Hindawi
Wireless Communications and Mobile Computing
Volume 2021, Article ID 2706462, 9 pages
https://doi.org/10.1155/2021/2706462

Research Article

Intelligent Point Cloud Edge Detection Method Based on Projection Transformation

Juan Zhu,[1] **Xiaofeng Yue**,[1] **Jipeng Huang**,[2] **and Zongwei Huang**[1]

[1]*School of Mechanical and Electrical Engineering, Changchun University of Technology, Changchun, China*
[2]*School of Physics, Northeast Normal University, Changchun, China*

Correspondence should be addressed to Xiaofeng Yue; yuexiaofeng@ccut.edu.cn and Jipeng Huang; huangjp848@nenu.edu.cn

Received 20 October 2021; Accepted 22 November 2021; Published 21 December 2021

Academic Editor: Vinoth Babu Kumaravelu

An edge detection method based on projection transformation is proposed. First, the vertical projection transformation is carried out on the target point cloud. Data X and data Y are normalized to the width and height of the image, respectively. Data Z is normalized to the range of 0-255, and the depth represents the gray level of the image. Then, the Canny algorithm is used to detect the edge of the projection transformed image, and the detected edge data is back projected to extract the edge point cloud in the point cloud. Evaluate the performance by calculating the normal vector of the edge point cloud. Compared with the normal vector of the whole data point cloud of the target, the normal vector of the edge point cloud can well express the characteristics of the target, and the calculation time is reduced to 10% of the original.

1. Introduction

As a key automation technology, machine vision is very important to the modernization of the economy. Machine vision has been widely studied by scholars. Machine vision uses machines to measure the size of target or detect the surface of target instead of eyes. Machine vision mainly uses computers to simulate the function of human visual and reproduce certain intelligent behaviors related to human vision. Information is extracted from the image of an objective object, processed and understood, and finally used for practical detection and control. Machine vision started from statistical pattern recognition in the 1950s. The main work is focused on two-dimensional image analysis, recognition, and understanding. In recent years, various noncontact research results emerged [1–4]. Machine vision in the industrial field can be divided into four aspects. Surface detection is always used in product quality inspection and product classification. A camera and a robot are combined to package products. Feature detection is always used in robot positioning. Civil Machine Vision Technology is widely used in intelligent transportation, safety protection, character recognition, identity verification, medical equipment, etc. In the field of scientific research, machine vision can be used for material analysis, biological analysis, chemical analysis, and life science. In the military field, it can be used in aerospace, aviation, weapons, and mapping. Its technology mainly includes image processing, mechanical engineering, control, and optical imaging.

With the rapid development of 3D acquisition technology, 3D sensors are becoming more available and affordable, including various types of 3D scanners, LiDAR, and RGB-D cameras (such as Kinect, RealSense, and Apple depth cameras). The 3D data from these sensors can provide rich geometry, shape, and scale information. Complementing 2D images, 3D data provides an opportunity to better understand the environment around the machine. 3D data can often be represented in different formats, including depth images, point clouds, grids, and volumetric grids. As a common format, the point cloud representation preserves the original geometry in 3D space without any discretization. It is the preferred notation for understanding related applications in many scenarios. 3D point cloud detection is widely researched by scholars. Ali et al. [5] build on the success of the one-shot regression meta-architecture in the 2D perspective image space and extend it to generate oriented

3D object bounding boxes from LiDAR point cloud. Zhou et al. [6] remove the need of manual feature engineering for 3D point clouds and propose VoxelNet, a generic 3D detection network that unifies feature extraction and bounding box prediction into a single-stage, end-to-end trainable deep network. Meyer et al. [7] present LaserNet, a computationally efficient method for 3D object detection from LiDAR data for autonomous driving. Beltran et al. [8] present a LiDAR-based 3D object detection pipeline entailing three stages. Minemura et al. [9] employ dilated convolutions to gradually increase the perceptive field as depth increases; this helps to reduce the computation time by about 30%. Asvadi et al. [10] address the problem of vehicle detection using Deep Convolutional Neural Network (ConvNet) and 3D-LIDAR data with application in advanced driver assistance systems and autonomous driving. Propose a vehicle detection system based on the Hypothesis Generation (HG) and Verification (HV) paradigms. Simon et al. [11] propose a specific Euler-Region-Proposal Network (E-RPN) to estimate the pose of the object by adding an imaginary and real fraction to the regression network.

Edge detection is a key technology to detect the target [12]. Most of the information of the image exists in the edge of the image, which is mainly represented by the discontinuity of the local features of the image. Edge detection is first proposed for a two-dimensional digital image; the purpose is to identify and detect the position where the image characteristics change [13]. Point cloud edge refers to some edge measurement points that can express the target features. Point cloud edge can not only express the geometric characteristics of the object but also play an important role in the quality and accuracy of object recognition and surface model reconstruction [14, 15]. As an important research field of image analysis and computer vision, edge detection has attracted the attention of many scholars. A variety of mature edge detection algorithms have been developed.

