electronics

MDPI

Article

A Four-Point Orientation Method for Scene-to-Model Point Cloud Registration of Engine Blades

Duanjiao Li [1], Ying Zhang [1], Ziran Jia [1], Zhiyu Wang [2,*], Qiu Fang [3,*] and Xiaogang Zhang [3]

[1] Aircraft Patrol Management Center, Guangdong Power Grid Co., Ltd., Guangzhou 510630, China; liduanjiao@gd.csg.cn (D.L.); zhangying0806@gd.csg.cn (Y.Z.); jiaziran@gd.csg.cn (Z.J.)
[2] Shanxi Intelligent Transportation Research Institute Co., Ltd., Taiyuan 030032, China
[3] College of Electrical and Information Engineering, Hunan University, Changsha 410082, China; zhangxg@hnu.edu.cn
* Correspondence: zhixiong@hnu.edu.cn (Z.W.); qfang@hnu.edu.cn (Q.F.)

Abstract: The use of 3D optical equipment for multi-view scanning is a promising approach to assessing the processing errors of engine blades. However, incomplete scanned point cloud data may impact the accuracy of point cloud registration (PCR). This paper proposes a four-point orientation point cloud registration method to improve the efficiency and accuracy of the coarse registration of turbine blades and prevent PCR failure. First, the point cloud is divided into four labeling blocks based on a principal component analysis. Second, keypoints are detected in each block based on their distance from the plane formed by the principal axes and described with a location-label descriptor based on their position. Third, a keypoint pair set is chosen based on the descriptor, and a suitable keypoint base is selected through singular value decomposition to obtain the final rigid transformation. To verify the effectiveness of the method, experiments are conducted on different blades. The results demonstrate the improved performance and efficiency of the proposed method of coarse registration for turbine blades.

Keywords: three-dimensional pointcloud; four-point congruent sets (4PCSs); keypoints detector; detector and descriptors; principal component analysis (PCA)

Citation: Li, D.; Zhang, Y.; Jia, Z.; Wang, Z.; Fang, Q.; Zhang, X. A Four-Point Orientation Method for Scene-to-Model Point Cloud Registration of Engine Blades. *Electronics* **2024**, *13*, 4634. https://doi.org/10.3390/electronics13234634

Academic Editor: Beiwen Li

Received: 29 October 2024
Revised: 16 November 2024
Accepted: 18 November 2024
Published: 25 November 2024

1. Introduction

Engine blades are renowned for their intricate and precise structure. In recent years, significant focus has been placed on improving the technology used to measure manufacturing errors in blade production. While the coordinate measuring machine (CMM) is a traditional and widely used measuring technique due to its high precision [1], it has limitations in terms of efficiency due to the fact that a CMM requires manual intervention and is a time-consuming process.

An industrial manufacturing solution has been developed using a 3D structured light device which scans objects from multiple viewpoints to produce a significant amount of point cloud data. To fully utilize these data, point cloud registration (PCR) [1] has been developed, which involves two main steps. The first step is called "part-to-whole registration", which stitches together the point cloud data from multiple views to create a 3D scene cloud. The second step, called "scene-to-model registration", matches the point clouds of the model with the rebuilt scene. After PCR, a global color-coded deviation spectrum is generated to visualize the global error of the manufactured workpiece. To analyze errors for different parts, a specific blade profile is extracted for detailed error analysis. Figure 1 shows the general flowchart for measuring manufacturing errors in blades using a 3D structured light device.

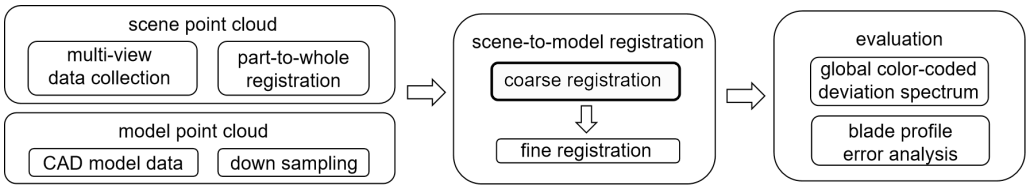


Figure 1. The flowchart for measuring blade manufacturing errors using a 3D structured light device.

Currently, the part-to-whole registration process often requires maker points to ensure rebuild accuracy. Figure 2 shows two methods for scanning objects with maker points: placing maker points directly on the object, which can result in more or fewer holes on the body of the scene's point cloud, or placing the object on a fixture table, with pillars being used to attach the maker points, which may prevent the fixed part from being scanned. Both of these methods can result in missing parts in the scene point cloud, which can inevitably impact the results of PCR. This paper focuses on the problem of coarse registration of scene point clouds with missing parts to model point clouds, assuming that multi-viewpoint cloud data have been collected and reconstructed using maker points. The missing parts can be the holes on the body of the blade caused by the maker points or the missing part at the bottom of the blades caused by the fixture design.

