Autonomous and Cost-effective Defect Detection System for Molded Pulp Products

Haochen Wang[†]
Shandong University
China
202220770@mail.sdu.edu.cn

Zhiwei Shi[†]
DeepCode Robotics
China
shizw@deepcode.cc

Yafei Qiao
Shandong University
China
202215120@mail.sdu.edu.cn

Fan Yang
DeepCode Robotics
China
fanyang@deepcode.cc

Yuzhe He
DeepCode Robotics
China
heyz@deepcode.cc

Dong Xuan
Shandong University
China
dongx2012@sdu.edu.cn

Wei Zhao
Shenzhen Institutes of Advanced Technology
China
weizhao86@outlook.com

ABSTRACT

Molded pulp products, such as dinnerware, containers, packaging boxes, etc., have gained increasing popularity due to their eco-friendly features. One critical step in their production process is detecting their defects. In this paper, we present an autonomous defect detection system for such products. In the system design, we face four challenges: first, molded pulp products come in various forms and sizes; second, defects are typically small and appear in different forms; third, detection must be fast enough to achieve desired high production rates; fourth, low cost is a key consideration in the pulp molding industry. To overcome these challenges, we have designed a defect detection system with an enhanced YOLOV5s + DeepLabV3Plus backbone and specific modules. Particularly, we design a lightweight YOLOV5s network with an attention mechanism to improve YOLOV5s' accuracy and speed, for roughly detecting and identifying the type and position of defects. We then utilize the DeepLabV3Plus segmentation model for precise detection. We deploy multiple cameras to handle the products of different sizes and forms, and design a spatial-information based method to eliminate the duplication in detection by different cameras. We have implemented our detection system in a real-world pulp molding manufacturing line, using cost-effective hardware. We have conducted extensive evaluation on our system, demonstrating that our system can meet molded pulp production requirements.

[†] The first two authors are co-primary authors.

ICCPS '23, May 9–12, 2023, San Antonio, TX, USA

ACM ISBN 979-8-4007-0036-1/23/05…$15.00
https://doi.org/10.1145/3576841.3585918

KEYWORDS

molded pulp, defect detection, YOLO, attention mechanism

1 INTRODUCTION

All nations in the world today are concerned about environmental protection, and many of them are actively promoting eco-friendly enterprises in their policies. Molded pulp products enjoy an extensive market as they are made of environmentally sustainable materials: plant fibers and water are the primary ingredients. Molded pulp is shaped into various products like bowls, plates and lunchboxes through a process that includes sterilizing, compression molding, and drying. These products are recyclable and biodegradable. However, raw materials inevitably harbor contaminants, which are the major cause of defects in molded pulp products. At present, defects of the products are mostly detected by human labor, which is inefficient, expensive, and does not fulfill the needs of large-scale detection, severely impeding the growth of the molded pulp business. To address this issue, we have developed an autonomous molded pulp defect detection system using machine vision. It can replace the manual detection completely by automation and remove unqualified items. There are several challenges in developing this defect detection system for molded pulp products:

- Molded pulp products come in a variety of shapes and sizes. As illustrated in Figure 1, the oval bowl is approximately 8cm tall, and the disc-shaped plate is approximately 1cm tall; the shape of the folded box causes uneven illumination in the camera images. The detection algorithm must be generalizable to all product dimensions.
- The presence of impurities in the raw materials during the production of molded pulp products can cause various types of defects on the product surface, such as hairlike defects, large and tiny speckles, as shown in Figure 2. Product qualification is defined clearly in the molded pulp industry. Qualified products cannot contain hairlike defects or speckles with

a diameter greater than 1mm (large speckle), and the number of speckles with a diameter less than 1mm (tiny speckle) cannot be greater than 3. Our system must detect both hairlike defects and speckles, while distinguishing between two speckle types: large and tiny.
- The defect detection system must run in real-time and maximize detection efficiency while ensuring detection accuracy.
- Low cost is always a requirement in industrial applications. Detection algorithms cannot be too demanding on hardware. We need to accomplish effective detection with moderate hardware complexity.

(a) **(b)**

Figure 1: Molded pulp product samples

(a) Hairlike defect **(b) Tiny speckle** **(c) Large speckle**

Figure 2: Three types of defects

The above four challenges collectively form a unique hurdle in developing a desired defect detection system for molded pulp products. To the best of our knowledge, there is no work published on such systems. We have developed a defect detection system for molded pulp products, which overcomes the above challenges. The detection system has already been used in real-world molded pulp production lines, as shown in Figure 3. In summary, our contributions are as follows:

- We have developed a deep-learning based defect detection system for molded pulp products. The system can detect and remove unqualified products automatically and effectively.
- We have designed a detection pipeline with improved YOLOV5s[1] + DeepLabV3Plus[2]. It can be used to detect defects on various molded pulp products. The pipeline begins with coarse detection of defect types and positions using improved YOLOV5s, then moves on to fine detection using DeepLabV3Plus to determine the precise positions of speckles. Finally it counts the number of defects on a single product with detection results of multiple cameras, using a spatial-information based detection duplication-elimination method.
- We have implemented the system on low-cost hardware. We have also conducted extensive experiments to evaluate its performance. Our evaluation data demonstrate that our system can achieve high defect detection accuracy under the required time limits.

Our system has a generalizable significance. Its modules can be readily extended to detect defects on other products, such as electronic parts. We have also uploaded a short anonymised video of our defect detection system on YouTube to demonstrate the status of its implementation (https://youtu.be/FNdvkcrpzWY).

2 SYSTEM DESIGN

In this section, we describe the design of our defect detection system for molded pulp products. The description has two parts: system architecture and defect detection algorithms.

2.1 System Architecture

The defect detection system is part of the whole molded pulp production line. We will first introduce the whole production line briefly and then discuss the detection system in details.