Different point cloud data models have different edge feature extraction methods, which can be roughly divided into grid-based and scattered point cloud-based feature extraction methods [16, 17]. In feature extraction based on mesh, firstly, the point cloud is gridded, and then, the edge features of the point cloud are obtained by traversing the triangulated point cloud and threshold constraints. Among them, the most famous algorithm of Delaunay is simple and intuitive. But in the process of triangulation, we need to evaluate the Euclidean distance between point clouds. If the Euclidean distance is not suitable, holes will be generated. In addition, if the method is applied to three-dimensional point clouds, it needs to use the normal direction of each point cloud to determine the projection direction, so the algorithm is more suitable for uniform and smooth point clouds. The feature extraction based on scattered point cloud mainly extracts some regular points, lines, surfaces, and other features from this type of point cloud, so it pays more attention to local features. Song et al. [18] take the vector of each point in the point cloud and the vector of K adjacent points as the root mean square as the standard of edge feature extraction, although this method well reflects the relationship between the normal direction of each point in the point cloud and its adjacent points, the nonedge points adjacent to the edge will be detected in the result of edge extraction. Han et al. [19] keep the edge feature by using the feature that the normal direction of the boundary point is different from the nonboundary normal direction, but the density of the edge is the same as that of the nonedge part. Chen et al. [20] proposed a feature extraction algorithm with multiparameter constraints. Feature points are determined by normal, curvature, and Euclidean distance. In [21, 22], principal component analysis (PCA) and normal method were used to extract edge feature points.

Referring to the edge detection algorithm of two-dimensional image, this paper proposes a Canny operator based on projection transformation for edge detection of point cloud data and obtains the normal vector of the edge point cloud after edge detection. The point cloud can better reflect the characteristics of the target, and the speed of solving the normal vector is greatly improved. Compared with the normal vector of the whole data point cloud of the target, the normal vector of the edge point cloud can well express the characteristics of the target, and the calculation time is reduced to 10% of the original.

2. Methodology Vertical Projection of Point Data

Projection transformation is the process of transforming the coordinates of one map projection point into the coordinates of another map projection point. 3D point cloud is a massive set of points that express the spatial distribution and surface characteristics of targets in the same spatial reference system. It is a collection of points after obtaining the spatial coordinates of each sampling point on the object surface. Compared with a 2D image, 3D point cloud usually only has X, Y, Z coordinate information. Its spatial information is redundant to a 2D image. The corresponding geometric structure is more complex, and the neighbour structure of point cloud data is more complex.

In this paper, the Canny edge detection algorithm based on projection transformation is proposed to detect the edge of point cloud data. The point cloud data is projected along the vertical direction to the XY two-dimensional plane, and the projected point cloud data is normalized. The X direction represents the width, the Y direction represents the height, and the Z value represents the gray value of the pixel in the image. The edge of the transformed data is detected, and then, the final edge point cloud is obtained by inverse transformation.

Supposing the number of point clouds is m, all point clouds are represented as

$$P = \begin{bmatrix} x_1, x_2, \cdots, x_m \\ y_1, y_2, \cdots, y_m \\ z_1, z_2, \cdots, z_m \end{bmatrix}. \tag{1}$$

Here, $P_i = [x_i, y_i, z_i]^T$ is the three-dimensional coordinates of point i.

The point cloud data is vertically projected in the Z direction, and the Z value is converted into the depth value of the current point. The projected point set is expressed as

$$f(x_i, y_i) = z_i. \tag{2}$$

For the projected point set, the maximum value and minimum value of X and Y directions are counted, which are marked as follows: $x_{\min}$, $x_{\max}$, $y_{\min}$, and $y_{\max}$. The X, Y axis data is quantized into a form corresponding to the image width W and height H. Then, the abscissa and ordinate of x_i, y_i are as

$$i = W * \frac{x_i - x_{\min}}{x_{\max} - x_{\min}} + 1, \tag{3}$$

$$j = H * \frac{y_i - y_{\min}}{y_{\max} - y_{\min}} + 1. \tag{4}$$

A linear transformation is performed on the matrix z_i. In order to realize the corresponding relationship between point cloud and image, statistically, the minimum and maximum values of Z are marked as follows: $z_{\min}$ and $z_{\max}$. The Z value is linearly transformed to the range of 100~255. For a coordinate where there is no point cloud, it is represented by 0. The Z value is like the gray value of an image. The projection transformation is shown in

$$\text{gray}(z_i) = 100 + (z_i - z_{\min}) * \frac{155}{z_{\max} - z_{\min}}. \tag{5}$$

The point set after quantization is expressed as

$$f(i, j) = \text{gray}(z_i). \tag{6}$$

As an example, the industrial part target is vertically projected, and the projection result is shown in Figure 1.

Figures 1(a) and 1(d) are the original target of tee and elbow. Figures 1(b) and 1(e) are the point cloud of tee and elbow. Figures 1(c) and 1(f) are the vertical projection of tee and elbow. Because the angle of view is the Z direction when shooting the target, the data shape of the target point cloud data after vertical projection in the Z direction is consistent with the original image features.

