ICIC Express Letters
Volume **10**, Number **9**, September **2016**

 ISSN 1881-803X
pp. **2283–2289**

POINT CLOUD COARSE MATCHING METHOD FOR SIMILAR OBJECTS

Qiang Zhao[1,2], Xijuan Guo[1,2], Buying Zhang[1,2] and Xinyu Bai[3]

[1]Key Laboratory for Computer Virtual Technology and System Integration of Hebei Province
[2]College of Information Science and Engineering
Yanshan University
No. 438, Hebei Ave., Qinhuangdao 066004, P. R. China
hmoe@vip.qq.com; xjguo@ysu.edu.cn

[3]Technical Center
Shenyang Aircraft Corporation
Shenyang 110850, P. R. China

Received February 2016; accepted May 2016

Abstract. *Point cloud matching is much more difficult when an initial estimate of the relative pose is not known. In order to reduce the effect of relative pose on matching operation and improve the computing speed, this article presents a method to fast match the pose of two similar objects through the covariance of their point cloud coordinates. This paper contains three parts. In the first part, a matrix is constructed through the values of point coordinates of one object and then a set of orthogonal vectors will be obtained through eigenvector of the matrix. Rotation matrix will be obtained by comparing the vectors of two similar objects. In the second part, a description of procedures is given and the complexity of the algorithm is analyzed. In the last part, the experiment shows the proposed method has high computing efficiency and the matching results are acceptable.*
Keywords: Point cloud, Covariance matrix, Pose adjustment, Coarse matching

1. **Introduction.** 3D modeling, object detection, and pose estimation are three of the most challenging tasks in the area of 3D computer vision. Most present recognition methods identify two objects through feature points and specific shape matching. The pose estimation through unordered point-clouds has already been used in many fields, like man face or industrial parts pose recognition. Barros et al. [1] presented a novel approach to estimate the human pose from a body-scanned point cloud. Figueiredon et al. [2] proposed algorithms for 3D object recognition from 3D point clouds of rotationally symmetric objects. Although it shortens the computing time, it can only recognize the symmetric objects. Berner et al. [3] enhanced a probabilistic graph-matching approach that detects objects by using 3D shape primitives with distinct 2D primitives such as circular contours. The method depends on a composition of shape primitives like cylinders, planes, cones and spheres. Nguyen et al. [4] used point-pair feature for matching instead of traditional approaches using local feature descriptors. The object model is a set of point pair descriptors computed from CAD model.

Point cloud matching is a central problem in the pose estimation. Although the problem is well studied in the case when an initial estimate of the relative pose is known (fine matching), the problem becomes much more difficult when a priori knowledge is not available (coarse matching). Guo et al. [5] reconstructed the model for a set of input point-clouds in the presence of clutter and occlusion using a novel model growing technique. The algorithm does not rely on any prior information about the objects in the scene, but it needs a lot of computing. In order to speed up the matching process, many methods have been presented. Diez et al. [6] introduced a novel technique to speed up coarse matching algorithms for point clouds. Cheung et al. [7] reduced the problem to 3 dimensions using

contours computed from horizontal slices. However, the method only suits objects lying stably on a planar surface.

Among so many methods, Fehr et al. [8] introduced various descriptors to match features in a point cloud and proved covariance to be a successful method in point cloud matching processing. However, covariance is used directly to look for features in point clouds; therefore, the algorithm is complex and a lot of computing time is required. In order to speed up the coarse matching process, this paper uses covariance to make two objects have the same pose, which need to be recognized. The whole article is divided into three parts. The principle of the coarse matching method is denoted in the first part. Then, the algorithm based on the principle is given and the complexity of the program is analyzed. At last, several similar 3D models are used to test the efficiency of the method.

2. Coarse Matching Principle.

During object detection process, most matching methods need to find the corresponding features. That is difficult because the pose of objects needed to be detected is uncertain. Therefore, many other methods take much time to suppress interference caused by pose. The coarse matching method is designed to work out a rotation matrix used to change the pose before the detection process.

In this paper, 3D point sets are the data to be processed. These elements of point sets can be got from a 3D scanner or other equipments, and they also can be the points in 3D models. The point coordinates are the only information required in the method.

2.1. Covariance matrix.

The coarse matching between objects can be got through the object contour, which can be obtained by the method of OBB (Oriented Bounding Box) or convex hull. Because the convex hull is too complex and OBB requires polygons information, a method only requiring the point coordinates is designed.