The whole production line is shown in Figure 12. It includes molding machine, trimming machine, defect detection system, packing machine and autonomous guided vehicles (AGVs in short). The molding machine compresses liquid pulp into products. The trimming machine then removes the excess areas around the products. The unqualified products are detected and removed by the defect detection system. Finally, qualified products are packed using the packing machine and unqualified products are thrown into a waste bin.The defect detection system is shown in Figure 13. Its input is one mold of products from the trimming machine in the production line. The detection system includes the visual detection components and the auxiliary mechanical parts. The visual detection component is the core of the detection system. Visual detection is conducted in two steps: external and internal detection, detecting defects on the external and internal surfaces of the products respectively. It has several external and internal detection stations, as well as an auxiliary lighting module. The external detection station contains multiple cameras for detecting outer surface defects. The internal detection stations consists of cameras facing the internal center of the products.

The auxiliary mechanical parts include a conveyor belt, suction cups, slide rails, and a packing machine, etc. The conveyor belt moves products of one mold into the detection stations. Products' movements from the conveyor belt to the external detection stations and then to the internal detection stations are conducted by slide rails and suction cups.

The whole detection process can be divided into six steps: Transfer 1 ($T1$), External detection (ED), Transfer 2 ($T2$), Internal detection (ID), Transfer 3 ($T3$), Discarding or Packing (DP). $T1$ takes the mold of products from the trimming machine to the external detection station, and ED conducts external detection and so on.

The molding machine in the production line finishes one mold in a certain time interval roughly in 10 to 20 seconds, depending on various factors such as the quality of molded pulp and the type of products etc. Naturally, the manufacturers require the components

(a) Molded pulp production line (b) Outside of defect detection system (c) Inside of defect detection system

Figure 3: Molded pulp production line and defect detection system

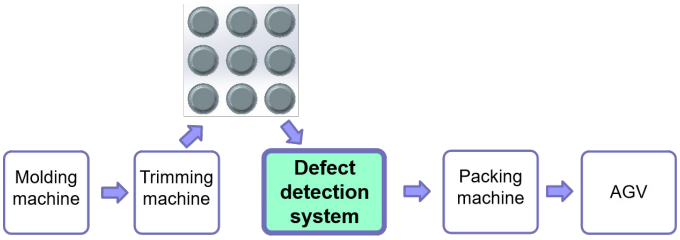


Figure 4: Molded pulp production line

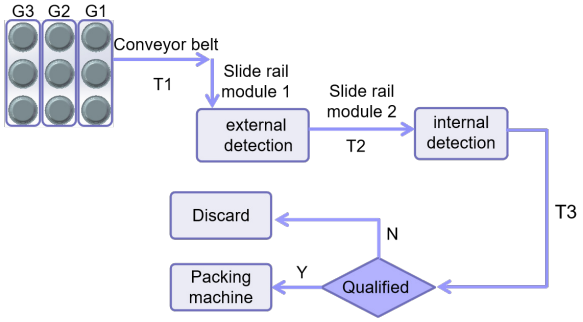
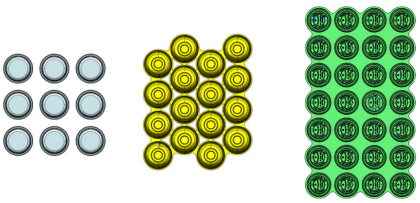


Figure 5: Defect detection system



(a) Plates (b) Bowls (c) Lids

Figure 6: One mold products

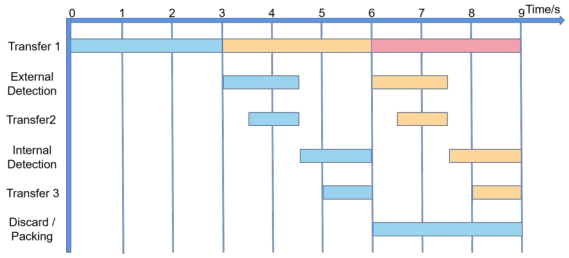


Figure 7: Parallel defect detection

following the molding machine in the production line would not delay the production process. In our working scenarios, the molding machine finishes one mold in every 18 seconds. $T1$ takes about 3 seconds, $T2$ and $T3$ takes about 1 second respectively.

Parallel Defect Detection: Recall that the input of the defect detection system is one (whole) mold from the trimming machine in the production line. One mold contains a number of pulp products, such as 9 plates, 16 bowls, or 28 lids, etc. as shown in Figure 6, according to different shapes and sizes. Products are firstly transported to the detection area. There are two potential solutions to detect the whole mold of products:

- One by one: we do defect detection on one molded pulp product each time. After the current product's defect detection is done, the next product is moved into the detection area. This type of serialized processing can greatly reduce the number of cameras. However, detecting defects one by one would make the overall system run very slowly.
- All together: we place all the products in one mold into the detection area at a time, and both external and internal detection are performed. This type of processing can significantly increase the overall speed, because we can complete the defect external and internal detection in parallel. However, this approach necessitates more cameras and high performance hardware due to parallel processing.
- Our solution: in order to achieve the fastest detection speed in the detection system with low cost, we make a compromise between

the above two naive solutions. The products in one mold are divided into several groups. While the first group is undergoing external detection and internal detection, the second group is on its way to the detection area. We design a parallel defect detection pipeline based on this idea. We achieve a balance between cost and speed by adjusting the number of products in each group that are detected at the same time.

Taking a mold containing 9 molded pulp products in Figure 6a as an example, we set the products in a 3x3 pattern. The mold is divided into three groups (G1, G2, G3), each with 3 products. The groups are processed in a parallel pipeline mode. The Gantt chart of the whole process is shown in Figure 7. Recall in our working scenario, $T1$ takes about 3 seconds, $T2$ and $T3$ takes about 1 second respectively. In the first three seconds, G1 is transferred ($T1$) to the detection area. In the second three seconds, while G2 is on the way to the detection area ($T1$), G1 begins to take ED, $T2$, ID and $T3$. In the next three seconds, while G3 is on the way to the detection area ($T1$), G1 is to be packed if it is qualified (DP) and G2 repeats the steps that G1 has done.

In this situation, the time left for ED and ID together is about 4 seconds. Similarly, for the mold in Figure 6b, the time left for ED and ID together is about 3 seconds, and for the mold in Figure 6c, the time left for ED and ID together is about 3 seconds. We will have to keep these time requirements in mind in the design of detection algorithms.

2.2 Detection Algorithms

The workflow of the algorithm is shown in Figure 8. The detection pipeline runs in three steps: coarse detection, fine detection and counting.