3. Edge Detection with Canny Operator

Canny edge detection was first proposed by John Canny in the paper with a computational approach to edge detection in 1986. Canny edge detection is a technology to extract useful structural information from different visual objects and greatly reduce the amount of data to be processed. It has been widely used in various computer vision systems. Canny found that the requirements of edge detection in different vision systems are similar, so it can achieve a widely used edge detection technology. The Canny algorithm is based on three basic objectives. (1) In the low error rate, all edges should be found with no pseudoresponse. Capture as many edges as possible in the image as accurately as possible. (2) The detected edge should be accurately located in the center of the real edge. (3) In single edge point response, the detector should not point out multiple pixel edges where there is only one single edge point. In order to meet these requirements, Canny uses the variation method. The optimal function in Canny detector is described by the sum of four exponential terms, which can be approximated by the first derivative of Gaussian function. The Canny edge detection algorithm can be divided into the following five steps. (1) Gaussian filter is used to smooth the image and remove the noise. (2) Calculate the gradient intensity and direction of each pixel point in the image. (3) Nonmaximum suppression is applied to eliminate the spurious response caused by edge detection. (4) Double threshold detection is applied to determine real and potential edges. (5) Finally, the edge detection is completed by suppressing the isolated weak edge.

3.1. Gaussian Smoothing. Gaussian smoothing is a 2D convolution operation, which is applied to blurred images to remove details and noise. In order to reduce the influence of noise on the result of edge detection as much as possible, the noise must be filtered to prevent false detection caused by noise. In order to smoothen the image, Gaussian filter is used to convolute the image. In this step, the image is smoothed to reduce the obvious noise effect on the edge detector. The generation equation of Gaussian filter kernel with size of $(2k+1) * (2k+1)$ is given by

$$H_{ij} = \frac{1}{2\pi\sigma^2} e^{-\left((i-(k+1))^2 + (j-(k+1))^2\right)/2\sigma^2}, \quad 1 \le i, j \le 2k+1. \tag{7}$$

Here, σ is the standard deviation of the distribution $K = i + j$. Assume that the mean of the distribution is 0; that is, its center is on the line $x = 0$. The distribution of two-dimension Gaussian filter kernel is shown in Figure 2.

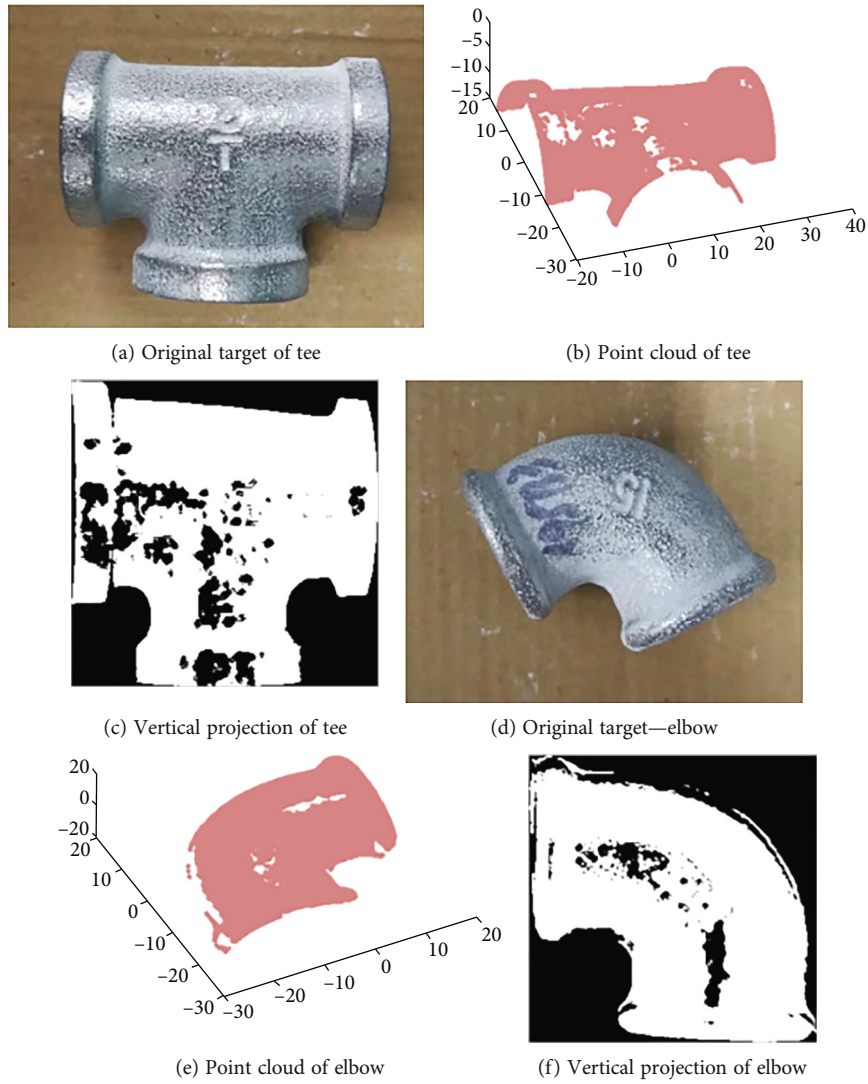
When $\sigma = 1.4$, $k = i$, and the size of Gaussian filter kernel is $3 * 3$, the corresponding Gauss kernel is shown in

$$\begin{bmatrix} 0.0924 & 0.1192 & 0.0924 \\ 0.1192 & 0.1538 & 0.1192 \\ 0.0924 & 0.1192 & 0.0924 \end{bmatrix}. \tag{8}$$