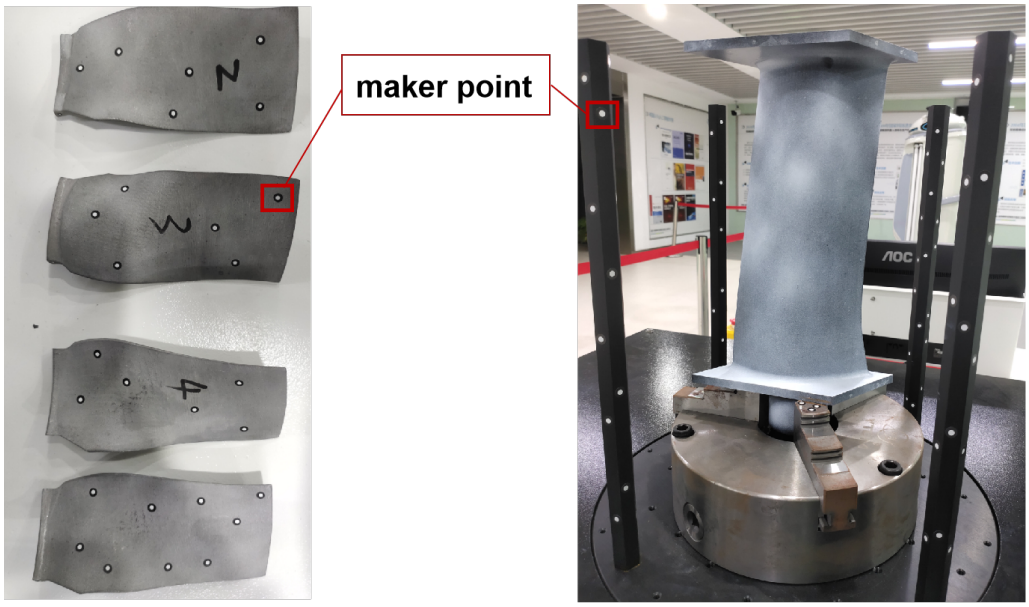


Figure 2. Two ways of scanning objects with maker points. The left is placing maker points on the object to be scanned. The right is placing objects on the fixture table with maker points attached on the pillars.

The current coarse registration process involves three steps: detection, description, and searching strategies [2]. With a large number of points in the cloud, the first step involves detecting keypoints in order to extract areas with distinctive features, which improves processing efficiency. Next, descriptors are defined to accurately identify points within a certain range of features surrounding the keypoints. Normal and curvature are commonly used features for extraction and are valid in certain registration procedures. However, many current coarse registration methods require manual parameter adjustments based on the shape of the scanned objects, which is an inefficient practice.

In this paper, a four-point orientation point cloud registration method is proposed to improve the accuracy and efficiency of coarse registration for turbine blades. The main contributions of this paper are summarized as follows:

- Point cloud data are divided into specific blocks based on principal component analysis, and a location-label descriptor is present;

- The selection of four points over three for a keypoint base is intended to reduce the likelihood of mismatches that could occur due to the presence of similar structural features;
- A location-label descriptor that considers the location of keypoints instead of features is presented to avoid manual parameter adjustments.

Our proposed method's efficiency and advancement are demonstrated in scene-to-model registration experiments involving five types of turbine blades.

The rest of the paper is organized as follows. Related work is shown in Section 2. In Section 3, the proposed registration method is presented in detail. Section 4 shows the experimental data and results. The concluding remarks are given in Section 5.

2. Related Work

The iterative closest point (ICP) [3] and its variants [4–8] are widely used in point cloud registration due to their simplicity and effectiveness. Although the development of ICPs has significantly improved [9], the process of iteratively finding the nearest points and performing transformations can be time-consuming and prone to local optima. Additionally, when the initial positions of the two point clouds to be aligned differ significantly, the ICP has a higher probability of alignment failure. Therefore, an ICP and its variants are generally used as a fine registration method to improve accuracy after a coarse registration has been performed.

Coarse registration is an essential step in achieving good initial values for fine registration, which requires fast and robust methods for handling large transformations. There are three types of methods commonly used for coarse registration. The first method is random sample consensus (RANSAC) [10–13], which randomly selects three points in each of the two point clouds and computes the transformation matrix, the number of inlier sets and the error. This process is repeated multiple times, and the best result is selected as the final transformation. The four-point congruent sets (4PCSs) [14] method takes advantage of rigid invariance and selects four-point base sets that are coplanar to resolve the transformation between the two point clouds. Super 4PCSs [15] accelerate the search for congruent four-point bases by utilizing a smart index and adding angle features. Super generalized 4PCSs [16] further reduce the number of congruent bases by employing a generalized four-point base. The robustness and effectiveness of these algorithms have been demonstrated on multiple sets of range scans.

The second approach involves registering keypoints and descriptors. Keypoint detectors, such as those based on corner and inflection point features [17], identify distinctive points in an object. Descriptors, such as SHOT [18] and FPFHs [19], are then used to describe the distribution of neighboring points around the keypoints. However, both of these methods share a common drawback: to achieve optimal results, the parameters need to be appropriately adjusted based on the shape of the object.

The third approach is the use of a principal component analysis (PCA) [20] for registration. This method estimates the position and orientation of two point clouds by analyzing their principal axes. However, it relies on the completeness of the point cloud data, meaning that the scene point cloud must be almost identical to the model point cloud so that the two principal axes align closely. To address this limitation, MRPCA [21] was developed as an improvement on the PCA method. MRPCA selects three points manually on the boundary plane to avoid the need for completeness, but this technique is not an effective solution for automated processing and measurement.

Among the various coarse registration methods, registration based on descriptors is generally considered effective [22]. However, The Keypoint–Descriptor Driven Automatic Registration (KDDAR) we proposed has addressed the issues of manual parameter adjustment and point selection, thereby improving the accuracy and efficiency of coarse registration.

3. Proposed Method

3.1. Registration Problem Description

Define two point clouds denoted by $P = \{p_1, p_2, \ldots, p_m\}$ and $Q = \{q_1, q_2, \ldots, q_n\}$, where $p_i = \{x_i, y_i, z_i\}$ and $q_j = \{x_j, y_j, z_j\}$. Let P be a set of scene point cloud data that has undergone *part-to-whole registration*. Let Q be model data produced by computer-aided design (CAD) software SOLIDWORKS 2024 and sampled by the point cloud processing software CloudCompare V2.13.2. The goal of registration is to find a transformation matrix that can be used to align P and Q in the same coordinate system regardless of their initial position and pose. This transformation matrix enables the scene point cloud to be transformed so that both point clouds have the same pose and are located in the same position.

The matrix is a homogeneous transformation matrix consisting of a 3×3 rotation matrix R and a 3×1 translation vector t which is defined as

$$T = [R \quad t], \tag{1}$$

The point cloud is transformed as follows

$$p_i' = Rp_i + t, \tag{2}$$

where p_i' denotes the transformed point from the point p_i of the scene point cloud P and p_i' belongs to the transformed point cloud P'. Then, the difference between P' and Q is compared in the same coordinate system to evaluate the registration performance.

