Improvement of DBSCAN Algorithm Involving Automatic Parameters Estimation and Curvature Analysis in 3D Point Cloud of Piled Pipe

Alfan Rizaldy Pratama [1,2], Bima Sena Bayu Dewantara [3], Dewi Mutiara Sari [3], and Dadet Pramadihanto [3,*]

[1] Center for Research and Innovation on Advanced Transportation Electrification, Politeknik Elektronika Negeri Surabaya, Surabaya, Indonesia
[2] Data Science Department, Faculty of Computer Science, Universitas Pembangunan Nasional Veteran Jawa Timur, Surabaya, Indonesia
[3] Department of Informatics and Computer Engineering, Politeknik Elektronika Negeri Surabaya, Surabaya, Indonesia
Email: alfan.rizaldy@upnjatim.ac.id (A.R.P.); bima@pens.ac.id (B.S.B.D.); dewi.mutiara@pens.ac.id (D.M.S.); dadet@pens.ac.id (D.P.)
*Corresponding author

***Abstract*—Bin-picking in the industrial area is a challenging task since the object is piled in a box. The rapid development of 3D point cloud data in the bin-picking task has not fully addressed the robustness issue of handling objects in every circumstance of piled objects. Density-Based Spatial Clustering of Application with Noise (DBSCAN) as the algorithm that attempts to solve by its density still has a disadvantage like parameter-tuning and ignoring the unique shape of an object. This paper proposes a solution by providing curvature analysis in each point data to represent the shape of an object therefore called Curvature-Density-Based Spatial Clustering of Application with Noise (CVR-DBSCAN). Our improvement uses curvature to analyze object shapes in different placements and automatically estimates parameters like Eps and MinPts. Divided by three algorithms, we call it Auto-DBSCAN, CVR-DBSCAN-Avg, and CVR-DBSCAN-Disc. By using real-scanned Time-of-Flight camera datasets separated by three piled conditions that are well separated, well piled, and arbitrary piled to analyze all possibilities in placing objects. As a result, in well separated, Auto-DBSCAN leads by the stability and accuracy in 99.67% which draws as the DBSCAN using specified parameters. For well piled, CVR-DBSCAN-Avg gives the highest stability although the accuracy can be met with DBSCAN on specified parameters in 98.83%. Last, in arbitrary piled though CVR-DBSCAN-Avg in accuracy lower than DBSCAN which is 73.17% compared to 80.43% the stability is slightly higher with less outlier value. Deal with computational time higher than novel DBSCAN, our improvement made the simplicity and deep analysis in scene understanding.**

***Keywords*—bin-picking, point cloud, density-based clustering, automatic clustering, curvature analysis**

I. Introduction

Recently, the 4th industrial revolution has supported the growth of robot automation in many industrial activities. Dealing with an automatic role in an industrial environment such as bin-picking is the first wave of robots in the job. To be able to apply other processes like loading or unloading industrial parts, assembling parts, sorting, bin-picking, etc. [1–5]. These tasks are inseparable from the machine vision system, which uses computer vision technology to obtain the data. One of the camera devices that can produce point cloud data is the Time-of-Flight camera. By using this camera, the various lightning which is the major problem in machine vision can be resolved [6, 7]. Generally, the object used is a fitting pipe with complex placements [4, 8, 9]. This condition makes it difficult for the bin-picking system to process the data. Clustering is a key to success in a bin-picking pipeline system where the core task in bin-picking is how the multiple objects are separated from each other although these are overlapping. Afterward, the individual data object can be further processed by the system and pick the object. In common, Euclidean clustering is a distance-based method to separate each object into individuals [10]. But in the bin-picking case, distance information is not enough because the objects are stacked and overlap each other. With these configurations, it might happen the objects are overlapping each other making the information of an object incomplete which makes Euclidean clustering algorithms cannot handle overlapping objects that make occlusion on data results.

Another popular method that has been applied to solve this issue is using deep learning. Qi *et al.* [11] was the first architecture that can consume raw 3D point cloud directly and its architecture of deep learning can handle segmentation tasks. By taking the output of feature transform and global feature, the extended network that contains a multi-layer perceptron can segment objects into separated clusters. Because PointNet ignores local structures that are important in representing 3D shapes, continued by PointNet++ that be used to address this problem by creating sampling and grouping operations to

Manuscript received September 15, 2023; revised November 1, 2023; accepted November 17, 2023; published June 14, 2024.

extract features from point clusters in a hierarchical manner [12]. Other researchers also studied the development of deep learning in clustering tasks like using a Geometry Sharing Network [13] which learns the geometric information of the point cloud data and the method proposed by Song *et al.* [14] that proposes a Hybrid Semantic Affinity (HSA) learning method to leveraging the dependencies of many categories in 3D semantic segmentation. Also, Xu *et al.* [15] proposed instance segmentation based on point-wise embedded feature and centroid value that takes the x, y and z with each normal value using DGCNN as the backbone for feature extraction. However, the deep learning method tends to be performance-heavy.

Density-based clustering attempts to cluster data by the density of a region [16]. This algorithm is known well for grouping data not based only on their distance, but the density is considered to form a cluster of data. With the benefit of density information that can help to form a cluster though the object is randomly piled but still have a remaining problem that the parameters (*Eps* and *MinPts*) to consider the cluster is manually chosen by the user causing varying clustering result and taking time to consider the optimum parameter in DBSCAN [17]. There are many ways to estimate these parameters automatically [17–20]. Even like that, few of these have been implemented into piled industrial objects. Wang *et al.* [21] are implementing improved Density-Based Spatial Clustering of Application with Noise (DBSCAN) that automatically estimated the Eps parameter based on K^{th} nearest neighbor, polynomial fitting, and derivative method, but the K value for nearest neighbor estimation and *MinPts* parameter is still considered manually by the user. Gaonkar *et al.* [22] present automatically both estimations of *Eps* and *MinPts*, but again the K value is determined by the user and it leads to a different cluster result. Czerniawski *et al.* [23] and Guo *et al.* [8] present DBSCAN using industrial objects but still manually chosen parameters for its DBSCAN although Guo *et al.* [8] provide DBSCAN combined with region growing.

In this paper, we proposed to improve the clustering algorithm based on density and curvature named Curvature-Density-Based Spatial Clustering Application with Noise (CVR-DBSCAN), an improvement of the novel DBSCAN that brings the fully automatic parameter estimation in DBSCAN and curvature analysis for determining density. Due to the uncertainty of the parameter used in DBSCAN, CVR-DBSCAN is also followed by automatic estimation of the parameter used in the algorithm to avoid the dynamic piled position of objects. Then CVR-DBSCAN is divided into three types. Auto-DBSCAN which only has an automatic estimation parameter, CVRDBSCAN-Avg which added the curvature analysis based on average to give a decision, and CVR-DBSCAN-Disc which added the curvature analysis based on discontinuity to give a decision. First, the data taken from the Time-of-Flight camera is distinguished by three models of laying condition, there are well separated, well piled, and arbitrary piled. Then the obtained data is preprocessed by removing non-measured data and the passthrough filter. Followed by plane segmentation to separate the object and its planar. Then CVR-DBSCAN is applied to cluster the object.

