IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 5, 2020, accepted May 11, 2020, date of publication May 18, 2020, date of current version June 8, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2995369

A Local Feature Descriptor Based on Rotational Volume for Pairwise Registration of Point Clouds

XIONG FENGGUANG[1], DONG BIAO[1], HUO WANG[2], PANG MIN[1], KUANG LIQUN[1], AND HAN XIE[1]

[1]School of Data Science and Technology, North University of China, Taiyuan 030051, China
[2]Zhejiang Hanchine AI Tech. Company Ltd., Hangzhou 310000, China

Corresponding authors: Xiong Fengguang (hopenxfg@nuc.edu.cn) and Han Xie (hanxie@nuc.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61672473, in part by the Shanxi Province Key Research and Development Technology Project under Grant 201803D121081 and Grant 201903D121147, and in part by the Natural Science Foundation of Shanxi Province of China under Grant 201901D111150.

ABSTRACT Aiming to problems in the pairwise registration of point clouds, such as keypoints are difficult to describe accurately, corresponding points are difficult to match accurately and convergence speed is slow due to uncertainty of initial transformation matrix, we propose a novel feature descriptor based on ratio of rotational volume to describe effectively keypoints, and on the basis of the feature descriptor, we proposed an improved coarse-to-fine registration pipeline of point clouds, in which we use coarse registration to obtain a good initial transformation matrix and then use fine registration based on a modified ICP algorithm to obtain an accurate transformation matrix. Experimental results show that our proposed feature descriptor has a good robustness to rotation, noise, scale and varying mesh resolution, less storage space and faster running speed than PFH, FPFH, SHOT and RoPS descriptors, and our improved pairwise registration pipeline is very effective to solve the problems in the pairwise registration of point clouds.

INDEX TERMS Point cloud, feature descriptor, rotational volume, boundary point detection, transformation estimation.

I. INTRODUCTION AND RELATED WORK

With the development of laser scanning technology, the capacity to capture 3D spatial data has been enhanced greatly. However, limit to the field of view of the scanning device, each scanning can only capture partial point cloud of 3D object. In order to obtain complete 3D object, it is necessary to use registration technology to align partial point clouds into a global coordinate framework. The core work of registering partial point clouds is to find the corresponding position and orientation of a pairwise point clouds in a global coordinate framework, which is also called pairwise registration [1]. At present, the most popular pairwise registration is the Iterative Closest Point (ICP) algorithm [2], [3]. The ICP searches the nearest point pairs from a pair of point clouds, namely a source point cloud and a target point cloud, and estimate the rigid body transformation and then apply it into the source point cloud. The process of search and transformation is performed iteratively until the convergence is obtained.

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

Because input pairwise point clouds might be only partially overlapping, complete point-to-point correspondences are always hard to be found. Many efforts have been made into the field of feature selection [4] that could improve the correspondence problem. Robust feature descriptors such as integral volume descriptors [5], moment invariants [6] and spherical harmonic invariants [7] have been proposed and used for registration of pairwise point clouds. The above feature descriptors are invariant to translation and rotation, but are still sensitive to noise [8], so it is hard to find correct correspondences by using them. In recent years, people have also put forward some local feature descriptors such as point feature histogram (PFH) [9], rotational projection statistics (RoPS) [10]–[13], signature of histogram of orientations (SHOT) [14], multi-attribute statistics histograms (MaSH) [15], local feature statistic histogram (LFSH) [16], binary shape context descriptor [17], [18], voxel-based buffer-weighted binary descriptor [19], 3D descriptor with global structural frames and local signatures of histograms [20], signature of geometric centroids descriptor [21], etc. These local feature descriptors

have a certain improvement on finding correct point-to-point correspondences in noisy point clouds, and outperform global feature descriptors in pairwise registration of point clouds. Moreover, there are also methods generating descriptors based on machine learning, such as [22]–[25]. However, there also exist some problems for them, such as time efficiency or space storage. Therefore, finding a feature descriptor robust to nuisance and highly efficient is still a great challenge for researchers who have been studying pairwise registration of point clouds.

The contributions of this paper are summarized as follows:

- A feature descriptor based on rotational volume is presented, which simultaneously achieves high descriptiveness, robustness and time efficiency for the purpose of local surface representation and surface matching.
- A boundary point detection algorithm is presented, which can obtain effectively points on the boundary; and based on the algorithm, we can remove effectively keypoints near to boundary, which can enhance the descriptiveness of keypoints.
- An improved rigid transformation estimation via SAC is presented, which can get a transformation estimation according to a pair of keypoint and their local reference frames, and can get an optimal transformation estimation according to ratio of inliers.
- An improved coarse-to-fine pairwise registration pipeline for point clouds is proposed, which includes keypoint detection with boundary removed, feature description and matching, transformation estimation, fine registration.

The remainder of this paper is organized as follows. In Section 2, we describe the process of our proposed feature descriptor based on rotational volume in detail. In Section 3, we describe our improved coarse-to-fine pairwise registration of point clouds, especially including a boundary point detection algorithm and a robust rigid transformation estimation via SAC. Section 4 discusses the experimental results about our proposed feature descriptor, keypoint detection and pairwise registration before concluding in Section 5.

II. A LOCAL FEATURE DESCRIPTOR BASED ON ROTATIONAL VOLUME

Local feature-based methods for pairwise registration aligns point clouds by using consistent point-to-point correspondences which are usually generated by matching local feature descriptors between pairwise point clouds. Therefore, the local feature descriptor should be robust and distinctive to various nuisances such as noise, scale and varying point cloud resolution, in order to obtain sufficient correct point-to-point correspondences. Moreover, local feature descriptor should be invariant to rigid transformation including translation and rotation. In this section, a fast and robust local feature descriptor based on rotational volume will be first introduced and then similarity measure of the feature descriptor will be given.

In this section, we first generate a right-angled trapezoid by sequentially connecting two points within a neighborhood of a point cloud and their projected points in a special plane, which is a tangent plane of a neighborhood of a query point in a point cloud. Then, we compute the volume of a shape, which is generated by rotating the above trapezoid around a special axis which is the normal of the above special plane and is parallel with a right-angled side of the above trapezoid. We name the volume as rotational volume since it is obtained by rotation. Given a query point, its feature descriptor based on rotational volume describes accumulation of rotational volume of the query point's neighbors distributed on different regions. For each point on a point cloud, its local surface formed by its neighboring points has subtle fluctuations different from other local surfaces, which causes its feature descriptor based on rotational volume to be different from other point's feature descriptors. At the same time, for each point on a point cloud, its local surface will not change with rotation and translation of the point cloud, which also makes its feature descriptor based on rotational volume be invariant to rotation and translation. Moreover, for each point on a point cloud, geometric structure of its local surface will only have a little change when there are various nuisances such as noise, scale and varying point cloud resolution, which can make its feature descriptor based on rotational volume be robust to nuisance. The rest of this section will introduce how feature descriptor of a query point is generated and the similarity between our proposed descriptors.

1) SELECTION OF NEIGHBORHOOD AND GENERATION OF LOCAL REFERENCE FRAME

Assume that P is input point cloud, which contains N points $\{p_1, p_2, \ldots, p_N\}$. Given a query point $p_i (i \in [0, N-1])$, its neighborhood is a sphere of radius r centered at p_i, and all the points excluding p_i in this neighborhood are named as neighboring points or neighbors of the query point. The sphere of p_i is named as $\boldsymbol{S_i}$, and the set of the neighbors are named as $\boldsymbol{Nbhd}(\boldsymbol{p_i})$. Moreover, KD-Tree is used to search the neighbors of a query point in this paper, which can accelerate the searching speed.

For a query point $\boldsymbol{p_i}$, we denote its neighbors by $\boldsymbol{Nbhd}(\boldsymbol{p_i}) = \{\boldsymbol{p_i^j} | j = 0, 1, \ldots, k-1\}$, and with those neighbors, we can define a local reference frame $\boldsymbol{LRF}_i$ centered at the query point $\boldsymbol{p_i}$ by using eigen analysis of covariance matrix as follows.

(1) Compute a weight for $\boldsymbol{p_i}$ inversely related to its neighbors as follows

$$\boldsymbol{w_i^j} = \frac{1}{\left\| p_i^j - p_i \right\|}. \tag{1}$$

This weight is used to compensate for uneven sampling of the neighboring points, so that the neighboring points at sparsely sampled regions contribute more than the neighboring points at densely sampled regions.

(2) Compute a weighted covariance matrix $\boldsymbol{cov}(\boldsymbol{p_i})$ for $\boldsymbol{p_i}$ using its neighbors. The 3×3 covariance matrix is given by

$$\boldsymbol{cov}\left(\boldsymbol{p_i}\right)=\sum_{j=1}^{k}w_i^j\left(p_i^j-p_i\right)\left(p_i^j-p_i\right)^T/\sum_{j=1}^{k}w_i^j. \quad (2)$$

(3) Compute its eigenvalues $\{\lambda_i^1,\lambda_i^2,\lambda_i^3\}$ in the order of decreasing magnitude and their corresponding eigenvectors $\{\boldsymbol{e_i^1},\boldsymbol{e_i^2},\boldsymbol{e_i^3}\}$, which define repeatable, orthogonal directions in presence of noise and clutter. It is worth noting that, compared to [26] and [27], the third eigenvector e_i^3 no longer represents the normal direction of the total least squares estimation and sometimes it is obviously different from it. But this does not affect performance, since in the case of local feature descriptors what matters is a highly repeatable and robust triplet of orthogonal directions, and not its geometrical or topological meaning.

(4) Use p_i as the origin, and use e_i^1,e_i^3 and their cross product $e_i^1\times e_i^3$ as *u, w*, and *v* axes respectively to define a local reference frame $\boldsymbol{LRF}_i$ at $\boldsymbol{p_i}$. Since an eigenvector of the covariance matrix computes a direction in the 3D space based on the amount of point position variations, its orientation has a 180° ambiguity. Therefore, we have two choices for the orientation of *u* and *w* axis. We determine the sign on the *u* and *w* axis according to the principle which the sign of local axis should be coherent with the majority of the vectors formed by p_i and its neighbors. In the following, we refer to the three eigenvectors in decreasing eigenvalue order as the $u^+, v^+ and\ w^+$ axis respectively. With $u^-, v^- and\ w^-$ denote the opposite vectors of $u^+, v^+ and\ w^+$ respectively. The criterion that a vector formed by p_i and an its neighbor is coherent with u^+ axis is given by

$$Sum_{p_i^j}=\begin{cases}1, & (p_i^j-p_i)\cdot u^+\geq 0\\ -1, & (p_i^j-p_i)\cdot u^+<0\end{cases}, \quad (3)$$

in which, the vector is coherent with u^+ axis when $Sum_{p_i^j}$ is equal 1, and the vector is coherent with u^- axis when $Sum_{p_i^j}$ is equal -1. Therefore, *u* axis can be given by

$$u=\begin{cases}u^+, & \sum_{i=0}^{k-1}Sum_{p_i^j}\geq 0\\ u^-, & \sum_{i=0}^{k-1}Sum_{p_i^j}<0\end{cases}. \quad (4)$$

The same procedure is used to disambiguate the *w* axis. Finally, *v* axis is obtained as $w\times u$. As far, the local reference frame $\boldsymbol{LRF_i(p_i,u,v,w)}$ has been constructed.