Similar to the OBB computation algorithm, the coarse matching method uses the first and second order statistics to summarize the point coordinates. They are the mean, μ, and the covariance matrix, C, respectively. If the ith point coordinate is the $p_i(x_i, y_i, z_i)$, then the mean and covariance matrix can be expressed as

$$\mu = \begin{bmatrix} \bar{x} \\ \bar{y} \\ \bar{z} \end{bmatrix} = \frac{1}{n} \sum_{i=0}^{n} \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix}$$

$$C = \frac{1}{n} \sum_{i=0}^{n} \begin{bmatrix} x_i^2 & x_i y_i & x_i z_i \\ y_i x_i & y_i^2 & y_i z_i \\ z_i x_i & z_i y_i & z_i^2 \end{bmatrix}$$

where n is the number of points, $x_i = x_i - \bar{x}$, $y_i = y_i - \bar{y}$, $z_i = z_i - \bar{z}$.

The eigenvectors of a symmetric matrix, such as C, are mutually orthogonal. After being normalized, they are used as a basis. Then the extremal points along each axis of this basis are found. After the length of projection of the maximum and minimum points on the axis of one basis is compared, the direction of this axis is defined to be the direction of the point having longer projection. Finally, a set of vectors with new direction is obtained, such as $\{\vec{u}, \vec{v}, \vec{w}\}$. Different from the OBB algorithm, the more the points are, the more accurate the coarse matching method is.

2.2. Rotation matrix.

If there are two objects named object 1 and object 2, the covariance matrices of two objects can be obtained after their point clouds are got, and then their feature vectors can be computed by the upper subsection. Set the two sets of feature vectors as $\{\vec{u_1}, \vec{v_1}, \vec{w_1}\}$ and $\{\vec{u_2}, \vec{v_2}, \vec{w_2}\}$ respectively. Then two pairs of corresponding vectors are randomly selected to compute the rotation matrix. Rodrigues' Rotation Formula (RRF) is used to work out the rotation matrix; therefore, two rotation processes are needed to be computed.

If object 1 is fixed, object 2 needs to be rotated to the same pose as that of object 1. Then a pair of corresponding vectors $\vec{u_1}$ and $\vec{u_2}$ are used in the first rotation computation. Based on the *RRF*, the included angle θ, and the rotation axis $\vec{V}$ can be expressed as

$$\theta = \arccos\left(\frac{\vec{u_1}\cdot\vec{u_2}}{|\vec{u_1}||\vec{u_2}|}\right)$$

$$\vec{V} = \begin{bmatrix} V_x \\ V_y \\ V_z \end{bmatrix} = \begin{bmatrix} u_{2y}u_{1z} - u_{2z}u_{1y} \\ u_{2z}u_{1x} - u_{2x}u_{1z} \\ u_{2x}u_{1y} - u_{2y}u_{1x} \end{bmatrix}$$

where $\vec{u_1} = (u_{1x}, u_{1y}, u_{1z})$ and $\vec{u_2} = (u_{2x}, u_{2y}, u_{2z})$. Set $\vec{\lambda}(\lambda_x, \lambda_y, \lambda_z)$ as the unit vector of $\vec{V}$, $\vec{\lambda} = \vec{V}/|\vec{V}|$. Then the first rotation matrix $R^1_\lambda(\theta)$ can be shown as $R^1_\lambda(\theta) =$

$$\begin{bmatrix} \cos\theta + \lambda_x^2(1-\cos\theta) & \lambda_x\lambda_y(1-\cos\theta) - \lambda_z\sin\theta & \lambda_y\sin\theta + \lambda_x\lambda_z(1-\cos\theta) \\ \lambda_z\sin\theta + \lambda_x\lambda_y(1-\cos\theta) & \cos\theta + \lambda_y^2(1-\cos\theta) & -\lambda_x\sin\theta + \lambda_y\lambda_z(1-\cos\theta) \\ -\lambda_y\sin\theta + \lambda_x\lambda_z(1-\cos\theta) & \lambda_x\sin\theta + \lambda_y\lambda_z(1-\cos\theta) & \cos\theta + \lambda_z^2(1-\cos\theta) \end{bmatrix}$$