When $\sigma = 1.4$, $k = 2$, and the size of Gaussian filter kernel is $5 * 5$, the corresponding Gauss kernel is shown in

$$\begin{bmatrix} 0.0105 & 0.0227 & 0.0293 & 0.0227 & 0.0105 \\ 0.0227 & 0.0924 & 0.1192 & 0.0924 & 0.0227 \\ 0.0293 & 0.1192 & 0.1538 & 0.1192 & 0.0293 \\ 0.0227 & 0.0924 & 0.1192 & 0.0924 & 0.0227 \\ 0.0105 & 0.0227 & 0.0293 & 0.0227 & 0.0105 \end{bmatrix}. \tag{9}$$

When $\sigma = 2$, $k = 1$, and the size of Gaussian filter kernel



(a) Original target of tee

(b) Point cloud of tee

(c) Vertical projection of tee

(d) Original target—elbow

(e) Point cloud of elbow

(f) Vertical projection of elbow

Figure 1: Target point cloud and vertical projection.

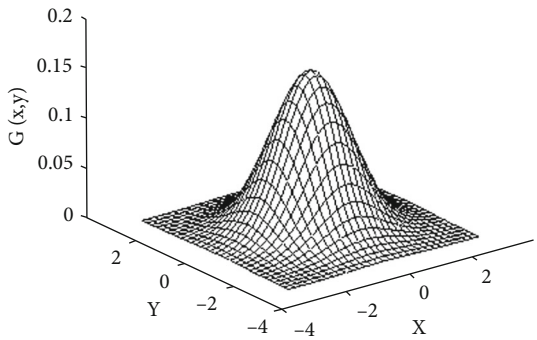


Figure 2: Distribution of 2D Gaussian filter kernel.

is 3 ∗ 3, the corresponding Gauss kernel is shown in

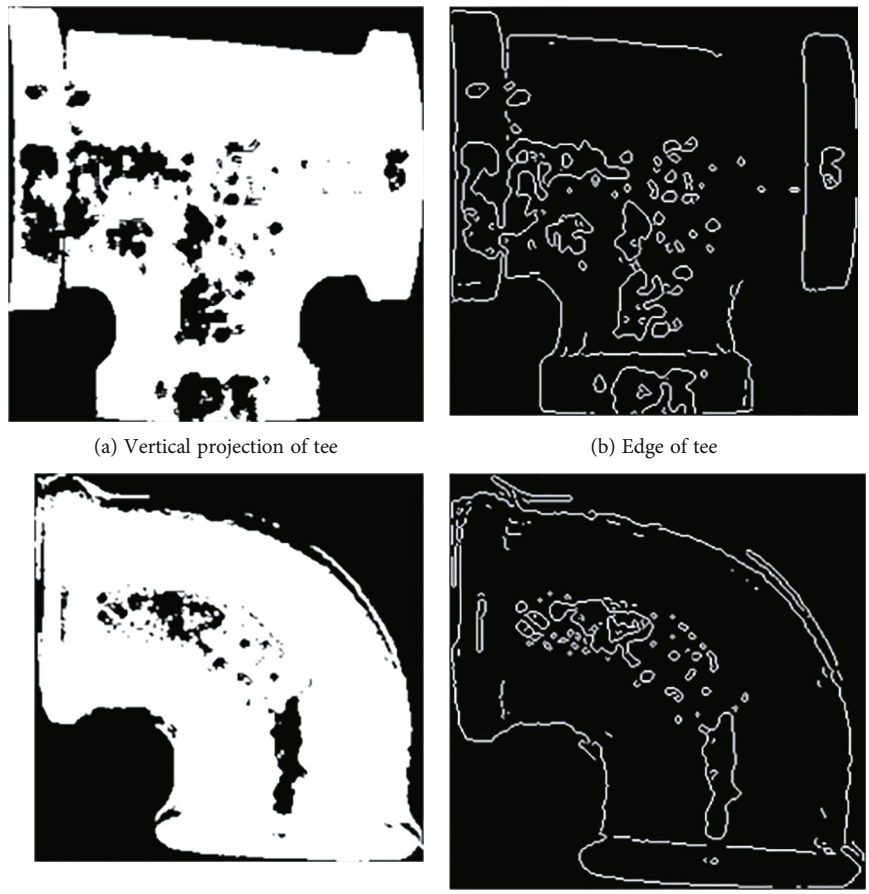
$$\begin{bmatrix} 0.0310 & 0.0351 & 0.0310 \\ 0.0351 & 0.0398 & 0.0351 \\ 0.0310 & 0.0351 & 0.0310 \end{bmatrix}. \quad (10)$$

When $\sigma = 2$, $k = 2$, and the size of Gaussian filter kernel is 5 ∗ 5, the corresponding Gauss kernel is shown

$$\begin{bmatrix} 0.0416 & 0.0213 & 0.0241 & 0.0213 & 0.0416 \\ 0.0213 & 0.0310 & 0.0351 & 0.0310 & 0.0213 \\ 0.0241 & 0.0351 & 0.0398 & 0.0351 & 0.0241 \\ 0.0213 & 0.0310 & 0.0351 & 0.0310 & 0.0213 \\ 0.0416 & 0.0213 & 0.0241 & 0.0213 & 0.0416 \end{bmatrix}. \quad (11)$$

The choice of Gaussian convolution kernel size will affect the performance of Canny detector. The larger the size, the lower the sensitivity of the detector to noise, but the positioning error of edge detection will increase slightly.