2.2.1 ***Coarse Detection.*** We use YOLOV5 as the baseline model for coarse detection with the following reasons:

- YOLOV5 performs Mosaic data augmentation on the original dataset by randomly scaling, cropping and stitching four images to get a brand new training sample. Mosaic augmentation enriches the dataset and improves the accuracy of small targets detection by random scaling of training samples.
- The SPPF module of YOLOV5 can convert feature maps of arbitrary size into feature vectors of fixed size. This can effectively eliminate image distortion caused by cropping and scaling, and solve the problem of repeated feature extraction of images caused by convolutional neural networks. SPPF greatly improves the speed of generating candidate bounding boxes and reduces costs.
- The C3 structure of YOLOV5 divides the input into two branches and performs convolutional operations separately before merging them together. This reduces the computational cost and allows the model to learn more features.

The above three factors are helpful to solve our problem (small defects, real-time).

There are four commonly used models of YOLOV5 according to their parameter numbers: YOLOV5s, YOLOV5m, YOLOV5l and YOLOV5x. We conduct experiments (see Chapter 4) and find that YOLOV5m, YOLOV5l and YOLOV5x have no significant improvement in accuracy compared to YOLOV5s, while the number of parameters increase substantially. Considering the real-time requirements of the detection system, we select the YOLOV5s model as our baseline model. The backbone of YOLOV5s is shown in Figure 9, and the CBS module consists of conv+bn+silu.

In order to meet the real-time requirement and to improve the detection accuracy of small defects, we make two improvements to YOLOV5s: (1) adding an attention module; (2) replacing parts of the network structure to make it lightweight.

- Attention module: in order to improve the accuracy of YOLOV5s for defect detection, especially for small defects, we add attention modules to the original YOLOV5s. Attention modules allow the model to better integrate contextual information and focus on the target to be detected. We insert SE[3], CBAM[4], ECA[5], SimAM[6], CA[7] and g^nConv[8] into the last layer of YOLOV5s backbone respectively, and find that the model with g^nConv achieves optimal accuracy with similar inference speed as the original model. The structure of the improved YOLOV5s is shown in Figure 10.

  Remark: traditional CNNs are translation invariant and localized, and lack the ability of global or long-range modeling. In contrast, recursive gated convolution (g^nConv performs higher-order spatial interactions through gated convolution and recursive design. The operation is highly flexible, customizable, and compatible with various convolutional variants. It extends second-order interactions in self-attentiveness to arbitrary orders without introducing significant additional computation. g^nConv can be used as a plug-and-play module to improve various visual transforms and convolution-based models.
- Lightweight model: in order to further improve the inference speed of the improved YOLOV5s, we try to replace the complex network modules in the improved YOLOV5s with network modules from two lightweight model, MobileNetV3[9] and PP-LCNet[10]. After tests, we find that the model using PP-LCNet is optimal in terms of accuracy and inference speed. The modified model structure is shown in Figure 11. We use DepthSepConv module in PP-LCNet to replace Conv module in YOLOV5s.

  Remark: PP-LCNet is a specific backbone network designed specifically for Intel CPU devices with its acceleration library MKLDNN. It uses the H-Swish function as its activation function, which removes the exponential operation and is faster with little impact on network accuracy. PP-LCNet also uses SE attention module, which can effectively improve network accuracy. PP-LCNet uses a 1x1 convolutional layer after GAP, so that the features after GAP do not directly go through the classification layer, but are fused firstly. The fused features are classified, which can greatly improve the accuracy without affecting the inference speed of the model.

2.2.2 ***Fine Detection.*** We find that the center of the defect and the corresponding prediction box of the coarse detection do not necessarily coincide. For example, in Figure **??**, the red dot is the center of the prediction box, while blue dot is the real center of the defect. If the center of the prediction box is used as the position of the defect, it will cause positioning error, which has a negative impact on the subsequent multi-view de-duplication algorithm in particular.

To solve this problem, we further refine the outputs of the coarse detection to improve the position accuracy of the defects. Considering the robustness and interpretability that the positioning algorithm should have, we use a semantic segmentation method

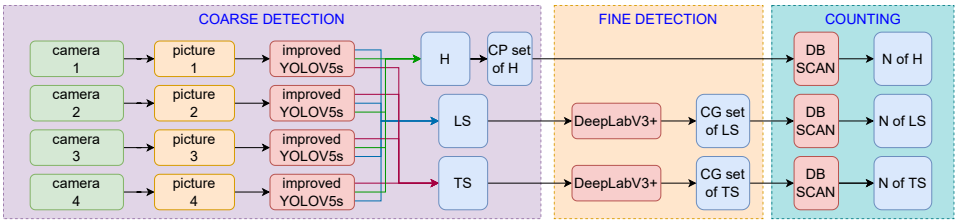


Figure 8: Workflow of detection algorithm
H: hairlike defects, LS: large speckles, TS: tiny speckles, N: number of defects.