Then, $\vec{u_1} = R^1_\lambda(\theta)\vec{u_2}$ can be known and the feature vector set $\{\vec{u_2}, \vec{v_2}, \vec{w_2}\}$ will be changed into $\{R^1_\lambda(\theta)\vec{u_2}, R^1_\lambda(\theta)\vec{v_2}, R^1_\lambda(\theta)\vec{w_2}\}$ after the first rotation. Therefore, the $\vec{v_1}$ and $R^1_\lambda(\theta)\vec{v_2}$ are the corresponding vector pair used in the second rotation matrix computation process. Set $R^1_\lambda(\theta)\vec{v_2}$ to be $\vec{v_r}$. Because the feature vectors are mutually orthogonal, the second rotation angle and rotation axis are expressed as

$$\theta = \arccos\left(\frac{\vec{v_1}\cdot\vec{v_r}}{|\vec{v_1}||\vec{v_r}|}\right)$$

$$\vec{V} = \vec{u_1} = R^1_\lambda(\theta)\vec{u_2}$$

The second rotation matrix $R^2_\lambda(\theta)$ can be obtained by the same formula above. Finally, the complete rotation matrix $R = R^2R^1$ is obtained.

Object 2 will be in the same pose with object 1 after each point in the point cloud of object 2 is multiplied by the rotation matrix R at the left side.

3. **Algorithm Description and Analysis.** According to Section 2, an algorithm is described to compute the rotation matrix and pseudocode is provided to analyze the complexity of the algorithm. The point sets of object 1 and object 2 are the only input parameters needed in this algorithm and a 3×3 matrix R is the result.

Algorithm 1 rotation matrix computation

Procedure: Rotation-Matrix(P1, P2)
Input: point sets of two objects *P1* and *P2*
Output: the rotation matrix R of object 2 to object 1
1. Compute covariance matrixes *M1* and *M2* based on Section 2.1
   using the input parameters *P1* and *P2*
2. *V1* $\leftarrow$ Jacobi-Eigenvectors(*M1*)
3. *V2* $\leftarrow$ Jacobi-Eigenvectors(*M2*)
4. Compute included angle A of corresponding vectors *V1*[0] and *V2*[0].
5. Compute cross product L of corresponding vectors *V1*[0] and *V2*[0].
6. Compute first rotation matrix $R1$ based on Section 2.2 using A and L.
7. *V2* $\leftarrow$ $R1 \times V2$
8. Compute included angle A of corresponding vectors *V1*[1] and *V2*[1].
9. Compute second rotation matrix $R2$ using A and *V1*[0].
10. $R \leftarrow R1 \times R2$
return R;

In Algorithm 1, there is a procedure named Jacobi-Eigenvectors used to compute the feature vectors of covariance matrixes. The algorithm is obtained according to the theory of Jacobi matrix and the specific steps are described as Algorithm 2. Due to the special nature of this paper, all the matrixes used in the algorithm are three factorial square matrixes.

Algorithm 2 feature vectors computation

Procedure: Jacobi-Eigenvectors(M)
Input: 3×3 covariance matrix M
Output: 3×3 vectors matrix V, in which each column is a feature vector
1. $circl \leftarrow 0$
2. Initialize V as an identity matrix
3. **while** $circl < num$ and $M[row][col] < precision$
4. $circl++$
5. Find the position (row, col) of the maximum value
in matrix M except its diagonal
6. $angle \leftarrow 0.5 * \arctan(2 * M[row][col]/(M[row][row] - M[col][col]))$
7. $M[row][row] \leftarrow M[row][row] * \cos(angle) * \cos(angle) + M[col][col]$
$* \sin(angle) * \sin(angle) + M[row][col] * \cos(angle) * \sin(angle)$
8. $M[col][col] \leftarrow M[row][row] * \sin(angle) * \sin(angle) + M[col][col]$
$* \cos(angle) * \cos(angle) - 2 * M[row][col] * \cos(angle) * \sin(angle)$
9. $M[row][col] \leftarrow 0.5 * (M[col][col] - M[row][row]) * \sin(2 * angle)$
$+M[row][row] * \cos(2 * angle)$
10. $M[col][row] \leftarrow M[row][col]$
11. **for** $i \leftarrow 0$ **to** 2
12. **if** $i \neq col$ and $i \neq row$
13. $temp \leftarrow M[i][row]$
14. $M[i][row] \leftarrow M[i][col] * \sin(angle) + temp * \cos(angle)$
15. $M[i][col] \leftarrow M[i][col] * \cos(angle) - temp * \sin(angle)$
16. **for** $i \leftarrow 0$ **to** 2
17. **if** $i \neq col$ and $i \neq row$
18. $temp \leftarrow M[row][i]$
19. $M[row][i] \leftarrow M[col][i] * \sin(angle) + temp * \cos(angle)$
20. $M[col][i] \leftarrow M[col][i] * \cos(angle) - temp * \sin(angle)$
21. **for** $i \leftarrow 0$ **to** 2
22. $temp \leftarrow V[i][row]$
23. $V[i][row] \leftarrow V[i][col] * \sin(angle) + temp * \cos(angle)$
24. $V[i][col] \leftarrow V[i][col] * \cos(angle) - temp * \sin(angle)$
return V;

