q}'$ on $\mathcal{P}$ and $\mathcal{Q}$ respectively.

#### 3) PARALLEL PFH DESCRIPTOR COMPUTATION FOR KEY POINTS

We propose a parallel implementation of PFH for key point sets $\mathcal{P}'$ and $\mathcal{Q}'$ in Figure 5. The PPFH computation mainly consists of parallel kd-tree searching and parallel normal vector estimation. Please refer to sections III for parallel implementation details.

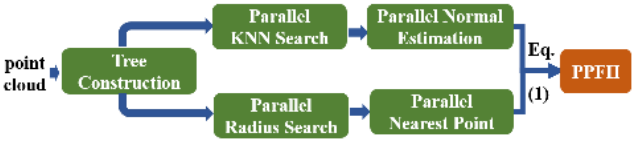



**FIGURE 5.** Details of PPFH descriptor implementation.

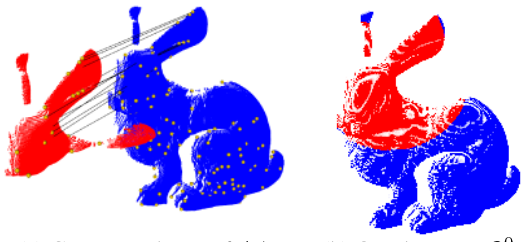
#### 4) COARSE ALIGNING

Here, with PFH feature, we search the most similar correspondent key point using the parallel brute force. We denote the set of key point pairs by

$$\mathcal{M} = \{(\boldsymbol{p}'', \boldsymbol{q}'') : \boldsymbol{p}'' \in \mathcal{P}'' \subseteq \mathcal{P}', \boldsymbol{q}'' \in \mathcal{Q}'' \subseteq \mathcal{Q}'\}.$$

Figure 6(a) illustrates the **correspondence** of key point pairs.

On the one hand, the correspondence of key point pairs $\mathcal{M} = \{(\boldsymbol{p}'', \boldsymbol{q}'')\}$ is used to identify the coarse overlap area and overlap ratio. Specifically, we utilize the parallel neighborhood search on $\mathcal{Q}$ for the key point center $\overline{\boldsymbol{q}} = \frac{1}{|\mathcal{Q}''|} \sum_{\boldsymbol{q}_i'' \in \mathcal{Q}''} \boldsymbol{q}_i''$ to get the **coarse overlap area** $\mathcal{Q}^0$, where $|\cdot|$ means the number of points. The searching range is about 1.1 times of the size of partial (source) point cloud $\mathcal{P}$. Figure 6(b) illustrates the overlap area (red) $\mathcal{Q}^0$. The coarse overlap ratio $r^0$ is estimated by $r^0 = \frac{|\mathcal{Q}^0|}{|\mathcal{Q}|}$.



(a) Correspondence of $\mathcal{M}$ (b) Overlap area $\mathcal{Q}^0$

**FIGURE 6.** The correspondence of key point pairs $\mathcal{M}$ and the overlap area $\mathcal{Q}^0$.

On the other hand, the correspondence of key point pairs $\mathcal{M}$ can be used to identify **the coarse transformation** $\mathcal{T}^0 = \{R^0, t^0\}$ [38], as summarized in Algorithm 3:

**Algorithm 3** Generation of Transformation $\mathcal{T}^0$

**Require:** Key point pair $\mathcal{M} = \{(\boldsymbol{p}'', \boldsymbol{q}'')\}$.

1: Find the barycenters $\overline{\boldsymbol{p}} = \frac{1}{|\mathcal{P}''|} \sum_i \boldsymbol{p}_i''$ and $\overline{\boldsymbol{q}} = \frac{1}{|\mathcal{Q}''|} \sum_i \boldsymbol{q}_i''$,
2: Transform key point sets $\bar{P}'' = \{\bar{\boldsymbol{p}}_i'' | \bar{\boldsymbol{p}}_i'' = \boldsymbol{p}_i'' - \overline{\boldsymbol{p}}\}$ and $\bar{Q}'' = \{\bar{\boldsymbol{q}}_i'' | \bar{\boldsymbol{q}}_i'' = \boldsymbol{q}_i'' - \overline{\boldsymbol{q}}\}$
3: Construct the covariance matrix $H = \sum_i \bar{\boldsymbol{p}}_i'' \bar{\boldsymbol{q}}_i''^T$
4: Conduct SVD on $H = U \Lambda V^T$
5: The rotation $R^0 = VU^T$ and translation $t^0 = \overline{\boldsymbol{q}} - R^0 \overline{\boldsymbol{p}}$.

### B. PARALLEL TRIMMED ICP FOR FINE PARTIAL REGISTRATION

With the coarse overlap area $\mathcal{Q}^0$ and initial rigid transformation $\mathcal{T}^0$, we conduct the fine registration based on the parallel trimmed ICP method under $\mathcal{Q}^0$ and the source point cloud $\mathcal{P}$. Denoting the transformation by $\mathcal{T}^\eta = \{R^\eta, t^\eta\}$ in the current $\eta$-th iteration, the transformation can consist of rotation $R^\eta$, translation $t^\eta$. For the current $\eta$-th iteration, the parallel trimmed ICP registration is conducted as follows:

#### 1) UPDATE CORRESPONDENCES

For each point $p_i$ in the source partial point cloud, find its correspondence $q_{c_i^\eta}$ via parallel implementation of the nearest

point searching. The index of point in $\mathcal{Q}$ that matches point $p_i$ is denoted by $c_i^\eta$:

$$c_i^\eta = \arg\min_j \|\mathcal{T}^{\eta-1} \cdot p_i - q_j\|^2, \quad i = 1 \dots M. \tag{4}$$

#### 2) UPDATE OVERLAP RATIO

Having the correspondences $\{(p_i, q_{c_i^\eta})\}_{i=1}^M$ fixed, parallelly compute the squared distances $\{(d_i^\eta)^2\}_{i=1}^M$ as

$$(d_i^\eta)^2 = \left(\mathcal{T}^{\eta-1} \cdot p_i - q_{c_i^\eta}\right)^\top \left(\mathcal{T}^{\eta-1} \cdot p_i - q_{c_i^\eta}\right), \tag{5}$$

and sort $\{(d_i^\eta)^2\}_{i=1}^M$ with the ascending order; then parallelly calculate the overlap ratio according to

$$r^\eta = \arg\min_r \sum_{i=1}^{M \times r} \frac{(d_i^\eta)^2}{e^{\lambda^\eta} \cdot M \cdot r^{\lambda^\eta}}, \tag{6}$$

where $\lambda^\eta = \lambda^{\eta-1} - \delta$ is the parameter of trimmed strategy, and the positive constants $\lambda^0$, $\delta$ are defined in advance. Besides, we parallelly pick up the first $M_r = \lfloor M \times r^\eta \rfloor$ corresponding point pairs, and define the error

$$\varepsilon^\eta = \sum_{i=1}^{M_r} (d_i^\eta)^2 / M_r \tag{7}$$

as the Trimmed Squared Distance (TSD).

#### 3) UPDATE TRANSFORMATION

$$\mathcal{T}^\eta = \arg\min_{\mathcal{T}} \sum_{i=1}^{M_r} \|\mathcal{T}^{\eta-1} \cdot p_i - q_{c_i}\|^2. \tag{8}$$

Repeat the above steps until the stopping criteria are satisfied. In our algorithm, the iteration stops when any of the following three conditions hold.

1) The maximum iteration number $\tau$ has reached;
2) The TSD error $\varepsilon^\eta$ is sufficiently small;
3) $|\varepsilon^\eta - \varepsilon^{\eta-1}|$ is sufficiently small.

### C. PARALLEL IMPLEMENTATION

Point cloud registration, especially for high-resolution data, usually suffers from the massive size problem, leading to long execution time. Some parallel implementation methods have been proposed to deal with this problem. In 2007, Nüchter proposed the parallel implementation of classical ICP with OpenMP and utilized this method in GraphSLAM [48]. In this decade, the GPU parallel mode began to be applied on ICP and EM-ICP [49], [50].

With the GPU development, the new generation GPU cores could handle massive point size in parallel mode. Based on this, we give the parallel execution introduction of our trimmed ICP method in Figure 7, which illustrates that most parts of the algorithm are implemented in the parallel mode, except for the simple computation such as the update of transformation matrix. The parallel implementation of bilateral filter is described in section III-A1. The fine registration could be divided into three main parts, among which the

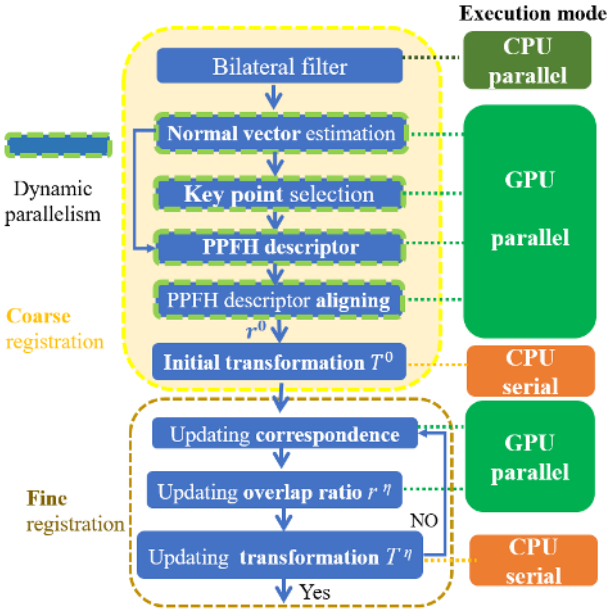



**FIGURE 7.** Parallel implementation of our algorithm.

correspondence and overlap ratio update are implemented in CUDA Thrust [51].