2) SPACE PARTITIONING OF NEIGHBORHOOD

Taking the intersection $\boldsymbol{p_i^{north}}$ between *w*'s positive direction and $\boldsymbol{S_i}$ as north pole, the intersection $\boldsymbol{p_i^{south}}$ between $\boldsymbol{w}$'s negative direction and $\boldsymbol{S_i}$ as south pole, and $\boldsymbol{w}$ as middle axis, we divide $\boldsymbol{S_i}$ into $\boldsymbol{m}$ uniform regions, along the longitude of $\boldsymbol{S_i}$. We denote a region as $\boldsymbol{Region_i^k}(k\in[0,m-1])$ and the set of $\boldsymbol{Region_i^k}$ as $\boldsymbol{Region_i}$. In order to determine the neighboring points included in each $\boldsymbol{Region_i^k}$, we project $\boldsymbol{Region_i}$ into a plane which is parallel to the $\boldsymbol{uv}$ plane and passes the point $\boldsymbol{p_i^{south}}$. We denote the projection of $\boldsymbol{S_i}$ as $\boldsymbol{C_{S_i}}$ which is a circle, and denote the projection of $\boldsymbol{Region_i^k}$ on $\boldsymbol{C_{S_i}}$ as $\boldsymbol{Sector_i^k}(k\in[0,m-1])$ which is a sector, and we denote the set of $\boldsymbol{Sector_i^k}$ as $\boldsymbol{Sector_i}$ in which a $\boldsymbol{Sector_i^k}$ is corresponded with a $\boldsymbol{Region_i^k}$. The space partitioning for the query point's neighborhood is shown as Fig. 1.

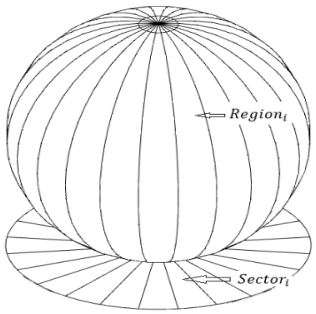


FIGURE 1. Space partitioning for query a point's neighborhood.

Then, for each neighboring point, we need to determine its corresponding $\boldsymbol{Region_i^k}$. The problem is hard to be solved in 3D space, so we project each neighboring point into 2D space and determine its projection point's corresponding $\boldsymbol{Sector_i^k}$. Thus, corresponding $\boldsymbol{Region_i^k}$ of each neighboring point can be determined according to the correspondence between $\boldsymbol{Sector_i^k}$ and $\boldsymbol{Region_i^k}$. The detail process is shown as follows:

(1) For a neighbor $\boldsymbol{p_i^j}$ in $\boldsymbol{Nbhd}(\boldsymbol{p_i})$, projecting it into the $\boldsymbol{uv}$ plane according to the following formula

$$q_i^j=p_i^j-\left(w\cdot\overrightarrow{p_ip_i^j}\right)*w, \quad (5)$$

in which $\boldsymbol{q_i^j}$ is projection point of $\boldsymbol{p_i^j}$ on the $\boldsymbol{uv}$ plane, and $*$ denotes a multiplication between a scalar and a vector.

(2) Connect $\boldsymbol{p_i}$ with $\boldsymbol{q_i^j}$ and compute angle $\boldsymbol{\alpha_i^j}$ between the vector $\overrightarrow{p_iq_i^j}$ and $\boldsymbol{u}$ axis according to the following formula

$$\boldsymbol{\alpha_i^j}=\begin{cases}arccos\left(\dfrac{\overrightarrow{p_iq_i^j}\cdot\vec{u}}{\left\|\overrightarrow{p_iq_i^j}\right\|}\right), & \left(\vec{u}\times\overrightarrow{p_iq_i^j}\right)\cdot\vec{w}\geq 0\\ 2\pi-arccos\left(\dfrac{\overrightarrow{p_iq_i^j}\cdot\vec{u}}{\left\|\overrightarrow{p_iq_i^j}\right\|}\right), & \left(\vec{u}\times\overrightarrow{p_iq_i^j}\right)\cdot\vec{w}<0\end{cases}, \quad (6)$$

and the schematic diagram is shown as Fig. 2.

(3) Determine the sector that $\boldsymbol{q_i^j}$ locates in according to the value of $\boldsymbol{\alpha_i^j}$. We divide $\boldsymbol{S_i}$ into $\boldsymbol{m}$ uniform regions, so $\boldsymbol{C_{S_i}}$, the projection plane of $\boldsymbol{S_i}$, is also divided into $\boldsymbol{m}$ uniform regions, and the value of central angle of each $\boldsymbol{Sector_i^k}$ is $\mathbf{360/m}$ that we denote as $\boldsymbol{\beta}$, and then the number of sector,

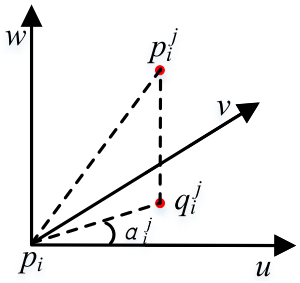


FIGURE 2. Projecting a neighboring point into *uv* plane.

which $\boldsymbol{q_i^j}$ locates in, can be defined as follow

$$n_i^j = \left\lfloor \alpha_i^j / \beta \right\rfloor + 1, \tag{7}$$

in which $\boldsymbol{n_i^j}$ is number of sectors in $\boldsymbol{Sector_i^k}$ for a neighbor $\boldsymbol{p_i^j}$ in $\boldsymbol{Nbhd}(\boldsymbol{p_i})$.

3) COMPUTATION OF ROTATIONAL VOLUME AND GENERATION OF FEATURE DESCRIPTOR

By space partitioning of neighborhood, we can determine the neighboring points included in each $\boldsymbol{Region_i^k}$, then we can calculate sum of rotational volume for each $\boldsymbol{Region_i^k}$ and $\boldsymbol{m}$ sums of rotational volume form a vector, and finally we are able to obtain the feature descriptor of the query point $\boldsymbol{p_i}$ **by** normalize the above vector. Therefore, the core, that generates a feature descriptor for a query point, is to calculate sum of rotational volume for each $\boldsymbol{Region_i^k}$ and its process is detailed as follows.

(1) Suppose a $\boldsymbol{Region_i^k}(k \in [0, m-1])$ including the neighboring points $\boldsymbol{Nbhd}\left(\boldsymbol{p_i^k}\right) = \{\boldsymbol{p_i^{k1}}, \boldsymbol{p_i^{k2}}, \ldots, \boldsymbol{p_i^{kt}}\}$, and assign zero to $\boldsymbol{V_i^k}$ which denotes sum of rotational volume in $\boldsymbol{Region_i^k}$.

(2) Take out two adjacent points in order from $\boldsymbol{Nbhd}\left(\boldsymbol{p_i^k}\right)$ and form ***t-1*** sequences such as $< \boldsymbol{p_i^{k1}}, \boldsymbol{p_i^{k2}} >$, $< \boldsymbol{p_i^{k2}}, \boldsymbol{p_i^{k3}} >, \ldots, < \boldsymbol{p_i^{k(t-1)}}, \boldsymbol{p_i^{kt}} >$.

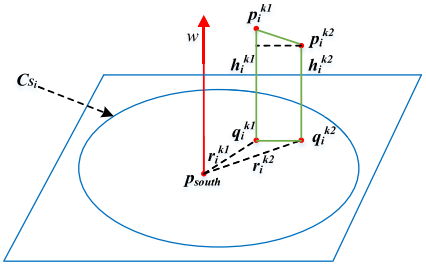


FIGURE 3. Space location of two neighboring points and their projection points.

(3) Take first sequence $< \boldsymbol{p_i^{k1}}, \boldsymbol{p_i^{k2}} >$ and their projection points $\boldsymbol{q_i^{k1}}$ and $\boldsymbol{q_i^{k2}}$ respectively in the plane where $\boldsymbol{C_{S_i}}$ locates. Fig. 3 shows space location of these points, in which $\boldsymbol{p_i^{k1}}$, $\boldsymbol{p_i^{k2}}$, $\boldsymbol{q_i^{k1}}$ and $\boldsymbol{q_i^{k2}}$ form a right-angled trapezoid $\boldsymbol{T_i^{k1k2}}$ which is marked out with green line segments, $\boldsymbol{h_i^{k1}}$ denotes the distance between $\boldsymbol{p_i^{k1}}$ and $\boldsymbol{q_i^{k1}}$, $\boldsymbol{h_i^{k2}}$ denotes the distance between $\boldsymbol{p_i^{k2}}$ and $\boldsymbol{q_i^{k2}}$, $\boldsymbol{r_i^{k1}}$ denotes the radius of a circle formed by $\boldsymbol{p_{south}}$ and $\boldsymbol{q_i^{k1}}$ and centered at $\boldsymbol{p_{south}}$, $\boldsymbol{r_i^{k2}}$ denotes the radius of a circle formed by $\boldsymbol{p_{south}}$ and $\boldsymbol{q_i^{k2}}$ and centered at $\boldsymbol{p_{south}}$. The right-angled trapezoid $\boldsymbol{T_i^{k1k2}}$ rotates 360 degree around $\boldsymbol{w}$ axis and it will generate a new shape that consists of a solid frustum, a solid cylinder and a hollow cylinder which are shown as Fig. 4. So, the volume $\boldsymbol{V_i^{tmp}}$ of this shape can be calculated as follows.

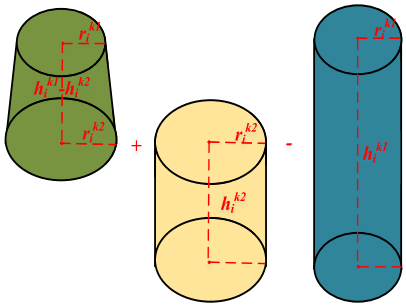


FIGURE 4. Three parts of a shape generated by rotation.

$$\begin{aligned} \boldsymbol{V_i^{tmp}} &= \boldsymbol{1/3 * \pi * \left(h_i^{k1} - h_i^{k2}\right)} \\ &\quad \boldsymbol{* \left(r_i^{k1} * r_i^{k1} + r_i^{k2} * r, _i^{k2} + r_i^{k1} * r_i^{k2}\right)} \\ &\quad \boldsymbol{+ \pi * h_i^{k2} * r_i^{k2} * r_i^{k2} - \pi * h_i^{k1} * r_i^{k1} * r_i^{k1}}, \end{aligned} \tag{8}$$

$$\boldsymbol{h_i^{k1}} = \overrightarrow{\boldsymbol{p_{south} p_i^{k1}}} \cdot \vec{\boldsymbol{w}}, \tag{9}$$

$$\boldsymbol{h_i^{k2}} = \overrightarrow{\boldsymbol{p_{south} p_i^{k2}}} \cdot \vec{\boldsymbol{w}}, \tag{10}$$

$$\boldsymbol{r_i^{k1}} = \sqrt[2]{\left\|\boldsymbol{p_{south} p_i^{k1}}\right\|^2 - \left\|\boldsymbol{h_i^{k1}}\right\|^2}, \tag{11}$$

$$\boldsymbol{r_i^{k2}} = \sqrt[2]{\left\|\boldsymbol{p_{south} p_i^{k2}}\right\|^2 - \left\|\boldsymbol{h_i^{k2}}\right\|^2}. \tag{12}$$

(4) Assign the result of $\boldsymbol{V_i^k}$ plus $\boldsymbol{V_i^{tmp}}$ to $\boldsymbol{V_i^k}$ and then take next sequence from $\boldsymbol{Nbhd}\left(\boldsymbol{p_i^k}\right)$ to execute step (3) until all the sequences have been handled.

