Application-Oriented Fusion Estimation Method for Road Slope Using RGB-D Robots: Postprint

Authors: Pre-dawn flight, Shuwen Dang, Chen Li

Date: 2022-06-06T00:00:00+00:00

Abstract

To improve the estimation accuracy of road slope for robots along their mobile paths, an application-oriented RGB-D (Red Green Blue-Depth) fusion-based road slope estimation method is proposed. First, a Random Sample Consensus algorithm is introduced for point cloud processing; second, an improved plane fitting method is adopted for normal vector estimation; finally, cosine clustering and cumulative averaging methods are employed to achieve high-precision road slope estimation. Experimental results demonstrate that the proposed algorithm reduces the estimation error by 1.21% and 2.13% compared with the least squares method and sparse subspace method on the dataset, respectively, and reduces the average error by 1.43° compared with the least squares method in real-world environments, which proves the feasibility and accuracy of the proposed method.

Full Text

Application-Oriented RGB-D Robot Road Slope Fusion Estimation Method

Ling Chenfei, Dang Shuwen†, Chen Li
(School of Air Transport, Shanghai University of Engineering Science, Shanghai 201600, China)

Abstract: To improve the estimation accuracy of road slope during robot movement, this paper proposes an application-oriented RGB-D (Red Green Blue-Depth) robot fusion slope estimation method. First, the Random Sample Consensus (RANSAC) algorithm is introduced for point cloud processing. Second, an improved plane fitting method is adopted for normal vector estimation. Finally, cosine clustering and cumulative averaging methods are employed to achieve high-precision road slope estimation. Experimental results show that compared with the least squares method and sparse subspace method on the

dataset, the estimation error is reduced by 1.21% and 2.13% respectively, and in real-world environments, the average error is reduced by 1.43° compared with the least squares method, which verifies the feasibility and accuracy of the proposed method.

Key words: RGB-D; robot; normal vector estimation; slope estimation

0 Introduction

Accurate and stable road slope estimation is a critical issue for mobile robots operating in complex terrain environments, particularly on sloped surfaces. Currently, environmental information detection based on laser [1,2] and vision [3,4] sensors serves as the primary means for mobile robots to accomplish map construction [5] and path planning [6]. Laser sensors are often applied in intelligent transportation, epidemic prevention, and even extraterrestrial exploration due to their high measurement accuracy. However, improving the precision and stability of road slope estimation remains an urgent challenge.

Traditional approaches in literature [7~10] employ the least squares method to directly fit the mean plane of regional terrain blocks, using the fitted plane' s inclination angle as the regional slope. While computationally simple and elegantly designed, this method is susceptible to outliers and yields large estimation errors. Li et al. [11] optimized the traditional least squares method by using sparse coefficients to partition subspaces, fitting planes within each subspace, and performing random searches to obtain the optimal plane for slope estimation. Although this approach effectively reduces outlier influence, it requires pre-setting search thresholds based on plane conditions, and the random search for optimal planes may lead to local optima, introducing randomness into slope estimation results. Bai et al. [12] utilized an adaptive threshold fitting algorithm in their midline fitting process, automatically calculating thresholds through point cloud quantity and coordinate information, enriching slope calculation methods for ski resort trails.

The aforementioned slope estimation methods adopt LiDAR solutions. While LiDAR measurement technology develops rapidly with wide measurement ranges [13,14], the 3D environmental information obtained is relatively sparse compared to visual cameras, losing many 3D features. Moreover, most vision cameras are compact and lightweight, exerting minimal impact on robot payload [15]. Currently, vision cameras are primarily applied to workpiece defect detection [16], object sorting [17], and visual positioning [18], but few solutions exist for mobile robot slope estimation. From a practical standpoint, vision-based mobile robots are more suitable for sloped terrain environments than laser sensors. However, existing vision-based mobile robot road slope estimation methods generally remain in simulation stages and suffer from large estimation errors when facing real-world applications.