Comparing with the nearest neighbor search (NNS) based parallel ICP method [49], [50], our contributions in parallel registration lie in: 1) we provide GPU parallelization for all modules (not limited to NNS) of the point cloud registration; 2) we implement the parallel trimmed ICP algorithm, which improves the robustness by adding the parallel overlap ratio estimation in each iteration; and 3) we use the high-level Thrust library (https://thrust.github.io/) in our parallel implementation, which strengthens the reliability of our algorithm.

Most of the parallel coarse-to-fine registration implementation is based on the parallel kd-tree search, which consists of two parts: tree construction in section III-C1 and the nearest point search in section III-C2.

We first elaborate the parallel kd-tree search implementation.

#### 1) PARALLEL TREE CONSTRUCTION

Given $L$ points for tree construction, and restrict each leaf node containing maximum one point, the complexity of sequential kd-tree construction is $\mathcal{O}(kL \log_2 L)$ [52], where $k$ is the dimension of kd-tree. If each leaf node contains maximum $\kappa$ points ($1 < \kappa \ll L$), there are approximately $\lfloor \frac{L}{\kappa} \rfloor$ leaf nodes and the tree depth is $(\log_2 \lfloor \frac{L}{\kappa} \rfloor)$. Therefore, the number of nodes is $1 + 2 + 4 + \cdots + \lfloor \frac{L}{\kappa} \rfloor = 2\lfloor \frac{L}{\kappa} \rfloor - 1$, and there are $\lfloor \frac{L}{\kappa} \rfloor - 1$ node splitting operations in serial processing.

In contrast, for the parallel mode, every thread in GPU manipulates one parent node and its corresponding left and right child nodes, i.e., all node splitting procedures in the same level of a kd-tree are executed parallelly. Hence the parallel processing needs $\log_2 \lfloor \frac{L}{\kappa} \rfloor$ splitting operations. As a

result, the parallel tree construction is about $\frac{\lfloor \frac{L}{\kappa} \rfloor - 1}{\log_2 \lfloor \frac{L}{\kappa} \rfloor}$ times faster than the sequential implementation.

#### 2) NEAREST POINT SEARCH

After parallelly constructing the tree structure in section III-C1, every query point finds its nearest point in the tree structure. The serial computation simply searches the nearest neighbor point by point, while in parallel mode GPU can easily schedule millions of threads under specific heterogeneous computing architecture to manipulate each point concurrently. For the point cloud of size $M$, the complexity of scale situations decreases from $\mathcal{O}(M \log_2 \lfloor \frac{L}{\kappa} \rfloor)$ to $\mathcal{O}(\log_2 \lfloor \frac{L}{\kappa} \rfloor)$, where $\log_2 \lfloor \frac{L}{\kappa} \rfloor$ is the tree depth.

#### 3) DYNAMIC PARALLELISM

Dynamic parallelism, as a programming structure template, is extensively used in our parallel normal vector estimation, key points selection, PFH descriptors estimation, and PFH descriptor alignment for key point pairs. In Figure 8, each thread in our parent GPU thread block manipulates some specific data, and each parent thread could create another GPU thread block, marked as 'child', which also executes under the parallel mode.

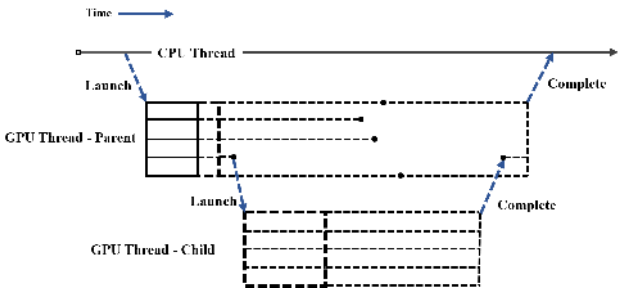



**FIGURE 8.** Dynamic parallelism programming structure.

#### 4) NORMAL VECTOR ESTIMATION

The normal vector estimation, based on the NNS results in section III-C2, is implemented via the dynamic parallelism mode. Each thread in the parent thread block manipulates a point $p$ in the point cloud, and each thread in the child thread block computes the dot product $p_{3\times1} \cdot q_{1\times3}$, where $q$ is one of the nearest points of $p$. The parallelism of child thread block reduces the complexity from $\mathcal{O}(N_0)$ to $\mathcal{O}(1)$, where $N_0$ is the neighborhood search range. The dot product results are added up by the parent thread in parallel, and the complexity of parallel adding up is $\mathcal{O}(\log_2 N_0)$ [53]. Hence we get the $3 \times 3$ covariance matrix $X_{cov}$. In the end, the parent thread utilizes the eigenvalue decomposition on $X_{cov}$ to get the unit normal vector of each point.

#### 5) KEY POINT SELECTION

As mentioned in section III-A2, the key point selection is based on the normal vector estimation and the neighborhood search. We utilize the dynamic parallelism mode for the key point selection. Given $M$ points for key point selection, each thread in the parent thread block manipulates one point, and schedules a corresponding child thread block for computing $\gamma(p)$ parallelly. Assuming that the nearest point range is $N_2$, the child thread block reduces the complexity of computing $\gamma(p)$ from $\mathcal{O}(N_2)$ to $\mathcal{O}(\log_2 N_2)$, and the entire dynamic parallelism reduces the complexity from $\mathcal{O}(MN_2)$ to $\mathcal{O}(\log_2 N_2)$.

#### 6) PARALLEL PFH DESCRIPTOR COMPUTATION

We propose a parallel implementation of PFH descriptor based on parallel kd-tree searching, normal vector estimation and key point selection.

In terms of constructing the histogram of features, we also utilize the dynamical parallelism mode. In the parent thread block of GPU, each thread manipulates the 3D point position coordinates and its corresponding normal vector of key point set, which means that the thread number of parent thread block is equal to the number of key points. In the child thread block, each thread manipulates the nearest indices of each key point inherited from its parent thread. Then each child thread computes the triplet $(\alpha, \phi, \theta)$ between each key point and one of its neighborhood points. This parallel mode reduces the complexity from $\mathcal{O}(L'N_3^2)$ to $\mathcal{O}(1)$, where $L'$ is the size of key point set, and $N_3$ is the nearest search range. The final statistic histogram of the point feature is implemented with the CUDA command atomicAdd in the sequential mode to avoid the multi-thread memory access conflict.

#### 7) KEY POINT PFH DESCRIPTOR ALIGNING

In the key point pair feature correspondence search part, each key point feature is a 125 dimensional vector. The kd-tree search method, however, is not suitable for the high dimensional feature correspondence search [54]. We utilize the parallel brute force search method, i.e., each thread in the parent GPU thread block manipulates one 125 dimensional feature vector for a key point; each thread creates another 'child' GPU thread block, and then searches for the most similar feature in another point cloud under the parallel mode. Given $M'$ key points, the complexity of parallel implementation for PFH descriptor aligning could be reduced from $\mathcal{O}(125 \cdot M')$ to $\mathcal{O}(1)$. After successfully finding the aligned key point pairs, our algorithm obtains the initial overlap ratio $r^0$ and the initial transformation $\mathcal{T}^0$.

## IV. EXPERIMENTAL RESULTS AND ANALYSIS

Our experiment is verified in terms of accuracy and runtime. The sequential program is written in C++ with Point Cloud Library (PCL) [55], and the parallel program is written in CUDA C/C++ [56]. All experiments are running on Intel® Core™ i7-8750H CPU @2.20GHz and NVIDIA® GeForce GTX 1060.

The data used in the experiments are Bunny, Dragon and Happy Buddha (Figure 9) from the Stanford 3D Scanning Repository (`http://graphics.stanford.edu/data/3Dscanrep/`). We also test our algorithm on

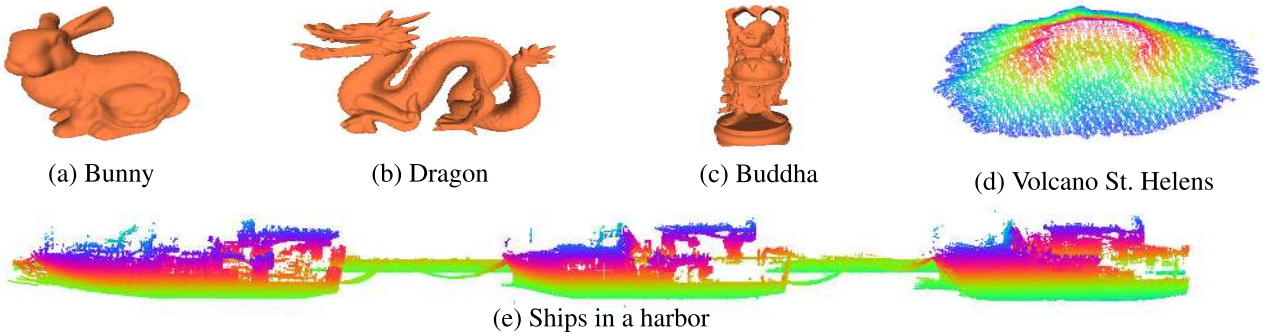



**FIGURE 9.** Point cloud data Bunny, Dragon, Buddha, Ship and Volcano.

remote sensing point clouds, for example, the volcano point cloud acquired by LiDAR over Mount St. Helens and the ship data acquired from a LiDAR point cloud of a harbor scene.