(5) Calculate $\boldsymbol{V_i^k}$ of each $\boldsymbol{Region_i^k}$ according to above steps and form a vector *v_ds* with $\boldsymbol{m}$ dimension.

(6) Normalize *v_ds* by making each element of *v_ds* be divided by 1-norm of *v_ds*. The new vector is the feature descriptor of the query point $\boldsymbol{p_i}$.

4) SIMILARITY MEASURE OF FEATURE DESCRIPTOR

What rotational volume describes is the ratio of volumes where each volume is generated by rotating a right-angled trapezoid which is formed by two points and its projected points. A neighborhood of a query point will be divided into $\boldsymbol{m}$ regions, and neighboring points in each region will generate a rotational volume. When geometric structure is different between a query point and a target point, their rotational volume in each region will be different with each other, which causes feature descriptors of these two points to be different with each other. When geometric structure is

same between a query point and a target point, their rotational volume will be same in each region, which causes feature descriptors of these two points to be same with each other. When geometric structure is similar between a query point and a target point, their rotational volume will be different in some regions and will be similar in some other regions, which causes feature descriptors of these two points to be similar. Geometric structure of two points determine whether their feature descriptors based on rotational volume are similar or not, which is the principle of exclusivity for two feature descriptors generated by rotational volume.

The similarity between two feature descriptors based on rotational volume can be described by the distance between two feature descriptors, and the closer the distance is, the more similar they are. Supposing two feature descriptors $\boldsymbol{Ds_1}=\{\boldsymbol{V_1^1},\boldsymbol{V_1^2},\ldots,\boldsymbol{V_1^m}\}$ and $\boldsymbol{Ds_2}=\{\boldsymbol{V_2^1},\boldsymbol{V_2^2},\ldots,\boldsymbol{V_2^m}\}$, their similarity can be defined as follow

$$\boldsymbol{d}(\boldsymbol{Ds_1},\boldsymbol{Ds_2})=\sqrt{\sum_{i=1}^{m}\left\|\boldsymbol{V_1^i}-\boldsymbol{V_2^i}\right\|^2}. \tag{13}$$

III. COARSE-TO-FINE PAIRWISE REGISTRATION OF POINT CLOUDS

Based on our proposed feature descriptor, an improved coarse-to-fine method for pairwise registration of point clouds is described in this section. It consists of five major modules: keypoint detection with boundary removed, feature descriptor generation, feature matching, robust transformation estimation and fine registration. Fig. 5 shows pairwise registration pipeline of point clouds.

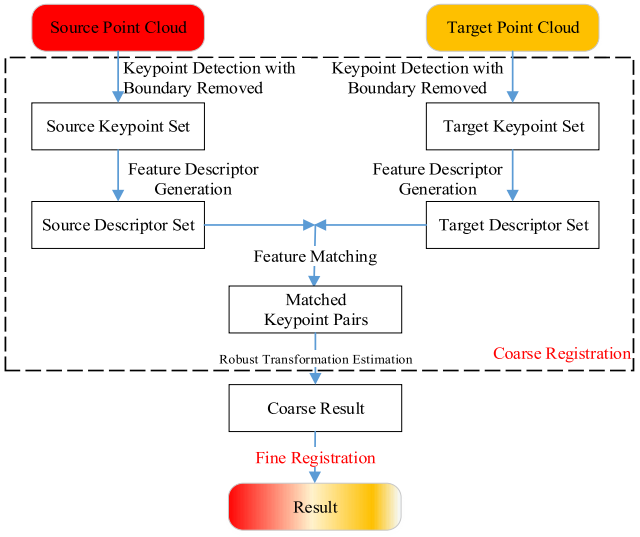


FIGURE 5. Pairwise registration pipeline of point cloud.

A. KEYPOINT DETECTION WITH BOUNDARY REMOVED

Based on ISS detector, keypoint detection with boundary removed will remove keypoints on the boundary from keypoint set. Fig. 6 shows the pipeline of keypoint detection.

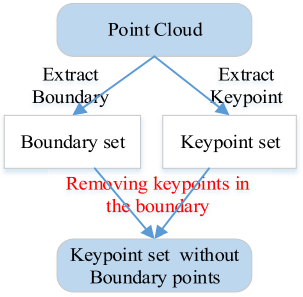


FIGURE 6. Pipeline of keypoint detection.

1) ISS DETECTOR

ISS detector is based on the Eigenvalue decomposition of the scatter matrix $\sum(\boldsymbol{p})$ of neighboring points centered at p, i.e.

$$\sum(p)=\frac{1}{N}\sum_{j=1}^{N}(p_j-\mu_p)(p_j-\mu_p)^T \text{ with}$$
$$\mu_p=\frac{1}{N}\sum_{j=1}^{N}p_j, \tag{14}$$

in which $\boldsymbol{p_j}$ is a neighboring point centered at $\boldsymbol{p}$.

Given $\sum(\boldsymbol{p})$, its eigenvalues in the order of decreasing magnitude are denoted $\boldsymbol{\lambda_1},\boldsymbol{\lambda_2},\boldsymbol{\lambda_3}$, and points whose ratio between two successive eigenvalues is below a threshold are retained, i.e.

$$\lambda_2(p)/\lambda_1(p)<th_{12}\wedge\lambda_3(p)/\lambda_2(p)<th_{23}. \tag{15}$$

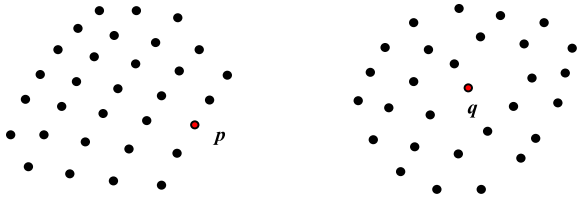
The rationale is to avoid detecting keypoints at points exhibiting a similar spread along the principal directions where a repeatable canonical reference frame cannot be established. Among remaining points, the keypoint is determined by the magnitude of the smallest eigenvalue is local minima or maxima, i.e.

$$\lambda_3(p)>\lambda_3(\hat{p})\vee\lambda_3(p)<\lambda_3(\hat{p}), \tag{16}$$

in which $\boldsymbol{\lambda_3}(\hat{p})$ is the smallest eigenvalue of any neighboring point centered at $\boldsymbol{p}$. in which $\boldsymbol{p_j}$ is a neighboring point centered at $\boldsymbol{p}$.

2) BOUNDARY POINT DETECTION ALGORITHM

Mian *et al.* [28] proposed a boundary detection based on that the number of points in the neighborhood of boundary point is much lower than the number of points in the neighborhood of non-boundary point. The method accords with people's recognition to boundary points, but in practice, its effect is not so good, and then the average number of neighboring points and the number of neighboring points of each point need to be calculated in advance, which is very time-consuming and is not robust to noise. The boundary point detection in this paper is since most neighboring points of a boundary point are distributed on one side of the boundary point, which is shown in Fig. 7. According to this fact, if a point is a boundary point, most of its neighboring points are distributed on one side



(a) $\boldsymbol{p}$ is a boundary point (b) $\boldsymbol{q}$ is a non-boundary point

FIGURE 7. Distribution of a boundary and a non-boundary point with their neighboring points.

which causes a notch on the other side of the boundary point. Therefore, when these neighboring points are projected on tangent plane of the boundary point, there will also be a notch in the line between the projection points and the boundary point. Detecting angle of the notch can determine whether the point is a boundary point or not.

The process of determining whether a point $\boldsymbol{p}$ in a point cloud $\mathcal{P}$ is a boundary point is shown as follows:

(1) Determine $\boldsymbol{p's}$ neighboring points centered at $\boldsymbol{p}$ with radius $\boldsymbol{r}$ and name them as $\boldsymbol{N(p)}$, i.e.

$$N(p) = \{p_j | p_j \in \mathrm{P}, \|p_j - p\| \leq r\}. \tag{17}$$

(2) Project $\boldsymbol{N(p)}$ to a tangent plane which is centered at $\boldsymbol{p}$ and whose normal is $\boldsymbol{n}$ with following formula

$$q_j = p_i - \left(n \cdot \overrightarrow{pp_j}\right) \times n, \tag{18}$$

and name these projection points as $\boldsymbol{N(q)}$.

(3) Find the nearest point to $\boldsymbol{p}$ and name it as $\boldsymbol{q_u}$, and then construct a local reference frame $\boldsymbol{(p, u, v, w)}$ whose $\boldsymbol{w}$ axis is $\boldsymbol{p's}$ normal $\boldsymbol{n}$, $\boldsymbol{u}$ axis is $\overrightarrow{pq_u}/\left|\overrightarrow{pq_u}\right|$, $\boldsymbol{v}$ axis is $\boldsymbol{u} \times \boldsymbol{w}$, origin point is $\boldsymbol{p}$.

(4) Calculate the included angle between $\boldsymbol{k}$ vectors $\overrightarrow{\boldsymbol{pq_j}}$ and $\boldsymbol{u}$ axis respectively along the clockwise direction, and name the angles as $\boldsymbol{S} = (\boldsymbol{\alpha_1, \alpha_2, \ldots, \alpha_k})$.

(5) Sort $\boldsymbol{S}$ in ascending order and name it as $\boldsymbol{S'} = (\boldsymbol{\alpha_1', \alpha_2', \ldots, \alpha_k'})$, and then calculate the included angle between adjacent angles, i.e.

$$L_i = \alpha_{i+1}^{'} - \alpha_i'. \tag{19}$$

in which $\boldsymbol{L_i}$ describes the included angle between two adjacent vectors on the tangent plane which is shown as Fig. 8.

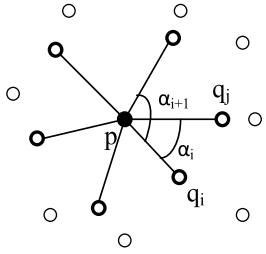


FIGURE 8. Included angle on tangent plane.

(6) Find maximum angle $\boldsymbol{L_{max}}$ and determine $\boldsymbol{p}$ is a boundary point if $\boldsymbol{L_{max} > L_{th}}$. Here, $\boldsymbol{L_{th}}$ is a threshold whose value can be adjusted.

3) REMOVING KEYPOINTS ON THE BOUNDARY

The process of removing keypoints on the boundary is show as follows:

(1) Traverse all points in a point cloud $\mathcal{P}$ and detect the entire boundary points into a set named as ***Boundaries*** according to the above boundary point detection algorithm.

(2) Traverse all points in a point cloud $\mathcal{P}$ and detect the entire keypoints into a set named as ***Keypoints*** according to the above ISS detector.

(3) For each keypoint kp in ***Keypoints***, find its nearest distance $\boldsymbol{d_{min}}$ to a boundary point in ***Boundaries*** and remove $\boldsymbol{kp}$ from ***Keypoints*** if $\boldsymbol{d_{min} < 5mr}$.

B. FEATURE DESCRIPTOR GENERATION AND FEATURE MATCHING

Let $\mathcal{P}^s$ and $\mathcal{P}^t$ respectively represent source and target point cloud. Before feature descriptor generation, we first detect respectively keypoint for $\mathcal{P}^s$ and $\mathcal{P}^t$, and keypoints are denoted by $\boldsymbol{KPS^s} = \{\boldsymbol{kp_1^s, kp_2^s, \ldots, kp_n^s}\}$ and $\boldsymbol{KPS^t} = \{\boldsymbol{kp_1^t, kp_2^t, \ldots, kp_m^t}\}$, respectively. Then, our proposed feature descriptor based on rotational volume for $\boldsymbol{KPS^s}$ and $\boldsymbol{KPS^t}$ are calculated respectively as $\boldsymbol{DS^s} = \{\boldsymbol{d_1^s, d_2^s, \ldots, d_n^s}\}$ and $\boldsymbol{DS^t} = \{\boldsymbol{d_1^t, d_2^t, \ldots, d_m^t}\}$.