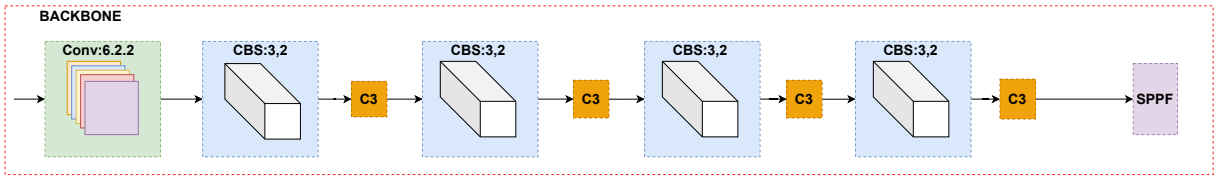


Figure 9: Backbone of YOLOV5s

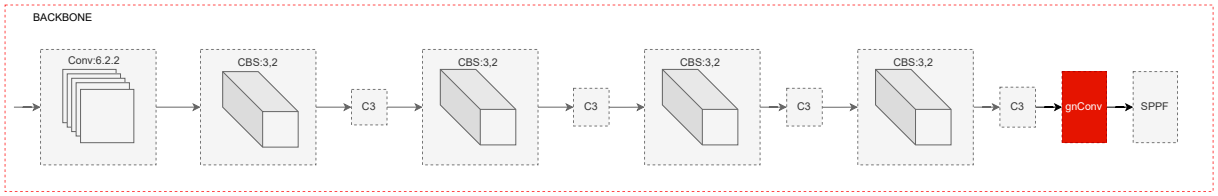


Figure 10: YOLOV5s backbone structure with gnConv

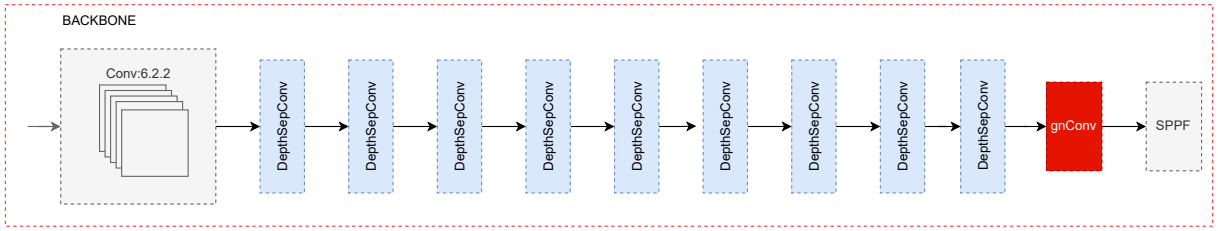


Figure 11: YOLOV5s backbone structure with gnConv and PP-LCNet

based on deep learning to obtain the mask of the defect at the original image resolution. We calculate the center of gravity(CG) of the mask and use it as the center of the defect. The workflow of the algorithm is shown in Figure **??**. We crop the original image according to the prediction box obtained with the improved YOLOV5s to get a sub-image containing only that defect, and use this sub-image as the input of the semantic segmentation model. After the processing of the semantic segmentation model, we get the mask of the defect, and a sample is shown in Figure 14. Finally, we calculate the CG of the mask and use it as the position of the defect. We only do the refined positioning for speckle defects, considering that hairlike defects are generally easy to handle. The semantic segmentation model we use here is DeepLabV3Plus, which is effective and widely used.

2.2.3 ***Counting***. In the previous "coarse + fine" detection, we already detect the type and position of defects on the molded pulp products. The next step is to count the number of defects on each product so that we can determine whether the product is qualified or not. In our defect detection system for molded pulp products, multiple cameras are placed around the product to collect surface information. The images from these cameras are fed into our defect detection model which detects the types and position of the defects in each image and then counts the number of defects. Since there are coverage overlaps between cameras, there may be some defects detected by multiple cameras at the same time. If we sum the number of defects detected by each camera, some defects are likely to be counted multiple times. Therefore, we need to de-duplicate the defects detected by multiple cameras.

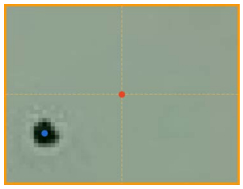


Figure 12: Position error effect

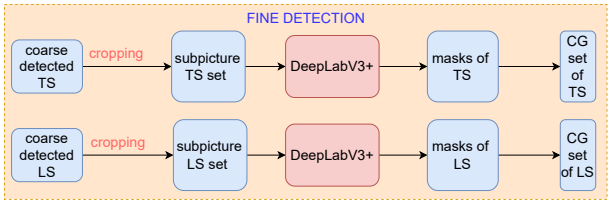


Figure 13: Fine detection algorithm workflow

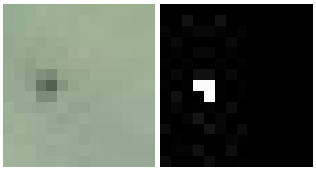
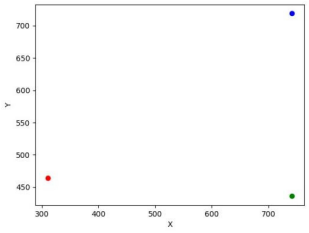
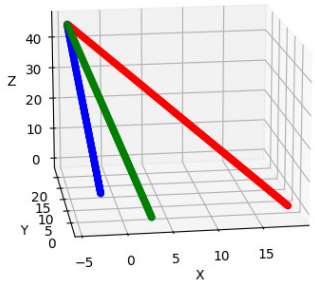


Figure 14: Semantic segmentation schematic



(a) Pixel coordinate system



(b) World coordinate system

Figure 15: Representation of defects in different coordinate systems

- Naive solution. We try to use DBSCAN to remove the defects that are repeatedly detected. DNSCAN is a density-based clustering algorithm which considers samples belonging to the same class are closely related to each other. DBSCAN is based on a set of neighborhoods to describe the closeness between samples, and the parameters ($Eps, MinPts$) are used to describe the closeness of the sample distribution in the neighborhoods. Eps represents the neighborhood distance threshold of one cluster's samples, and $MinPts$ represents the number threshold of one cluster's samples.

The defect detection algorithm allows us to obtain the position of each defect in the pixel coordinate system. This point in the pixel coordinate system can be converted to a line in the world coordinate system, which passes through the defect point and the optical center of the camera. In this way, each camera corresponds to a set of lines that pass through the detected defects. As shown in Figure 15, the left image shows the relative position of three defects detected by a camera in the pixel coordinate system, and the right image shows three lines corresponding to these defects in the world coordinate system, with the intersection point being the optical center of the camera.

Ideally, if multiple cameras detect the same defect, the defect point corresponds to a line under each camera, and these lines intersect at the defect point, the distance between these lines is 0. We use DBSCAN to perform a clustering of the distances between these lines to count the actual number of defects.

Directly using DBSCAN to do de-duplication may have the following two problems.