### A. EXPERIMENT DESIGN

Given a source 3D point cloud $\mathcal{P}$ and a target point cloud $\mathcal{Q}$, we conduct a predetermined random transformation $\mathcal{T}$ on $\mathcal{P}$. Here the transformation includes rotation, translation and scaling. As a result, a data pair $(\mathcal{P}, \mathcal{Q})$ is obtained with the ground truth of transformation $\mathcal{T}$.

The goal of the algorithm is to find the best transformation $\hat{\mathcal{T}}$ from $\mathcal{P}$ to $\mathcal{Q}$. We compare $\hat{\mathcal{T}} = \mathcal{T}^* \cdot \mathcal{T}_0$ under various conditions such as missing points, noise, outlier and isotropic scaling. The better the registration performs, the closer the transformation to the ground truth $\mathcal{T}$. We show both the visualization and the numerical results. All the quantity results include the average accuracy and runtime of 15 different sample pairs.

We compare our coarse-to-fine registration algorithm with the Super4PCS coarse algorithm [33], the robust trimmed ICP [35], [36], and the coarse-to-fine adaptive generalized-ICP algorithm (AGICP) [37] in various situations, including missing data, outliers and perturbation noise. For missing data, the source point cloud $\mathcal{P}$ sizes are 10,098, 11,449, 17,212, 76,395, 10,110; the target cloud $\mathcal{Q}$ sizes are 40,097, 41,841, 54,353, 244,996, 30,554; the corresponding overlap ratios are 25.18%, 27.36%, 31.67%, 31.18% and 33.09% for bunny, dragon, buddha, ship and volcano, respectively.

We measure the error by the average of point-to-point distances, marked as TSD, on the overlap area of $\mathcal{P}$ and $\mathcal{Q}$. Meanwhile, the stopping values of the trimmed ICP, AGICP and our fine registration method are: 1) The maximum iteration number $\tau = 200$; 2) The TSD error $\varepsilon^\eta < e^{-10}$; and 3) $|\varepsilon^\eta - \varepsilon^{\eta-1}| < e^{-10}$, respectively.

### B. MISSING POINTS

Figure 10 shows the results of missing points case, compared with the Super4PCS, original trimmed ICP and AGICP. The PPFH features of the key points match well on the overlap area. Figure 11 shows the zoom-in results of coarse and fine registrations, respectively.

Table 1 displays the detailed numerical results, indicating that our PPFH based coarse-to-fine registration method outperforms the original trimmed ICP, AGICP and Super4PCS methods in terms of the registration average accuracy and runtime. Note that even our coarse registration outperforms the trimmed ICP, AGICP and Super4PCS in terms of accuracy. Of course, our coarse-to-fine registration method outperforms our own coarse trimmed module, which is also shown in the table. It indicates that our fine trimmed module further improves the accuracy. Our method sets the neighborhood range $N_0 = 8$ for the point cloud normal vector estimation, and $N_3 = 32$ for the PFH descriptor computation in both $\mathcal{P}$ and $\mathcal{Q}$.

Table 1 also shows the runtime of coarse and fine registrations of parallel implementation. Owing to the accurate coarse registration transformation result, the iterations of fine registration are less than directly using trimmed ICP and AGICP. It verifies again the importance of coarse registration. Besides, the fine registration utilizes the GPU parallel mode to reduce the runtime in each iteration, which has significant improvement in accuracy and runtime. Particularly, we downsample the ship data for AGICP, since this algorithm requires a large amount of memory and cannot be executed for full size data.

### C. OUTLIERS

For comparing our method under different outlier levels, we add additional 10% to 40% outlier points on the missing data. Then, the actual overlap ratio becomes 19.2% to 24.9%. Figure 12 illustrates the comparison under different outlier levels, and the numerical results are given in Table 2.

Comparing to the other three methods, our PPFH based coarse-to-fine algorithm has much higher registration accuracy; owing to the accurate coarse estimation, the fine registration converges faster. In order to obtain similar PPFH features of the key points correctly, we need to reduce $N_3$ in KNN search to avoid capturing more random outlier information. Hence, the search region is about 3/4 of that in the raw data experiment.

### D. PERTURBATION NOISE

Furthermore, we verify our algorithm under the perturbation noise. The results are shown in Figure 13 and Table 3. We compare the registration performance with seven different methods:

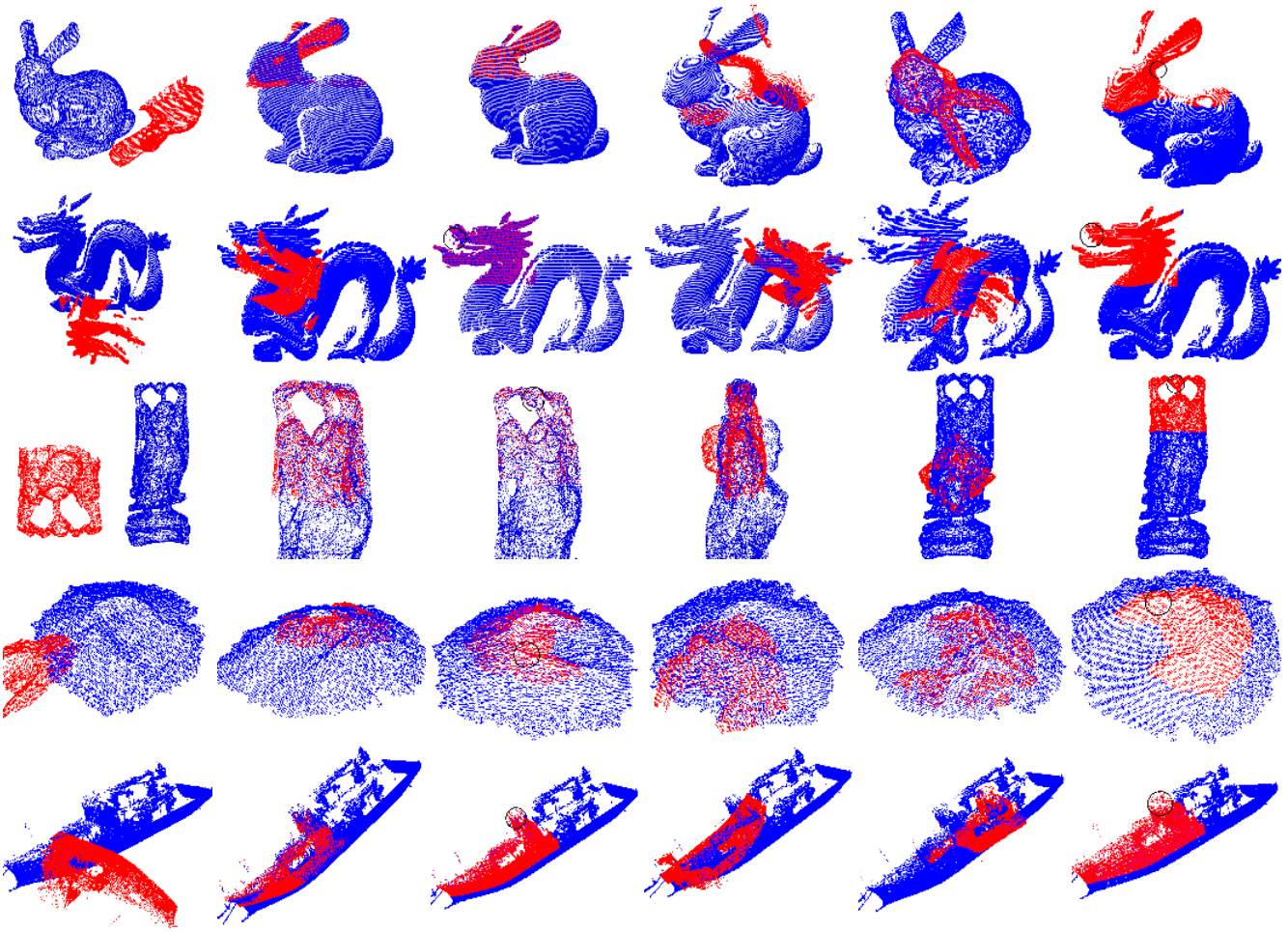



**FIGURE 10.** Comparison results for missing data. The circled areas in the fine registration column are enlarged in Figure 11.