After having generated feature descriptors for keypoints of point clouds, we need generate correspondence between $\boldsymbol{DS^s}$ and $\boldsymbol{DS^t}$ by feature matching. For each feature descriptor $\boldsymbol{d_i^s}$ in $\boldsymbol{DS^s}$, we find a closest feature descriptor $\boldsymbol{d_j^t}$ from $\boldsymbol{DS^t}$ as

$$d_j^t = \arg \min_{k=1,2,\ldots,m} (\left\| d_k^t - d_i^s \right\|). \tag{20}$$

By matched feature pair $< \boldsymbol{d_i^s, d_j^t} >$, we can find matched keypoint pair $< \boldsymbol{kp_i^s, kp_j^t} >$ which is a point-to-point correspondence. KD tree is here used to accelerated feature matching process. It is noted that $L2$ norm is always employed to measure the similarity between two feature descriptors, as in [14], [29]. Therefore, $\boldsymbol{n}$ correspondences, denoted by a set $\boldsymbol{C}$, can be obtained between $\boldsymbol{DS^s}$ and $\boldsymbol{DS^t}$. However, only a part of them is determined to be correct since $\mathcal{P}^s$ and $\mathcal{P}^t$ are usually partially overlapped.

C. ROBUST TRANSFORMATION ESTIMATION VIA SAC

As far, a correspondence set $\mathcal{C}$ is established between $\mathcal{P}^s$ and $\mathcal{P}^t$. The aim of robust transformation estimation is to estimate an optimal rigid transformation $\mathcal{T}$ from $\mathcal{C}$ in order to transform $\mathcal{P}^s$ to $\mathcal{P}^t$. Popular methods include the RANSAC algorithm [30], [31] and its variants [32], [33]. However, they always suffer from weak robustness to high time consumption. Here, we propose a sample consensus (SAC) to estimate $\mathcal{T}$ with a quicker speed and a higher accuracy.

Our proposed SAC method also estimates $\mathcal{T}$ from $\mathcal{C}$ within a fixed iteration. However, the difference is that our proposed method uses a correspondence to estimate a rigid transformation instead of three correspondences. Theoretically, a correspondence is insufficient to estimate a rigid transformation between 3D point clouds. However, if a correspondence is

orientated with LRF, it is sufficient to estimate a rigid transformation [34]. The formulas of estimating a rigid transformation are shown as follows

$$R = \left(LRF_i^s\right)^T \left(LRF_j^t\right), \tag{21}$$

$$t = kp_j^t - kp_i^s R, \tag{22}$$

in which $< \boldsymbol{kp_j^t}, \boldsymbol{kp_i^s} >$ is a keypoint pair, $\boldsymbol{LRF_i^s}$ and $\boldsymbol{LRF_j^t}$ are local reference frame of $\boldsymbol{kp_i^s}$ and $\boldsymbol{kp_j^t}$ respectively, $\boldsymbol{R}$ and $\boldsymbol{t}$ denote a rotation matrix and a translation vector of a rigid transformation. It is noted that LRF of each keypoint does not be calculated here, and it has been calculated in the process of feature descriptor generation, because our proposed feature descriptor based on rotational volume need to construct LRF for each keypoint in advance. In contrast to classical RANSAC, the advantage of a correspondence is that the computational complex is reduced from $\boldsymbol{O(n^3)}$ to $\boldsymbol{O(n)}$ where $\boldsymbol{n}$ is the number of correspondences.

By above method, we can achieve n rigid transformations by $\boldsymbol{n}$ correspondences, and the optimal rigid transformation is satisfied with maximized sample consensus. We use ratio of inliers to measure the level of sample consensus and so the maximized sample consensus is corresponding to the biggest ratio of inliers. In order to calculate a ratio of inliers for a correspondence, we first calculate a rigid transformation by formula 21-22 and name it as $\boldsymbol{t_i}$; then, we transform $\mathcal{P}^s$ via $\boldsymbol{t_i}$ and name the transformed point cloud as $\mathcal{P}^{s'}$; and then, we search its nearest point in $\mathcal{P}^t$ for each point in $\mathcal{P}^{s'}$, and we regard a point $\boldsymbol{p_i^{s'}}$ in $\mathcal{P}^{s'}$ as a inlier if the distance between $\boldsymbol{p_i^{s'}}$ and its nearest point $\boldsymbol{p_j^t}$ in $\mathcal{P}^t$ is less than a threshold, i.e.

$$\left\| p_i^{s'} - p_j^t \right\| < \epsilon_d, \tag{23}$$

and finally the ratio of inliers can be defined as follow

$$rt_{inliers} = n_{inlines} / n_s, \tag{24}$$

in which $\boldsymbol{n_{inlines}}$ is denoted as the number of inliers, and $\boldsymbol{n_s}$ is denoted as the number of points in $\mathcal{P}^s$. Therefore, we can achieve the optimal rigid transformation $\mathcal{T}^*$ and then $\mathcal{T}^*$ is applied to transform $\mathcal{P}^s$ and generate a transformed point cloud $\mathcal{P}^{s^*}$. Here, coarse registration of pairwise point clouds has been complete.

D. FINE REGISTRATION

After coarse registration, pairwise point clouds almost have been aligned well and further refining pairwise point clouds uses a modified ICP algorithm [35] proposed recently, which can exhibit accurate performance even when a good initialization is not reliably available. Via the fine registration, we can merge the input pairwise pint clouds into a seamless point cloud.

IV. EXPERIMENTS

In this section, experiments include feature descriptor based on rotational volume (section 4.1), keypoint detection with boundary removed (section 4.2) and coarse pairwise registration (section 4.3). In experiments of keypoint detection, we will show detection process of keypoints, including the keypoints detected by ISS detector, and boundary points detected by our proposed boundary point detection algorithm, and ultimate keypoints after removing keypoints near to boundary. In experiments of pairwise registration, we will show results of pairwise matching by keypoint detection, feature descriptor generation and feature matching, and will give registration error of our proposed pairwise registration, and will compare our proposed feature descriptor with state-of-the-art feature descriptors based on our improved pairwise registration pipeline.

A. EXPERMENTS OF FEATURE DESCRIPTOR BASED ON ROTATIONAL VOLUME

The task of this section is to compare our proposed feature descriptor with PFH, FPFH, SHOT, RoPS in terms of run time, robustness to rotation, noise, scale and varying mesh resolution. PFH, FPFH, SHOT and RoPS are implemented in PCL [36](Point Cloud Library). In these experiments, the parameter m in our proposed descriptor is set to 24, and a normal vector of a point in the point cloud is computed on the nearest 10 neighboring points.

1) DATASET

In these experiments, we use the Stanford 3D Scanning Repository [37] to test these feature descriptors. There are a total of nine 3D models in the Stanford 3D Scanning Repository: Bunny, Drill bit, Happy Buddha, Dragon, Armadillo, Lucy, Asian Dragon, Vellum manuscript and Thai Statue. These models were scanned using a Cyberware 3030 MS scanner, Stanford Large Statue Scanner or XYZ RGB auto-synchronized camera with a resolution of 100 microns. However, only three models, namely Armadillo, Bunny and Happy Buddha, are used in our experiments because these three models were all scanned by a Cyberware 3030 MS scanner and have same mesh resolution ($\boldsymbol{mr}$), which is helpful for analyzing and comparing the performance of these feature descriptors. These three models are shown in Fig. 9. It is worth noting that we convert the model's data format from original ply (Polygon File Format) format to pcd (point cloud data) format using CloudCompare Software so that those models can be used in our algorithm.

(a)Armadillo (c) Bunny (d) Happy Buddha

FIGURE 9. Models used in these experiments.

2) RUN TIME

Table 1 shows the number of points in each model and the number of keypoints in each model, and these keypoints are

TABLE 1. Model information.

Model	The number of	The number of key point
Bunny	35947	50
Armadillo	172974	192
Happy buddha	144647	92

TABLE 2. Runtime of generating descriptors of each model's key points with different descriptors.

Method	Model	Dimension	Run time (ms)
Our proposed descriptor	Bunny	24	85
	Armadillo		422
	Happy buddha		448
SHOT	Bunny	352	252
	Armadillo		1247
	Happy buddha		1066
FPFH	Bunny	33	5395
	Armadillo		32193
	Happy buddha		14530
PFH	Bunny	125	3153
	Armadillo		17004
	Happy buddha		6606
RoPS	Bunny	135	8257
	Armadillo		49905
	Happy buddha		25714

obtained by ISS [38](Intrinsic Shape Signatures) which has been implemented in PCL. Table 2 shows the information of dimension of five feature descriptors and run time that generates descriptors for each model's keypoints.

It can be seen from Table 2 that the feature descriptor proposed in this paper is low-dimensional and high-speed. It means that the required storage space is smaller, and computation efficiency is higher for our proposed feature descriptor. The deep reason is that our proposed featture descriptor only needs to calculate the local coordinate system and the ratio of rational volume around each keypoint's neighborhood. It is noted that in this experiment, PFH is faster than FPFH. The reason is that the search radius of FPFH is twice than that of PFH, which causes FPFH's time cost will increase more quickly than PFH's time cost while the number of points used to calculate descriptors is small. Finally, it should be also noted that RoPS descriptor could not be generated direct from point cloud, and we need first turn point cloud into triangle mesh using the method of greedy projection triangulation before generating RoPS descriptor. Therefore, it is very time-consuming to generate RoPS descriptor and we can also see clearly from Table 2 that run time of generating RoPS descriptor is the longest.

3) ROBUSTNESS TO ROTATION

In order to evaluate the robustness of these feature descriptors to rotation, we rotate models in an increasing angle from 0 to 90. For the feature descriptors with robust rotation, their values do not change or have a small change. The process of experiment is shown as follows:

(1) Suppose a model $\boldsymbol{M}$ which comes from models used in this experiment and a ground-truth rotation matrix $\boldsymbol{R_{gt}}$.

(2) Rotate $\boldsymbol{M}$ according to rotation matrix $\boldsymbol{R_{gt}}$ and generate a new model $\boldsymbol{M'}$.

(3) Calculate keypoints for $\boldsymbol{M}$ and denote the keypoints as $\boldsymbol{KPS}$.

(4) From $\boldsymbol{M'}$, find corresponding keypoint for each keypoint in $\boldsymbol{KPS}$ according to $\boldsymbol{R_{gt}}$ and denote the keypoints as $\boldsymbol{KPS'}$. In this step, we first rotate a keypoint $\boldsymbol{kp}$ in $\boldsymbol{KPS}$ according to $\boldsymbol{R_{gt}}$ and generate a keypoint $\boldsymbol{kp_{tmp}'}$, and then we find the nearest point of $\boldsymbol{kp_{tmp}'}$ from $\boldsymbol{M'}$ and denote it as $\boldsymbol{kp'}$. The point $\boldsymbol{kp'}$ is a corresponding keypoint of $\boldsymbol{kp}$. According to this operation, other keypoints in $\boldsymbol{KPS}$ can find their corresponding keypoints in $\boldsymbol{KPS'}$.

(5) Calculate its descriptor for each keypoint in $\boldsymbol{KPS}$ and denote the descriptors as $\boldsymbol{DS}$, and calculate its descriptor for each keypoint in $\boldsymbol{KPS'}$ and denote the descriptors as $\boldsymbol{DS'}$.