This paper proposes an application-oriented visual RGB-D robot road slope fusion estimation method. The method systematically constructs a fusion al-

gorithm pipeline comprising point cloud filtering, plane segmentation, normal vector extraction, cosine clustering, and cumulative averaging filtering, and conducts road slope estimation experiments in both dataset and real-world environments. Comparative analysis with traditional methods demonstrates that the proposed method offers significant advantages in reducing estimation error and improving slope estimation accuracy.

1 Algorithm Flow

The algorithm flow is illustrated in Figure 1 [Figure 1: see original paper]. The process involves: (a) For acquired point cloud information, statistical filtering is first applied to remove outliers from the point cloud surface captured by the visual sensor. RANSAC is then used for plane segmentation to extract both inclined and horizontal planes. To improve processing efficiency, a data threshold of 10,000 points is set for plane point clouds. When data points exceed this threshold, a pass-through filter reduces the point count, and a random least squares method fits the point cloud plane to extract normal vectors. (b) For extracted normal vectors, cosine clustering first eliminates outlier vectors, followed by cumulative fitting to obtain more precise results.

2.1 Filtering Processing

When acquiring point clouds using visual cameras, depth data is susceptible to external light interference, resulting in abnormal noise in captured images. Therefore, statistical filtering is introduced to perform statistical analysis on the neighborhood of each point in the point cloud, estimating its average distance to all neighboring points. Assuming a point cloud is represented as $\{p_i | i = 1, 2, ..., n\}$, where n denotes the total number of points and p_i represents the 3D coordinates of point i, the distance d_i from point p_i to its neighbors is calculated as:

$$d_i = \frac{1}{m} \sum_{j=1}^{m} \|p_i - p_j\|$$

where m is the number of neighboring points for p_i, and $\|p_i - p_j\|$ represents the Euclidean distance between p_i and its neighbor p_j, calculated as:

$$\|p_i - p_j\| = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2}$$

The mean distance avg_d and standard deviation std_d are:

$$\text{avg}_d = \frac{1}{n} \sum_{i=1}^{n} d_i$$

$$\text{std}_d = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(d_i - \text{avg}_d)^2}$$

A point is identified as an outlier and removed from the set if its distance exceeds $\text{avg}_d + \alpha \times \text{std}_d$ [19,20]. Through multiple experimental comparisons, statistical filtering parameters are set to $m = 20$ and $\alpha = 2$ for more effective outlier removal [21].

2.2 Plane Segmentation

Assuming that among the two planes forming the road angle, the wall surface is the inclined plane and the ground is the horizontal plane, the Random Sample Consensus (RANSAC) algorithm is introduced to extract point clouds for both planes. The extraction algorithm proceeds as follows: three non-collinear points are randomly selected from the point cloud to determine a plane equation $Ax + By + Cz + E = 0$, where A, B, C, E are the plane equation parameters. The distance d_i from other points to this plane is:

$$d_i = \frac{|Ax_i + By_i + Cz_i + E|}{\sqrt{A^2 + B^2 + C^2}}$$

2.3 Normal Vector Extraction

For the two segmented planes, normal vectors must be extracted before angle estimation. The point cloud normal vector extraction process consists of normal vector estimation and orientation.

2.3.1 Normal Vector Estimation

Based on randomly selected neighboring points $\{p_{i_k} | k = 1, 2, ..., K\}$ within a neighborhood and setting the fitting neighborhood value K, a fitted plane can be obtained. The plane' s normal line serves as the point' s normal vector. With neighboring points' 3D coordinates $(x_{i_k}, y_{i_k}, z_{i_k})$, the fitted plane' s equation parameters (F, G, H, I) satisfy:

$$Fx_{i_k} + Gy_{i_k} + Hz_{i_k} + I = 0$$

subject to $F^2 + G^2 + H^2 = 1$. Using random least squares fitting, we solve for the minimum:

$$\min_{F,G,H,I} \sum_{k=1}^{K} (Fx_{i_k} + Gy_{i_k} + Hz_{i_k} + I)^2$$

The covariance matrix M is:

$$M = \frac{1}{K}\sum_{k=1}^{K}(p_{i_k} - \bar{p})(p_{i_k} - \bar{p})^T$$

where $\bar{p}$ is the neighborhood centroid. With eigenvalues $\lambda^{(j)}$ and eigenvectors $v^{(j)}$ $(j = 1, 2, 3)$, if the eigenvalues satisfy $\lambda^{(1)} \leq \lambda^{(2)} \leq \lambda^{(3)}$, then $v^{(1)}$ is the normal vector n_i of point p_i.

2.3.2 Normal Vector Orientation

Considering the uncertainty in the estimated normal vector n_i direction, the line-of-sight direction is adopted for uniform orientation to ensure consistency. Given the viewpoint V, all normal vectors are oriented to face the viewpoint direction, satisfying:

$$n_i \cdot (V - p_i) > 0$$

3 Road Slope Estimation

The sparse subspace plane optimization method can effectively reduce outlier influence on slope estimation, but its optimal plane is obtained through random search, which is highly likely to yield local optimal solutions and increase plane iteration counts. To reduce the impact of random value selection, this section introduces cosine distance into cosine k-means clustering to address the random initial point selection problem. Table 1 lists the definitions of important formula variables used in this section.

As shown in Figure 2 [Figure 2: see original paper], when the distance threshold t is set, points beyond this threshold are considered outliers and deleted; otherwise, they are inliers and retained, with the total number of inliers counted [22]. The plane point set with the maximum number of inliers is identified, and the corresponding inlier indices are returned as a plane region. The obtained inliers are removed from the scene point cloud, and the process repeats until the remaining points in the scene are fewer than a preset threshold or the last obtained inlier count is below the threshold [23,24].

Table 1 Formula variable definition

Variable	Definition
$\mathrm{dist}(a, b)$	Euclidean distance between any two normal vectors a and b in a single plane
$\cos(a, b)$	Cosine similarity between a and b
$\tilde{a}, \tilde{b}$	Normalized vectors of a and b
$\mathrm{cdist}(b, a)$	Cosine distance between a and b

3.1 Cosine Clustering

For computational convenience, data points in each segmented plane point cloud are evaluated. When data points exceed 10,000, a pass-through filter removes some points.

Among normal vectors extracted from a single plane, any two vectors a and b are represented in Euclidean space as $a = (a_1, a_2, ..., a_n)$ and $b = (b_1, b_2, ..., b_n)$. The cosine angle between a and b is defined as cosine similarity. Since vector orientation has been unified in the normal vector orientation section, values now distribute within $[0, 1]$.

Cosine similarity uses the cosine of the vector angle to characterize similarity, emphasizing relative differences between dimensions [25]. Continuing the derivation: first normalize a and b:

$$\tilde{a} = \frac{a}{\|a\|}, \quad \tilde{b} = \frac{b}{\|b\|}$$

Substituting normalized vectors into the cosine similarity formula yields:

$$\cos(a, b) = \frac{\sum_{k=1}^{n} a_{kb}k}{\|a\|\|b\|} = \sum_{k=1}^{n} \tilde{a}_k \tilde{b}_k = \cos(\tilde{a}, \tilde{b})$$

Thus, cosine similarity remains unchanged before and after normalization. With vector length becoming 1, the formula simplifies to $\cos(a, b) = \sum_{k=1}^{n} \tilde{a}_k \tilde{b}_k$, significantly improving computational efficiency.

Initial point selection is a key research issue in Euclidean distance-based k-means algorithms [26]. The core involves predicting categories during the first iteration to select points closer to ideal values, reducing clustering iterations and computational overhead. Initial point selection follows the principle: samples with large distances have low probability of belonging to the same class, while those with small distances have high probability [27]. Therefore, the initial iteration should select K samples with relatively large mutual distances for better results.