3.2. Detector and Descriptors

3.2.1. Principal Axis Extraction

For a point cloud, the location coordinates of every point involve a wealth of information. Instead of dealing with all the information, PCA simplifies the point cloud data by extracting the eigenvalues and eigenvectors of the covariance matrix for the point cloud, i.e., $\lambda_P = \{\lambda_{P,1}, \lambda_{P,2}, \lambda_{P,3}\}$ and $\lambda_Q = \{\lambda_{Q,1}, \lambda_{Q,2}, \lambda_{Q,3}\}$, where λ_P is the eigenvalue of point cloud P and satisfies $\lambda_{P,1} > \lambda_{P,2} > \lambda_{P,3}$ and λ_Q is the eigenvalue of point cloud Q and satisfies $\lambda_{Q,1} > \lambda_{Q,2} > \lambda_{Q,3}$. Corresponding to λ_P, $e_P = \{e_{P,1}, e_{P,2}, e_{P,3}\}$ is the eigenvector of point cloud P, as is e_Q. Accordingly, we set $e_{P,1}$, $e_{P,2}$, $e_{P,3}$ as the z-axis, y-axis and x-axis of point cloud P, respectively, and $e_{Q,1}$, $e_{Q,2}$, $e_{Q,3}$ as z-axis, y-axis and x-axis of point cloud Q, respectively. The z-axis represents the direction of the largest variance in the point cloud. Then, the centroid of the point cloud is set as its coordinate origin. Figure 3 is an example of a PCA-computed coordinate axis of a point cloud of the blade. The left is the scene point cloud (rendered in red). The right is the model point cloud (rendered in green). The red axis is the x-axis, the yellow axis is the y-axis and the blue axis is the z-axis.

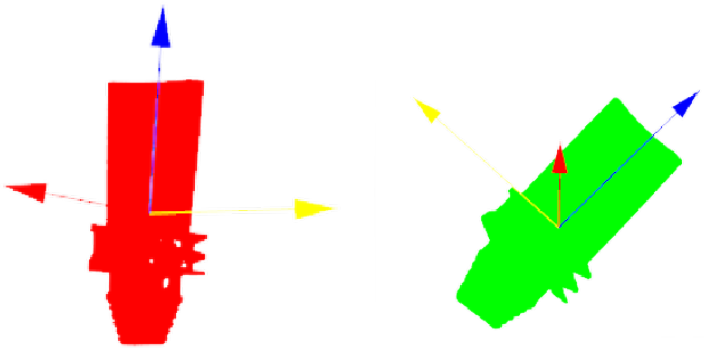


Figure 3. An example of a PCA-computed coordinate axis of the point cloud of blade.

3.2.2. Keypoint Detection

In general, rigid transformations, e.g., RANSAC [10], require at least three point pairs to produce a result. When a structure in the point cloud, such as Figure 4 has similar features, the approach is prone to producing an inaccurate result.

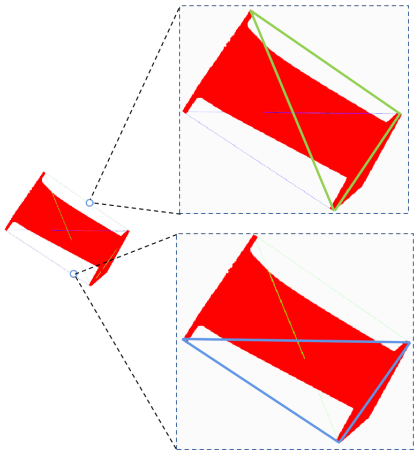


Figure 4. Two three-point bases with similar structures.

The figure depicts the registration process, where a blue three-point base from the scene point cloud is mapped to a green three-point base in the model point cloud. This traditional approach often fails due to its sensitivity to noise and structural similarities. To overcome these limitations, our study adopts the 4PCSs method, which uses four keypoints as a base for more robust registration. By expanding the base to four points, the method significantly improves the success rate of coarse registration and provides a more reliable transformation.

During the measurement process of the blade, marker points are often attached to the blade body in a scattered manner, resulting in no missing holes on the boundary areas of the scene point cloud. Additionally, there are numerous feature points at the boundary of each point cloud that can be used for feature extraction. The location information of these feature points can be utilized to classify and identify the distribution of the point cloud. Based on these factors, keypoints are detected in the boundary areas of the blade point cloud. However, for turbine blades, vertices with similar neighborhood point distributions may cause issues when selecting point pairs. As illustrated in Figure 5, an example of incorrect point pairs can occur when the top keypoints of the scene point cloud (rendered in red) are incorrectly mapped to the bottom keypoints of the model point cloud (rendered in green).

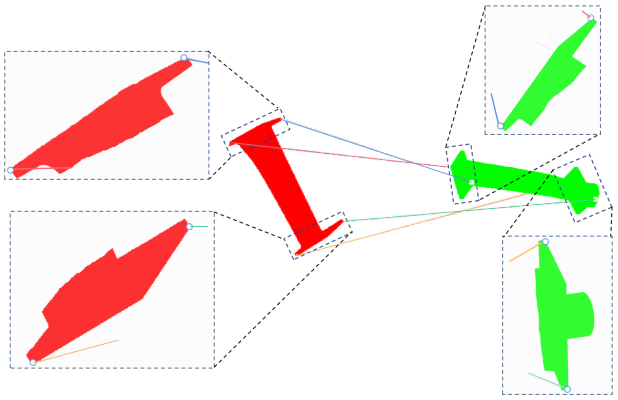


Figure 5. Incorrect keypoints pairing.

To avoid incorrect point pairing, a combination of location and distance features can effectively extract keypoints. For the distance feature, keypoints are detected as the farthest points from the two planes where the axis with the greatest variance lies (i.e., the plane XOZ and YOZ). Regarding the location feature, the point cloud is divided into four blocks based on the two principal axes, the y-axis and z-axis, corresponding to the two largest eigenvalues, as the characteristic change is not significant along the x-axis and can be disregarded. This proposal is particularly relevant for point clouds with holes, where dividing the point cloud along the x-axis may lead to incorrect results. The detailed detection process is described below.