(6) For each descriptor $\boldsymbol{d}$ in $\boldsymbol{DS}$, find the nearest descriptor $\boldsymbol{d'}$ from $\boldsymbol{DS'}$, and denote $\boldsymbol{d}$ and $\boldsymbol{d'}$ as an ordinal pair $<\boldsymbol{d}, \boldsymbol{d'}>$.

(7) According to $<\boldsymbol{d}, \boldsymbol{d'}>$, find corresponding keypoint's ordinal pair $<\boldsymbol{kp}, \boldsymbol{kp'}>$ in which $\boldsymbol{kp}$ comes from $\boldsymbol{M}$ and $\boldsymbol{kp'}$ comes from $\boldsymbol{M'}$.

(8) Judge the correspondence between $\boldsymbol{kp}$ and $\boldsymbol{kp'}$ according to the following formula

$$\left\| R_{gt} kp - kp' \right\| < 0.5mr, \tag{25}$$

which determines whether a descriptor does not change or has a small change with rotation.

(9) Calculate accuracy of matched descriptors according to the following formula

$$accuracy = n_{acc}/n, \tag{26}$$

in which $\boldsymbol{n_{acc}}$ denotes the number of matched descriptors, and $\boldsymbol{n}$ denotes the total of descriptors. The accuracy can reflect the robustness of feature descriptor to rotation. The higher the accuracy is, the more robust the feature descriptor is to rotation.

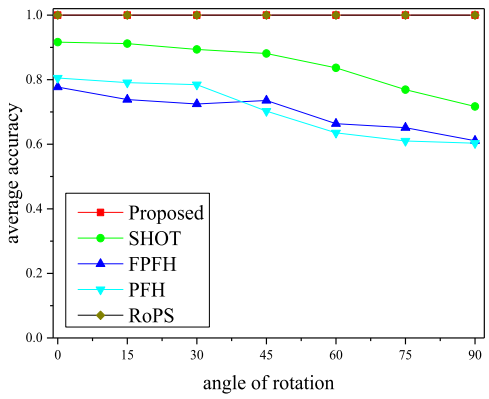
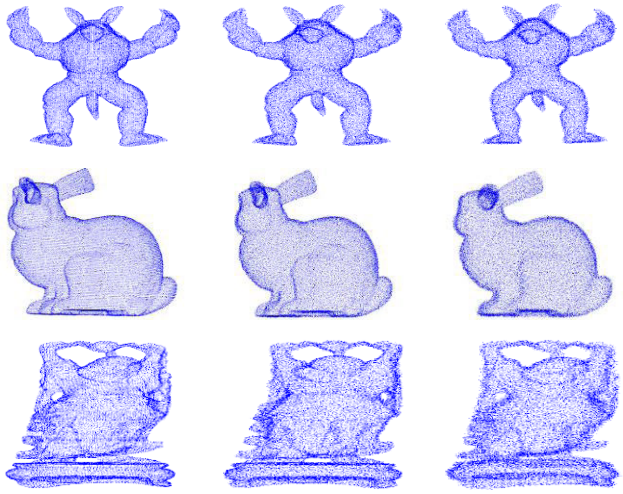


FIGURE 10. Accuracy under different angles of rotation for three models.

The ***average accuracy*** under different rotation angle for three models is shown as Fig. 10, in which it is very clear that

our proposed descriptor achieves the best accuracy in most feature descriptors and it also shows our proposed descriptor has the best robustness to rotation. There are at least two reasons for this. First, our proposed feature descriptor is based on a robust local reference frame which is invariant to rotation. Second, our proposed descriptor describes ratio of rotational volume of points in different partitions and the ratio of rotational volume is also invariant to rotation.



(a) σ=0.1*mr* (b) σ=0.2*mr* (c) σ=0.3*mr*

FIGURE 11. Scenes used in these experiments.

4) ROBUSTNESS TO NOISE

In order to evaluate the robustness of these feature descriptors to noise, we added Gaussian noise with increasing standard deviation of 0.1,0.2 and 0.3 ***mr*** to models and denotes them as scenes which are shown as Fig. 11. For the feature descriptors with robust noise, their values do not change or have a small change. The process of experiment is shown as follows:

(1) Suppose a model $\boldsymbol{M}$ which comes from models used in this experiment and a ground-truth matrix Rt_{gt} which includes a rotation matrix R_{gt} and a translate vector t_{gt}.

(2) Select a corresponding scene S_σ which comes from scenes used in this experiment and is generated by adding Gaussian noise with standard deviation σ to M.

(3) Rotate and translate $\boldsymbol{S_\sigma}$ according to the matrix $\boldsymbol{Rt_{gt}}$ and the translate vector $\boldsymbol{t_{gt}}$, and then generate a new scene $\boldsymbol{S_\sigma}'$.

(4) Calculate keypoints for $\boldsymbol{M}$ and denote the keypoints as ***KPS***.

(5) From $\boldsymbol{S_\sigma'}$, find corresponding keypoint for each keypoint in ***KPS*** according to $\boldsymbol{Rt_{gt}}$ and $\boldsymbol{t_{gt}}$, and denote the keypoints as ***KPS'***. In this step, we first rotate and translate a keypoint ***kp*** in ***KPS*** according to Rt_{gt} and $\boldsymbol{t_{gt}}$, and generate a keypoint $\boldsymbol{kp_{tmp}'}$, and then we find the nearest point of $\boldsymbol{kp_{tmp}}'$ from $\boldsymbol{S_\sigma'}$ and denote it as ***kp'***. The point ***kp'*** is a corresponding keypoint of ***kp***. According to this operation, other keypoints in ***KPS*** can find their corresponding keypoints in ***KPS'***.

(6) Calculate its descriptor for each keypoint in ***KPS*** and denote the descriptors as ***DS***, and calculate its descriptor for each keypoint in ***KPS'*** and denote the descriptors as ***DS'***.

(7) For each descriptor ***d*** in ***DS***, find the nearest descriptor ***d'*** from ***DS'*** , and denote ***d*** and ***d'*** as an ordinal pair $<\boldsymbol{d}, \boldsymbol{d'}>$.

(8) According to $<\boldsymbol{d}, \boldsymbol{d'}>$, find corresponding keypoint' s ordinal pair $<\boldsymbol{kp}, \boldsymbol{kp'}>$ in which ***kp*** comes from ***M*** and ***kp'*** comes from ***M'***.

(9) Judge the correspondence between ***kp*** and ***kp'*** according to the following formula

$$\left\| R_{gt} kp + t_{gt} - kp' \right\| < \sigma \cdot mr, \tag{27}$$

in which $\boldsymbol{\sigma}$ denotes standard deviation of Gaussian noise between $\boldsymbol{M}$ and $\boldsymbol{S_\sigma'}$. Formula (27) determines whether a descriptor does not change or has a small change with noise.

(10) Calculate accuracy of matched descriptors according to formulas (26). Here, the accuracy can reflect the robustness of feature descriptor to noise. The higher the accuracy is, the more robust the feature descriptor is to noise.

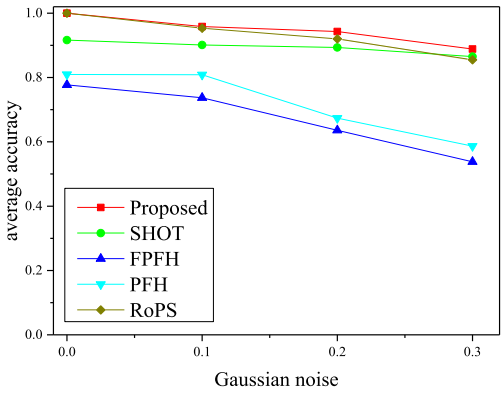


FIGURE 12. Accuracy under different Gaussian noise for three models.

The ***average accuracy*** under different Gaussian noise for three models is shown as Fig. 12, in which it is very clear that our proposed descriptor achieves the best accuracy in most feature descriptors, and is followed by RoPS. By Fig. 12, it shows our proposed feature descriptor has the best robustness to noise. The deep-seated reason is that our proposed feature descriptor describes ratio of rotational volume, which denotes space structure of keypoint and its neighboring points around the keypoint will keep invariant as long as noises do not change the geometric structure of neighborhood of keypoint.

5) ROBUSTNESS TO VARYING MESH RESOLUTION

In real production and life, different hardware and configuration of sampling equipment will lead to different mesh resolutions. Therefore, it is very important to test the performance of a feature descriptor based on varying mesh resolution. The process of experiment on robustness to varying mesh resolution is shown as follows:

(1) Suppose a model M which comes from models used in this experiment and a ground-truth rotation matrix R_{gt}.

(2) Resample M to δ of its original mesh resolution and name the new model as M_δ.

(3) Calculate *accuracy* between M and M_δ based on the above experiment of robustness to rotation.

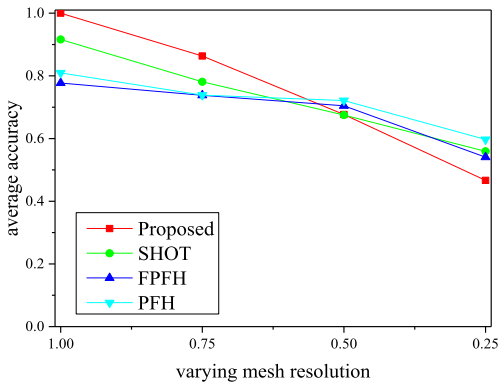


FIGURE 13. Accuracy under varying mesh resolution for three models.

The ***average accuracy*** under varying mesh resolution for three models is shown as Fig. 13. It can be clearly seen from Fig. 13 that our proposed feature descriptor has the best accuracy when mesh is resampled down to 0.75 of their original mesh resolution, and the ***average accuracy*** of our proposed feature descriptor is close to other feature descriptors when mesh is resampled down to 0.5 of their original mesh resolution, and the its ***average accuracy*** is worse than other descriptors when mesh is resampled down to 0.25 of their original mesh resolution whose reason is that our proposed feature descriptor has great changes for internal geometry structure and volume when mesh resolution varies greatly, and then large error is made on the process of calculating ratio of volume.

Though our proposed feature descriptor is not so good when mesh is resampled to 0.25, it is very rare for this kind of varying mesh resolution in real production and life. Meanwhile, our proposed feature descriptor is good when mesh is resampled to 0.75 and 0.5. Therefore, our proposed feature descriptor is robust to varying mesh resolution.

6) ROBUSTNESS TO SCALE

In order to evaluate the robustness of these feature descriptors to scale, we need make a point cloud multiply by a scale factor. The process of experiment is shown as follows:

(1) Suppose a model ***M*** which comes from models used in this experiment and a ground-truth scale factor $\boldsymbol{\omega}$.

(2) Multiply ***M*** to scale factor $\boldsymbol{\omega}$ and generate a new model ***M'***.

(3) Calculate keypoints for ***M*** and denote the keypoints as ***KPS***.

(4) From ***M'***, find corresponding keypoint for each keypoint in ***KPS*** according to $\boldsymbol{\omega}$ and denote the keypoints as ***KPS'***. In this step, we first multiply a keypoint ***kp*** in ***KPS*** ***by*** $\boldsymbol{\omega}$ and generate a keypoint $\boldsymbol{kp}'_{\boldsymbol{tmp}}$, and then we find the nearest point of $\boldsymbol{kp}'_{\boldsymbol{tmp}}$ from ***M'*** and denote it as ***kp'***. The point ***kp'*** is a corresponding keypoint of ***kp***. According to this operation, other keypoints in ***KPS*** can find their corresponding keypoints in ***KPS'*** .