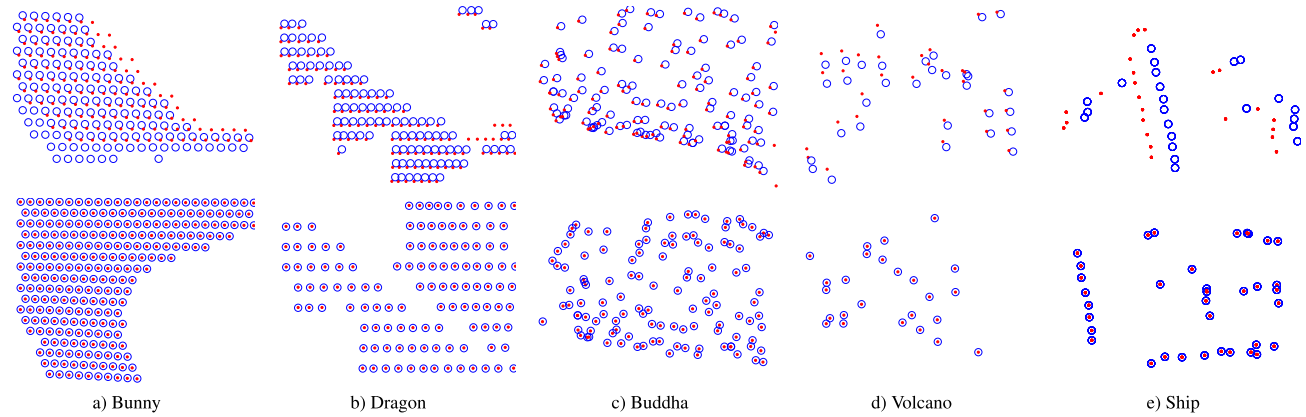



**FIGURE 11.** Enlarged local areas of Figure 10. The first row is the zoom-in of the coarse registration result, and the second row is that of the fine registration.

1) Super4PCS [33],
2) our coarse trimmed module without denoising (named coarse w/o denoising),
3) our coarse trimmed module (named our coarse method),
4) the trimmed ICP [35],
5) AGICP [37],
6) our coarse-to-fine registration without denoising (named our method w/o denoising),

**TABLE 1. Ablation analysis on accuracy and runtime (ms) under the low overlap caused by missing points.**

| Methods | Modules | | | | | Bunny | | Dragon | | Buddha | | Volcano | | Ship | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Bilateral Filter | Feature Descriptor | Coarse Registration | Plane to Plane aligning | Trimmed ICP | TSD | Time | TSD | Time | TSD | Time | TSD | Time | TSD | Time |
| Super4PCS | | | ✓ | | | $2.14e^{-3}$ | 9136 | $3.20e^{-2}$ | 713 | $2.07e^{-3}$ | 8760 | 6631.88 | 1274 | 1.94 | 368 |
| Our coarse method | ✓ | ✓ | ✓ | | | $3.17e^{-7}$ | 426 | $3.14e^{-7}$ | 412 | $2.82e^{-7}$ | 632 | 42.4 | 417 | 0.483 | 4931 |
| Trimmed ICP | | | | | ✓ | $6.68e^{-2}$ | 7106 | 0.116 | 8145 | $8.15e^{-2}$ | 12116 | 7599 | 9210 | 25.5 | 80956 |
| Coarse-to-fine AGICP | | | | ✓ | ✓ | $5.98e^{-2}$ | 1408 | $6.94e^{-2}$ | 1167 | $9.01e^{-2}$ | 1386 | 6445 | 1676 | 21.5 | - |
| Our coarse-to-fine method | ✓ | ✓ | ✓ | | ✓ | $\mathbf{6.62e^{-11}}$ | 519 | $\mathbf{1.35e^{-11}}$ | 509 | $\mathbf{8.85e^{-12}}$ | 728 | **0.143** | 521 | $\mathbf{2.24e^{-5}}$ | 5283 |

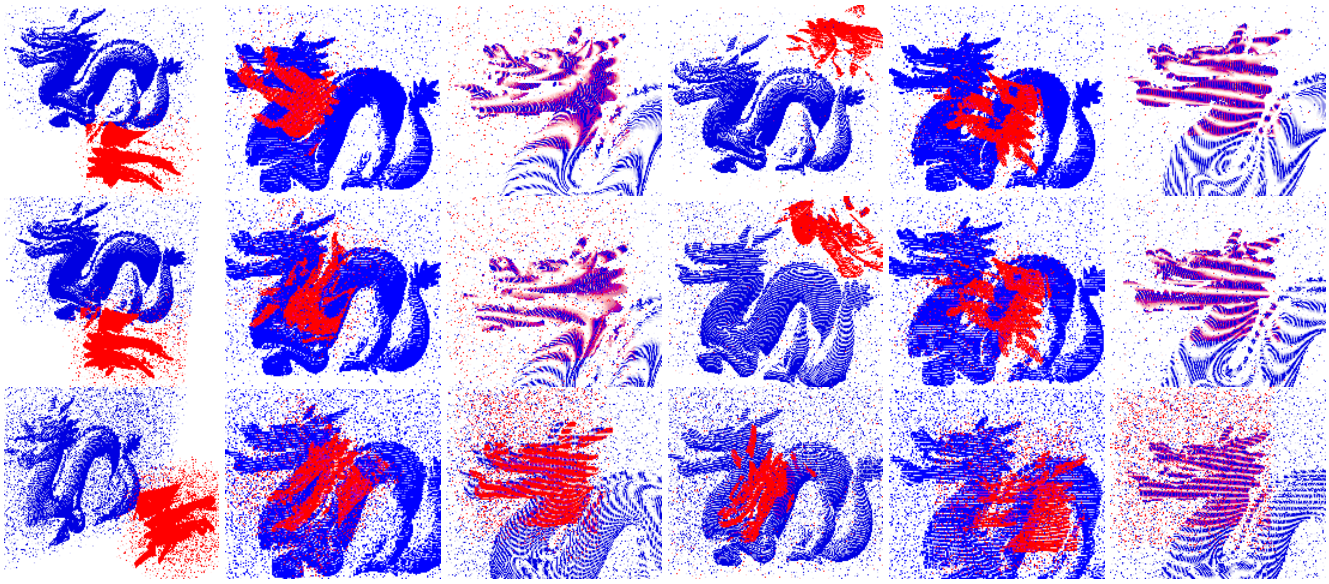



**FIGURE 12. Comparison results under different levels of outliers. In the first row, the outlier is 10% of point cloud size, the second one is 20%, and the last row is 40%.**

**TABLE 2. Ablation analysis on accuracy and runtime (ms) under different levels of outliers.**

| Methods | Modules | | | | | 10% Outliers | | 20% Outliers | | 40% Outliers | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | Bilateral Filter | Feature Descriptor | Coarse Registration | Plane to Plane aligning | Trimmed ICP | TSD | Time | TSD | Time | TSD | Time |
| Super4PCS | | | ✓ | | | $2.81e^{-2}$ | 1561 | $6.39e^{-2}$ | 30128 | $4.51e^{-2}$ | 56620 |
| Our coarse method | ✓ | ✓ | ✓ | | | $1.37e^{-7}$ | 524 | $1.55e^{-7}$ | 475 | $8.17e^{-3}$ | 679 |
| Trimmed ICP | | | | | ✓ | 0.107 | 10623 | 0.138 | 10832 | $8.78e^{-2}$ | 13451 |
| Coarse-to-fine AGICP | | | | ✓ | ✓ | $7.00e^{-2}$ | 696 | $7.22e^{-2}$ | 720 | $6.43e^{-2}$ | 999 |
| Our coarse-to-fine method | ✓ | ✓ | ✓ | | ✓ | $\mathbf{1.35e^{-11}}$ | 597 | $\mathbf{1.35e^{-11}}$ | 543 | $\mathbf{1.37e^{-11}}$ | 781 |

7) our coarse-to-fine registration method.

In Figure 13, the first row is the data contaminated by noise $N(0, 0.0003)$, and the second row is contaminated by $N(0, 0.0007)$. Super4PCS, Trimmed ICP, AGICP and coarse w/o denoising suffer significantly from noisy and missing data.

Table 3 shows the registration accuracy and runtime on the different noise levels. Comparing the coarse registrations with and without denoising, the bilateral filter efficiently reduces the noise influence on PPFH feature. Using trimmed ICP directly suffers from the low overlap ratio problem, leading to more iteration steps or wrong transformation results.

Although the runtime slightly increases with the bilateral filter in our method, the registration accuracy is much better than not using it. This outcome benefits from not only the parallel execution mode but also the denoising procedure, which reduces the iteration steps in the modified trimmed ICP.

### E. SCALING

In this part, we give the extension of our coarse-to-fine trimmed partial registration method to the isotropic scaling case. To register in the scaling case successfully, we need to estimate the scale $s^0$ in the coarse trimmed module too.

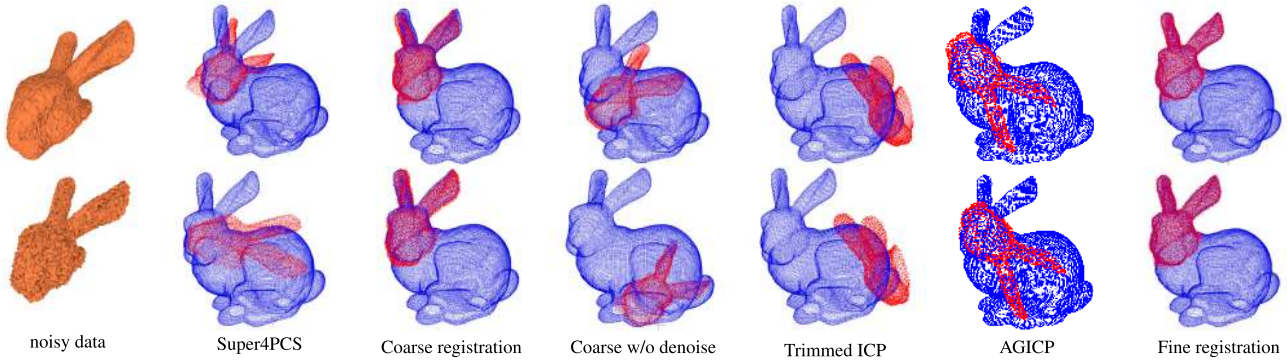



**FIGURE 13.** Comparison results under different noise levels. The noise levels are $N(0, 0.0003)$ and $N(0, 0.0007)$ in the first and second row, respectively.