For every point p_i in the point cloud P, we compute the vector $\overrightarrow{p_o p_i}$, where p_o is the centroid of the point cloud P. The point cloud P can be divided into four blocks along $e_{P,1}$ and $e_{P,2}$, respectively, by computing the angle between vector $\overrightarrow{p_o p_i}$ and $e_{P,1}$ or $e_{P,2}$. An example of a divided-block scene and model point cloud is displayed in Figure 6. The left is the scene point cloud and the right is the model point cloud.

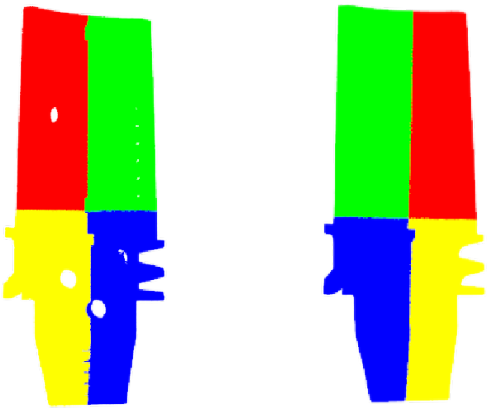


Figure 6. An example of a divided-block point cloud.

By combining location and distance feature, in every block, Equation (3) computes the distance $d_{\mathrm{XOZ},i}$ between point p_i and plane XOZ, as well as and $d_{\mathrm{YOZ},i}$ between point p_i and plane YOZ. θ_{y} is the angle between vector $\overrightarrow{p_o p_i}$ and the y-axis. θ_{x} is the angle between vector $\overrightarrow{p_o p_i}$ and the x-axis.

$$d_{\mathrm{XOZ},i} = \left\|\overrightarrow{p_o p_i}\right\| \cos\theta_{\mathrm{y}}, \quad d_{\mathrm{YOZ},i} = \left\|\overrightarrow{p_o p_i}\right\| \cos\theta_{\mathrm{x}}. \tag{3}$$

Using $d_{\mathrm{XOZ},i}$ and $d_{\mathrm{YOZ},i}$, the two farthest points in each block from the plane XOZ and YOZ are determined. This process results in eight keypoints, which form a keypoints set S_P for the scene point cloud P, and another set S_Q is obtained for the point cloud Q.

3.2.3. Location-Label Descriptors

Two location labels, L_{z} and L_{y}, are defined to describe the position of the extracted keypoints. These labels correspond to the basis used for dividing blocks in the keypoint detection step and indicate the position of each point relative to the z-axis and y-axis in the principal axis extraction step. Specifically, for point p_i, it will have a label $L_{\mathrm{P}i,\mathrm{z}}$ set to "true" when the angle between vector $\overrightarrow{p_o p_i}$ and $e_{\mathrm{P},1}$ is acute and a label $L_{\mathrm{P}i,\mathrm{y}}$ set to "true" when the angle between vector $\overrightarrow{p_o p_i}$ and $e_{\mathrm{P},2}$ is acute. Conversely, these labels will be set to "false" when the angle is obtuse. Consequently, each keypoint is assigned two labels representing its location feature descriptor. The pseudocode of the points detection and location-label descriptors is shown in Algorithm 1.

Algorithm 1 Keypoints Detector and Location-label Descriptors (taking the point cloud P as an example)

Require: The source point cloud P
Ensure: The keypoints set S_P in which every point is attached with two bool labels $L_{Pi,z}$ and $L_{Pi,y}$
// principal axis extraction
Compute principal axis $e_P = \{e_{P,1}, e_{P,2}, e_{P,3}\}$ and centroids p_0;
$e_{P,1}$ as z-axis of P; $e_{P,2}$ as y-axis of P; $e_{P,3}$ as x-axis of P;
// keypoint detection
for each $p_i \in P$ **do**
  compute θ_x between vector $\overline{p_0 p_i}$ and x-axis; θ_y between vector $\overline{p_0 p_i}$ and y-axis;
  compute the distance $d_{XOZ,i}$ and $d_{YOZ,i}$ according to Equation (3);
end for
Divide the point cloud P into blocks according to the z-axis and y-axis;
for each block in P **do**
  extract the largest $d_{XOZ,i}$, $p_i \rightarrow S_P$;
  extract the largest $d_{YOZ,i}$, $p_i \rightarrow S_P$;
end for
// location-label descriptors
for each keypoints $p_i \in S_P$ **do**
  if the angle between vector $\overline{p_0 p_i}$ and y-axis $< \pi/2$ **then**
    $L_{Pi,y} \leftarrow$ true;
  else
    $L_{Pi,y} \leftarrow$ false;
  end if
  if the angle between vector $\overline{p_0 p_i}$ and z-axis $< \pi/2$ **then**
    $L_{Pi,z} \leftarrow$ true;
  else
    $L_{Pi,z} \leftarrow$ false;
  end if
end for

3.3. Pair Searching Mechanism

Two pairing mechanisms are introduced for pair searching: block pairing and point pairing. These two searching strategies are implemented sequentially as shown in Algorithm 2.

Algorithm 2 Pair Searching Mechanism

Require: The keypoints set S_P and S_Q
Ensure: A pairing-base set S_B
// block pairing searching
for each block pairing situation in Table 1 **do**
  // point pairing searching
  for each block in point cloud P and correspoinding block in point cloud Q **do**
    Choose a point p_i and q_j as a point pair;
  end for
  Four point pairs as a pairing base;
  Add the pairing base to S_B;
  // iterate through all point pair cases
end for

Table 1. Four kinds of situations of block pairing.