(5) Calculate its descriptor for each keypoint in ***KPS*** and denote the descriptors as ***DS***, and calculate its descriptor for each keypoint in ***KPS'*** and denote the descriptors as ***DS'***.

(6) For each descriptor ***d*** in ***DS***, find the nearest descriptor ***d'*** from ***DS'***, and denote ***d*** and ***d'*** as an ordinal pair $<\boldsymbol{d}, \boldsymbol{d'}>$.

(7) According to $<\boldsymbol{d}, \boldsymbol{d'}>$, find corresponding keypoint' s ordinal pair $<\boldsymbol{kp}, \boldsymbol{kp'}>$ in which ***kp*** comes from ***M*** and ***kp'*** comes from ***M'***.

(8) Judge the correspondence between ***kp*** and ***kp'*** according to the following formula

$$\left\| kp - kp^{'}/\boldsymbol{\omega} \right\| < 0.5mr, \tag{28}$$

which determines whether a descriptor does not change or has a small change under different scales.

(9) Calculate ***average accuracy*** between M and ***M'*** based on the above experiment of robustness to rotation.

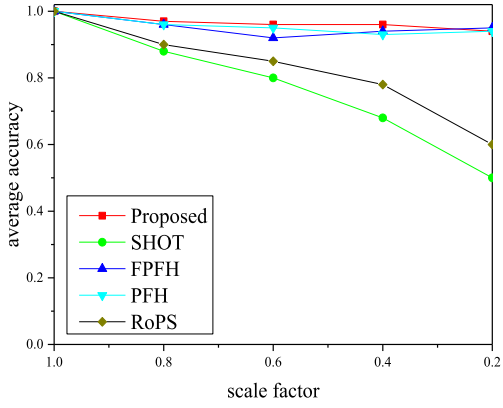


FIGURE 14. Average accuracy under different scale factors for three models.

The ***average accuracy*** under different scale factors for three models is shown as Fig. 14. It is very clear that our proposed feature descriptor achieves the best accuracy in most feature descriptors and is followed by PFH and FPFH. The reason that our proposed feature descriptor is robust to scale factor is that we normalize our proposed feature descriptor and make it almost unaffected for scale change. The reason that PFH and FPFH are almost unaffected by scale change is they both describe relation of angle between different sides, which is also robust to scale change.

B. EXPERIMENTS OF KEYPOINT DETECTION WITH BOUNDARY REMOVED

The task of this section is to validate whether our keypoint detection with boundary removed can detect boundary points

in a point cloud and can obtain keypoints far away from boundary.

1) DATASET

In these experiments, we first use partial point clouds of Armadillo, Bunny and Buddha which come from the Stanford 3D Scanning Repository to perform our experiments. For Armadillo, we select its 11 partial point clouds into our dataset and they are named from ArmadilloBack_0 to ArmadilloBack_330 with 60-degree intervals between them. For Bunny, we select 6 partial point clouds into our dataset and they are named as Bun000, Bun045, Bun090, Bun180, Bun270 and Bun315 respectively. For Happy buddha, we select 15 partial point clouds into our dataset and they are named from HappyStandRight_0 to HappyStandRight_336 with 24-degree intervals between them. In addition, we also use partial point clouds of Bremen which comes from Robotic 3D Repository to perform our experiments. For Bremen, we select its 13 partial point clouds into our dataset and they are named from scan000 to scan012. Fig. 15 shows part of point clouds in our dataset.

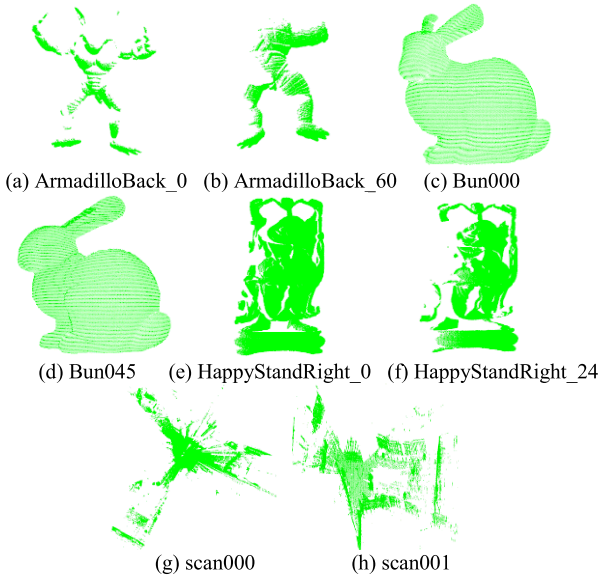


FIGURE 15. Part of point clouds in our dataset.

2) RESULTS OF KEYPOINT DETECTION WITH BOUNDARY REMOVED

Before obtaining keypoints, a key step is to obtain boundary points. In the process of determining whether a point is a boundary point, we choose its neighboring points within radius 4 ***mr*** and set different $\boldsymbol{L_{max}}$. Fig. 16 shows boundary points of point clouds listed in Fig. 15 under different $\boldsymbol{L_{max}}$ and the red points are boundary points and green point is original points of point clouds in Fig. 15. From Fig. 16, it is very clear that the number of boundary points is smallest when L_{max} is set to π and the boundary points are discrete. By contrast, there are more boundary points when L_{max} is between $\pi/2$ and $\pi/4$. Especially, when L_{max} is set to $\pi/3$ or $\pi/4$, boundary points overlap with each other and form several clusters. In later experiments, we set L_{max} to $\pi/2$, which is tradeoff that too many boundary points will decrease time efficiency of detecting keypoints and too little boundary points will decrease effect of detecting keypoints.

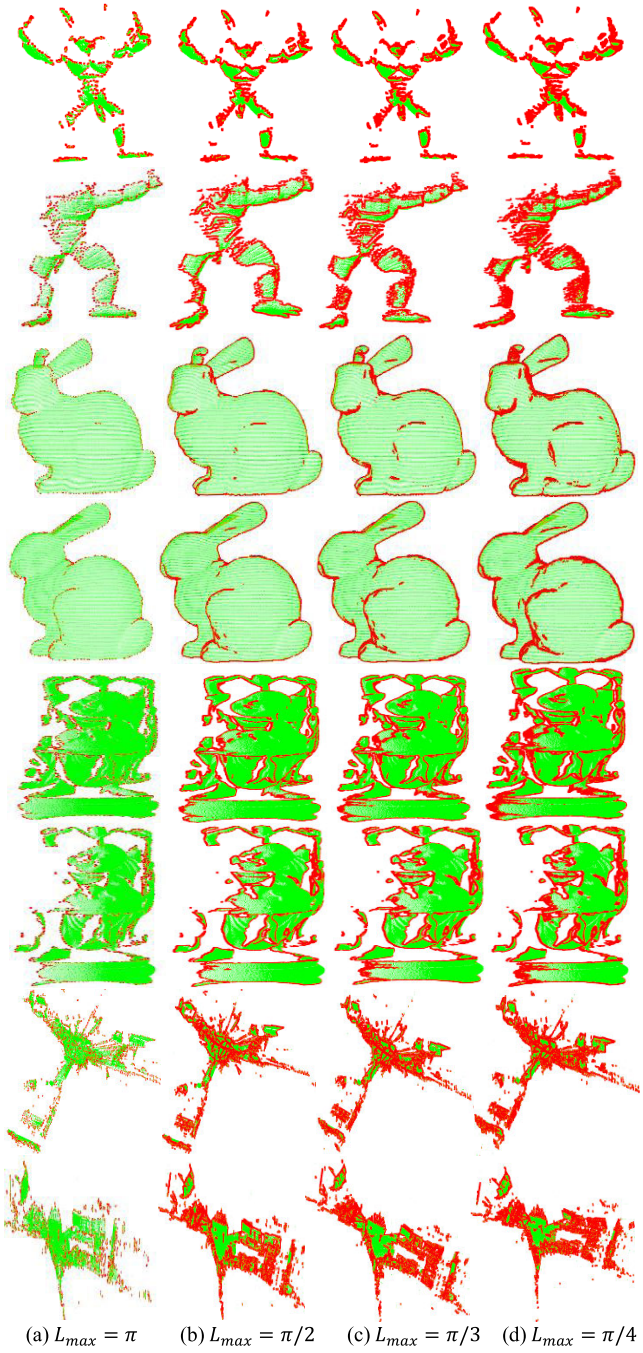


FIGURE 16. Boundary points on different point clouds.

After obtaining boundary points for a point cloud, we can obtain original keypoints by ISS detector and remove keypoints on the boundary from the original keypoints. In the process of determining whether a point is a keypoint, we choose its neighboring points within radius 4 ***mr*** in order

to calculate scatter matrix $\sum(p)$, and we set $\boldsymbol{th_{12}}$ and $\boldsymbol{th_{23}}$ to 0.975 (which is default value in ISS detector).

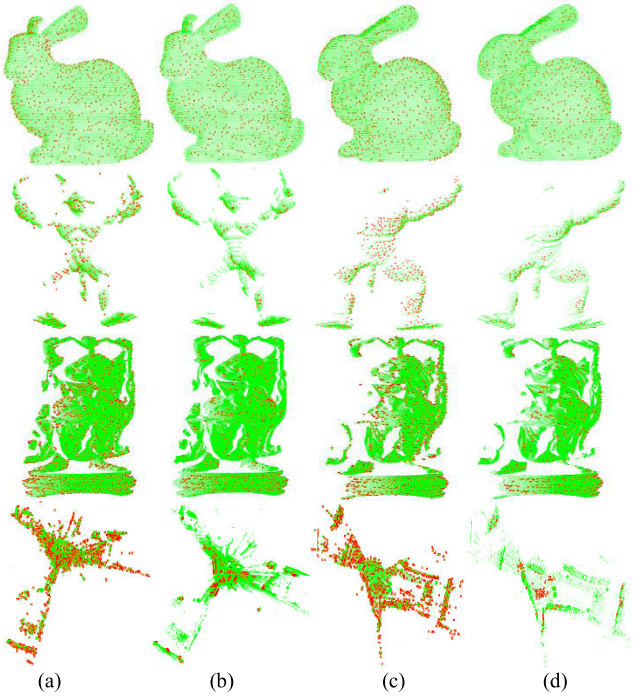


FIGURE 17. Keypoints on different point clouds.

In Fig. 17, column (a) and column (c) show keypoints by ISS detector for different point clouds listed in Fig. 15, and red points are keypoints and green points are original points of point clouds. After obtaining original keypoints, we will remove each keypoint whose distance with the nearest boundary point is less than 5 *mr*. In Fig. 17, column (b) and column (d) show current keypoints after removing those keypoints on the boundary from original keypoints. Compare to keypoints on column (a) and column (b), column (c) and column (d) in Fig. 17, it is clearly observed that our algorithm of keypoint detection with boundary removed can obtain keypoints on non-boundary, which is very import to generate feature descriptor.

C. EXPERIMENTS OF PAIRWISE REGISTRATION

After obtaining keypoints from point cloud, feature descriptor generation, feature matching, robust transformation estimation using SAC and fine ICP are then performed, and the result of pairwise registration can be obtained. Here, we use the same dataset used in section 4.2. In those steps, more important step is pairwise matching of point clouds, including keypoint detection, feature descriptor generation and feature matching. By pairwise matching of point clouds, lots of correspondences between point clouds are able to be obtained, which is helpful for a good coarse pairwise registration, and meanwhile, a good coarse pairwise registration will be helpful for a fine registration. It is specially noted value of radius r need be set in many cases, such as detecting keypoints without boundary removed, and constructing LRF, and generating our proposed descriptor, and so on. So, we set uniformly radius r to $\mathbf{4}\,\boldsymbol{mr}$ in order to decrease the number of parameters.