II. Material and Method

The method used to develop the bin-picking vision systems follows a sequence of steps shown in Fig. 1.

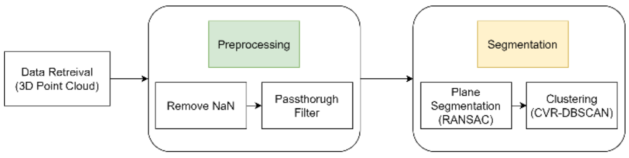


Fig. 1. System diagram.

A. Preprocessing

The raw point cloud data in *x*, *y* and *z* produced by the Time-of-Flight camera CamBoard pico flexx needs some configuration with preprocessing where *P* is a point cloud and *S* is the whole scene captured by the camera. To remove its non-measured data Eq. (2) is used where all points that are not a *NaN* can be incorporated into point cloud data *P*. Then, by applying the Passthrough filter to remove unused data outside the box using Eq. (3) for focusing on data in the box only which is bounded by two thresholds less than *LT* and greater than *GT*.

$$P \in S \tag{1}$$

$$P_i = \begin{cases} p_i, p_i \neq NaN \\ null, otherwise \end{cases} \tag{2}$$

$$P_i = \begin{cases} p_i, LT > p_{i(x,y)} > GT \\ null, otherwise \end{cases} \tag{3}$$

B. Segmentation

The key process of the proposed pipeline is segmentation. In this step, we do the separating objects between their plane and each other. Plane segmentation is intended to obtain the object's data only. Using the RAndom SAmple Consensus (RANSAC) algorithm which resampling technique that results points candidate as a model iteratively. Working by choosing 3 random points from data to form a plane equation in Eq. (4).

$$ax + by + cz + d = 0 \tag{4}$$

where *a*, *b* and *c* are constant values for random points *x*, *y* and *z* that can be obtained using Eqs. (5)–(7).

$$a = [(y_2 - y_1) \times (z_3 - z_1) - (z_2 - z_1) \times (y_3 - y_2)] \tag{5}$$

$$b = [(z_2 - z_1) \times (x_3 - x_1) - (x_2 - x_1) \times (z_3 - z_2)] \tag{6}$$

$$c = [(x_2 - x_1) \times (y_3 - y_1) - (y_2 - y_1) \times (x_3 - x_2)] \tag{7}$$

After the constant value is obtained, *d* which a normal vector can be obtained using Eq. (8).

$$d = -(ax + by + cz) \tag{8}$$

Then, determine the distance between all points against the plane. Using Eq. (9) to know whether the new point belongs to the plane or not.

$$dist_{np} = \frac{ax_4 + by_4 + cz_4 + d}{\sqrt{a^2 + b^2 + c^2}} \tag{9}$$

where $dist_{np}$ is the distance and (x_4, y_4, z_4) is the new point. A threshold is given to determine whether points are included as a plane or an object.

The next step is clustering. By obtaining the objects, they must be separated from each other and then can be further processed. Before we go through to the CVR-DBSCAN, we discuss the brief explanation of the DBSCAN algorithm which is the basis of our improvement. Based on density, DBSCAN works by clustering data according to two parameters: epsilon Eps and minimum points $MinPts$, so it can be called that DBSCAN is an automatic clustering because it doesn't need to determine the preliminary number of clusters. Eps is a region that limits the search for core points, the main key to forming a cluster. In addition, there is also a limitation in the form of $MinPts$, which provides a minimum limit for a collection of points in Eps to meet the criteria as a core point candidate. DBSCAN depend on some definition explained below.

- Eps-neighborhood, when two points p and q met the requirement of Eq. (10).

$$N_{eps}(p) = \{q \in P | dist(p, q) \leq Eps\} \tag{10}$$

where $dist(p, q)$ can be obtained using Eq. (11).

$$dist(p, q) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2} \tag{11}$$

- Directly density-reachable, the relation of point p and other points w.r.t. Eps and $MinPts$. Point q is directly density-reachable with point p if $q \in N_{eps}(p)$ and point p is a core point that meets Eq. (12).

$$N_{eps}(p) \geq MinPts \tag{12}$$

- Density-reachable, the relation between point p and q w.r.t. Eps and $MinPts$ if there is a chain $p_1, p_2, p_3, \dots, p_n$ where $p_1 = q, p_n = q,$ and p_{i+1} is directly density-reachable from p_i.
- Density-connected, the relation between point p and q w.r.t. Eps and $MinPts$ if there are core point that are density-reachable into both two points p and q.
- Cluster, the group of points C w.r.t. Eps and $MinPts$ that meet the stated condition below:

Maximality: $\forall p, q : p \in c$ and q are density-reachable from p w.r.t. Eps and $MinPts$, then $q \in C$.

Connectivity: $\forall p, q : C : p$ is density-connected into q w.r.t. Eps and $MinPts$.

DBSCAN was still dependent on Eps and $MinPts$ parameters to perform clustering. This condition makes it difficult when the condition of objects is unpredictable. Curvature-Density-Based Density Spatial Clustering Application with Noise (CVR-DBSCAN) are improved DBSCAN algorithm proposed by the author which offers automatic estimation parameter of DBSCAN which Eps and $MinPts$, also curvature analysis for the sake of curved object shape. Previous research also takes up DBSCAN in the object recognition pipeline [4]. A brief illustration of the DBSCAN and CVR-DBSCAN is shown in Figs. 2 and 3, respectively.

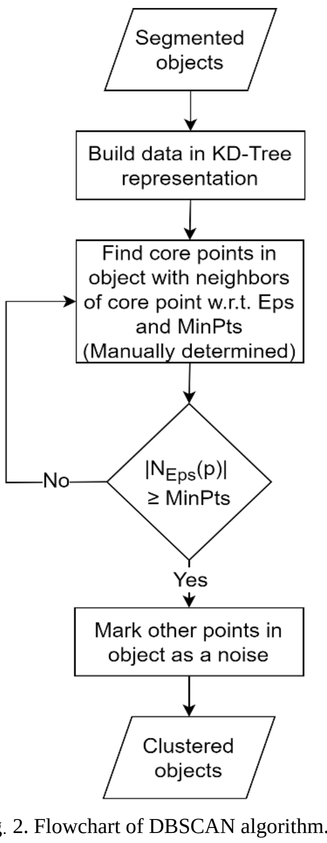

Segmented objects
Build data in KD-Tree representation
Find core points in object with neighbors of core point w.r.t. Eps and MinPts (Manually determined)
$|N_{Eps}(p)| \geq$ MinPts
No
Yes
Mark other points in object as a noise
Clustered objects