The challenge becomes how to perform the first-round point selection for spatial vectors in cosine clustering. Using small cosine similarity as the initial selection criterion makes it extremely difficult to find two points with maximized distance [28], because in Euclidean distance, the third point selection uses the maximum sum or product of distances to the first two points, yet sum or product is meaningless in cosine space and cannot represent large separation. Therefore, cosine distance $\text{cdist}(b, a) = 1 - \cos(a, b)$ is introduced. Since $\cos(a, b)$ distributes in $[0, 1]$, $\text{cdist}(b, a)$ distributes in $[0, 1]$, and cosine distance positively correlates with the spatial angle between vectors—smaller angles yield smaller cosine distance values, making vectors more likely to be classified together. Thus, cosine distance serves as a distance metric.

For any two normalized vectors c and d in Euclidean space, the relationship between Euclidean distance $\mathrm{dist}(c,d)$, cosine similarity $\cos(c,d)$, and cosine distance $\mathrm{cdist}(d,c)$ can be derived. Substituting normalized vectors and combining formulas yields:

$$\mathrm{dist}(c,d) = \sqrt{2 - 2\cos(c,d)}$$

Combining with the cosine distance definition gives:

$$\mathrm{dist}(c,d) = \sqrt{2 \times \mathrm{cdist}(d,c)}$$

This reveals a direct functional relationship between cosine similarity and Euclidean distance, indicating a close connection. The relationship among the three metrics is:

$$\mathrm{dist}(c,d) = \sqrt{2 - 2\cos(c,d)} = \sqrt{2 \times \mathrm{cdist}(d,c)}$$

The introduction of cosine distance compensates for the lack of distance metrics in cosine k-means clustering. Simultaneously, it serves as a bridge to migrate Euclidean distance-based k-means algorithms to the proposed cosine clustering method. Equation (25) shows that cosine distance aligns with Euclidean distance in characterizing distance and maintains positive correlation, thus expressing equivalent distance meaning. Moreover, cosine distance is entirely determined by cosine similarity, ensuring consistent clustering effects.

Therefore, the principle of selecting initial points with large Euclidean distances is applied to cosine k-means clustering by sequentially seeking K points with relatively large cosine distances. This approach helps cosine k-means clustering conditionally select initial cluster centers, mitigating adverse effects from random initial values. The clustering flowchart is designed as shown in Figure 3 [Figure 3: see original paper].

3.2 Cumulative Fitting

After cosine clustering, the angle θ between inclined and horizontal plane normal vectors can be defined as [29]:

$$\theta = \arctan\left(\frac{n_i \cdot n_j}{\|n_i\|\|n_j\|}\right)$$

where n_i represents the normal vector of any point p_i on the slope surface, and n_j represents the normal vector of any point p_j on the ground.

Since slope estimation results using random value selection often exhibit large uncertainty, a cumulative fitting method is adopted. The angle value r_θ is calculated as:

$$r_\theta = \frac{1}{n}\sum_{i=1}^{n}\theta_i$$

4.1 Experimental Platform

The mobile robot structure used in experiments is shown in Figure 4 [Figure 4: see original paper]. The chassis employs a four-wheel configuration with two driving wheels and two driven wheels—the driving wheels control direction adjustment while the driven wheels provide propulsion. The robot is equipped with a second-generation Kinect-based RGB-D vision camera for environmental perception and a Core i7 host for data processing and motion control.

4.2 Comparison with Common Methods

To verify the superiority of the proposed fusion algorithm, comparisons were conducted against the traditional least squares slope estimation method and the sparse subspace slope estimation method. The Washington University public dataset Rgb-scenes-v2 was used, selecting wall-ground pairs at 90° angles (simulating 90° road slopes) for simulation experiments. The dataset point cloud is shown in Figure 5 [Figure 5: see original paper].