In line 1 of Algorithm 1, the computation time is determined by the number n of point sets and time spent on traversing all elements in point set is $O(n)$. All the computation in lines 4 to 9 of Algorithm 1 is obtained by one cycle and the spent time is $O(1)$. Lines 2 and 3 of Algorithm 1 are shown as Algorithm 2, wherein the cycle index is determined by the parameter *num* and *precision*. The smaller the *precision* is, the more cycle times are. Therefore, the time spent on this algorithm is $O(m)$ and m is the maximum cycle times.

According to above analysis, the time spent in the coarse matching method is determined by the maximum value between m and n. Generally, the number of elements in point set is large enough, and the accuracy can meet the requirements when $m < 10$. Therefore, the whole time spent is $O(n)$ and time complexity is acceptable.

4. **Experiment and Analysis.** Three airfoil 3D models are used in the algorithm experiment. These three models are Airbus-A320, B747 and F16 respectively. Their point clouds picked up from models are used as the input of Algorithm 1. As shown in Figure 1, there are two airbuses with different poses on the left of the figure and they are put together on the top right. After the computation, they will be in the pose as shown on the lower right.

When the objects are different, the coarse matching method is also useful. In Figure 2, the pose of A320 is on the top left, the pose of B747 after computation will be shown on the lower of the figure. The parameter *precision* in Algorithm 2 decides the differences between the two lower poses. The two poses are much closer when the *precision* is much higher.

Even though there is quite a difference between the two airfoils, like A320 and F16, the matching result is still effective as shown in Figure 3.