- Problem 1: in practice there is error in both the camera calibrations and defect detection results. The corresponding lines passing through one defect of multiple cameras may not intersect at the defect perfectly.
- Problem 2: if the distance between two defects is less than Eps, DBSCAN may cluster the two defects into one defect. As in Figure 17, camera A detects points $P1$ and $P2$. Camera B detects points $P2$ and $P3$. Then a total of four lines will be generated, which are $A1$, $A2$, $B2$ and $B3$. When clustering the line $A2$, the algorithm will calculate the distance between $A2$ and $B3$. If their distance is less than the EPS, DBSCAN will incorrectly cluster points $P2$ and $P3$ into one point.

	l0	l1	l2	.....	ln
l0	inf	inf	dis(l0,l2)	.....	dis(l0,ln)
l1	inf	inf	dis(l1,l2)	.....	dis(l1,ln)
l2	dis(l0,l2)	dis(l1,l2)	inf	.....	dis(l2,ln)
.....	.....	.....	.....	inf	.....
ln	dis(l0,ln)	dis(l1,ln)	dis(l2,ln)	.....	inf

Figure 16: distance table

- Our solution: for Problem 1, we consider that two lines intersect if their distance is less than the threshold, and the intersection point is the midpoint of their common vertical line. We also need to estimate whether the intersection point is on the product's surface. Here we set a threshold range for the z-coordinate value of the intersection point. Only if the z-coordinate of the intersection point is within the range, we consider the point to be a valid defect. To increase the DBSCAN clustering speed, we set the distance between these two lines to infinity. Finally, we can get a table of distances, as shown

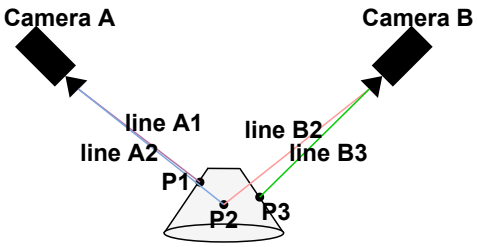


Figure 17: A special case of de-duplication

in Figure 16, where lines labeled with the same color indicates that they correspond to the same camera. Then we use the DBSCAN to cluster this distance table to exclude the defect repeatedly detected and get the actual number of defects.

For Problem 2, we make some improvements to DBSCAN. When finding the cluster of a line Lp in distance table, we only put the lines of every other cameras with the closest distance to Lp into neighboring line set N. Then we perform clustering, which can effectively avoid misclassifying two points that are close to each other as the same point.

The improved DBSCAN is presented in Algorithm 1. The algorithm for determining the set of neighborhood lines N is presented in Algorithm 2 (please refer to Appendix A for the algorithms).

3 SYSTEM IMPLEMENTATION

We have implemented the defect detection system based on the design presented in Section 2. As described in Section 1, the system has been used in real-world molded pulp production line (Figure 3).

The mechanical hardware of the system mainly includes conveyor belt, pneumatic suction cup, slide rail, cylinder, motor, and packaging machine, etc. It supports multiple external and internal detection stations. Each external defect detection station (see Figure 18) is covered with four downward-sloping cameras, which are evenly distributed above and around the pallet, and the optical axis of each camera is $-45°$ to the horizontal plane. The internal defect detection station (see Figure 19) is covered by one camera facing the center of the molded pulp product. The camera resolution is 1280x720, with a 4mm-focal-length lens. The main control unit of the mechanical hardware is an STM32H743 microcontroller with 400Mhz frequency and a dual 1Mbps rate CAN bus to control motors and air cylinders to transfer the products.

The system is equipped with an i5 processor and 16G memory. It mainly runs the detection software. The detection software is programmed in C++. It is multi-threaded and parallel, containing a main thread, multiple image acquisition threads, multiple image processing threads, and a communication thread. The main thread schedules all the functions. The image acquisition threads collect images from each camera. The image processing threads detect the defects of each image and count defects. The communication thread communicates with the underlying microcontroller, receiving the detection signal from the microcontroller and sending the results of whether the product is qualified or not to the actuator of the physical module to make the corresponding action.

4 EVALUATION

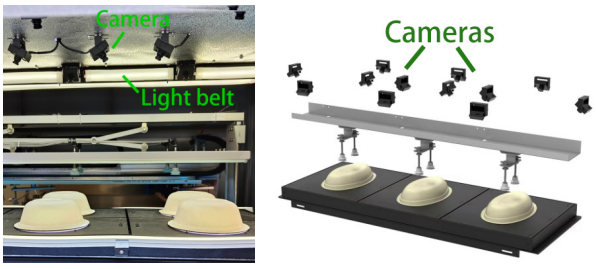
We build a test experimental environment that recreates the real situation inside the defect detection system in order to evaluate the performance of our design algorithms. In this test experiment environment, we collected a total of 1685 images, including 553 plates, 546 bowls and 586 lunchboxes. The defects of all molded pulp products can be classified into 3 classes, namely hairlike defects, tiny speckle defects and large speckle defects.

In coarse detection, we use 845 images (plates: 273, bowls: 266, and lunchboxes: 306) as training data (748 as training set, 84 as validation set, and 13 images without defects), and 120 images (40 each for plates, bowls, and lunch boxes) as test data. In fine detection, we used 300 real box cropped image data (extracted from coarse detection training data) as training data and 100 real box cropped image data (extracted from coarse detection test data) to test fine detection. In the part of counting and overall method testing, we use 720 images (60 lunchboxes×4, 60 plates×4 and 60 bowls×4) as test data for our proposed method.

We use evaluation metrics such as $recall$, $precision$, mAP, and $FPS(frames per second)$ to judge the performance of the various models we choose.The formulas are as follows: $recall = \frac{TP}{TP+FN}$, $precision = \frac{TP}{TP+FP}$, $AP = \int_0^1 PdR$, $mAP = \frac{\sum_{i=1}^{N}}{AP_i}$. TP denotes the number of actual positive samples that are predicted to be positive; FN denotes the number of actual positive samples that are predicted to be negative; FP denotes the number of actual negative samples that are predicted to be positive; $Precision$ measures the model's ability to predict accurately. AP represents the area of the $PR-curve$ composed on the x-axis. Each class corresponds to an AP value. mAP is the average of the AP values of all classes. N represents the number of classes.