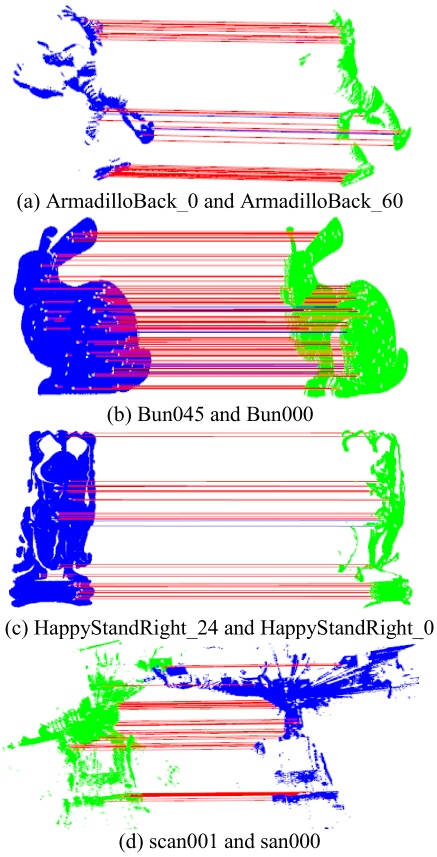


FIGURE 18. Matching illustrations between point clouds.

Fig. 18 presents four matching illustrations for point clouds listed in Fig. 15. For each figure in Fig. 18, the blue point cloud represents source point cloud, the green point cloud represents target point cloud, and a line represents a pair of keypoints between source point cloud and target point cloud. It is noted specially that the original lines are not parallel, but rather crisscross because rotation transformation exists between source point cloud and target point cloud. However, to easily evaluate the accuracy of matching pairs, we rotate source point cloud and its keypoints according to the ground-truth rotation between source point cloud and target point cloud. Based on this rotation, if the matching lines are more parallel and the number of matching lines is larger, our proposed method is more effective. It is visually obvious that there are many parallel matching lines in Fig. 18(a), 18(b) and 18 (c) between source point cloud and target point cloud. At the same time, we verify further the accuracy for a matching line according to the following steps: firstly, for a pair of keypoint $< \boldsymbol{kp_j^t}, \boldsymbol{kp_i^s} >$ represented by a line, we transform $\boldsymbol{kp_i^s}$ according to the ground-truth transformation between source point cloud and target point cloud,

and name the transformed keypoint as $\boldsymbol{kp}_i^{s'}$; secondly, we calculate the distance between $\boldsymbol{kp}_i^{s'}$ and $\boldsymbol{kp}_j^t$; finally, if the distance is less than 0.5 $\boldsymbol{mr}$, we regard $\boldsymbol{kp}_j^t$ and $\boldsymbol{kp}_i^s$ as a true matching keypoint pair, and the matching line will be shown as a red line, and on the contrary false matching line will be shown as a blue line. From Fig. 18 (a) to 18 (c), it is very clear that most matching lines are red and it shows further our proposed pairwise matching of point clouds is more effective.

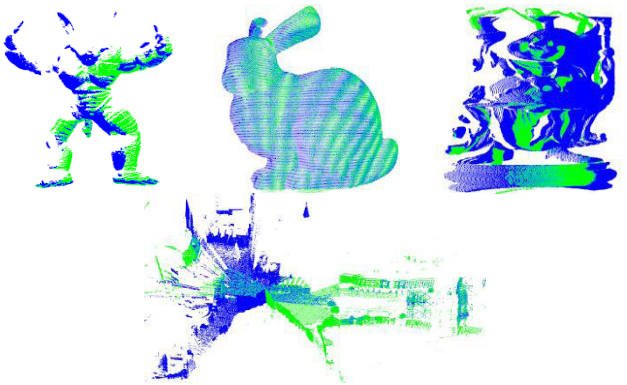


FIGURE 19. Pairwise registration illustrations.

Four results of pairwise registration are represented in Fig. 19, in which blue points represent source point cloud and green points represent target point cloud. It is visually obvious that source point cloud can be registered seamlessly to target point cloud and there is a complete fuse between source point cloud and target point cloud.

To further verify our pairwise registration, two criteria are employed for quantitative assessment of registration's results, i.e., the rotation error $\boldsymbol{\theta_r}$ and the translation error $\boldsymbol{\theta_t}$ which are used in [15], [34]. $\boldsymbol{\theta_r}$ represents error between the ground truth rotation matrix $\boldsymbol{R_{GT}}$ and the actual matrix $\boldsymbol{R_A}$. $\boldsymbol{\theta_t}$ represents error between the ground truth vector $\boldsymbol{t_{GT}}$ and the actual translation vector $\boldsymbol{t_A}$. They are defined as follows

$$\theta_r = cos^{-1}\left(\frac{trace\left(R_A R_{GT}^{-1}\right) - 1}{2}\right) * \frac{180}{\pi}, \tag{29}$$

$$\theta_t = \frac{\|t_A - t_{GT}\|}{d_{res}}, \tag{30}$$

where $\boldsymbol{d_{res}}$ is 1 $\boldsymbol{mr}$, $\boldsymbol{R_{GT}}$ and $\boldsymbol{t_{GT}}$ are known in advance. It should be noted that how we can obtain $\boldsymbol{R_A}$ and $\boldsymbol{t_A}$. In the process of we can use SAC to estimate a coarse transformation matrix $\boldsymbol{Rt_C}$ between source point cloud and target point cloud, and then we can use fine registration to obtain a fine transformation matrix $\boldsymbol{Rt_F}$, so final transformation matrix $\boldsymbol{Rt}$ is the result that $\boldsymbol{Rt_F}$ multiplies $\boldsymbol{Rt_C}$. We denote $\boldsymbol{Rt_F}$ and $\boldsymbol{Rt_C}$ as follow

$$Rt_C = \begin{bmatrix} R_c & t_C \\ 0 & 1 \end{bmatrix}, \tag{31}$$

$$Rt_F = \begin{bmatrix} R_F & t_F \\ 0 & 1 \end{bmatrix}, \tag{32}$$

where $\boldsymbol{R_c}$ and $\boldsymbol{t_C}$ represent rotation matrix and translation vector of $\boldsymbol{Rt_C}$ respectively, $\boldsymbol{R_F}$ and $\boldsymbol{t_F}$ represent rotation matrix and translation vector of $\boldsymbol{Rt_F}$ respectively. Therefore, $\boldsymbol{Rt}$ can be obtained as follow

$$Rt = Rt_F Rt_C = \begin{bmatrix} R_F & t_F \\ 0 & 1 \end{bmatrix}\begin{bmatrix} R_c & t_C \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} R_F R_C & R_F t_C + t_F \\ 0 & 1 \end{bmatrix}, \tag{33}$$

where $\boldsymbol{R_F R_C}$ represents $\boldsymbol{R_A}$ and $\boldsymbol{R_F t_C + t_F}$ represents $\boldsymbol{t_A}$.

TABLE 3. Registration errors of whole pairs of point clouds in our dataset.

Model	Source point cloud	Target point cloud	θ_r	θ_t
Armadillo	ArmadilloBack_60	ArmadilloBack_0	0.7823	2.9776
	ArmadilloBack_90	ArmadilloBack_60	0.1187	4.0772
	ArmadilloBack_120	ArmadilloBack_90	0.0685	1.7575
	ArmadilloBack_150	ArmadilloBack_120	0.7401	1.5037
	ArmadilloBack_180	ArmadilloBack_150	0.5158	2.4766
	ArmadilloBack_210	ArmadilloBack_180	0.7913	4.7557
	ArmadilloBack_240	ArmadilloBack_210	2.1219	7.4275
	ArmadilloBack_270	ArmadilloBack_240	0.3753	1.4457
	ArmadilloBack_300	ArmadilloBack_270	0.8393	3.8522
	ArmadilloBack_330	ArmadilloBack_300	1.0025	1.9785
Bunny	Bun045	Bun000	0.9405	2.4058
	Bun090	Bun045	2.0422	5.18616
	Bun180	Bun090	2.0076	4.0886
	Bun270	Bun180	2.6941	3.0838
	Bun315	Bun270	1.2468	2.8650
Happy buddha	HappyStandRight_24	HappyStandRight_0	1.6179	3.2518
	HappyStandRight_48	HappyStandRight_24	1.2449	2.5768
	HappyStandRight_72	HappyStandRight_48	0.2564	0.3574
	HappyStandRight_96	HappyStandRight_72	0.4597	0.8114
	HappyStandRight_120	HappyStandRight_96	0.3380	0.6676
	HappyStandRight_144	HappyStandRight_120	0.2167	0.6834
	HappyStandRight_168	HappyStandRight_144	0.2093	0.5684
	HappyStandRight_192	HappyStandRight_168	0.2533	0.7627
	HappyStandRight_216	HappyStandRight_192	0.3115	0.9549
	HappyStandRight_240	HappyStandRight_216	0.5189	2.2352
	HappyStandRight_264	HappyStandRight_240	0.3648	1.1011
	HappyStandRight_288	HappyStandRight_264	0.3128	0.8905
	HappyStandRight_312	HappyStandRight_288	0.3298	0.9372
	HappyStandRight_336	HappyStandRight_312	0.2961	0.6725
Bremen	Scan1	Scan0	0.1265	0.0907
	Scan2	Scan1	1.0839	7.1061
	Scan3	Scan2	0.2782	0.0294
	Scan4	Scan3	0.7381	0.0788
	Scan5	Scan4	0.0485	0.1445
	Scan6	Scan5	0.0685	0.1331
	Scan7	Scan6	0.1312	0.0409
	Scan8	Scan7	0.0685	0.0792
	Scan9	Scan8	1.3605	9.3226
	Scan10	Scan9	0.4956	0.0455
	Scan11	Scan10	0.0559	0.0211
	Scan12	Scan11	0.2450	0.0192

Theoretically, when $\boldsymbol{\theta_r}$ and $\boldsymbol{\theta_t}$ are smaller, the accuracy of pairwise registration is higher. In our dataset used in section 4.2, there are total 45 pieces of point clouds and 41 pairs of point clouds are formed. The registration errors of whole 41 pairs of point clouds are shown in Table 3. One can see that, most pairs of point clouds have been accurately registered (i.e., with small rotation error and translation error). To judge the correctness of a pairwise registration, thresholds are required as principles. For example, we regard a pairwise registration as correctness only when its rotation error and translation error are both smaller than 5 degree and 5 $\boldsymbol{mr}$ respectively. Based on current thresholds, our pairwise registration achieves a registration accuracy of 90.2% on our dataset. The deep reason is that our proposed feature descriptor in pairwise registration pipeline is robust to

TABLE 4. Average registration errors for pairwise registration based on different feature descriptor.

Method	Average θ_r	Average θ_t
based on our proposed feature descripto	0.6760	2.0357
based on SHOT	1.4271	2.9023
based on FPFH	1.5396	4.7471
based on PFH	3.5537	4.6589
Based on RoPS	1.2655	2.4037

rotation and noise. Therefore, based on our improved pairwise registration pipeline, we compare our feature descriptor with state-of-the-art feature descriptors. Table 4 shows average registration errors of whole 41 pairs of point clouds in our dataset. From Table 4, it is clearly shown that average registration error for pairwise registration based on our proposed feature descriptor is the smallest, followed by based on based on RoPS and SHOT. The deep-seated reason is that our proposed feature descriptor is more invariant to rotation and more robust to noise than other feature descriptors.