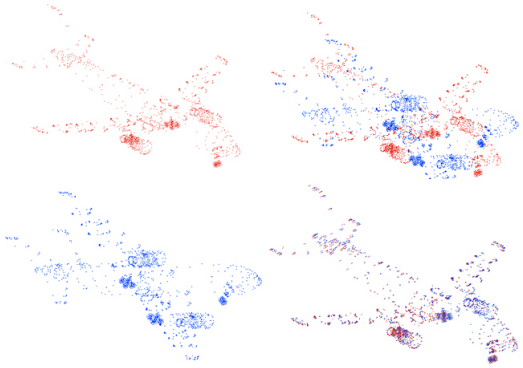

FIGURE 1. The coarse matching process of two Airbus-A320

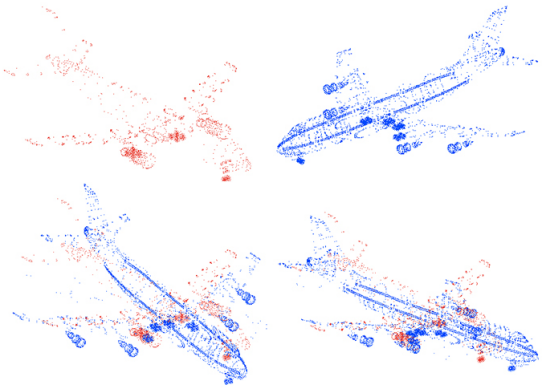

FIGURE 2. The coarse matching process between Airbus-A320 and B747

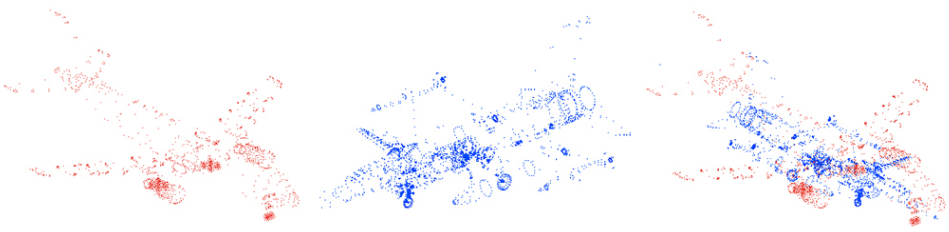

FIGURE 3. The coarse matching process between Airbus-A320 and F16

FIGURE 4. The coarse matching between points from 3D model and 3D scanner

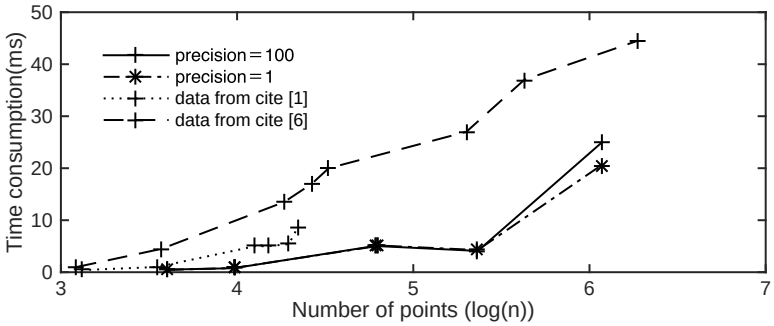

FIGURE 5. The relation between time consumption and number of points

This coarse matching method also can be used to match point clouds obtained through 3D scanner. As shown in Figure 4, the left point cloud is obtained from 3D model and the middle one is obtained from 3D scanner. They are achieved from the same component of an airfoil. All the figures of matching results are better than those in cite [6] and cite [9].

The efficiency of the algorithm only depends on the number of points. As shown in Figure 5, the larger the number of points is, the more the time consumption is, and the time consumption is almost the same when $precision = 100$ or $precision = 1$. According to the data from cite [1] and cite [6], the time consumption is less than that in cite [1] and cite [6] obviously. This further illustrates that the algorithm proposed in this paper speeds up the coarse matching process.

5. **Conclusion.** A coarse matching method is proposed in this paper to adjust the pose of objects through their point clouds. The method speeds up the matching process and it can be extensively used for pose estimation of most objects. At last, the experiments verify that the coarse matching results are acceptable and the time consumption is less than other methods. In the future work, the efficiency of this method will be further improved and it will be used in making wing skin and wing butt joint.

Acknowledgement. This work is supported by the Science and Technology Support of Hebei Province under Grant No. 15211019D.

REFERENCES

[1] J. M. D. Barros, F. Garcia and D. Sidibe, Real-time human pose estimation from body-scanned point clouds, *The 10th International Conference on Computer Vision Theory and Applications*, Berlin, Germany, pp.553-560, 2015.

[2] R. P. de Figueiredon, P. Moreno and A. Bernardino, Efficient pose estimation of rotationally symmetric objects, *Neurocomputing*, vol.150, pp.126-135, 2015.

[3] A. Berner, J. Li, D. Holz et al., Combining contour and shape primitives for object detection and pose estimation of prefabricated parts, *The 20th IEEE International Conference on Image Processing*, Melbourne, Australia, pp.3326-3330, 2013.

[4] D. D. Nguyen, J. P. Ko and J. W. Jeon, Determination of 3D object pose in point cloud with CAD model, *The 21st Korea-Japan Joint Workshop on Frontiers of Computer Vision*, Mokpo, South Korea, pp.1-6, 2015.

[5] Y. Guo, M. Bennamoun, F. Sohel et al., An integrated framework for 3-D modeling, object detection, and pose estimation from point-clouds, *IEEE Trans. Instrumentation and Measurement*, vol.64, no.3, pp.683-693, 2014.

[6] Y. Diez, J. Mart and J. Salvi, Hierarchical normal space sampling to speed up point cloud coarse matching, *Pattern Recognition Letters*, vol.33, pp.2127-2133, 2012.

[7] E. C. H. Cheung, C. Chao and W. S. Newman, Initial pose estimation using cross-section contours, *IEEE International Conference on Robotics and Biomimetics*, Bali, Indonesia, pp.878-883, 2014.

[8] D. Fehr, W. J. Beksi, D. Zermas et al., Covariance based point cloud descriptors for object detection and recognition, *Computer Vision and Image Understanding*, pp.1-14, 2015.

[9] C. Torre-Ferrero, S. Robla, E. G. Sarabia et al., A coarse-to-fine algorithm for 3D registration based on wavelet decomposition, *WSEAS Trans. Systems*, vol.7, no.7, pp.655-664, 2008.