**TABLE 3.** Ablation analysis on accuracy and runtime (ms) under different noise levels.

| Methods | Modules | | | | | *N(0,0.0003)* | | *N(0,0.0005)* | | *N(0,0.0007)* | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | Bilateral Filter | Feature Descriptor | Coarse Registration | Plane to Plane aligning | Trimmed ICP | TSD | Time | TSD | Time | TSD | Time |
| Super4PCS | | | ✓ | | | $2.29e^{-2}$ | 32260 | $3.12e^{-2}$ | 41737 | $5.60e^{-2}$ | 45071 |
| Coarse w/o denoise | | ✓ | ✓ | | | $3.73e^{-2}$ | 517 | $4.78e^{-2}$ | 522 | 0.103 | 535 |
| Our coarse method | ✓ | ✓ | ✓ | | | $3.09e^{-3}$ | 1157 | $4.93e^{-3}$ | 1239 | $1.09e^{-2}$ | 1505 |
| Trimmed ICP | | | | | ✓ | 0.132 | 6062 | 0.129 | 7360 | 0.134 | 8271 |
| Coarse-to-fine AGICP | | | | ✓ | ✓ | $5.96e^{-2}$ | 1453 | $6.03e^{-2}$ | 1295 | $6.13e^{-2}$ | 1292 |
| Our method w/o denoise | | ✓ | ✓ | | ✓ | $4.14e^{-2}$ | 941 | $6.21e^{-2}$ | 953 | $9.63e^{-2}$ | 961 |
| Our coarse-to-fine method | ✓ | ✓ | ✓ | | ✓ | $\mathbf{7.72e^{-7}}$ | 1476 | $\mathbf{1.39e^{-6}}$ | 1558 | $\mathbf{5.73e^{-6}}$ | 1846 |

Then, we conduct the trimmed scale ICP [36] in GPU parallel model:

$$\hat{\mathcal{T}} = \arg\min \|s \cdot R \cdot \mathcal{P} + t - \mathcal{Q}\|^2. \tag{9}$$

The detail could be summarized in Algorithm 4.

**Algorithm 4** Partial Registration for Isotropic Scaling

**Require:** Point clouds $\mathcal{P}$ and $\mathcal{Q}$.

1: Estimate the correspondence and coarse overlap area referred in section III-A4.
2: Estimate the coarse transformation $\{s^0, R^0, t^0\}$ by SVD
3: **for** each iteration $\eta$ in fine registration **do**
4: Find the nearest points of $\mathcal{P}$ in $\mathcal{Q}$ with parallel mode: $C^\eta = \arg\min \|s^{\eta-1}R^{\eta-1}\mathcal{P} + t^{\eta-1} - \mathcal{Q}\|^2$.
5: Update the overlap ratio via (6).
6: Compute the barycenters $\overline{P}$ and $\overline{Q}$ of the overlap area.
7: Update the scale, rotation and translation: $[s^\eta, R^\eta, t^\eta] = \arg\min \|s \cdot R \cdot \mathcal{P} + t - \mathcal{Q}\|^2$. Estimate the scale $s^\eta$ via the division of the singular values of SVD $(\overline{\mathcal{P}} \cdot \overline{\mathcal{P}}^T)$ and SVD $(\overline{\mathcal{Q}} \cdot \overline{\mathcal{Q}}^T)$. Compute the rotation $R^\eta$ via Algorithm 3 and translation $t^\eta = \overline{\boldsymbol{q}} - s^\eta \cdot R^\eta \cdot \overline{\boldsymbol{p}}$
8: Repeat the above steps until the stopping criteria are satisfied.
9: **end for**

In order to reduce the possibility of scale degeneration in non-rigid transformation, we set the upper and lower scale constraints $[s_L, s_U]$ for each iteration.

To verify the proposed method under scaling for real data, we conduct different methods on the Ship and Volcano data. Super4PCS is not suitable for scaling situation due to its reliance on point distance. Experimental results are shown in Figures 14 and 16.

Figure 14 shows that directly conducting the trimmed ICP easily fails for the low overlap ratio and the scale 2:1. In contrast, PPFH based coarse registration provides a reasonable coarse transformation including rotation, translation and scaling. This can be taken as a good initial transformation for the fine registration.

Notice that the PPFH based coarse registration gives an accurate transformation estimation, as shown in Table 4, which is better than directly conducting the trimmed ICP and AGICP. Besides, with the help of coarse registration, the fine registration converges faster than directly conducting the trimmed ICP and AGICP.

Figure 16 provides the comparison for partial registration with outliers (10%) under the scale 2:1. Obviously, PPFH based coarse registration gives a reasonable coarse transformation. Figure 17 shows the zoom-in result of our fine registration algorithm. In particular, we only downsample the ship data for AGICP, since this algorithm requires a large amount of memory and cannot be executed for full size data.

Table 5 further verifies that the PPFH based coarse-to-fine registration algorithm is also suitable for data with outliers (additional 10%) under the scale 2:1.

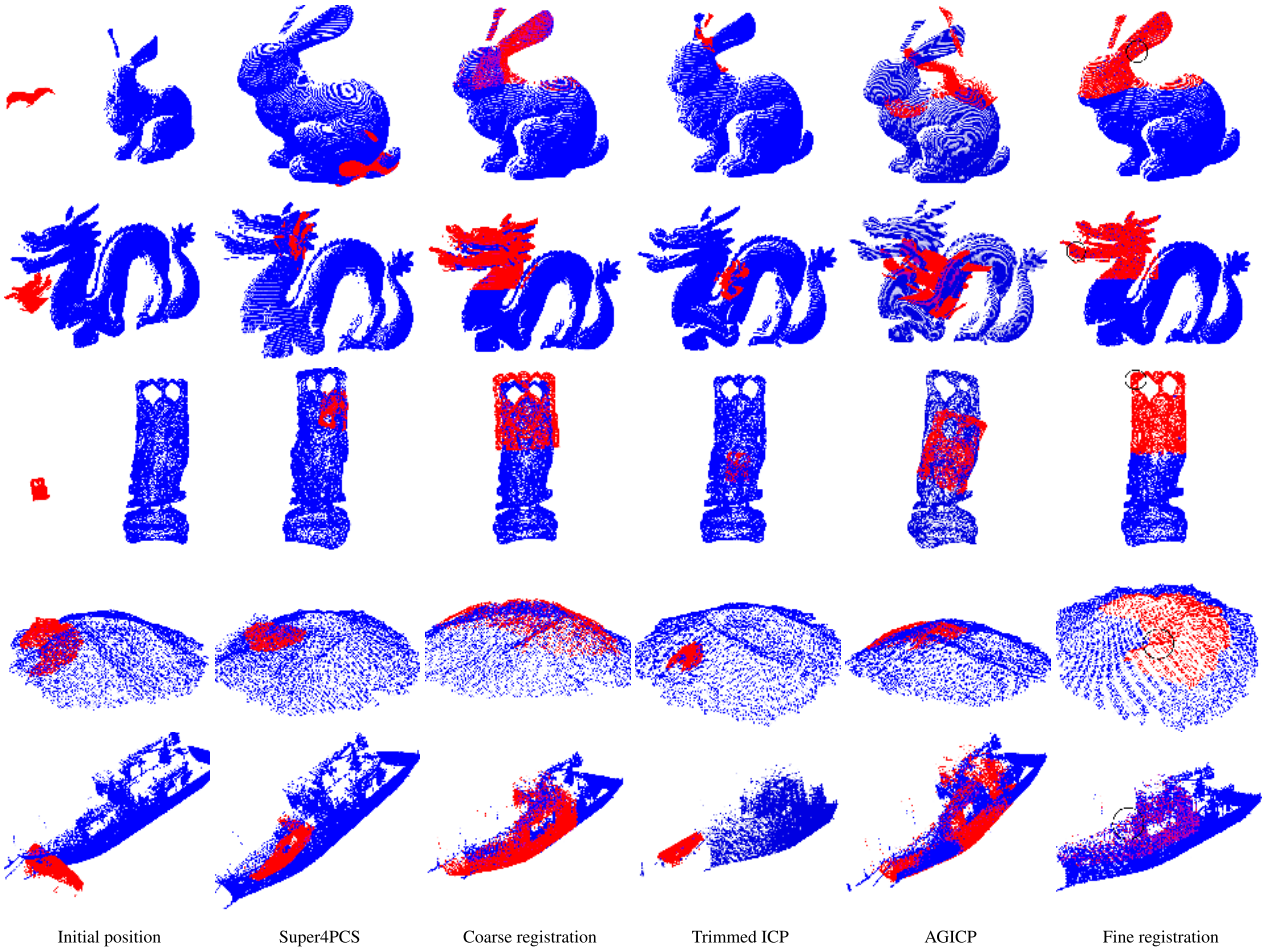



**FIGURE 14.** Comparison results under the scale 2 : 1. The circled areas in the fine registration column are enlarged in Figure 15.