Situation	Scene Point Cloud	Model Point Cloud
1	$L_{Pi,z}$&& $L_{Pi,y}$	$L_{Qi,z}$&& $L_{Qi,y}$
2	$L_{Pi,z}$&&$L_{Pi,y}$	$L_{Qi,z}$&& !$L_{Qi,y}$
3	$L_{Pi,z}$&& $L_{Pi,y}$	!$L_{Qi,z}$&& $L_{Qi,y}$
4	$L_{Pi,z}$&& $L_{Pi,y}$	!$L_{Qi,z}$&& !$L_{Qi,y}$

3.3.1. Block Pairing Searching

The axes of the scene point cloud P obtained by the PCA may differ from those of the model point cloud Q due to missing parts. The directions of the two eigenvectors may be shifted or even reversed. For example, in Figure 3, the direction of the y-axis in the scene point cloud P is opposite to that in the model point cloud Q. Therefore, the same label is attached to a red block of the scene point cloud and a green block of the model point cloud in Figure 6. As the point cloud is divided along the y-axis and z-axis, the directions of these two axes need to be taken into account. Hence, there are four block pairing situations, as shown in Table 1. $L_{\mathrm{P}i,\mathrm{z}}$ and $L_{\mathrm{P}i,\mathrm{y}}$ represent the Boolean values of the z-axis label and y-axis label for the point cloud P, respectively, which are introduced in the location-label descriptors step in Section 3. $L_{\mathrm{Q}i,\mathrm{z}}$ and $L_{\mathrm{Q}i,\mathrm{y}}$ are defined similarly. The symbol ! before L indicates the *negation* operation, while && denotes the *with* operation.

3.3.2. Point Pairing Searching

In each block pairing situation, the finest point pair needs to be found in every corresponding block for the two point clouds. For point pairing, the finest corner point in every block is selected from the keypoints set at S_{P} and S_{Q}, which is introduced in the keypoint detection step in Section 3. Consequently, for every block, the keypoints in the scene point cloud are paired with the keypoints in the model point cloud, resulting in four pairs in each block. The most suitable point pair can be selected from the four pairs.

For each block pairing situation, there are $\left(C_4^1\right)^4$ possible pairing bases formed by the extracted point pairs, which constitute a set of pairing bases S_B. The finest pairing base in S_B is selected as the final result. C represents an *unordered* operation.

3.4. Rigid Transformation Estimation

We iterate through each base in the pairing-base set and use singular value decomposition (SVD) [23] to compute their related transformation matrices in order to identify the optimum four-keypoint base. By comparing the mean square error (MSE) between the transformed scene cloud and the model cloud, the finest pairing base in the set S_B is determined. The following is the MSE calculation equation:

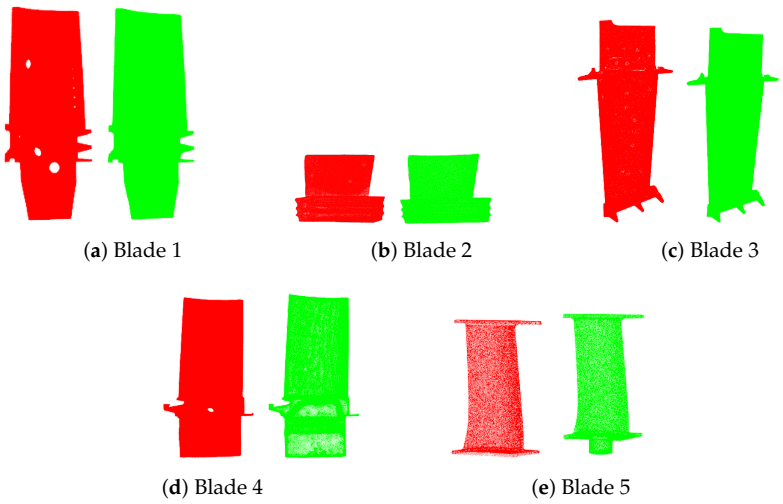
$$MSE = \frac{\sum_{i=1}^{m} \epsilon_{ij}^2}{m}, \tag{4}$$

where ϵ_{ij} represents the distance between p_i' and q_j, i.e., $\epsilon_{ij} = \left\| p_i' - q_j \right\|$, and q_j is the closest point from p_i' by searching KD-tree [24] in the point cloud Q and m is the number of the scene point cloud.

4. Experiments

4.1. Experiment Setup

The experiments were conducted using five blades, as shown in Figure 7. Details of the blades are given in Table 2. Blades 1–4 were scanned with marker points on the body since their sizes were not large enough to be clamped. However, as a result of the scanning process, there are missing parts (holes) on the blade body of Blades 1–4 due to the occlusion of marker points. Blade 5 was placed on a fixture table during scanning, resulting in a missing part at the bottom. For the model point cloud, CAD data of the five blades were sampled by the point cloud processing software CloudCompare. The number of points and the size of the blades are also shown in Table 2. The bounding box dimensions represent the length, width and height of the point cloud, respectively.



(**a**) Blade 1 (**b**) Blade 2 (**c**) Blade 3

(**d**) Blade 4 (**e**) Blade 5

Figure 7. Original experiment data. The left is the scene point cloud. The right is the model point cloud.

Table 2. Experiment data for the scene point cloud and model point cloud of five blades.

Point Cloud	Number of Points		Bounding Box Dimensions		
	Scene	Model	Length (mm)	Width (mm)	Height (mm)
Blade 1	393,419	812,541	98.9793	42.8394	30.0117
Blade 2	505,177	235,153	77.1192	61.0042	22.3977
Blade 3	316,136	651,775	122.395	56.6682	46.0813
Blade 4	475,620	400,243	98.5815	56.4624	33.6747
Blade 5	4,687,185	5,000,000	314.598	206.38	101.325

The proposed algorithm was implemented in C++ on a PC equipped with an Intel i5 2.6 GHz processor, 8 GB of memory and a NVIDlA GeForce GTX 1650 SUPER. A KD tree search algorithm has high efficiency, adaptability and simplicity during a multi-dimensional spatial data search. Therefore, we choose the KD tree search in the point cloud library (PCL) [25]. It can recursively divide space in each dimension, build a tree structure, and quickly locate the nearest neighbor points in the cloud, effectively process large-scale point cloud data, and provide a good initial pose for subsequent fine registration. The algorithm was tested on the five blades described earlier. Due to the large number of points in Blade 5, which reached millions, the computational and memory burdens were high. Therefore, the point clouds were downsampled to 19,322 and 29,265 points for preprocessing. To improve computational efficiency, the MSE was computed at intervals of one-hundredth of the number of points in the source point cloud.