4.1 Coarse Detection

- Comparison of four YOLOV5 models: Figure 20 shows the mAP value of YOLOV5s, YOLOV5m, YOLOV5l and YOLOV5x in the detection of hairlike, tiny speckle and large speckle defects respectively, as well as mAP and FPS of each model. The four models show a sequential increase in the number of model parameters, which are related to the spatial and computing complexity. The mAP value of all the four models for hairlike defects can reach above 0.979, while the detection accuracy for tiny speckles is the lowest, with the highest mAP being 0.734 with the YOLOV5x model. Among them, YOLOV5s has the lowest mAP of 0.843 and YOLOV5x has the highest mAP value of 0.858. The accuracy of the m, l, and x models is marginally better than that of the s model, but their performance in FPS is much worse than the former three models. Considering the above observations and the high real-time requirements of our defect detection system, we decide to use YOLOV5s.
- Comparison of YOLOV5s after adding various attention module:Figure 21 shows the mAP values of YOLOV5s after adding various attention mechanisms in the detection of three types of defects, as well as the overall mAP and FPS values of the models. As can be seen from Figure 21, the addition of the attention module generally improves the mAP of the model, except for the addition of the CA and SE model, which result in a decrease of 0.05 and 0.017 compared to the original YOLOV5s. The g^nConv model significantly improves the detection accuracy of tiny speckle defects with the highest mAP,

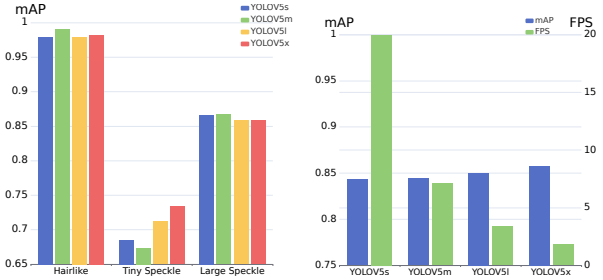


(a) Structure (b) Camera setup

Figure 18: External detection station

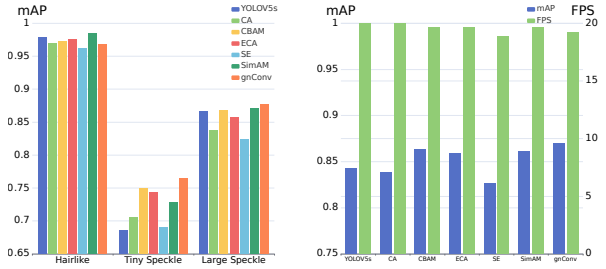
(a) Structure (b) Camera setup

Figure 19: Internal detection station



(a) mAP of different defects (b) mAP and FPS of each model

Figure 20: YOLO comparison



(a) mAP of different defects (b) mAP and FPS of each model

Figure 21: Attention comparison

which increases by 0.027 compared to the original YOLOV5s's *mAP*. The addition of the g^nConv attention module does not overly slow down the inference speed of the model, whose FPS is about 19, note that the FPS of YOLOV5s without the attention module is about 20. Based on the above observations, we choose to add the g^nConv attention module to YOLOV5s.

- Comparison of YOLOV5s using various lightweight methods after adding the attention mechanism: we try two lightweight methods, mobile3s and PP-LCNet, respectively. Figure 22 shows the *mAP* values of YOLOV5s with the addition of g^nConv attention module and two lightweighting methods in the detection of three defects, and the *mAP* values of each model. From the figure, we can observe that the *mAP* values of the model for hairlike defects is reduced under the PP-LCNet lightweighting method from 0.979 with the original YOLOV5s to 0.932 with the addition of PP-LCNet. For the detection of speckle defects, the model using PP-LCNet has a higher *mAP* and *FPS*, which are 0.849 and 35.714 respectively, compared to the other two models. Based on the above observations, we choose to perform PP-LCNet lightweighting on the YOLOV5s model with the addition of the g^nConv attention mechanism.

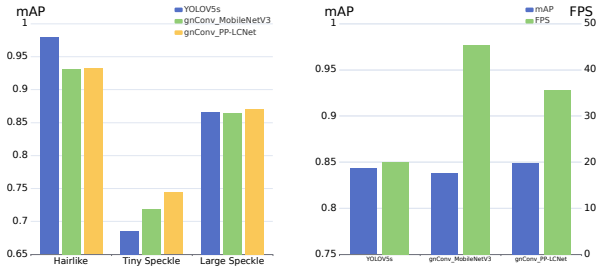
4.2 Fine Detection

To contrast the accuracy of the model with and without fine detection, we do some experiments. We detect and locate the defects in both cases, and then calculate the average of the Euclidean distance between the predicted defects and the corresponding groundtruths in the pixel coordinate system. The results are shown in Figure 23a. We can see that without DeepLabV3Plus, the mean value of the Euclidean distance between the detected defects and the corresponding groundtruths is about 2.2 pixels, while with DeepLabV3Plus, the mean distance is reduced by nearly 0.9. DeepLabV3Plus takes about 25ms to process an image on the CPU. Considering that the input images are independent of each other in fine detection, we can process multiple images simultaneously in parallel. The image processing speed of DeepLabV3Plus can meet our requirements. Based on these observations, we choose to add DeepLabV3Plus after coarse detection to further pinpoint the defects position.

4.3 Counting

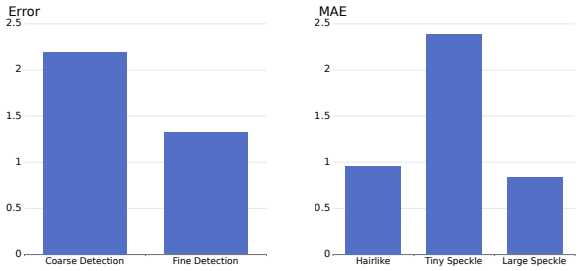
We calculate the mean absolute error(*MAE*) between the results obtained by our method and the ground truth. The calculation formula is $MAE = \frac{\sum_{i=1}^{N} abs(err_i)}{N}$. where N represents the number of molded pulp products. As can be seen from Figure 23b, our model has a small error in counting hairlike defects and large speckles, with a mean error within 1. The detection accuracy for tiny speckles is significantly lower than the other two classes, reaching about 2.387. This is in part because small object recognition is challenging and manual annotation is more prone to errors on such defects.

Recall that our defect detection has two parts: external detection and internal detection. In our system implementation, the external detection part has 12 cameras while the internal one has 6 cameras. According to the above performance data, the external detection roughly takes 1.2 seconds and the internal detection roughly takes



(a) mAP of different defects **(b) mAP and FPS of each model**

Figure 22: Lightweight comparison



(a) Position error **(b) Mean absolute error**

Figure 23: Counting effect comparison

0.6 seconds. Our design meets the molded pulp production real-time requirements discussed in Section 2.1.