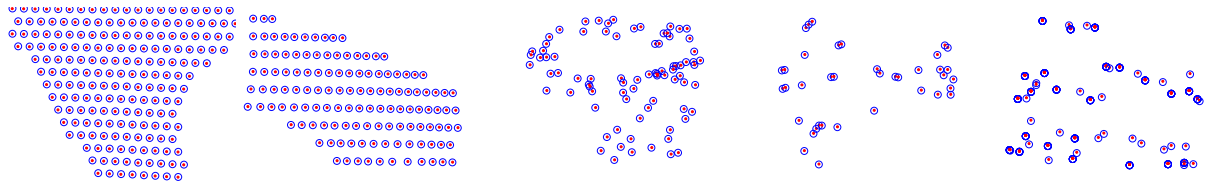



**FIGURE 15.** Enlarged areas of Figure 14.

**TABLE 4.** Ablation analysis on accuracy and runtime (ms) of low overlap ratio registration under the scale 2:1.

| Methods | Modules | | | | | Bunny | | Dragon | | Buddha | | Volcano | | Ship | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Bilateral Filter | Feature Descriptor | Coarse Registration | Plane to Plane Property | Trimmed ICP | TSD | Time | TSD | Time | TSD | Time | TSD | Time | TSD | Time |
| Super4PCS | | | ✓ | | | 0.128 | 15971 | $3.91e^{-2}$ | 7860 | $4.15e^{-2}$ | 12487 | 5221 | 7143 | 8.71 | 239620 |
| Our coarse method | ✓ | ✓ | ✓ | | | $2.21e^{-4}$ | 397 | $8.31e^{-3}$ | 414 | $1.74e^{-2}$ | 635 | 498 | 423 | 7.47 | 4372 |
| Trimmed ICP | | | | | ✓ | $6.44e^{-2}$ | 8739 | $8.91e^{-2}$ | 8365 | $7.95e^{-2}$ | 12330 | 7176 | 8667 | 28.4 | 69340 |
| Coarse-to-fine AGICP | | | | ✓ | ✓ | $8.64e^{-2}$ | 1441 | $7.63e^{-2}$ | 1483 | $8.64e^{-2}$ | 2193 | 3272 | 1633 | 9.54 | - |
| Our coarse-to-fine method | ✓ | ✓ | ✓ | | ✓ | $< \mathbf{1e^{-10}}$ | 486 | $< \mathbf{1e^{-10}}$ | 532 | $< \mathbf{1e^{-10}}$ | 779 | **0.526** | 602 | $\mathbf{3.06e^{-2}}$ | 4748 |

### *F. TIME CONSUMING ANALYSIS OF PARALLEL IMPLEMENTATION*

We compare the runtime of normal vector estimation, PFH computation and tree construction in parallel and sequential implementations, respectively. The results are shown in Figure 18 and Table 6.

In Figure 18(a), the blue and orange bars represent the sequential runtime of the normal vector estimation and PFH estimation, respectively. Similarly, the purple bar represents the parallel PFH computation time, and the faint yellow 'PNormal' means the parallel normal vector estimation. Notice that the key point size significantly influences the

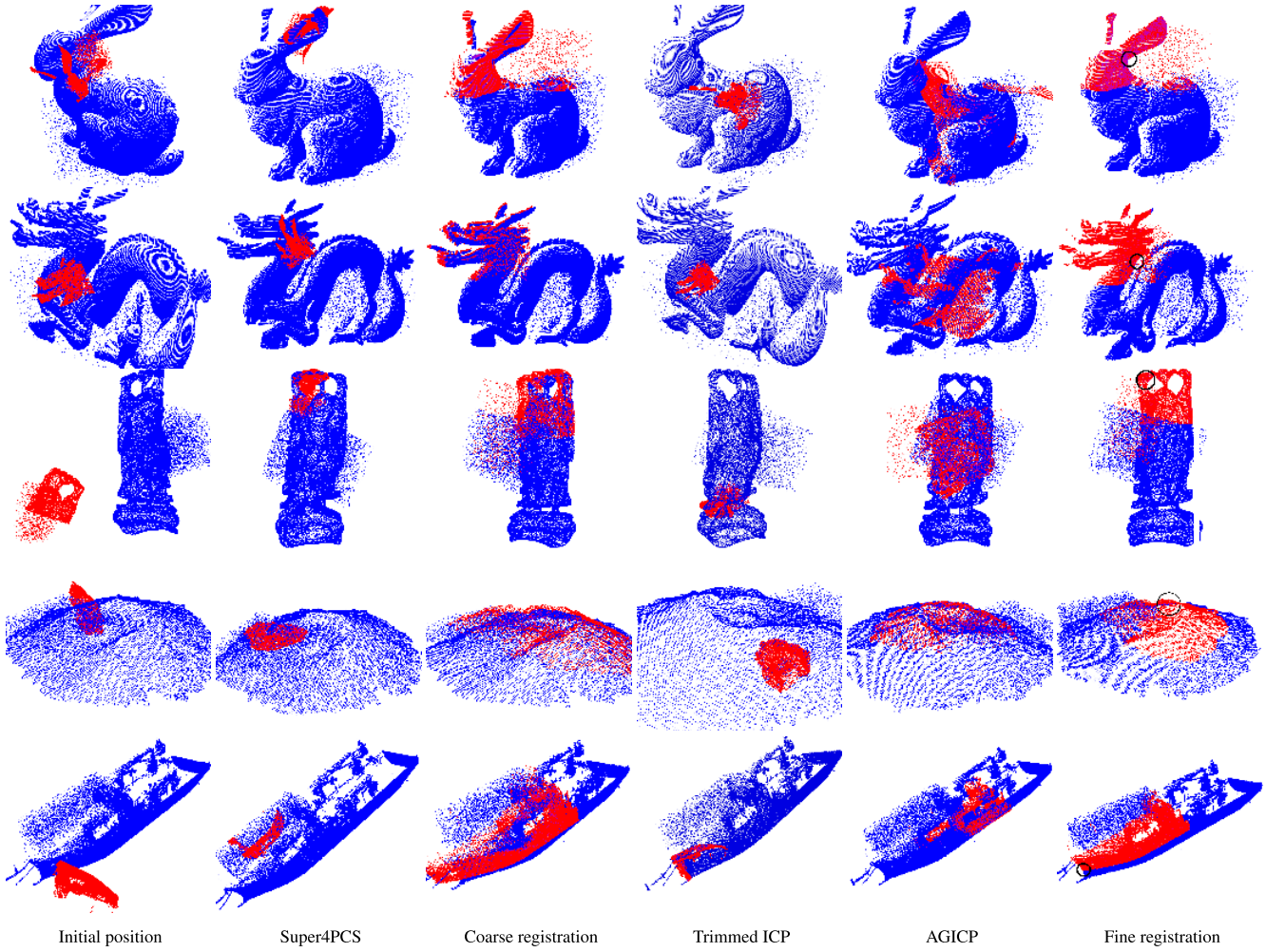



**FIGURE 16.** Comparison results for data with outliers (10%) under the scale 2:1. The circled areas in the fine registration column are enlarged in Figure 17.

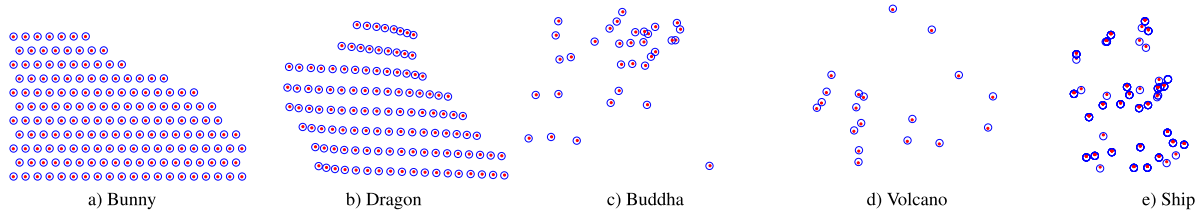



**FIGURE 17.** Enlarged local areas of Figure 16.

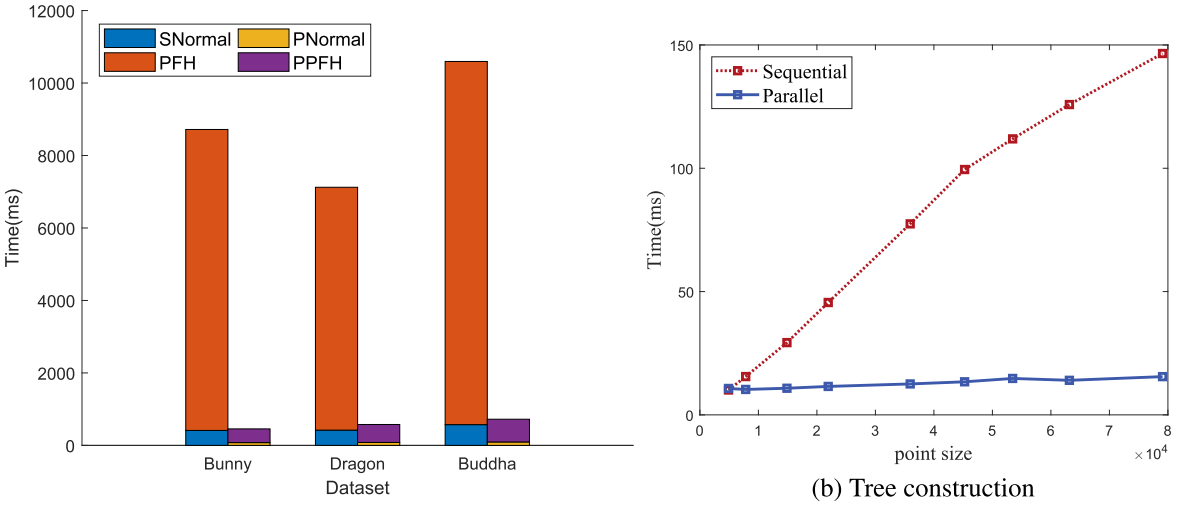
**TABLE 5.** Ablation analysis on accuracy and runtime (ms) under low overlap situation with outliers (10%) under the scale 2:1.