The experimental procedure for the proposed fusion algorithm is as follows: First, plane segmentation is performed on the filtered dataset. Figure 6 [Figure 6: see original paper] shows the segmented inclined plane (blue) and horizontal plane (red). The fitting neighborhood value K for vector extraction is discussed: when K is sufficiently large, better normal vectors with less noise are obtained, but at the cost of exponentially increased computation time and pressure. To ensure real-time performance and accuracy, experiments were conducted on dataset point clouds with different K values: 25, 35, 50, 60, 100, 1000, and 2000. Figure 7 [Figure 7: see original paper] shows the ramp angle results for different K values. As K increases, the angle stabilizes and measurements approach true values.

Experimental results indicate that when K is less than 100, the angle error exceeds 1° with numerous outliers. When K exceeds 100, the error stabilizes around 0.5°. Considering both factors, $K = 100$ is selected as the threshold. Normal vectors for inclined and horizontal planes are extracted based on this fitting neighborhood value, as shown in Figures 8 [Figure 8: see original paper] and 9 [Figure 9: see original paper].

After cosine clustering on single-plane vectors, cumulative fitting is applied. The fitting iteration count is set to 100 after comparison with random value selection. Partial results are listed in Table 2 , with full comparisons shown in Figure 10 [Figure 10: see original paper]. The cumulative fitting method stabilizes after 100 iterations, while random value selection still shows large fluctuations,

proving that cumulative fitting significantly ensures the estimation accuracy of the fusion algorithm.

Table 2 Comparison between cumulative fitting and random value

Iteration	Random Value	Cumulative Fitting
1	89.2°	89.2°
10	88.5°	89.8°
50	91.3°	90.4°
100	87.9°	90.5°

The final algorithm result for the 90° wall slope in dataset Rgb-scenes-v2 is 90.55°, with a relative error of 0.55°. Comparison with other algorithms is shown in Table 3 . Figures 11 [Figure 11: see original paper] and 12 [Figure 12: see original paper] show normal vectors extracted using the traditional least squares method. Sparse subspace clustering results reference data from literature [11]. The comparison demonstrates that the proposed fusion algorithm achieves higher estimation accuracy than least squares and sparse subspace methods, reducing relative error by 1.21% and 2.13% respectively.

Table 3 Analysis of dataset measurement results

Method	Measurement Result	Relative Error (°)	Relative Error (%)
Least Squares	91.76°	1.76	1.96
Sparse Subspace	92.68°	2.68	2.98
Proposed Method	90.55°	0.55	0.61

4.3 Actual Environment Slope Recognition

To further verify the effectiveness and feasibility of the fusion algorithm in outdoor environments, semi-physical simulation experiments were conducted on an RGB-D mobile robot equipped with ROS in an outdoor ramp environment, as shown in Figure 14 [Figure 14: see original paper].

The unprocessed ramp point cloud (Figure 15 [Figure 15: see original paper]) contains visible outliers at the edges, which are detrimental to normal vector extraction and require filtering. Using statistical filtering with parameters $m = 20$ and $\alpha = 2$, the filtered point cloud is shown in Figure 16 [Figure 16: see original paper], with removed outliers displayed in Figure 17 [Figure 17: see original paper], demonstrating effective outlier removal.

The filtered point cloud contains excessive data points, requiring downsampling during segmentation. Segmented results are shown in Figure 18 [Figure 18: see

original paper], where the horizontal plane appears red and the ramp blue. The green box marks a step protrusion on the ramp that interferes with normal vector generation. After plane segmentation, both planes retain some protrusions. With 12,405 points exceeding the threshold, pass-through filtering removes the protrusion points until the count falls below 10,000.

After plane segmentation, normal vectors are extracted with threshold $K = 100$. Results are shown in Figures 19 [Figure 19: see original paper] and 20 [Figure 20: see original paper], extracting 4,329 inclined plane vectors and 4,830 horizontal plane vectors. Outliers are eliminated through cosine clustering, followed by cumulative fitting with 100 iterations, yielding a final ramp angle of 12.5°.