5 RELATED WORK

Our defect detection system mainly incorporates three technologies: automated defect detection, lightweight neural networks, and multi-view detection. We discuss their related work below.

In recent years, industrial manufacture is gradually developing in intelligence and automation. Autonomous detection systems based on machine vision have become the mainstream detection solutions. Luo et al. developed an autonomous optical inspection (AOI) system for surface defects of hot-rolled flat steel[11]. Li et al. designed an automatic defect detection system based on AOI technology for PCB[12]. It can detect open-circuit, short-circuit as well as gaps, voids, scratches, and other defects. The fusion of machine vision and deep learning has been widely applied to defect detection in textiles[13, 14], metallic materials[15], plastic products[16], etc. Tao et al. proposed a multi-task aggregation neural network for defect detection in spring wire socket wires[17]. Cheng et al. developed an automatic method based on faster R-CNN for sewer defect detection[18]. These systems usually deploy one camera to acquire the information of detected products. However, molded pulp products can be large, and only one camera can't fully cover the products. Li et al. improved YOLO[19] with 27 convolutional layers, providing an end-to-end solution for surface defect detection of steel strips. Zhang et al. proposed an improved YOLOV5 algorithm for solar cell defect detection[20]. Huang et al. proposed a defect detection system for ceramic substrates[21]. Their system can only detect relatively large defects and cannot locate the defects accurately.

To enhance YOLO model structure, attention modules and lightweight networks are frequently used. Attention modules are widely used in various vision models to improve their performances. SE proposed by Hu et al. automatically acquires the importance of each feature channel by learning. CBAM proposed by Woo et al. integrates the Channel Attention Module (CAM) and the Spatial Attention Module (SAM). ECA proposed by Wang et al. implements a local cross-channel interaction strategy without dimensionality reduction by 1D convolutions. SimAM proposed by Yang et al. derives 3D attention weights for feature maps without additional parameters. CA proposed by Hou et al. embeds the location. g^nConv proposed by Rao et al. performs higher-order spatial interactions by gated convolution and recursive design, which is highly flexible, customizable, and compatible with various convolution variables. There are two ways to lightweight the model: novel structure design, such as MobileNetV3, ShuffleNet[22], and PP-LCNet, and model compression, including knowledge distillation, pruning, quantization, and low-rank decomposition[23, 24]. We test two network structures, MobileNetV3 and PP-LCNet, in our experiments.

One common method to process multi-view images is image stitching. Zhang et al. proposed a pyramid ORB-based endoscopic image stitching method to obtain a wider field of view and clearer image[25]. Megha et al. proposed a satellite image stitching technique based on accelerated robust feature algorithm[26]. Zhang et al. proposed a SURF-based image stitching method for constructing remote sensing image mosaics. The above methods basically perform image fusion based on features[27]. However, the feature matching methods are usually computationally expensive in the feature extraction process. In our work, we use spatial geometric information to achieve de-duplication.

6 DISCUSSION

In this section, we discuss the limitation of our detection system and our future work.

Full coverage of the target object can be achieved with a reasonable cameras location and orientation. In the multi-camera system, there is also a clearest image acquisition area. If the target object exists in this area, we can attain the clearest image. The cameras are arranged rather simply in our current work and intricate spatial interactions are not taken into account. Later, we'll think about how to employ the fewest cameras possible to fully cover the target object[28] and achieve the finest image clarity while attempting to cut expenses and boost defect detection precision.

Our models are trained based on labeled data. After the model is deployed on the production line, a lot of unlabeled data will be continuously collected by the cameras. The accuracy of the defect detection model can be improved if the data can be utilized efficiently. We therefore consider to add a semi-supervised learning strategy, such as a pseudo-label-based semi-supervised detection method[29], into the current defect detection system in our future work. Using labeled data to train a model, this model then can predict unlabeled data. After NMS reduction of redundant frames, a pseudo-label is selected for the data using a threshold value. The model continues to be trained using this pseudo-label.

The de-duplication approach now in use is purely based on the location of the defects in space, which could lead to de-duplication being inaccurate as a result of mistakes made during the camera calibration procedure. In the future, we'll try to incorporate the feature point matching technique based on the original research. The feature points are able to maintain the same features even when the camera is moved and rotated. They not only consider the position information, but also the appearance information of the pixels around the feature points. The feature points do not change as the image is zoomed in or out or rotated, and are insensitive to noise and illumination.

Currently our detection system can detect three types of molded pulp products such as bowls, plates and lunchboxes. In reality, it also needs to detect products such as cup lids, different size round plates, hot dog boxes, etc. In the future we will collect dataset for other types of products and train our model to be able to handle a wider variety of products.

7 CONCLUSION

In this paper, we introduced a defect detection system for molded pulp products. The system uses the improved YOLOV5s with DeepLabV3Plus for coarse and fine defect detection and a DBSCAN-based defect de-duplication method for defect counting under multi-view conditions. The system is fully autonomous without the need of human involvement in defect detection. It is also cost effective only using low price cameras and computing device. The system has been tested in a real-world production environment. The evaluation data demonstrates that our system can meet molded pulp production requirements in detection accuracy and time.

Our current system still has some limitations. In the near future, we plan to further improve the system in various directions such as optimizing visual camera deployment, enhancing our defect detection algorithm by incorporating semi-supervised learning etc.

ACKNOWLEDGEMENT

We sincerely thank anonymous reviewers and the area chair for their valuable comments. We acknowledge the support of Fan Yang's colleagues such as Wenjie Wang, Sixian Ye and Xin Xi on this pulp molding project at DeepCode and its collaborative companies.