If a 3 ∗ 3 window in the image is f and the Pixel to be filtered is $f(i, j)$, then after Gauss filtering, the value of pixel



(a) Vertical projection of tee (b) Edge of tee

(c) Vertical projection of elbow (d) Edge of elbow

Figure 3: Edge detection of vertical projection images.

$f(i,j)$ is shown in

$$f(i,j) = H * f = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} * \begin{bmatrix} f(i-1,j-1) & f(i-1,j) & f(i-1,j+1) \\ f(i,j-1) & f(i,j) & f(i,j+1) \\ f(i+1,j-1) & f(i+1,j) & f(i+1,j+1) \end{bmatrix}. \tag{12}$$

In equation (12), $*$ is a convolution symbol.

3.2. Calculate the Intensity and Direction of the Gradient. Using a discrete difference operator, convolution operation is carried out from the x-axis and y-axis, respectively. The gray change value and direction in the horizontal and vertical directions are obtained. Determine the gradient amplitude and direction with

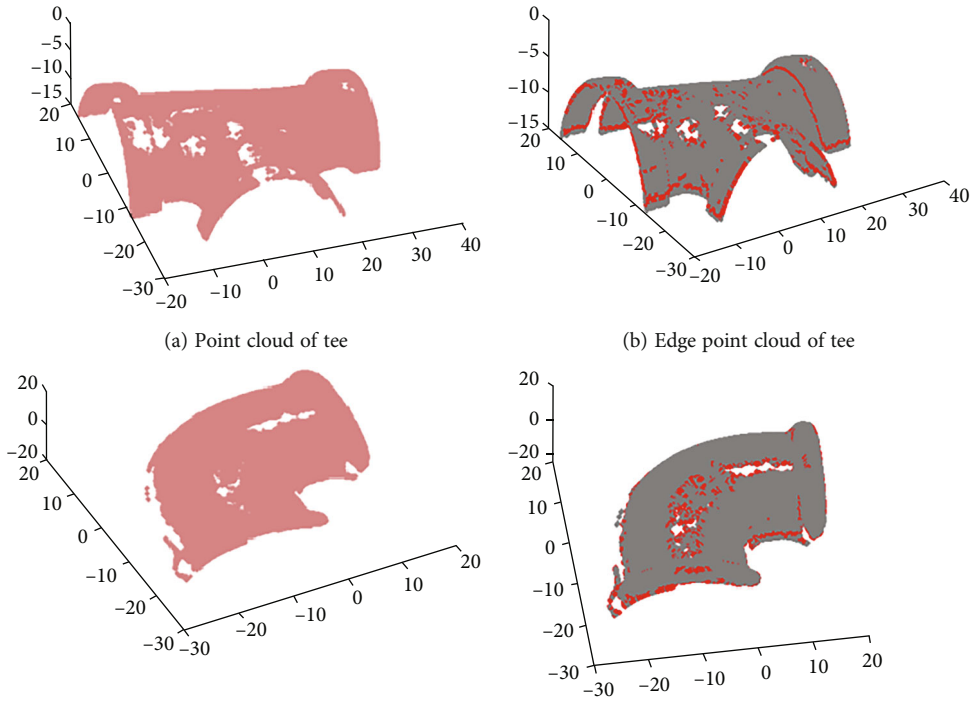
$$M(i,j) = \sqrt{G_x^2(i,j) + G_y^2(i,j)}, \tag{13}$$

```
pc=[];
for i=1:k
   m = find(abs(f(:,1)-newx(i))<0.1);
   n = find(abs(f(:,2)-newy(i))<0.1);
   R = intersect(m(:),n(:));
   [a,b]=size(R);
   if a&&b
      for j=1:a
         temp(j,1)=f(R(j),1);
         temp(j,2)=f(R(j),2);
         temp(j,3)=f(R(j),3);
      end
      pc=[pc;temp];
   end
end
```

Code 1

$$\theta(i,j) = \arctan \frac{G_y(i,j)}{G_x(i,j)}. \tag{14}$$

Here, $G_x(i,j)$ and $G_y(i,j)$ are the first derivatives of horizontal and vertical directions, respectively. Suppose G_x and



(a) Point cloud of tee

(b) Edge point cloud of tee

(c) Point cloud of elbow

(d) Edge point cloud of elbow

Figure 4: Back projection transform of edge detection data.