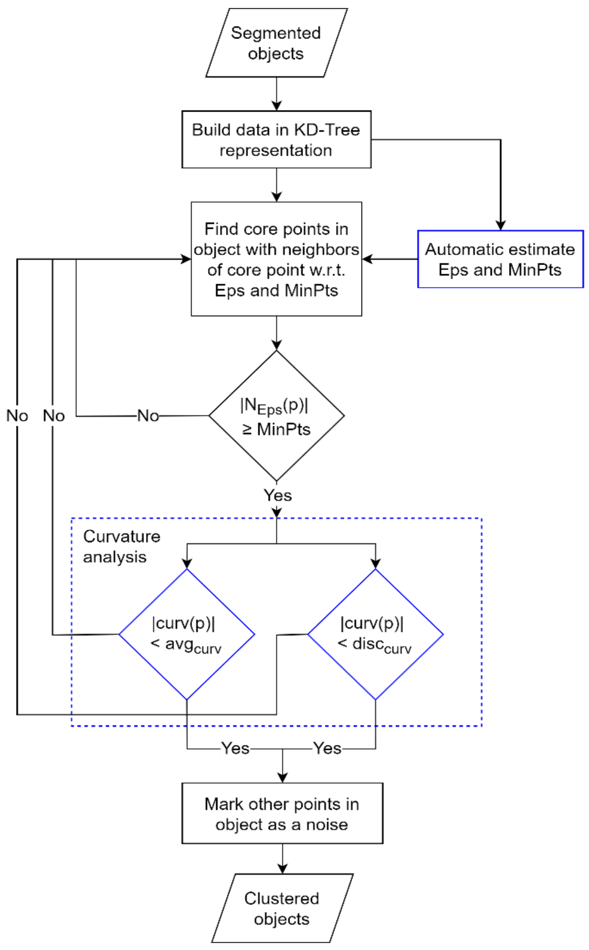
Fig. 2. Flowchart of DBSCAN algorithm.

By using k-Nearest Neighbor (k-NN) and rule-of-thumb of k-NN in determining k-value using Eq. (13).

$$k = \sqrt{\frac{N}{2}} \tag{13}$$

where N is the sum of points in the point cloud P. Then followed by Eq. (14) to store the average distance between points p_i with their neighbors in avg_{dist_i}.

$$avg_{dist_i} = \frac{np_i}{k}, i = \{1, 2, \dots, N\} \tag{14}$$



Segmented objects
Build data in KD-Tree representation
Find core points in object with neighbors of core point w.r.t. Eps and MinPts
Automatic estimate Eps and MinPts
$|N_{Eps}(p)| \geq$ MinPts
No
No
No
Yes
Curvature analysis
$|curv(p)| < avg_{curv}$
$|curv(p)| < disc_{curv}$
Yes
Yes
Mark other points in object as a noise
Clustered objects

Fig. 3. Flowchart of CVR-DBSCAN algorithm.

Then save the avg_{dist_i} in a 2-dimensional matrix called mat_{dist} with its index in Eq. (15) and the illustration can be seen in Fig. 4.

$$mat_{dist} = \begin{pmatrix} idx_i & avg_{dist_i} \\ idx_n & avg_{dist_n} \end{pmatrix} \tag{15}$$

Next, data in mat_{dist} is normalized using Min-Max Normalization using Eqs. (16) and (17).

$$sc\_avg_{dist} = \frac{avg_{dist_i} - min(avg_{dist})}{max(avg_{dist}) - min(avg_{dist})} \tag{16}$$

$$sc\_idx_{dist} = \frac{idx_i - min(idx)}{max(idx) - min(idx)} \tag{17}$$

As a result, we can see in Fig. 5 that the normalization is well performance because the data pattern is the same as Fig. 4 as the input data.

To obtain the Eps parameter from the graph data of k-NN distance determined from maximum curvature point. The purpose of obtaining them is the same as finding the elbow of a curve then $normmat_{dist}$ need to be rotated and pick the minimum value of the rotated curve.

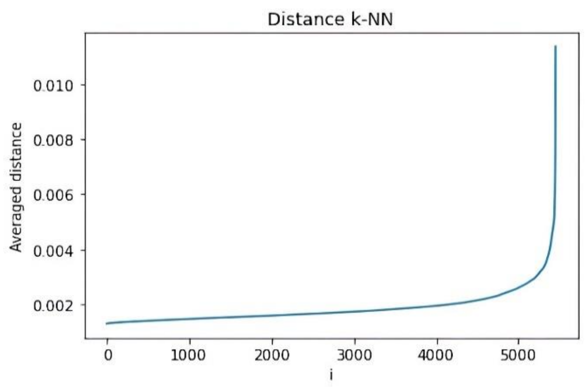


Fig. 4. Visualization of mat_{dist} data.

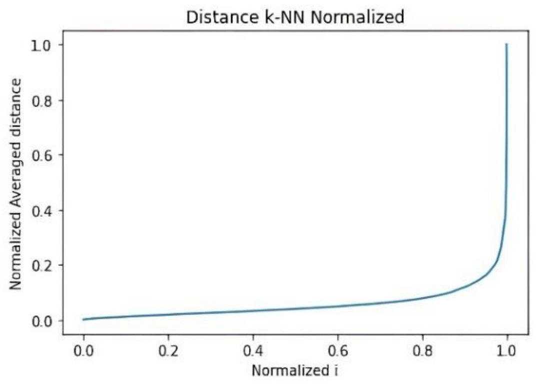


Fig. 5. Visualization of $normmat_{dist}$ data.

Erenow, firstly θ value for rotating picked from $normmat_{dist}$ using Eq. (18) until Eq. (21) by taking four values in each e, f, g, and h variable.

$$e = normmat_{dist}(0,0) \tag{18}$$

$$f = normmat_{dist}(N,0) \tag{19}$$

$$g = normmat_{dist}(0,1) \tag{20}$$

$$h = normmat_{dist}(N,1) \tag{21}$$

where all constant e, f, g and h acquired from value both in first-last at 1st and 2nd column. By using $\arctan 2$ in Eq. (22), theta for rotate $normmat_{dist}$ is obtained.

$$\theta = arctan\,2\,((h-g),(f,e)) \tag{22}$$

Uniquely, the obtained theta is always 45° because the given data for $\arctan 2$ is always valued (1, 1). With the obtained θ value, Eq. (23) is used for forming the rotation matrix and then mat_{rot} is multiplied with $normmat_{dist}$ and the result of multiplication depicted in Fig. 6 which is saved on $rot_{normmat_{dist}}$.

$$mat_{rot} = \begin{pmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{pmatrix} \tag{23}$$

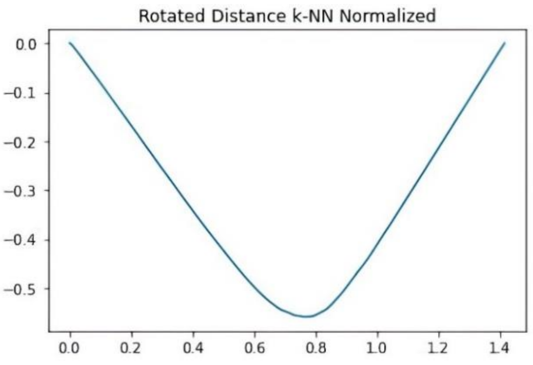


Fig. 6. Visualization of $rot_{normmat_{dist}}$ data.