Additionally, as shown in Figure 13 [Figure 13: see original paper], ramp planes at 16°, 47°, and 60° were constructed indoors to compare the performance of least squares and the proposed method across different slopes. Measurement results in Table 4 show that the proposed method maintains high accuracy in real-world scenarios, with an average error of 0.86° across the three slopes—1.43° lower than the least squares method' s average error.

Table 4 Analysis of actual environmental measurement results

Method	Ramp Angle	Measurement	Error (°)	Average Error (°)
Least Squares	16°	17.8°	1.8	2.29
	47°	49.2°	2.2	
	60°	62.8°	2.8	
Proposed	16°	16.3°	0.3	0.86
	47°	47.5°	0.5	
	60°	61.1°	1.1	

5 Conclusion

This paper proposes an application-oriented visual RGB-D robot road slope estimation fusion algorithm. Addressing the common limitation of slope estimation algorithms to simulation environments and their low accuracy in real-world conditions, the method designs and implements a fusion pipeline combining statistical filtering, plane segmentation, normal vector extraction, cosine clustering, and cumulative averaging. Experimental results demonstrate significant advantages in reducing estimation error and improving accuracy. On the dataset, the proposed fusion algorithm reduces slope estimation error by 1.21% and 2.13% compared to traditional least squares and sparse subspace methods. In real-world environments, the method reduces average error by 1.43° compared to least squares. This provides accurate environmental information assurance for subsequent mobile robot path planning and obstacle avoidance in real-world scenarios.

References

[1] Hess W, Kohler D, Rapp H, et al. Real-time loop closure in 2D LIDAR SLAM [C]// Proc of 2016 IEEE International Conference on Robotics and Automation. Stockholm: IEEE, 2016: 1271-1278.

[2] Pang Fan, Wei Shuangfeng, Shi Xianjie, et al. Odometry and mapping method for coupling LiDAR and IMU [J]. Application Research of Computers, 2021, 38 (7): 2188-2193.

[3] Mur A R, Tardos J D. ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras [J]. IEEE Trans on Robotics, 2017, 33 (5): 1255-1262.

[4] Campos C, Elvira R, Juan J, et al. ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual-Inertial and Multi-Map SLAM [J]. arXiv preprint arXiv: 2007.11898.

[5] Pan Linhao, Tian Fuqing, Ying Wenjian, et al. Survey on direct-method visual simultaneous localization and mapping [J]. Application Research of Computers, 2019, 36 (4): 961-966.

[6] Wang Ziqiang, Hu Xiaoguang, Li Xiaoxiao, et al. Overview of Global Path Planning Algorithms for Mobile Robots [J]. Computer Science, 2021, 48 (10): 19-29.

[7] Liang Dong, Wang Pengji, Liu Liangdong. A LIDAR-Based Autonomous Landing Site Selection Method for Pinpoint Lunar Soft Landing [J]. Aerospace Control and Application, 2009, 35 (6): 24-29.

[8] Zhang Zexu, Wang Weidong, Cui Pingyuan, et al. An algorithm of terrain hazard assessment for planetary soft landing [J]. Journal of Harbin Institute of Technology. 2011, 43 (5): 25-29.

[9] Wang Yalin, Liu Peng, Wu Huiyang, et al. Terrain Analysis and Simulation Verification on Rubble-Pile-Constructed Asteroid Surfaces [J]. Journal of Deep Space Exploration, 2019, 6 (5): 481-487.

[10] Li Boyuan, Zhang Jiawei, Du Haiping, et al. Two-layer structure based adaptive estimation for vehicle mass and road slope under longitudinal motion [J]. Measurement, 2017, 95: 439-455.

[11] Li Haibo, Cao Yunfeng, Ding Meng, et al. A Method of Slope Estimation Based on Clustering of Three-dimensional Point Cloud [J]. Acta Metrologica Sinica, 2018, 39 (3): 304-309.