G_y are the Sobel operators which are shown in

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}, \tag{15}$$

where G_x is the Sobel operator in the x direction, which is used to detect the edge in the y direction and G_y is the Sobel operator in the y direction, which is used to detect the edge in the x direction (edge direction is perpendicular to gradient direction). $G_x(i,j)$ and $G_y(i,j)$ are the convolution of $G_x(G_y)$ and the image data. The calculation formula is shown in

$$G_x(i,j) = -1 * f(i-1,j-1) - 2 * f(i,j-1) - f(i+1,j-1) + f(i-1,j+1) + 2 * f(i,j+1) + 2 * f(i+1,j+1), \tag{16}$$

$$G_y(i,j) = -1 * f(i-1,j-1) - 2 * f(i-1,j) - f(i-1,j+1) + f(i+1,j-1) + 2 * f(i+1,j) + 2 * f(i+1,j+1). \tag{17}$$

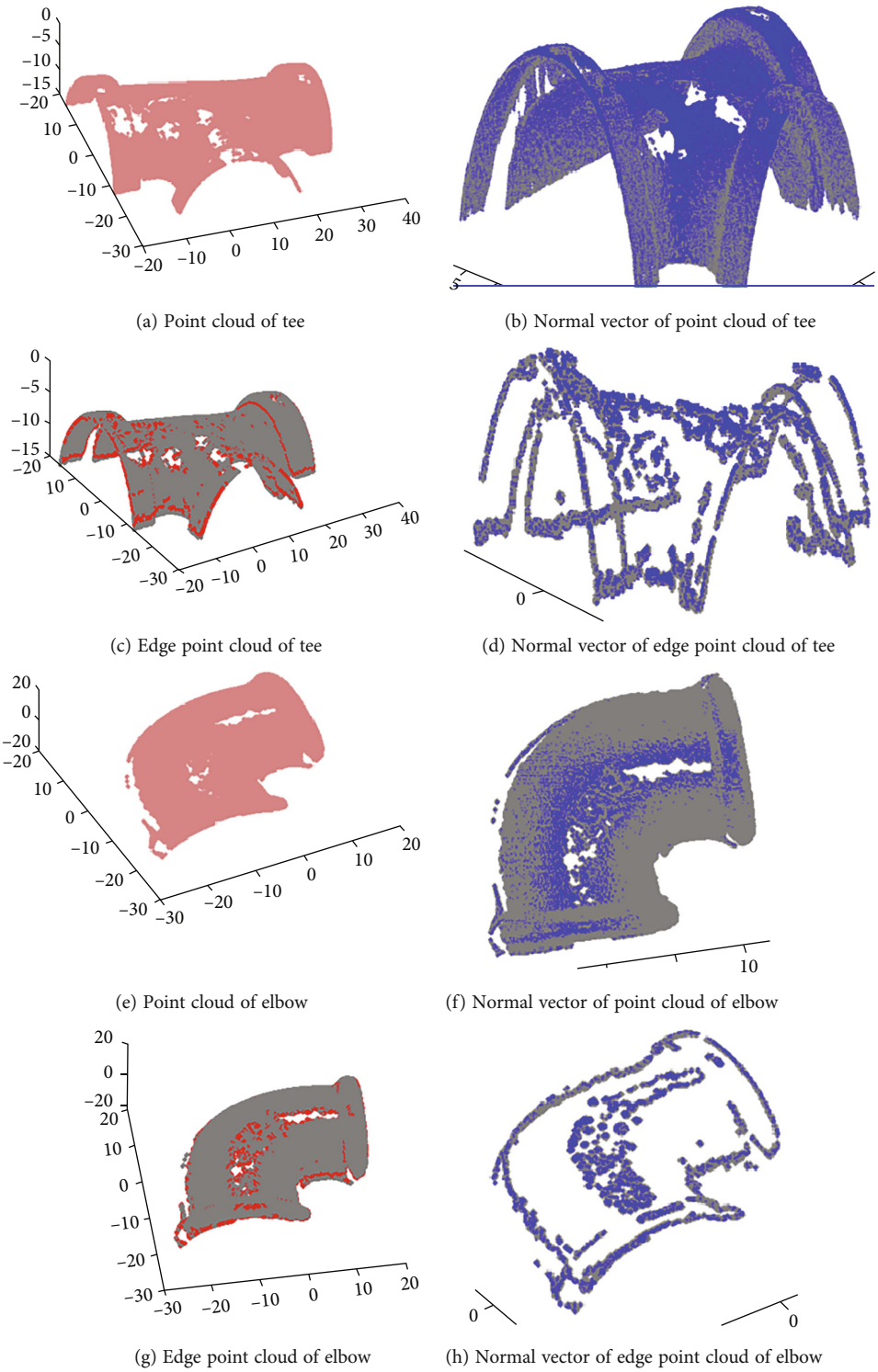
3.3. Nonmaximum Suppression. Nonmaximum suppression is a kind of edge sparsity technology. The effect of nonmaximum suppression lies in the "thin" edge. After calculating the gradient of the image, the edge extracted only based on the gradient value is still very fuzzy. The edge should have only one accurate response. Nonmaximum suppression can help to suppress all gradient values except local maximum to 0. The algorithm of nonmaximum suppression for each pixel in gradient image is as follows. (1) The gradient intensity of the current pixel is compared with two pixels along the positive and negative gradient direction. (2) If the gradient intensity of the current pixel is the largest compared with the other two pixels, the pixel will remain as the edge point; otherwise, the pixel will be suppressed.