From rotated data, we will search the minimum value using Eq. (24).

$$min_{val} = min\left(rot_{normmat_{dist}}(:,1)\right) \tag{24}$$

Next is the Eps parameter value that can be obtained using Eq. (25) where the index of minimum value is used as an index to search its value corresponding to mat_{dist}.

$$Eps = mat_{dist}(idx\_min_{val}) \tag{25}$$

Finally, the $MinPts$ parameter can be obtained which is the parameter for determining how many points in a radius will be satisfied and becoming one of them a core point. The calculation is shown in Eq. (26).

$$MinPts = \frac{\sum_i^N N_{Eps}(p_i)}{N} \tag{26}$$

Although all parameters are obtained, because the objects are stacked, this causes a lot of data to overlap with each other, so the shape of the object must be considered for separation. Based on these problems, curvature analysis is needed to provide improvement with additional information. This algorithm adds curvature analysis to determine a point that can be a core point which is a determinant of the formation of a cluster.

Before we can know the curvature value of each point, first of all, we must estimate its normal point based on its nearest neighbor. By taking k value automatically using rule-of-thumb in k-NN in Eq. (13). Then, we form a covariance matrix based on the neighbor of point p_i in P by using Eqs. (27)–(29).

$$C = \frac{1}{k} \cdot (p_i - \overline{p_i}) \cdot (p_i - \overline{p_i})^T \tag{27}$$

$$C = \frac{1}{k} \cdot [x_a \quad y_a \quad z_a] \cdot \begin{bmatrix} x_b \\ y_b \\ z_b \end{bmatrix} \tag{28}$$

$$C = \begin{bmatrix} \frac{x_a x_b}{k} & \frac{y_a x_b}{k} & \frac{z_a x_b}{k} \\ \frac{x_a y_b}{k} & \frac{y_a y_b}{k} & \frac{z_a y_b}{k} \\ \frac{x_a z_b}{k} & \frac{y_a z_b}{k} & \frac{z_a z_b}{k} \end{bmatrix} \tag{29}$$

After we got the covariance matrix C in Eq. (29), then continued with computing Principal Component Analysis (PCA) to obtain the eigenvalue and eigenvector. For example, λ and v are eigenvalue and eigenvector, respectively, then apply Eq. (30).

$$(C - \lambda I) \times v = 0 \tag{30}$$

where I is the identity matrix. Eigenvalue obtained by solving the characteristic equation as can be seen in Eq. (31).

$$det(C - \lambda I) = 0 \tag{31}$$

Generally, characteristic equation results in an 3rd order quadratic equation with constant value as shown in Eq. (32).

$$\lambda^3 - a_2\lambda^2 + a_1\lambda - a_0 = 0 \tag{32}$$

The solutions obtained from Eq. (32) are λ_1, λ_2, and λ_3 where each is an eigenvalue of the covariance matrix C. After the eigenvalue is obtained, continue by finding the eigenvector from the covariance matrix C by substituting each λ obtained from Eq. (32). Then, compute $C - \lambda I$ using Elementary Row Operations until we get the Reduced Row Echelon. Finally, the eigenvector can be obtained in Eq. (33).

$$eigenvector = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} \tag{33}$$

Normal point in $x, y,$ and z can be obtained from Eq. (33) that shown in Eqs. (34)–(36).

$$normal\_x = eigenvector[0] \tag{34}$$
$$normal\_y = eigenvector[1] \tag{35}$$
$$normal\_z = eigenvector[2] \tag{36}$$

After successfully obtaining the normal point, the curvature of a point is obtained using Eq. (37).

$$curvature = \left|\frac{\lambda}{C(0,0) + C(1,1) + C(2,2)}\right| \tag{37}$$

Discontinuity is a graph analysis technique used to determine the elbow of a curve which indicates a significant difference in value at that point. The algorithm to obtain the discontinuity is the same as the Eps parameter finding, in which the input data used is the curvature of each point in all eps-neighborhood from point p_i that is already sorted in ascending order depicted in Fig. 7.

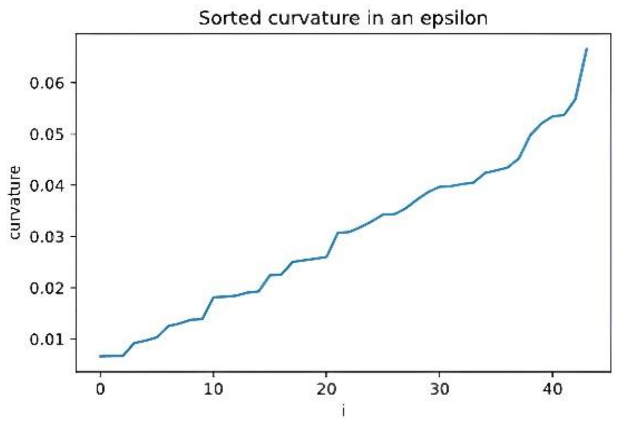


Fig. 7. Visualization of input curvature in eps-neighborhood data.

From the obtained data, we need to normalize the data using Min-Max Normalization in Eq. (38).

$$normed_{curv} = \frac{curv_i - min(curv)}{max(curv) - min(curv)} \tag{38}$$

where $curv$ is the curvature value of point in eps-neighborhood, and i is the index of point.

After normalized data is obtained, the next step is the same as the Eps finding algorithm and we get the $disc_{curv}$ value as the result. Then, the point with the highest curvature will be used as a threshold for determining the core point. Apart from using discontinuity, there is another calculation using an average of the curvature, where the curvature data from points within a radius of Eps will be averaged to be used as a threshold in determining the core point.

$$avg_{curv} = \frac{\sum_{i=1}^{N} curv_i}{N} \tag{39}$$

Lastly, instead of just using $MinPts$ as a determinant of the core point $\left|N_{eps}\right| \geq MinPts$, with curvature analysis Eq. (40) is used as an addition for the determination of the core point.

$$core_{point} = \begin{cases} core_{point}, |curv(p)| < avg_{curv} \| |curv(p)| < disc_{curv} \\ ignore, otherwise \end{cases} \tag{40}$$

Our improvement algorithm is divided into three. There are Auto-DBSCAN which involves the automatic estimation parameter, CVR-DBSCAN-Avg which adds the curvature analysis and takes the average for core point deciding, and CVR-DBSCAN-Disc which adds the curvature analysis and makes the discontinuity result as the core point determination. As a summary, we provide a comprehensive comparison shown in Table I which shows the main difference from the novel DBSCAN.