| Methods | Modules | | | | | Bunny | | Dragon | | Buddha | | Volcano | | Ship | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Bilateral Filter | Feature Descriptor | Coarse Registration | Plane to Plane aligning | Trimmed ICP | TSD | Time | TSD | Time | TSD | Time | TSD | Time | TSD | Time |
| Super4PCS | | | ✓ | | | $5.54e^{-2}$ | 5226 | $2.76e^{-2}$ | 7679 | $4.29e^{-2}$ | 3461 | 7279 | 7076 | 23.26 | 19273 |
| Our coarse method | ✓ | ✓ | ✓ | | | $9.82e^{-3}$ | 489 | $1.32e^{-3}$ | 442 | $9.64e^{-3}$ | 692 | 697.66 | 487 | 9.71 | 5782 |
| Trimmed ICP | | | | | ✓ | $7.94e^{-2}$ | 9017 | $7.43e^{-2}$ | 8474 | 0.164 | 13752 | 7105 | 9556 | 21.17 | 84793 |
| Coarse-to-fine AGICP | | | | ✓ | ✓ | $9.07e^{-2}$ | 1554 | $9.09e^{-2}$ | 1548 | $7.83e^{-2}$ | 2626 | 1082 | 1619 | 17.4 | - |
| Our coarse-to-fine method | ✓ | ✓ | ✓ | | ✓ | $< \mathbf{1e^{-10}}$ | 628 | $< \mathbf{1e^{-10}}$ | 576 | $< \mathbf{1e^{-10}}$ | 854 | **0.873** | 688 | $\mathbf{8.87e^{-2}}$ | 6155 |

runtime of sequential implemented PFH and normal vector estimation, but influences the parallel execution less.

Figure 18(b) compares the time of tree construction. The sequential tree construction time increases while the parallel one is very low and almost remains constant as the point size increases.

Table 6 illustrates the different stages of coarse-to-fine registration for data bunny, buddha and dragon. Although




(a) PFH descriptor computation

(b) Tree construction

**FIGURE 18.** Time comparison of normal vector estimation, PFH feature computation and tree construction.

**TABLE 6.** Parallel runtime (ms) table of different stages in the entire registration.

| *Data* | $\mathcal{P}$ size<br>$\mathcal{Q}$ size | Tree Construction | KNN match | Normal Vector | Radius Search | PPFH Calculation | Feature Pair Search | Fine Registration | Iteration |
|---|---|---|---|---|---|---|---|---|---|
| *Bunny* | 10098<br>40097 | 4.03<br>9.70 | 6.34<br>32.28 | 58.04<br>76.61 | 46.50<br>117.50 | 55.15<br>378.75 | 3.38 | 139.41 | 25 |
| *Dragon* | 11449<br>41841 | 3.79<br>8.79 | 8.17<br>32.75 | 49.47<br>81.43 | 76.87<br>131.93 | 135.80<br>495.57 | 5.11 | 131.72 | 19 |
| *Buddha* | 17212<br>54353 | 7.64<br>9.79 | 15.56<br>46.09 | 76.75<br>94.58 | 95.14<br>158.08 | 218.95<br>627.79 | 11.38 | 137.24 | 20 |

the PFH computation still occupies the main part of the registration time, the entire runtime decreases compared with the sequential mode.

## V. CONCLUSION

We have proposed an accurate and robust coarse-to-fine partial registration algorithm for two point clouds with the low overlap ratio. In methodology, this algorithm utilizes the noise robust PPFH matching to trim the outliers coarsely, estimates the overlap area and computes the coarse transformation in the coarse trimmed module. Then, the parallel trimmed ICP algorithm could be successfully conducted based on the coarse registration results. Moreover, This framework can be extended to isotropic scaling registration. In implementation, most part of the procedure is accelerated by CUDA and OpenMP.

## REFERENCES

[1] R. Huang, D. Hong, Y. Xu, W. Yao, and U. Stilla, "Multi-scale local context embedding for LiDAR point cloud classification," *IEEE Geosci. Remote Sens. Lett.*, vol. 17, no. 4, pp. 721–725, Apr. 2020.

[2] R. Huang, Y. Xu, D. Hong, W. Yao, P. Ghamisi, and U. Stilla, "Deep point embedding for urban classification using ALS point clouds: A new perspective from local to global," *ISPRS J. Photogramm. Remote Sens.*, vol. 163, pp. 62–81, May 2020.

[3] T. Wan, S. Du, W. Cui, Y. Yang, and C. Li, "Robust rigid registration algorithm based on correntropy and bi-directional distance," *IEEE Access*, vol. 8, pp. 22225–22234, 2020.

[4] M. Young, C. Pretty, S. Agostinho, R. Green, and X. Chen, "Loss of significance and its effect on point normal orientation and cloud registration," *Remote Sens.*, vol. 11, no. 11, p. 1329, Jun. 2019.

[5] T.-Y. Chuang and J.-J. Jaw, "Multi-feature registration of point clouds," *Remote Sens.*, vol. 9, no. 3, p. 281, Mar. 2017.

[6] H. Wu and H. Fan, "Registration of airborne LiDAR point clouds by matching the linear plane features of building roof facets," *Remote Sens.*, vol. 8, no. 6, p. 447, May 2016.

[7] L. Cheng, S. Chen, X. Liu, H. Xu, Y. Wu, M. Li, and Y. Chen, "Registration of laser scanning point clouds: A review," *Sensors*, vol. 18, no. 5, p. 1641, May 2018.

[8] P. Li, J. Wang, Y. Zhao, Y. Wang, and Y. Yao, "Improved algorithm for point cloud registration based on fast point feature histograms," *J. Appl. Remote Sens.*, vol. 10, no. 4, pp. 1–23, 2016.

[9] K.-H. Bae, "Evaluation of the convergence region of an automated registration method for 3D laser scanner point clouds," *Sensors*, vol. 9, no. 1, pp. 355–375, Jan. 2009.

[10] K.-H. Bae and D. D. Lichti, "A method for automated registration of unorganised point clouds," *ISPRS J. Photogramm. Remote Sens.*, vol. 63, no. 1, pp. 36–54, Jan. 2008.

[11] N. Fioraio, J. Taylor, A. Fitzgibbon, L. Di Stefano, and S. Izadi, "Large-scale and drift-free surface reconstruction using online subvolume registration," in *Proc. CVPR*, Jun. 2015, pp. 4475–4483.

[12] Y. Yang, D. Fan, S. Du, M. Wang, B. Chen, and Y. Gao, "Point set registration with similarity and affine transformations based on bidirectional KMPE loss," *IEEE Trans. Cybern.*, early access, Oct. 18, 2019, doi: 10.1109/TCYB.2019.2944171.

[13] A. Aldoma, Z.-C. Marton, F. Tombari, W. Wohlkinger, C. Potthast, B. Zeisl, R. B. Rusu, S. Gedikli, and M. Vincze, "Tutorial: Point cloud library: Three-dimensional object recognition and 6 DOF pose estimation," *IEEE Robot. Autom. Mag.*, vol. 19, no. 3, pp. 80–91, Sep. 2012.

[14] Y. Li, N. Snavely, D. Huttenlocher, and P. Fua, "Worldwide pose estimation using 3D point clouds," in *Proc. ECCV*, 2012, pp. 15–29.

[15] S. Liu, D. Gao, P. Wang, X. Guo, J. Xu, and D.-X. Liu, "A depth-based weighted point cloud registration for indoor scene," *Sensors*, vol. 18, no. 11, p. 3608, Oct. 2018.

[16] M. Vlaminck, H. Luong, W. Goeman, and W. Philips, "3D scene reconstruction using omnidirectional vision and LiDAR: A hybrid approach," *Sensors*, vol. 16, no. 11, p. 1923, Nov. 2016.

[17] M. Andriasyan, J. Moyano, J. E. Nieto-Julián, and D. Antón, "From point cloud data to building information modelling: An automatic parametric workflow for heritage," *Remote Sens.*, vol. 12, no. 7, p. 1094, Mar. 2020.