V. CONCLUSION

In this paper, we propose a novel local feature descriptor based on rotational volume, which describes a ratio of volume of a geometrical model which is generated by rotating a point and its neighboring points around their normal. Experimental results show that our proposed feature descriptor has a good robustness to rotation and noise, less storage space and faster running speed than PFH, FPFH, SHOT and RoPS descriptors. We also propose a boundary point detection algorithm in order to remove keypoints near to boundary, and a robust transformation estimation via SAC. Based on above works, we implement pairwise registration of point clouds, and experimental results show that our pairwise registration pipeline is very effective. Of course, there are also some limitations about our proposed method as follows: (1) Too many parameters need to be set and we intend to set these parameter by adaptive algorithm in the future. (2) Our proposed method only apply to rigid registration and it's invalid to non-rigid registration.

ACKNOWLEDGMENT

The authors thank the anonymous reviewers for their detailed comments and suggestions, which resulted in the improvement of this article.

REFERENCES

[1] D. Zai, J. Li, Y. Guo, M. Cheng, P. Huang, X. Cao, and C. Wang, "Pairwise registration of TLS point clouds using covariance descriptors and a non-cooperative game," *ISPRS J. Photogramm. Remote Sens.*, vol. 134, pp. 15–29, Dec. 2017.

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

[3] D. Svirko, P. Krsek, D. Stepanov, and D. Chetverikov, "The trimmed iterative closest point algorithm," in *Proc. Int. Conf. Pattern Recognit. (ICPR)*, vol. 3, Aug. 2002, pp. 545–548.

[4] G. C. Sharp, S. W. Lee, and D. K. Wehe, "ICP registration using invariant features," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 1, pp. 90–102, Jan. 2002.

[5] N. Gelfand, N. J. Mitra, L. J. Guibas, and H. Pottmann, "Robust global registration," in *Proc. Symp. Geometry Process.*, 2005, vol. 2, no. 3, p. 5.

[6] F. A. Sadjadi and E. L. Hall, "Three-dimensional moment invariants," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-2, no. 2, pp. 127–136, Mar. 1980.

[7] G. Burel and H. Hénocq, "Three-dimensional invariants and their application to object recognition," *Signal Process.*, vol. 45, no. 1, pp. 1–22, 1995.

[8] R. B. Rusu, N. Blodow, Z. C. Marton, and M. Beetz, "Aligning point cloud views using persistent feature histograms," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2008, pp. 3384–3391.

[9] R. B. Rusu, N. Blodow, Z. C. Marton, and M. Beetz, "Learning informative point classes for the acquisition of object model maps," in *Proc. 10th Int. Conf. Control, Automat., Robot. Vis.*, Dec. 2008, pp. 643–650.

[10] Y. Guo, F. Sohel, M. Bennamoun, M. Lu, and J. Wan, "Rotational projection statistics for 3D local surface description and object recognition," *Int. J. Comput. Vis.*, vol. 105, no. 1, pp. 63–86, 2013.

[11] Y. Guo, M. Bennamoun, F. A. Sohel, J. Wan, and M. Lu, "3D free form object recognition using rotational projection statistics," in *Proc. IEEE Workshop Appl. Comput. Vis. (WACV)*, Jan. 2013, pp. 1–8.

[12] Y. Guo, M. Bennamoun, F. A. Sohel, J. Wan, and M. Lu, "RoPS: A local feature descriptor for 3D rigid objects based on rotational projection statistics," in *Proc. 1st Int. Conf. Commun., Signal Process., Appl. (ICCSPA)*. Sharjah, United Arab Emirates: IEEE, Feb. 2013, pp. 1–6.

[13] Y. Guo, M. Bennamoun, F. Sohel, M. Lu, J. Wan, and N. M. Kwok, "A comprehensive performance evaluation of 3D local feature descriptors," *Int. J. Comput. Vis.*, vol. 116, no. 1, pp. 66–89, 2016.

[14] S. Salti, F. Tombari, and L. Di Stefano, "SHOT: Unique signatures of histograms for surface and texture description," *Comput. Vis. Image Understand.*, vol. 125, no. 8, pp. 251–264, Aug. 2014.

[15] J. Yang, Q. Zhang, and Z. Cao, "Multi-attribute statistics histograms for accurate and robust pairwise registration of range images," *Neurocomputing*, vol. 251, pp. 54–67, Aug. 2017.

[16] J. Yang, Z. Cao, and Q. Zhang, "A fast and robust local descriptor for 3D point cloud registration," *Inf. Sci.*, vol. 346, pp. 163–179, Jun. 2016.

[17] Z. Dong, B. Yang, F. Liang, R. Huang, and S. Scherer, "Hierarchical registration of unordered TLS point clouds based on binary shape context descriptor," *ISPRS J. Photogramm. Remote Sens.*, vol. 144, pp. 61–79, Oct. 2018.

[18] Z. Dong, B. Yang, Y. Liu, F. Liang, B. Li, and Y. Zang, "A novel binary shape context for 3D local surface description," *ISPRS J. Photogramm. Remote Sens.*, vol. 130, pp. 431–452, Aug. 2017.

[19] R. Zhou, X. Li, and W. Jiang, "3D surface matching by a voxel-based buffer-weighted binary descriptor," *IEEE Access*, vol. 7, pp. 86635–86650, 2019.

[20] Z. Shen, X. Ma, and Y. Li, "A hybrid 3D descriptor with global structural frames and local signatures of histograms," *IEEE Access*, vol. 6, pp. 39261–39272, 2018.

[21] K. Tang, S. Peng, and X. Chen, "3D object recognition in cluttered scenes with robust shape description and correspondence selection," *IEEE Access*, vol. 5, no. 99, pp. 1833–1845, 2017.

[22] Z. Zhang, L. Sun, R. Zhong, D. Chen, Z. Xu, C. Wang, C.-Z. Qin, H. Sun, and R. Li, "3-D deep feature construction for mobile laser scanning point cloud registration," *IEEE Geosci. Remote Sens. Lett.*, vol. 16, no. 12, pp. 1904–1908, Dec. 2019.

[23] Z. Zhang, L. Zhang, X. Tong, P. T. Mathiopoulos, B. Guo, X. Huang, and Z. Wang, "A multilevel point-cluster-based discriminative feature for ALS point cloud classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 6, pp. 3309–3321, Jun. 2016.

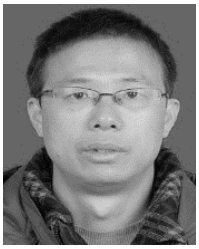
[24] Z. Zhang, L. Zhang, X. Tong, B. Guo, L. Zhang, and X. Xing, "Discriminative-dictionary-learning-based multilevel point-cluster features for ALS point-cloud classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 12, pp. 7309–7322, Dec. 2016.

[25] A. Zeng, S. Song, M. Nießner, M. Fisher, J. Xiao, and T. Funkhouser, "3DMatch: Learning local geometric descriptors from RGB-D reconstructions," in *Proc. 30th IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 199–208.

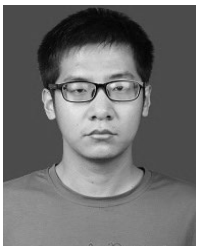
[26] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, "Surface reconstruction from unorganized points," in *Proc. Conf. Comput. Graph. Interact. Techn.* New York, NY, USA: ACM, 1992, pp. 71–78.

[27] N. J. Mitra and N. Guyen, "Estimating surface normals in noisy point cloud data," *Int. J. Comput. Geometry Appl.*, vol. 14, nos. 4–5, pp. 261–276, 2004.

[28] A. Mian, M. Bennamoun, and R. Owens, "On the repeatability and quality of keypoints for local feature-based 3D object retrieval from cluttered scenes," *Int. J. Comput. Vis.*, vol. 89, no. 2, pp. 348–361, Sep. 2010.
[29] F. Tombari, S. Salti, and S. L. Di, "Unique signatures of histograms for local surface description," in *Proc. Eur. Conf. Comput. Vis.* Berlin, Germany: Springer, 2010, pp. 356–369.
[30] M. A. Fischler and R. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, 1981.
[31] J. Yang, K. Xian, Y. Xiao, and Z. Cao, "Performance evaluation of 3D correspondence grouping algorithms," in *Proc. Int. Conf. 3D Vis. (3DV)*, Oct. 2017, pp. 467–476.
[32] R. Schnabel, R. Wahl, and R. Klein, "Efficient RANSAC for point-cloud shape detection," in *Computer Graphics Group*, vol. 26, no. 2. Oxford, U.K.: Blackwell, 2007, pp. 214–226.
[33] O. Chum, J. Matas, and J. Kittler, "Locally optimized RANSAC," in *Proc. Joint Pattern Recognit. Symp.* Berlin, Germany: Springer, 2003, pp. 236–243.
[34] A. S. Mian, M. Bennamoun, and R. A. Owens, "A novel representation and feature matching algorithm for automatic pairwise registration of range images," *Int. J. Comput. Vis.*, vol. 66, no. 1, pp. 19–40, 2006.
[35] J. Yang, H. Li, D. Campbell, and Y. Jia, "Go-ICP: A globally optimal solution to 3D ICP point-set registration," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 11, pp. 2241–2254, Nov. 2016.
[36] D. Holz, A. E. Ichim, F. Tombari, R. B. Rusu, and S. Behnke, "Registration with the point cloud library: A modular framework for aligning in 3-D," *IEEE Robot. Autom. Mag.*, vol. 22, no. 4, pp. 110–124, Dec. 2015.
[37] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," in *Proc. Int. Conf. Comput. Graph. Interact. Techn.*, 1996, pp. 303–312.
[38] Y. Zhong, "Intrinsic shape signatures: A shape descriptor for 3D object recognition," in *Proc. 12th Int. Conf. Comput. Vis. Workshops (ICCV Workshops)*, Sep./Oct. 2009, pp. 689–696.



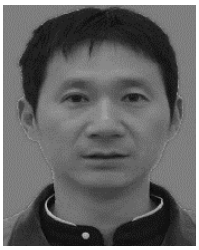
XIONG FENGGUANG was born in China, in 1979. He received the master's degree in computer application technology from the North University of China, China, in 2005, and the Ph.D. degree from the Institute of Information Engineering, North University of China, in 2018. His current research interests include computer vision, simulation, and visualization.



DONG BIAO was born in China, in 1995. He received the bachelor's degree in network engineering from the North University of China, China, in 2017, where he is currently pursuing the master's degree in computer science and technology.

HUO WANG was born in China, in 1991. He received the master's degree in computer application technology from the North University of China, China, in 2018. He is currently with Zhejiang Hanchine AI Tech. Company Ltd., Hangzhou, China.

PANG MIN was born in China, in 1979. She received the master's degree in computer application technology from the North University of China (NUC), China, in 2009. Her current research interests include computer vision, simulation, and visualization.



KUANG LIQUN was born in China, in 1976. He received the master's degree in computer application technology from the North University of China, China, in 2003, and the Ph.D. degree from the Institute of Information Engineering, North University of China, in 2019. His current research interests include computer vision and pattern recognition.

HAN XIE was born in China, in 1964. She received the master's degree in computer science and technology from the North China Institute of Technology (NCIT), China, and the Ph.D. degree from the Institute of Information Engineering, University of Science and Technology, Beijing, China, in 2002. Her current research interests include computer vision, simulation, and visualization.