TABLE I. COMPARISON OF DBSCAN ALGORITHM

Factors	DBSCAN	CVR-DBSCAN
Parameters Selection	Manual	Automatic
Deep Analysis of Object's Shape	Doesn't exist	Curvature Analysis

III. RESULT AND DISCUSSION

A. Experimental Setup and Data Acquisition

Using the Time-of-Flight CamBoard pico flexx camera to obtain the data, in this research several experimental parameters and setup were used. The camera is placed in a fixed position and the height perpendicular to the plane is 42 cm. Then the object is placed in a box sized 30.3×20.8×8.2 cm as depicted in Fig. 8.

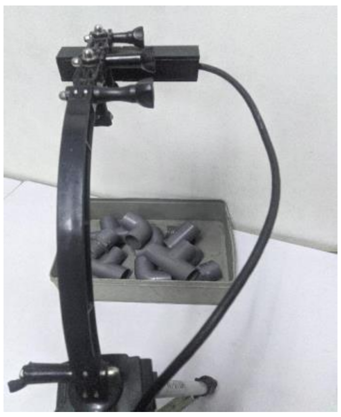


Fig. 8. Visualization of the experimental setup.

To represent the real condition in industry, variety in placing the objects divided by well separated, well piled, and arbitrary piled. Fig. 9 shows the example in real condition of the objects and its point cloud data obtained by CamBoard pico flexx Time-of-Flight camera. To evaluate the proposed systems, 50 data of each condition have been taken.

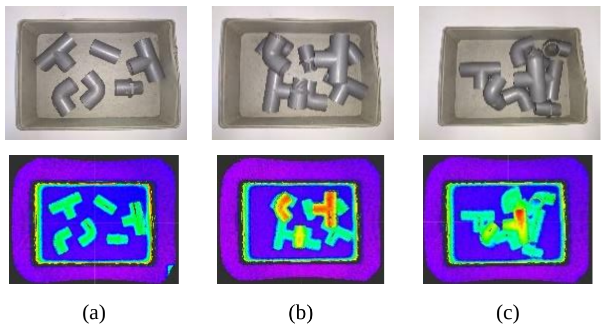
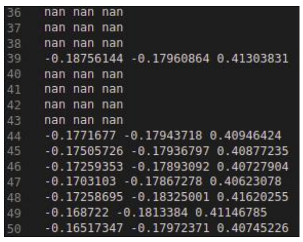


Fig. 9. Variety of placing objects (a) Well separated, (b) Well piled, (c) Arbitrary piled.

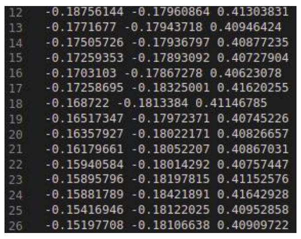
B. Preprocessing

Obtained data is inseparable from non-measured data which is unreadable and has been removed using Remove NaN data which the representation can be seen in Fig. 10.



```
nan nan nan
nan nan nan
nan nan nan
-0.18756144 -0.17960864 0.41303831
nan nan nan
nan nan nan
nan nan nan
nan nan nan
-0.1771677 -0.17943718 0.40946424
-0.17505726 -0.17936797 0.40877235
-0.17259353 -0.17893092 0.40727904
-0.1703103 -0.17867278 0.40623078
-0.17258695 -0.18325001 0.41620255
-0.168722 -0.1813384 0.41146785
-0.16517347 -0.17972371 0.40745226
```

(a)

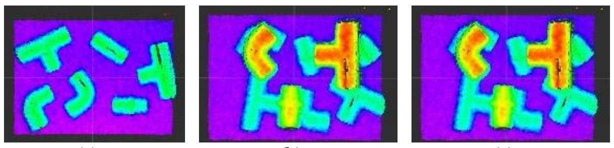


```
-0.18756144 -0.17960864 0.41303831
-0.1771677 -0.17943718 0.40946424
-0.17505726 -0.17936797 0.40877235
-0.17259353 -0.17893092 0.40727904
-0.1703103 -0.17867278 0.40623078
-0.17258695 -0.18325001 0.41620255
-0.168722 -0.1813384 0.41146785
-0.16517347 -0.17972371 0.40745226
-0.16357927 -0.18022171 0.40826657
-0.16179661 -0.18052207 0.40867031
-0.15940584 -0.18014292 0.40757447
-0.15895796 -0.18197815 0.41152576
-0.15881789 -0.18421891 0.41642928
-0.15416946 -0.18122025 0.40952858
-0.15197708 -0.18106638 0.40909722
```

(b)

Fig. 10. Point cloud data representation (a) Before remove NaN, (b) After remove NaN.

However, the result of the Passthrough Filter that eliminates the data outside the box can be seen in Fig. 11.



(a) (b) (c)

Fig. 11. Sample visualization result of passthrough filter (a) Well separated, (b) Well piled, (c) Arbitrary piled.

The performance of preprocessing is within 5–10 ms and depicted in Fig. 12 mostly stable in 5 ms although the placement condition is different. Also preprocessing can reduce around 60% of total points where the original point cloud obtained from the camera is 38,304 points. With 100% accuracy in preprocessing made it possible to go on to the next step of all data.

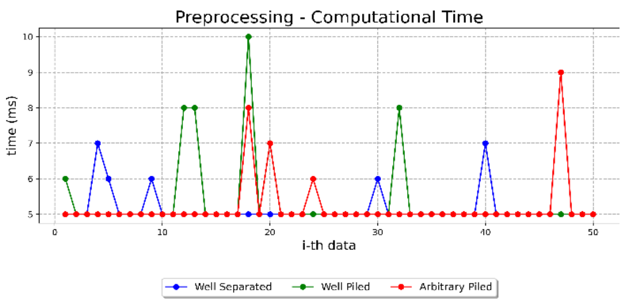


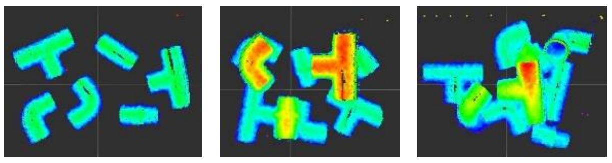
Fig. 12. Computational time of preprocessing.

C. Segmentation

For obtaining individual data, the object must be separated both by its plane and each other.

1) Plane segmentation

By using RANSAC, the object in a scene fully succeeds separated by its plane which makes it possible to process only the object. Fig. 13 shows the example result of plane segmentation in all conditions.



(a) (b) (c)

Fig. 13. Sample visualization result of plane segmentation (a) Well separated (b) Well piled (c) Arbitrary piled.

The computational time in well separated conditions is the fastest among other conditions. This happened because the complexity of placements in well piled and arbitrary piled is higher than well separated. Nevertheless, execution time is still relatively fast with a 10ms maximum computational time taken from Fig. 14.