Table 1: The number of original point and edge point.

Target type	Original point	Edge point
Tee	647984	60015
Elbow	520915	53353

3.4. Double Threshold Detection and Suppress Isolated Low Threshold Points. In order to delete the edge pixels that are caused by noise, delete the edge pixels with weak gradient, and retain edge pixels with high gradient through selecting high and low thresholds. The gradient of weak edge pixel is less than the high threshold and greater than the low threshold that should be suppressed.

Generally, weak edge pixels caused by real edges are connected to strong edge pixels, but noise response is not connected, in order to track the edge connection, by looking at



(a) Point cloud of tee

(b) Normal vector of point cloud of tee

(c) Edge point cloud of tee

(d) Normal vector of edge point cloud of tee

(e) Point cloud of elbow

(f) Normal vector of point cloud of elbow

(g) Edge point cloud of elbow

(h) Normal vector of edge point cloud of elbow

Figure 5: Normal vector of tee and elbow.

the weak edge pixel and its eight neighbours. When one of them is a strong edge pixel, the weak edge point can be retained as a real edge.

Edge detection is performed on the image after the point cloud is vertically projected, and the result is shown in Figure 3.

Figures 3(a) and 3(c) are the vertical projection images of tee and elbow. Figures 3(b) and 3(d) are the edges of vertical projection images. Through edge detection, the edge feature of target can be detected. Compared with the target point cloud, the edge feature can represents the target very well and the point number is reduced greatly.

Table 2: The calculation time of normal vector.

Order number	Target	Calculation time (s)
1	Point cloud of tee	53.03
2	Edge point cloud of tee	4.94
3	Point cloud of elbow	42.03
4	Edge point cloud of elbow	4.37

4. Back Projection Transform of Edge Detection Data

In order to express the three-dimensional information of edge points, the edge detection data is back projected to the origin cloud. The points in the edge map after projection transformation of point cloud are represented as $f(i, j)$, in which i is the horizontal ordinate. The value range is $1 \sim W$. j is the vertical ordinate. The value range is $1 \sim H$. When $f(i, j) = 1$, that means the point is the edge point. The corresponding point in point cloud should be found. First, calculate the x and y of point cloud. The transformation method is as follows:

$$x_i = (i-1) * \frac{x_{\max} - x_{\min}}{W} + x_{\min}, \tag{18}$$

$$y_i = (j-1) * \frac{y_{\max} - y_{\min}}{H} + y_{\min}. \tag{19}$$

In the origin point cloud data, search the horizontal ordinate and vertical ordinate. Then, mark the points as edge points. The search code is as follows:

The effect of the edge point cloud after the inverse transformation of the edge points is shown in Figure 4, in which gray represents the target and red represents the edge point cloud.

Figures 4(a) and 4(c) are the point cloud of tee and elbow. Figures 4(b) and 4(d) are the edge point cloud of them. Calculate the number of original point and edge point. The number is shown in Table 1.

The edge point of each target is around 10% of the original point. After extracting the edge points, it can provide a reliable basis for the subsequent point cloud normal vector calculation and point cloud registration.

5. Normal Vector Calculation of Edge Points Based on PCA

After edge detection, the normal vector of the local fitting plane is taken as the normal vector of the point. Suppose the edge point is $P_i = [x_i, y_i, z_i]^T$, K-Neighbourhood search in origin cloud dataset. Calculate the best fit plane $Ax + By + Cz - D = 0$. Here, A, B, C is the normal vector of a plane equation. D represents the distance from the origin to the plane, $A^2 + B^2 + C^2 = 1$, and $D > 0$. The distance from each point to the plane is $d_i = |Ax_i + By_i + Cz_i - d|$. The sum of the distance from each point to the best fit plane is the smallest, that is,

$$e = \sum_{i=1}^{n} d_i^2 \longrightarrow \min. \tag{20}$$

The normal vector estimation of the edge point cloud is shown in Figure 5.

Because the number of points corresponding to edge features is greatly reduced, the calculation time of using edge features to calculate normal vector is also greatly shortened. The calculation time is shown in Table 2.

It can be seen from the figure that compared with the normal vector of all data point clouds of the target, the normal vector of the edge point cloud can well express the characteristics of the target, and the processing time of the tee target is reduced from 53.03 s to 4.94 s, and the elbow target is reduced from 42.03 s to 4.37 s, which is around 10% of the original.

6. Conclusions

The data of point cloud is large and contains a lot of invalid information. It is very important to extract the characteristics of point cloud. Edge features can well express the geometric features of the target, so it is very important to extract edge point cloud. This paper proposes an edge detection algorithm based on projection transformation. Firstly, the target point cloud is projected vertically. Then, the Canny algorithm is used to detect the edge of the image. The detected edge data is back projected to extract the edge point cloud. By calculating the normal vector of the edge point cloud, compared with the normal vector of the whole data point cloud of the target, the normal vector of the edge point cloud can well express the characteristics of the target, and the calculation time is reduced to 10% of the original, which greatly saves the calculation time. This paper only detects 3D tee and elbow. Give the advantages of edge features in normal vector calculation. The advantages of edge point cloud in point cloud matching need to be further studied.