4.2. Results and Evaluation

In order to compare the performance of the proposed algorithm, we also implemented PCA [20], Super 4PCSs [15] and RANSAC [12], as shown in Figure 8 and Table 3, respectively. RANSAC combines Harris 3D [17] as a keypoint detector and FPFHs [19] as a descriptor for point cloud registration. In Figure 8, for Blade 2, the PCA fails with a large inclination angle. There is a converse result for Blade 4. The registration result of the PCA for Blade 5 is a deviation in height because the bottom of the scene point cloud is missing. Super 4PCSs produce a better result than the PCA in all blades. However, they all have similar deviations. RANSAC has an unstable alignment effect and needs to adjust the parameters several times according to the experimental object to obtain better registration results.

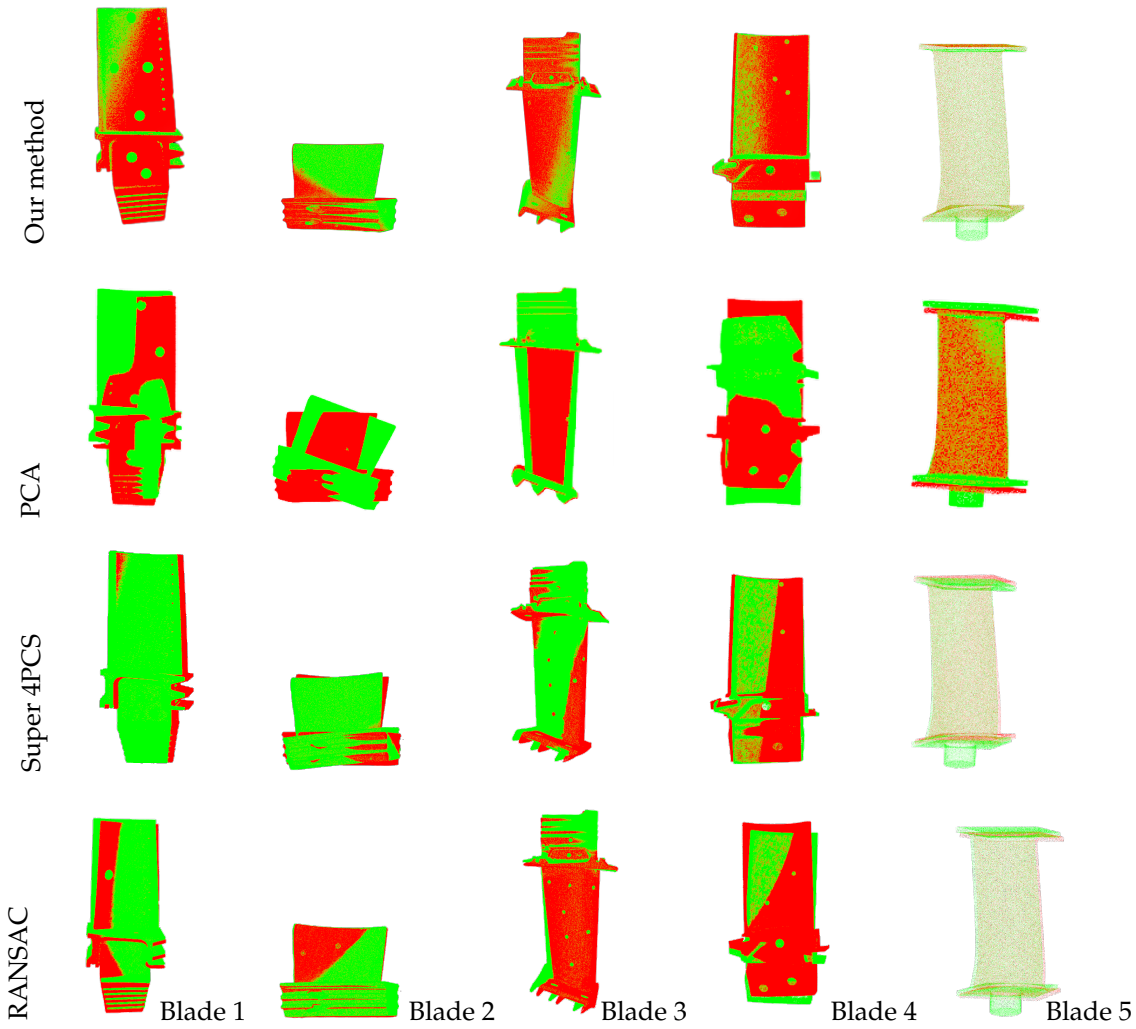


Figure 8. Registration experiment result, red is the scene point cloud and green is the model point cloud. From left to right are Blade 1 to Blade 5, respectively.

Table 3. The MSE comparing of KDDAR (ours), PCA, Super 4PCS and RANSAC.

Point Cloud	MSE (mm^2)			
	KDDAR (Ours)	PCA	Super 4PCS	RANSAC
Blade 1	0.00877	8.30541	1.13038	0.802914
Blade 2	0.1200	15.9042	1.39444	0.526328
Blade 3	0.04067	2.28668	1.05332	2.77349
Blade 4	0.03157	24.2637	1.73901	6.91692
Blade 5	0.81494	7.79815	9.93981	15.3917

Based on the results shown in Figure 8 and Table 3, we observed that the PCA had the largest error for Blade 2 and Blade 4, while Super 4PCSs demonstrated improved performance for Blade 1 to Blade 4. RANSAC performed better for Blade 1 and Blade 2. In contrast, our algorithm consistently achieved a lower MSE for all five blades compared to PCA, Super 4PCSs and RANSAC. Our method outperformed the other methods in terms of accuracy according to both qualitative and quantitative analysis. It is important to note that both Super 4PCSs and RANSAC require parameter adjustments depending on the scanned object. For our experiments, we determined the optimal parameters for Super 4PCSs and RANSAC as follows: for all blades, we set the overlap rate to 0.9 for Super 4PCSs and the delta values to 2, 3, 5, 5, and 10 for Blade 1 to Blade 5, respectively. For RANSAC, the radius was set to 1 for Blade 1–2 and 5 and to 3 for Blade 3–4 in Harris 3D keypoint

detection. In the FPFH descriptor step, we set the K-search to 10 for the source point cloud and 15 for the target point cloud for Blade 1–4 and to 15 and 20, respectively, for Blade 5.