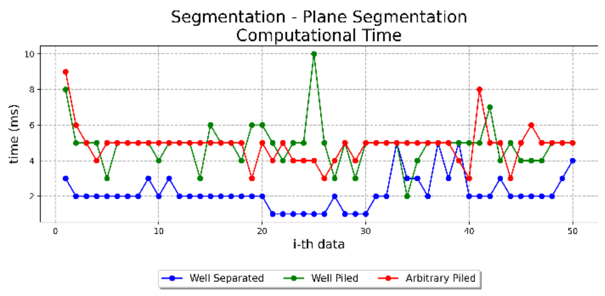
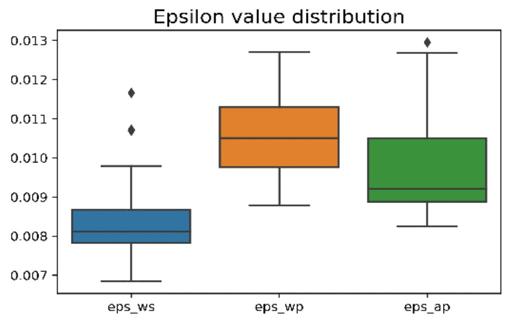


Fig. 14. Computational time of plane segmentation.

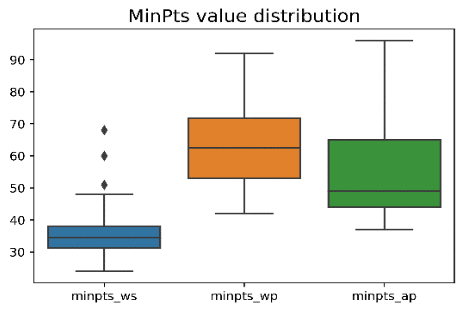
2) Clustering

After we have separated the object against its plane, before moving to the next step the segmented object must be individually separated from each other. Here, we present CVR-DBSCAN for the clustering and will be compared by a common algorithm like Euclidean clustering and novel DBSCAN.

CVR-DBSCAN generates Eps and $MinPts$ parameter values automatically, with a distribution that is shown in Fig. 15. As can be seen, the results of the given parameters give varying values, this happens because the entire dataset has a different arrangement from one another. As a stage that has high urgency, determining parameters for successful clustering is important because the data given is not always ideal and diverse, this is one of the advantages of CVR-DBSCAN.



(a)



(b)

Fig. 15. Distribution value of estimated parameters (a) Eps, (b) $MinPts$.

To analyze and validate the performance of clustering, we use qualitative analysis which compares the clustering result with the actual object that was seen in the real world. We compare the performance using a common algorithm like Euclidean clustering [10] and novel DBSCAN based on our previous research [4]. The visual comparison example can be seen in Fig. 16.

Unlike our algorithm where the parameters are given automatically and can adjust the existing data. But in computational time, Euclidean clustering got the fastest among others because of its simplicity in calculating and processing.As the result is well separated, overall, the algorithm can work extremely well with average accuracy up to 99% with the highest accuracy gained by DBSCAN (*Eps* = 0.01, *MinPts* = 50) and our Auto-DBSCAN. It can happen because all the objects in a scene are naturally separated so that the systems can easily group the object individually. As well as the stability of the accuracy given, the two algorithms have the highest stability among the others. It can be seen in Fig. 17(a), that both algorithms have a denser distribution of accuracy which indicates performance stability.

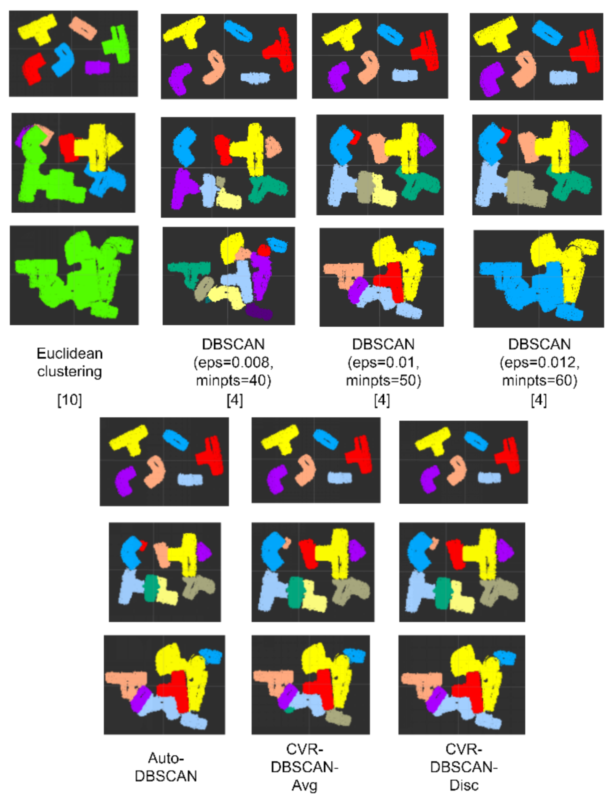
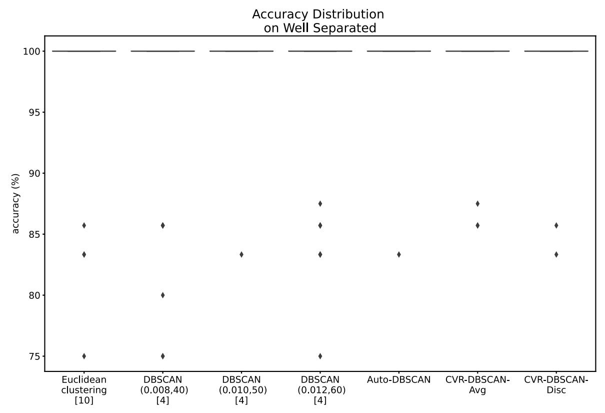
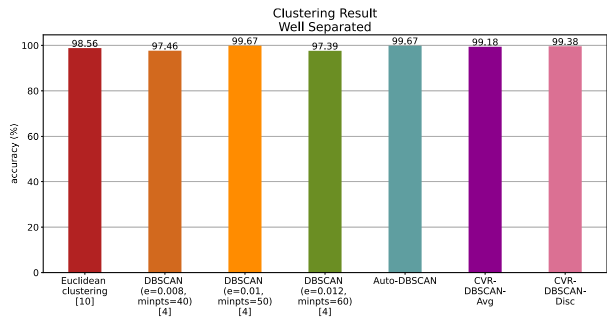

Euclidean clustering [10]
DBSCAN (eps=0.008, minpts=40) [4]
DBSCAN (eps=0.01, minpts=50) [4]
DBSCAN (eps=0.012, minpts=60) [4]
Auto-DBSCAN
CVR-DBSCAN-Avg
CVR-DBSCAN-Disc

Fig. 16. Visual comparison result on clustering.