[18] J. Prankl, A. Aldoma, A. Svejda, and M. Vincze, "RGB-D object modelling for object recognition and tracking," in *Proc. IROS*, vol. 2015, pp. 96–103.

[19] H. Kim, S. Song, and H. Myung, "GP-ICP: Ground plane ICP for mobile robots," *IEEE Access*, vol. 7, pp. 76599–76610, 2019.

[20] P. J. Besl and N. D. McKay, "A method for registration of 3-D shapes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, no. 2, pp. 239–256, Feb. 1992.

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

[22] A. Segal, D. Haehnel, and S. Thrun, "Generalized-ICP," in *Proc. RSS*, Seattle, WA, USA, Jun. 2009, pp. 26–27.

[23] F. Pomerleau, F. Colas, and R. Siegwart, "A review of point cloud registration algorithms for mobile robotics," *Found. Trends Robot.*, vol. 4, no. 1, pp. 1–104, 2015.

[24] S. Du, N. Zheng, S. Ying, and J. Liu, "Affine iterative closest point algorithm for point set registration," *Pattern Recognit. Lett.*, vol. 31, no. 9, pp. 791–799, Jul. 2010.

[25] S. Du, J. Liu, C. Zhang, J. Zhu, and K. Li, "Probability iterative closest point algorithm for m-D point set registration with noise," *Neurocomputing*, vol. 157, pp. 187–198, Jun. 2015.

[26] S. Du, G. Xu, S. Zhang, X. Zhang, Y. Gao, and B. Chen, "Robust rigid registration algorithm based on pointwise correspondence and correntropy," *Pattern Recognit. Lett.*, vol. 132, pp. 91–98, Apr. 2020.

[27] D. Hong, N. Yokoya, J. Chanussot, and X. X. Zhu, "An augmented linear mixing model to address spectral variability for hyperspectral unmixing," *IEEE Trans. Image Process.*, vol. 28, no. 4, pp. 1923–1938, Apr. 2019.

[28] X.-F. Han, J. S. Jin, M.-J. Wang, W. Jiang, L. Gao, and L. Xiao, "A review of algorithms for filtering the 3D point cloud," *Signal Process., Image Commun.*, vol. 57, pp. 103–112, Sep. 2017.

[29] O. Schall, A. Belyaev, and H.-P. Seidel, "Robust filtering of noisy scattered point data," in *Proc. SPBG*, 2005, pp. 71–144.

[30] E. Narváez and N. E. L. Narváez, "Point cloud denoising using robust principal component analysis," in *Proc. GRAPP*, 2006, pp. 51–58.

[31] S. Orts-Escolano, V. Morell, J. García-Rodríguez, and M. Cazorla, "Point cloud data filtering and downsampling using growing neural gas," in *Proc. IJCNN*, Aug. 2013, pp. 1–8.

[32] D. Aiger, N. J. Mitra, and D. Cohen-Or, "4-points congruent sets for robust pairwise surface registration," *ACM Trans. Graph.*, vol. 27, no. 3, pp. 1–10, Aug. 2008.

[33] N. Mellado, D. Aiger, and N. J. Mitra, "Super 4PCS fast global pointcloud registration via smart indexing," *Comput. Graph. Forum*, vol. 33, no. 5, pp. 205–215, Aug. 2014.

[34] D. Chetverikov, D. Svirko, D. Stepanov, and P. Krsek, "The trimmed iterative closest point algorithm," in *Proc. ICPR*, vol. 3, 2002, pp. 545–548.

[35] S. Du, J. Zhu, N. Zheng, Y. Liu, and L. Ce, "Robust iterative closest point algorithm for registration of point sets with outliers," *Opt. Eng.*, vol. 50, no. 8, Aug. 2011, Art. no. 087001.

[36] J. Dong, Y. Peng, S. Ying, and Z. Hu, "LieTrICP: An improvement of trimmed iterative closest point algorithm," *Neurocomputing*, vol. 140, pp. 67–76, Sep. 2014.

[37] X. Wang, Y. Li, Y. Peng, and S. Ying, "A coarse-to-fine generalized-ICP algorithm with trimmed strategy," *IEEE Access*, vol. 8, pp. 40692–40703, 2020.

[38] K. S. Arun, T. S. Huang, and S. D. Blostein, "Least-squares fitting of two 3-D point sets," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-9, no. 5, pp. 698–700, Sep. 1987.

[39] H. Zha, M. Ikuta, and T. Hasegawa, "Registration of range images with different scanning resolutions," in *Proc. IEEE Int. Conf. Sys., Man, Cyber.*, Jun. 2000, pp. 1495–1500.

[40] J. Han, P. Yin, Y. He, and F. Gu, "Enhanced ICP for the registration of large-scale 3D environment models: An experimental study," *Sensors*, vol. 16, no. 2, p. 228, Feb. 2016.

[41] S. Fleishman, I. Drori, and D. Cohen-Or, "Bilateral mesh denoising," in *Proc. ACM SIGGRAPH*, 2003, pp. 950–953.

[42] J. Digne and C. de Franchis, "The bilateral filter for point clouds," *IPOL*, vol. 7, pp. 278–287, 2017.

[43] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, Nov. 2004.

[44] R. B. Rusu, N. Blodow, Z. C. Marton, and M. Beetz, "Aligning point cloud views using persistent feature histograms," in *Proc. IROS*, Sep. 2008, pp. 3384–3391.

[45] R. B. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (FPFH) for 3D registration," in *Proc. ICRA*, May 2009, pp. 3212–3217.

[46] L. Dagum and R. Menon, "OpenMP: An industry standard API for shared-memory programming," *IEEE Comput. Sci. Eng.*, vol. 5, no. 1, pp. 46–55, Jan. 1998.

[47] D. S. Meek and D. J. Walton, "On surface normal and Gaussian curvature approximations given data sampled from a smooth surface," *Comput. Aided Geometric Des.*, vol. 17, no. 6, pp. 521–543, Jul. 2000.

[48] A. Nüchter, "Parallelization of scan matching for robotic 3D mapping," in *Proc. 3rd Eur. Conf. Mobile Robots*, 2007.

[49] D. Qiu, S. May, and A. Nüchter, "GPU-accelerated nearest neighbor search for 3D registration," in *Proc. Comput. Vis. Syst.* Berlin, Germany: Springer, 2009, pp. 194–203.

[50] T. Tamaki, M. Abe, B. Raytchev, and K. Kaneda, "Softassign and EM-ICP on GPU," in *Proc. 1st Int. Conf. Netw. Comput.*, Nov. 2010, pp. 179–183.

[51] B. Nathan and H. Jared, *Thrust: A Productivity-Oriented Library for CUDA* (Applications of GPU Computing Series). Boston, MA, USA: Morgan Kaufmann, 2012, ch. 26, pp. 359–371.

[52] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Commun. ACM*, vol. 18, no. 9, pp. 509–517, Sep. 1975.

[53] P. J. Martin, L. F. Ayuso, R. Torres, and A. Gavilanes, "Algorithmic strategies for optimizing the parallel reduction primitive in CUDA," in *Proc. HPCS*, Jul. 2012, pp. 511–519.

[54] J. O'Rourke and J. E. Goodman, *Handbook of Discrete and Computational Geometry*. Boca Raton, FL, USA: CRC Press, 2004.

[55] R. B. Rusu and S. Cousins, "3D is here: Point cloud library (PCL)," in *Proc. ICRA*, May 2011, pp. 1–4.

[56] J. Nickolls, I. Buck, M. Garland, and K. Skadron, "Scalable parallel programming with CUDA," *Queue*, vol. 6, no. 2, pp. 40–53, Mar. 2008.


**XIN WANG** received the B.Sc. degree in mathematics from Nanjing Normal University, Nanjing, China, in 2003, and the Ph.D. degree in mathematics from the Institute of Computational Mathematics and Scientific/Engineering Computing (ICMSEC), Chinese Academy of Sciences, Beijing, China, in 2008. She is currently a Lecturer with the Department of Mathematics, School of Science, Shanghai University, Shanghai, China. Her research interests include multiscale analysis, point cloud, and image processing.

**XIAOHUANG ZHU** is currently pursuing the M.S. degree with the Department of Mathematics, School of Science, Shanghai University. His research interests include point set registration, computer vision, and parallel computation.

**SHIHUI YING** (Member, IEEE) received the B.Eng. degree in mechanical engineering and the Ph.D. degree in applied mathematics from Xi'an Jiaotong University, Xi'an, China, in 2001 and 2008, respectively. He held a postdoctoral position at the Biomedical Research Imaging Center, The University of North Carolina at Chapel Hill, Chapel Hill, NC, USA, from 2012 to 2013. He is currently a Professor with the Department of Mathematics, School of Science, Shanghai University, Shanghai, China. His current research interests include geometric theory and methods for medical image processing and machine learning.

**CHAOMIN SHEN** received the B.Sc. degree in mathematics from Fudan University, Shanghai, China, in 1991, the M.Sc. degree in mathematics from the National University of Singapore, Singapore, in 1998, and the Ph.D. degree in mathematics from East China Normal University (ECNU), Shanghai, in 2009. He is currently an Associate Professor with the School of Computer Science and Technology, ECNU. His research interests include image processing and machine learning.