Data Availability

The datasets used and/or analyzed during the current study are available from the corresponding author on reasonable request.

Conflicts of Interest

The authors declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Acknowledgments

This paper is supported by Development and Reform Commission of Jilin Province (2020C018-3) and Jilin Provincial Department of Education (JJKH20210726KJ).

References

[1] S. W. Lee, S. Sarp, D. J. Jeon, and J. H. Kim, "Smart water grid: the future water management platform," *Desalination and Water Treatment*, vol. 55, no. 2, pp. 339–346, 2015.

[2] J. Zhang, J. J. Cao, X. Liu, H. Chen, B. Li, and L. Liu, "Multi-normal estimation via pair consistency voting," *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 4, pp. 1693–1706, 2019.

[3] W. Dong, "Building point cloud feature extraction using geometric features of adjacent points," *Laser and Optoelectronics*, vol. 55, no. 7, pp. 181–188, 2018.

[4] J. Zhu, J. P. Huang, and L. M. Wang, "Laser printing files detection method based on double features," *International Journal of Pattern Recognition and Artificial Intelligence.*, vol. 32, no. 10, p. 1854028, 2018.

[5] W. Ali, S. Abdelkarim, and M. Zidan, "Yolo3d: end-to-end real-time 3d oriented object bounding box detection from LiDAR point cloud," in *Lecture Notes in Computer Science*, pp. 716–728, Springer, Cham, 2019.

[6] Y. Zhou and O. Tuzel, "Voxelnet: end-to-end learning for point cloud based 3d object detection," 2017, https://arxiv.org/abs/1711.06396.

[7] G. P. Meyer, A. Laddha, E. Kee, C. Vallespi-Gonzalez, and C. K. Wellington, "LaserNet: an efficient probabilistic 3D object detector for autonomous driving," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, CA, USA, 2019.

[8] J. Beltran, C. Guindel, F. M. Moreno, D. Cruzado, F. Garcia, and A. De La Escalera, "BirdNet: a 3D object detection framework from LiDAR information," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, Maui, HI, USA, 2018.

[9] K. Minemura, H. Liau, A. Monrroy, and S. Kato, "LMNet: real-time multiclass object detection on CPU using 3D LiDARs," in *2018 3rd Asia-Pacific Conference on Intelligent Robot Systems (ACIRS)*, Singapore, 2018.

[10] A. Asvadi, L. Garrote, C. Premebida, P. Peixoto, and U. J. Nunes, "DepthCN: vehicle detection using 3d-lidar and convnet," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, Yokohama, Japan, 2017.

[11] M. Simon, S. Milz, K. Amende, and H. M. Gross, "Complex-yolo: real-time 3d object detection on point clouds," 2018, https://arxiv.org/abs/1803.06199.

[12] S. H. B. Xia and W. R. SH, "A fast edge extraction method for mobile LiDAR point clouds," *IEEE Geoscience and Remote Sensing Letters*, vol. 14, no. 8, pp. 1288–1292, 2017.

[13] C. H. J. Ding, G. Sun, and L. L. Yin, "Boundary extraction of scattered point cloud," *Computer Technology and Development*, vol. 27, no. 7, pp. 83–86, 2017.

[14] J. X. Li, E. Q. Wu, and Y. L. Ke, "3D reconstruction of small diameter pipes inner surface based on structural light," *Chinese Journal of Scientific Instrument*, vol. 27, no. 3, pp. 254–258, 2006.

[15] J. Ye, Z. Gao, X. Liu, W. Wang, and C. Zhang, "Freeform surfaces reconstruction based on Zernike polynomials and radial basis function," *Acta Optical Sinica*, vol. 34, no. 8, pp. 0822003–0822241, 2014.

[16] J. Z. Zhou and Y. J. Yan, "Research on the feature extraction technology of the point cloud in reverse engineering," *Equipment Manufacturing Technology*, vol. 8, no. 13-17, p. 33, 2019.

[17] Z. H. X. Duan, *Research on data reduction and surface reconstruction of 3D laser scanning and its application*, China University of mining and technology, 2019.

[18] H. Song, H. Y. Feng, and D. S. Ouyang, "Automatic detection of tangential discontinuities in point cloud data," *Journal of Computing and Information Science in Engineering*, vol. 8, no. 2, pp. 1–10, 2008.

[19] H. Y. Han, X. Han, and S. F. SH, "Point cloud simplification with preserved edge based on normal vector," *International Journal for Light and Electron Optics*, vol. 126, no. 19, pp. 2157–2162, 2015.

[20] L. Chen, Y. Cai, and J. S. H. Zhang, "Feature extraction of scattered point cloud based on hybrid method of multiple discriminant parameters," *Computer Application Research*, vol. 34, no. 9, pp. 2867–2870, 2017.

[21] T. Hackel, J. D. Wegenr, and K. Schindler, "Contour detection in unstructured 3D point clouds," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1610–1618, 2016.

[22] S. H. Y. Pei, N. Du, and L. Wang, "Feature extraction of building point cloud based on moving least squares normal vector estimation," *Bulletin of Surveying and Mapping*, vol. 4, pp. 73–77, 2018.