To enable a clearer comparison of experimental results, we selected Blade 3 for a comparison of color deviation graphs, as shown in Figure 9. The maximum and minimum ranges of the color fields are set to 1 mm and −1 mm, respectively, and the gray-blue part in the figure indicates the area outside the color display range. A tolerance of 0.1 mm is specified, and, as a result, 61%, 58%, 48% and 47% of points fall within the tolerance for our method, PCA coarse registration, Super 4PCS coarse registration and RANSAC coarse registration, respectively.

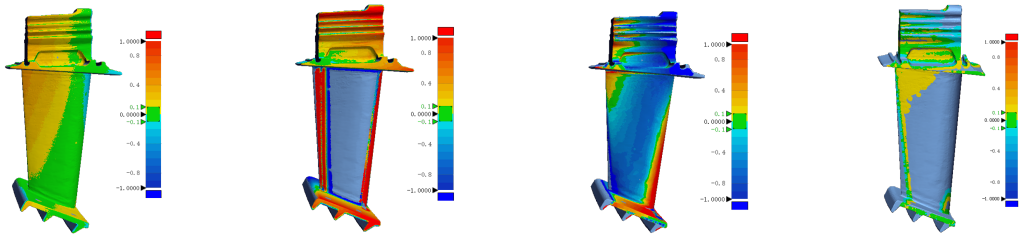


Figure 9. Color deviation graph comparison for Blade 3 about three common methods. From left to right are KDDAR (ours), PCA, Super 4PCSs, and RANSAC, respectively.

To compare the deviations of a certain cross-section of Blade 3, we show the deviation comparison in Figure 10. The connecting lines between the points indicate the corresponding point pairs used for deviation calculation. Due to the influence of the main direction sign on the rotation matrix, PCA registration performs the worst for Blade 3, resulting in the opposite direction of the blade after registration. Table 4 presents the quantitative analysis of the registration deviation for a certain cross-section of Blade 3, including the maximum deviation, minimum deviation, and mean deviation.

Table 4. Quantitative analysis of the registration deviation of a certain cross section of Blade 3.

Method	Deviation (mm)		
	Max	Min	Mean
KDDAR (ours)	0.3322	−0.3126	0.0149
PCA	3.9349	−3.0225	0.9351
Super 4PCS	1.6723	−1.1075	0.0284
RANSAC	3.0186	−3.0209	0.241

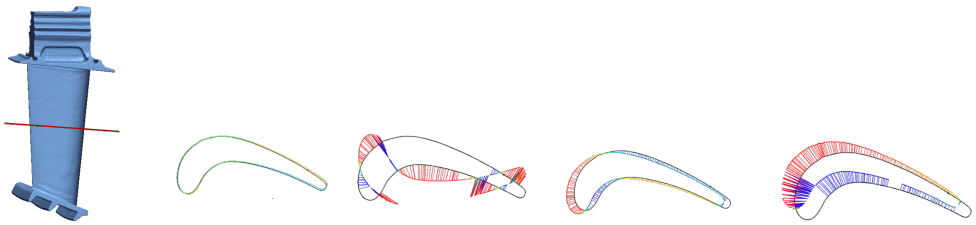


Figure 10. Comparison of the registration deviation of a certain cross-section of Blade 3. From left to right are KDDAR (ours), PCA, Super 4PCSs and RANSAC, respectively.

To conclude, simply comparing the MSE of each coarse registration method is not sufficient. A combination of tolerance ratios and intuitive visual comparisons allows for a better judgment of the effectiveness of the registration method. Our method yields good results, as seen in both visual and quantitative analyses, providing good initial poses for later fine registration. This reduces the number of iterations needed for the fine registration algorithm and improves registration efficiency.

5. Conclusions

This article presents a novel algorithm for the coarse registration of engine blades' scene-to-model registration process, particularly for free-form surfaces. To address the issue of PCR failure caused by missing parts in the point cloud, the paper proposes a four-point orientation point cloud registration method which incorporates a location-label descriptor and keypoint detector. The experimental results demonstrate that our method can effectively provide a good initial pose for the later fine registration, and it is robust enough to be utilized with different shapes of measured subjects without the need for manual parameter adjustments. In future work, we plan to combine our proposed coarse registration method with ICP fine registration to evaluate the processing error of engine blades more accurately.

Author Contributions: Conceptualization, Z.J., Z.W., Q.F. and X.Z.; Methodology, D.L., Y.Z., Z.J., Z.W., Q.F. and X.Z.; Formal analysis, Y.Z.; Investigation, D.L. and Y.Z.; Data curation, D.L., Y.Z. and Z.J.; Writing—original draft, D.L. and Z.W.; Writing—review & editing, Z.J., Z.W., Q.F. and X.Z.; Supervision, Z.W. and Q.F. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the National Natural Science Foundation of China under Grant (U23A20385, 62171184, 62273139), the Key scientific research projects of China Southern Power Grid (031200KC23120030), the Special funding support for the construction of innovative provinces in Hunan Province (2023JJ30162, 2021GK1010, 2024JJ8247), Hunan Provincial Department of Education Scientific Research Project (23B0029), the Major Project of Yuelushan Industrial Innovation Center (2023YCII0102).

Data Availability Statement: Data will be made available on request.