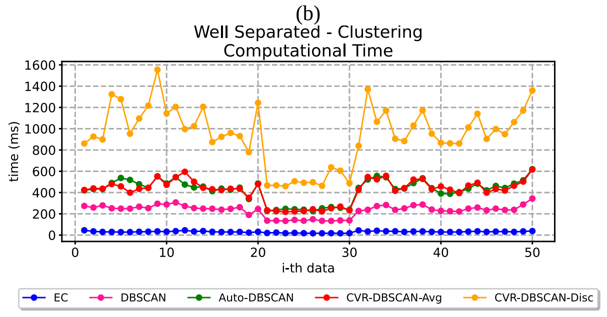
However, it must be remembered that by using DBSCAN, we must set the correct parameter values, because parameters other than (*Eps* = 0.01, *MinPts* = 50) have a more diverse distribution of accuracy values. This makes it difficult for users to tune manually because it takes a lot of time to find the best parameters.



(a)



(b)



(c)

Fig. 17. Clustering result on well separated (a) Accuracy distribution value, (b) Average accuracy, (c) Computational time.

Move to well piled placement, here the advantages of DBSCAN begin to appear. In piled objects, density calculation becomes important because the point cloud data is overlapping and makes it difficult if we use only the distance to consider the cluster like Euclidean clustering. The accuracy of that algorithm drops dramatically to 31.9% where the DBSCAN still can reach up to 75% accuracy. By using the right parameters (*Eps* = 0.008, *MinPts* = 40) DBSCAN can provide the highest accuracy to match CVR-DBSCAN-Avg with an average accuracy of 98.83%. However, the stability of accuracy provided by DBSCAN which can be seen in Fig. 18(a) at accuracy distribution is below Auto-DBSCAN and CVR-DBSCAN. With this result, our algorithm provides the advantage of not only high accuracy results but also provides accuracy stability on the entire test data because the parameters given can adjust and are not fixated on only one parameter value for the entire test data which may not be suitable although with a relatively higher computation time than the novel DBSCAN.

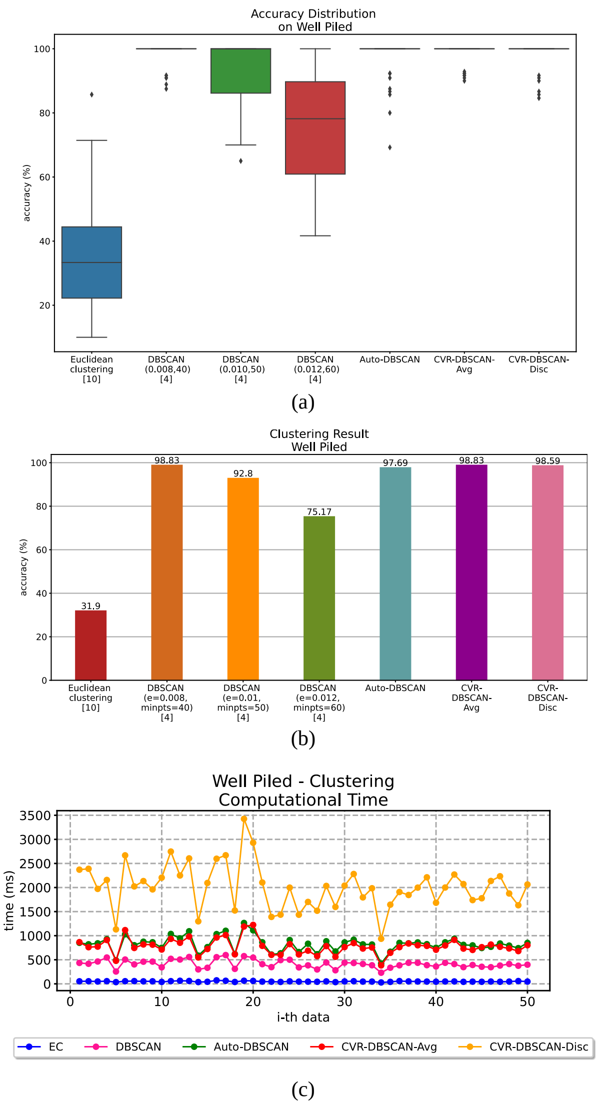


Fig. 18. Clustering result on well piled (a) Accuracy distribution value (b), Average accuracy, (c) Computational time.

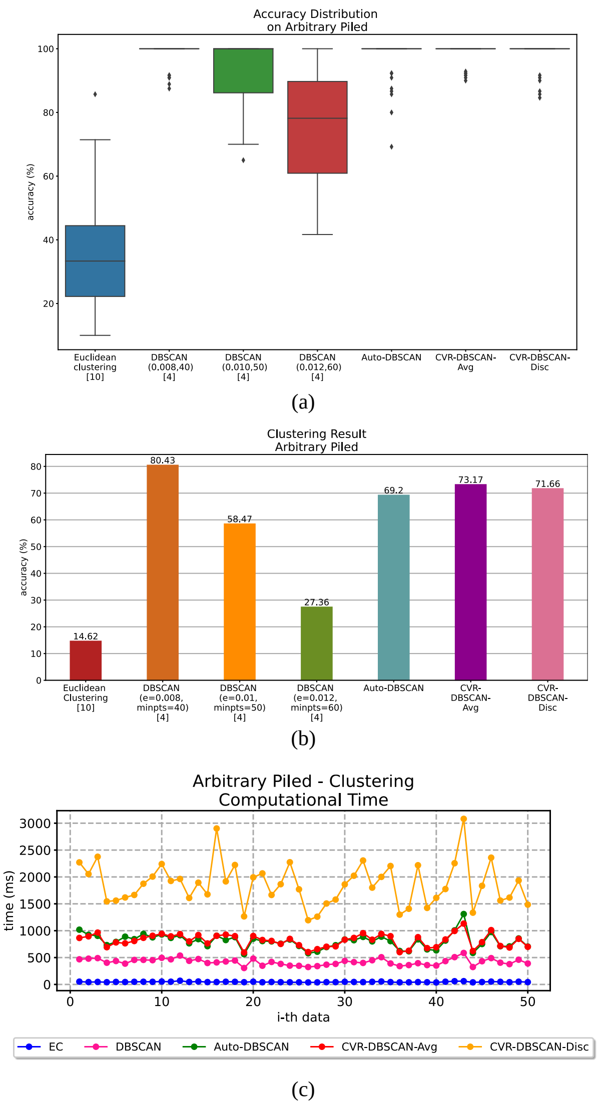


Fig. 19. Clustering result on arbitrary piled (a) Accuracy distribution value, (b) Average accuracy, (c) Computational time.

Finally, in arbitrary piled placement, CVR-DBSCAN continues to show its advantages. With the more complicated placement of randomly placed and stacked objects, CVR-DBSCAN-Avg can still handle well and resulting in an average accuracy of 73.17%. Compared to Euclidean clustering whose accuracy drops significantly at 14.62% and surprisingly DBSCAN (*Eps* = 0.008, *MinPts* = 40) gives the best average accuracy at 80.43% which is superior when compared to CVR-DBSCAN-Avg with the best average accuracy of 73.17%. This can occur due to the correct parameter selection on the test data where CVR-DBSCAN provides dynamic parameter estimation following the given data.