[12] Bai Shasha, Zhang Haiyang, Xu Shidong, et al. Slope extraction algorithm of ski tracks based on airborne LIDAR point cloud [J]. Journal of Applied Optics, 2021, 42 (3): 481-487.

[13] Debeunne C, Vivet D. A Review of Visual-LiDAR Fusion based Simultaneous Localization and Mapping [J]. Sensors, 2020, 20 (7): 2068-2091.

[14] Zhou Zhiguo, Cao Jiangwei, Di Shunfan. Overview of 3D Lidar SLAM algorithms [J]. Chinese Journal of Scientific Instrument, 2021, 42 (9): 13-27.

[15] Li Shangcong. Off-Road Environmental Traversability Analyzing for Quadruped Robots Based on Depth Sensing Information [D]. Jinan: Shandong University, 2020.

[16] Lyu Ning, Xiao Jian, Gao Jian, et al. Image segmentation algorithm on contour defects for stamping part based on improved MRF [J]. Forging & Stamping Technology, 2022, 47 (4): 101-109.

[17] Han Song, Liu Xiaoping, Han Xing, et al. Visual Sorting of Express Parcels Based on Multi-Task Deep Learning [J]. Sensors, 2020, 20 (23): 6943-6957.

[18] Young J K, Seon I K, Yeol D Y, et al. Visual Positioning System Based on 6D Object Pose Estimation Using Mobile Web [J]. Electronics, 2022, 11 (6): 865-879.

[19] Peng Yuhui, Zheng Weihong, Zhang Jianfeng. Deep learning-based on-road obstacle detection method [J]. Journal of Computer Applications, 2020, 40 (8): 2428-2433.

[20] Qi C R, Su H, Mo K, et al. PointNet: Deep learning on point Sets for 3D Classification and Segmentation [C]// Proc of 2017 IEEE Conference on Computer Vision and Pattern Recognition. Piscataway: IEEE, 2017: 77-85.

[21] Wang Xinjie, Zhang Ying, Zhang Dongbo, et al. Point cloud noise processing in path planning of autonomous mobile robot [J]. CAAI Trans on Intelligent Systems, 2021, 16 (4): 699-706.

[22] Wu Qinghua, Zhu Sisi, Zhou Yang, et al. Regular sticks pose position and orientation recognition based on RANSAC [J]. Transducer and Microsystem Technologies, 2019, 38 (9): 137-140.

[23] Lu Rongrong, Zhu Feng, Wu Qingxiao, et al. A Fast Segmenting Method for Scenes with Stacked Plate-Shaped Objects [J]. Acta Optica Sinica, 2019, 39 (4): 151-160.

[24] Schnabel R, Wahl R, Klein R. Efficient RANSAC for Point-Cloud Shape Detection [J]. Computer graphics forum, 2010: 214-226.

[25] Jain A K. Data clustering: 50 years beyond K-means [J]. Pattern Recognition Letters, 2010, 31 (8): 651-666.

[26] Zhang Yonglai, Zhou Yaojian. Review of clustering algorithms [J]. Journal of Computer Applications, 2019, 39 (7): 1869-1882.

[27] Zhai Donghai, Yu Jiang, Gao Fei, et al. K-means text clustering algorithm based on initial cluster centers selection according to maximum distance [J]. Application Research of Computers, 2014, 31 (3): 713-715.

[28] Wang Binyu, Liu Wenfen, Hu Xuexian, et al. Research on text clustering for selecting initial cluster center based on Cosine distance [J]. Computer

Engineering and Applications, 2018, 54 (10): 11-18.

[29] Wang Fei, Liu Rufei, Ren Hongwei, et al. Multi-Stage Vehicle-Mounted Laser Point Cloud Registration Using Road Target Features [J]. Journal of Geomatics Science and Technology, 2020, 37(5): 496-502.

Note: Figure translations are in progress. See original paper for figures.