Conflicts of Interest: Author Duanjiao Li, Ying Zhang and Ziran Jia were employed by the company Aircraft Patrol Management Center, Guangdong Power Grid Co., Ltd. Author Zhiyu Wang was employed by the company Shanxi Intelligent Transportation Research Institute Co., Ltd. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

References

1. Wu, Z.; Wang, Y.; Mo, Y.; Zhu, Q.; Xie, H.; Wu, H.; Feng, M.; Mian, A. Multiview Point Cloud Registration Based on Minimum Potential Energy for Free-Form Blade Measurement. *IEEE Trans. Instrum. Meas.* **2022**, *71*, 5011014. [CrossRef]
2. Diez, Y.; Roure, F.; Lladó, X.; Salvi, J. A qualitative review on 3D coarse registration methods. *ACM Comput. Surv. (CSUR)* **2015**, *47*, 1–36. [CrossRef]
3. Besl, P.; McKay, N.D. A method for registration of 3-D shapes. *IEEE Trans. Pattern Anal. Mach. Intell.* **1992**, *14*, 239–256. [CrossRef]
4. Chetverikov, D.; Svirko, D.; Stepanov, D.; Krsek, P. The trimmed iterative closest point algorithm. In Proceedings of the 2002 International Conference on Pattern Recognition, Quebec City, QC, USA, 11–15 August 2002; Volume 3, pp. 545–548.
5. Segal, A.; Haehnel, D.; Thrun, S. Generalized-icp. In Proceedings of the Robotics: Science and Systems, Seattle, WA, USA, 28 June–1 July 2009; Volume 2, p. 435.
6. Yang, J.; Li, H.; Jia, Y. Go-icp: Solving 3d registration efficiently and globally optimally. In Proceedings of the IEEE International Conference on Computer Vision, Sydney, Australia, 1–8 December 2013; pp. 1457–1464.
7. Koide, K.; Yokozuka, M.; Oishi, S.; Banno, A. Voxelized gicp for fast and accurate 3d point cloud registration. In Proceedings of the 2021 IEEE International Conference on Robotics and Automation (ICRA), Xi'an, China, 30 May–5 June 2021; pp. 11054–11059.
8. Zhang, J.; Yao, Y.; Deng, B. Fast and robust iterative closest point. *IEEE Trans. Pattern Anal. Mach. Intell.* **2021**, *44*, 3450–3466. [CrossRef] [PubMed]
9. Wu, J. Rigid 3-D registration: A simple method free of SVD and eigendecomposition. *IEEE Trans. Instrum. Meas.* **2020**, *69*, 8288–8303.
10. Fischler, M.A.; Bolles, R.C. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM* **1981**, *24*, 381–395. [CrossRef]
11. Arena, P.; Baglio, S.; Fortuna, L.; Manganaro, G. Self-organization in a two-layer CNN. *IEEE Trans. Circuits Syst. I Fundam. Theory Appl.* **1998**, *45*, 157–162. [CrossRef]
12. Quan, S.; Yang, J. Compatibility-guided sampling consensus for 3-d point cloud registration. *IEEE Trans. Geosci. Remote Sens.* **2020**, *58*, 7380–7392. [CrossRef]
13. Wu, H.; Wang, Y.; Vela, P.A.; Zhang, H.; Yuan, X.; Zhou, X. Geometric Inlier Selection for Robust Rigid Registration with Application to Blade Surfaces. *IEEE Trans. Ind. Electron.* **2021**, *69*, 9206–9215. [CrossRef]

14. Aiger, D.; Mitra, N.J.; Cohen-Or, D. 4-points congruent sets for robust pairwise surface registration. In *ACM SIGGRAPH 2008 Papers*; Association for Computing Machinery: New York, NY, USA, 2008; pp. 1–10.
15. Mellado, N.; Aiger, D.; Mitra, N.J. Super 4pcs fast global pointcloud registration via smart indexing. In *Computer Graphics Forum*; Wiley Online Library: Hoboken, NJ, USA, 2014; Volume 33, pp. 205–215.
16. Mohamad, M.; Ahmed, M.T.; Rappaport, D.; Greenspan, M. Super generalized 4pcs for 3d registration. In Proceedings of the 2015 International Conference on 3D Vision, Lyon, France, 19–20 October 2015; pp. 598–606.
17. Sipiran, I.; Bustos, B. Harris 3D: A robust extension of the Harris operator for interest point detection on 3D meshes. *Vis. Comput.* **2011**, *27*, 963–976. [CrossRef]
18. Salti, S.; Tombari, F.; Di Stefano, L. SHOT: Unique signatures of histograms for surface and texture description. *Comput. Vis. Image Underst.* **2014**, *125*, 251–264. [CrossRef]
19. Rusu, R.B.; Blodow, N.; Beetz, M. Fast point feature histograms (FPFH) for 3D registration. In Proceedings of the 2009 IEEE International Conference on Robotics and Automation, Kobe, Japan, 12–17 May 2009; pp. 3212–3217.
20. Kim, C.; Son, H.; Kim, C. Fully automated registration of 3D data to a 3D CAD model for project progress monitoring. *Autom. Constr.* **2013**, *35*, 587–594. [CrossRef]
21. Liu, J.; He, X.; Huang, X. An improved registration strategy for aligning incomplete blade measurement data to its model. *Optik* **2021**, *243*, 167304. [CrossRef]
22. Lei, H.; Jiang, G.; Quan, L. Fast descriptors and correspondence propagation for robust global point cloud registration. *IEEE Trans. Image Process.* **2017**, *26*, 3614–3623. [CrossRef] [PubMed]
23. Sorkine, O. Least-squares rigid motion using svd. *Technol. Notes* **2009**, *120*, 52.
24. Liu, Y.; Xiong, Y.l. Algorithm for searching nearest-neighbor based on the bounded kd tree. *J. Huazhong Univ. Sci. Technol. (Nat. Sci. Ed.* **2008**, *36*, 73.
25. Rusu, R.B.; Cousins, S. 3d is here: Point cloud library (pcl). In Proceedings of the 2011 IEEE International Conference on Robotics and Automation, Shanghai, China, 9–13 May 2011; pp. 1–4.