From stability, Fig. 19(a) gives the accuracy distribution on the test data where CVR-DBSCAN-Avg has fairly high stability where there is only one accuracy value that is far from the population. In DBSCAN, if the parameter selection is not correct, it will have an impact on accuracy and stability, in contrast to CVR-DBSCAN, our three algorithms provide consistent stable accuracy throughout the test data.

By performing curvature analysis which is useful for random and stacked objects that have a variety of curves.

IV. Conclusion

By using the improvement of the DBSCAN algorithm in terms of parameter estimation automation and curvature analysis, it is proven to provide improved clustering performance which can handle objects that have various placements. Although there is one case where the average accuracy of CVR-DBSCAN is still below that of the novel DBSCAN in arbitrary piled, in terms of accuracy stability in the overall data CVR DBSCAN provides good results. This is all made possible by applying the improvement on CVR DBSCAN. Also, in computational time, our improvement gives a competitive result that is close to novel DBSCAN which can be implemented in the real system. Since the algorithm is running well and suits several conditions in reality, what should be of concern is the computational time that should be improved as fast as possible. With combination of good accuracy and fast computational time makes the CVR-DBSCAN algorithm even better.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

A.R.P.: Methodology, Software, Writing—Original Draft; B.S.B.D.: Formal Analysis, Validation, Supervision, Writing—Review and Editing; D.M.S.: Investigation, Data Curation, Formal Analysis, Supervision, Writing—Review and Editing; D.P.: Conceptualization, Methodology, Supervision, Visualization, Writing—Review and Editing. All authors have read and agreed to the published version of the manuscript.

Funding

This research is supported by Center for Research and Innovation on Advanced Transportation Electrification (CReATE) Lab and Politeknik Elektronika Negeri Surabaya (PENS).

References

[1] S. Hwang, J. Park, J. Won, Y. Kwon, and Y. Kim, "Object detection for cargo unloading system based on fuzzy c means," *Computers, Materials and Continua*, vol. 71, no. 2, 2022.

[2] L. D. Hanh and H. B. Tu, "Computer vision for industrial robot in planar bin picking application," *Advances in Science, Technology and Engineering Systems*, vol. 5, no. 6, pp. 1244–1249, 2020.

[3] S. D'Avella, P. Tripicchio, and C. A. Avizzano, "A study on picking objects in cluttered environments: Exploiting depth features for a custom low-cost universal jamming gripper," *Robotics and Computer-Integrated Manufacturing*, vol. 63, 2020.

[4] A. R. Pratama, B. S. B. Dewantara, D. M. Sari, and D. Pramadihanto, "Density-based clustering for 3D stacked pipe object recognition using directly-given point cloud data on convolutional neural network," *EMITTER International Journal of Engineering Technology*, vol. 10, no. 1, pp. 153–169, 2022.

[5] T. Höfer, F. Shamsafar, N. Benbarka, and A. Zell, "Object detection and autoencoder-based 6D pose estimation for highly cluttered bin picking," in *Proc. 2021 IEEE International Conference on Image Processing (ICIP)*, 2021, pp. 704–708.

[6] D. M. Sari, A. R. Pratama, D. Pramadihanto, and B. S. Marta, "3D object detection based on point cloud data," *Inform: Jurnal Ilmiah Bidang Teknologi Informasi dan Komunikasi*, vol. 7, no. 1, pp. 59–66, 2022.

[7] M. Hansard, S. Lee, O. Choi, and R. Horaud, *Time of Flight Cameras: Principles, Methods, and Applications*, 2012.

[8] J. Guo, L. Fu, M. Jia, K. Wang, and S. Liu, "Fast and robust bin-picking system for densely piled industrial objects," in *Proc. 2020 Chinese Automation Congress*, 2020, pp. 2845–2850.

[9] A. R. Pratama, B. S. B. Dewantara, D. M. Sari, and D. Pramadihanto, "3D object pose estimation using local features based for industrial appliance," in *Proc. 2023 International Electronics Symposium (IES)*, pp. 440–445, 2023.

[10] R. B. Rusu, "Semantic 3D object maps for everyday manipulation in human living environments," *Künstl Intell*, vol. 24, no. 4, pp. 345–348, 2010.

[11] C. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *Proc. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 77–85, 2017.

[12] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep hierarchical feature learning on point sets in a metric space," in *Proc. the 31st International Conference on Neural Information Processing Systems*, 2017, pp. 5105–5114.

[13] M. Xu, Z. Zhou, and Y. Qiao, "Geometry sharing network for 3D point cloud classification and segmentation," in *Proc. the AAAI Conference on Artificial Intelligence*, 2020, vol. 34, no. 7, pp. 12500–12507.

[14] Z. Song, L. Zhao, and J. Zhou, "Learning hybrid semantic affinity for point cloud segmentation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 32, no. 7, pp. 4599–4612, 2022.

[15] Y. Xu, S. Arai, D. Liu, F. Lin, and K. Kosuge, "FPCC: Fast point cloud clustering-based instance segmentation for industrial bin-picking," *Neurocomputing*, vol. 494, pp. 255–268, 2022.

[16] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. the Second International Conference on Knowledge Discovery and Data Mining*, 1996, pp. 226–231.

[17] K. Sawant, "Adaptive methods for determining DBSCAN parameters," *International Journal of Innovative Science, Engineering & Technology*, vol. 1, no. 4, pp. 329–334, 2014.

[18] N. Rahmah and I. S. Sitanggang, "Determination of optimal Epsilon (Eps) value on DBSCAN Algorithm to clustering data on peatland hotspots in sumatra," *IOP Conference Series: Earth and Environmental Science*, 2016.

[19] A. Starczewski, P. Goetzen, and M. J. Er, "A new method for automatic determining of the DBSCAN parameters," *Journal of Artificial Intelligence and Soft Computing Research*, vol. 10, no. 3, pp. 209–221, 2020.

[20] N. Soni and A. Ganatra, "AGED (Automatic Generation of Eps for DBSCAN)," *International Journal of Computer Science and Information Security*, vol. 14, no. 5, 536, 2016.

[21] C. Wang, M. Ji, J. Wang, W. Wen, T. Li, and Y. Sun, "An improved DBSCAN method for LiDAR data segmentation with automatic Eps estimation," *Sensors*, vol. 19, no. 1, 2019.

[22] M. Naik Gaonkar and K. Sawant, "AutoEpsDBSCAN : DBSCAN with Eps automatic for large dataset," *International Journal on Advanced Computer Theory and Engineering*, vol. 2, no. 2, pp. 11-16, 2013.

[23] T. Czerniawski, M. Nahangi, S. Walbridge, and C. Haas, "Automated removal of planar clutter from 3D point clouds for improving industrial object recognition," in *Proc. the 33rd International Symposium on Automation and Robotics in Construction (ISARC)*, 2016, pp. 357–365.