REFERENCES

[1] Glenn Jocher. Yolov5 by ultralytics. https://github.com/ultralytics/yolov5, 2020.
[2] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818, 2018.
[3] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018.
[4] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. Cbam: Convolutional block attention module. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018.
[5] Pengfei Zhu Peihua Li Wangmeng Zuo Qilong Wang, Banggu Wu and Qinghua Hu. Eca-net: Efficient channel attention for deep convolutional neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.
[6] Lingxiao Yang, Ru-Yuan Zhang, Lida Li, and Xiaohua Xie. Simam: A simple, parameter-free attention module for convolutional neural networks. In *International conference on machine learning*, pages 11863–11874. PMLR, 2021.
[7] Qibin Hou, Daquan Zhou, and Jiashi Feng. Coordinate attention for efficient mobile network design. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13713–13722, 2021.
[8] Yongming Rao, Wenliang Zhao, Yansong Tang, Jie Zhou, Ser-Nam Lim, and Jiwen Lu. Hornet: Efficient high-order spatial interactions with recursive gated convolutions. *arXiv preprint arXiv:2207.14284*, 2022.
[9] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1314–1324, 2019.
[10] Cheng Cui, Tingquan Gao, Shengyu Wei, Yuning Du, Ruoyu Guo, Shuilong Dong, Bin Lu, Ying Zhou, Xueying Lv, Qiwen Liu, et al. Pp-lcnet: A lightweight cpu convolutional neural network. *arXiv preprint arXiv:2109.15099*, 2021.
[11] Qiwu Luo and Yigang He. A cost-effective and automatic surface defect inspection system for hot-rolled flat steel. *Robotics and Computer-Integrated Manufacturing*, 38:16–30, 2016.
[12] Ziyin Li and Qi Yang. System design for pcb defects detection based on aoi technology. In *2011 4th International Congress on Image and Signal Processing*, volume 4, pages 1988–1991. IEEE, 2011.
[13] Jiaqi Zhang, Junfeng Jing, Pengwen Lu, and Shaojun Song. Improved mobilenetv2-ssdlite for automatic fabric defect detection system based on cloud-edge computing. *Measurement*, 201:111665, 2022.
[14] Aqsa Rasheed, Bushra Zafar, Amina Rasheed, Nouman Ali, Muhammad Sajid, Saadat Hanif Dar, Usman Habib, Tehmina Shehryar, and Muhammad Tariq Mahmood. Fabric defect detection using computer vision techniques: a comprehensive review. *Mathematical Problems in Engineering*, 2020, 2020.
[15] Yiming Xu, Kai Zhang, and Li Wang. Metal surface defect detection using modified yolo. *Algorithms*, 14(9):257, 2021.
[16] Yi-Fan Chen, Fu-Sheng Yang, Eugene Su, and Chao-Ching Ho. Automatic defect detection system based on deep convolutional neural networks. In *2019 International Conference on Engineering, Science, and Industrial Applications (ICESI)*, pages 1–4. IEEE, 2019.
[17] Xian Tao, Zihao Wang, Zhengtao Zhang, Dapeng Zhang, De Xu, Xinyi Gong, and Lei Zhang. Wire defect recognition of spring-wire socket using multitask convolutional neural networks. *IEEE Transactions on Components, Packaging and Manufacturing Technology*, 8(4):689–698, 2018.
[18] Jack CP Cheng and Mingzhu Wang. Automated detection of sewer pipe defects in closed-circuit television images using deep learning techniques. *Automation in Construction*, 95:155–171, 2018.
[19] Jiangyun Li, Zhenfeng Su, Jiahui Geng, and Yixin Yin. Real-time detection of steel strip surface defects based on improved yolo detection network. *IFAC-PapersOnLine*, 51(21):76–81, 2018.
[20] Meng Zhang and Liju Yin. Solar cell surface defect detection based on improved yolo v5. *IEEE Access*, 10:80804–80815, 2022.
[21] Chien-Yi Huang, I-Chen Lin, and Yuan-Lien Liu. Applying deep learning to construct a defect detection system for ceramic substrates. *Applied Sciences*, 12(5):2269, 2022.
[22] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6848–6856, 2018.
[23] Yu Cheng, Duo Wang, Pan Zhou, and Tao Zhang. A survey of model compression and acceleration for deep neural networks. *arXiv preprint arXiv:1710.09282*, 2017.
[24] Jian Cheng, Pei-song Wang, Gang Li, Qing-hao Hu, and Han-qing Lu. Recent advances in efficient computation of deep convolutional neural networks. *Frontiers of Information Technology & Electronic Engineering*, 19(1):64–77, 2018.
[25] Ziyan Zhang, Lixiao Wang, Wenfeng Zheng, Lirong Yin, Rongrong Hu, and Bo Yang. Endoscope image mosaic based on pyramid orb. *Biomedical Signal Processing and Control*, 71:103261, 2022.
[26] V Megha and KK Rajkumar. Automatic satellite image stitching based on speeded up robust feature. In *2021 International Conference on Artificial Intelligence and Machine Vision (AIMV)*, pages 1–6. IEEE, 2021.
[27] Tian Zhang, Rui Zhao, and Zhongsheng Chen. Application of migration image registration algorithm based on improved surf in remote sensing image mosaic. *IEEE Access*, 8:163637–163645, 2020.
[28] Zuoming Yu, Fan Yang, Jin Teng, Adam C Champion, and Dong Xuan. Local face-view barrier coverage in camera sensor networks. In *2015 IEEE Conference on Computer Communications (INFOCOM)*, pages 684–692. IEEE, 2015.
[29] Kihyuk Sohn, Zizhao Zhang, Chun-Liang Li, Han Zhang, Chen-Yu Lee, and Tomas Pfister. A simple semi-supervised learning framework for object detection. *arXiv preprint arXiv:2005.04757*, 2020.

A COUNTING ALGORITHMS

At the beginning of the algorithm we mark all the lines in the set of *vecs* as unvisited. One unvisited line p is randomly selected and marked as visited. Then the neighboring line set N of line p is found according to Algorithm 2. If the number of lines in N exceeds the $MinPts$ threshold, we consider line p to be a core line and mark p as a new cluster. After that we iterate through each neighboring line pi of p. If pi is unvisited, mark it as visited and determine whether pi is a core point. If so, merging the neighboring lines of pi into p. If pi does not belong to any cluster, group pi into the cluster of p. To determine the set of neighboring lines N, we maintain a neighborhood line set *neighborhood_list*, a list of camera ids, *idx_list*, corresponding to each line in *neighborhood_list* and a list of distances *dist_list* between each line in *neighborhood_list* and core object p. For any line l in *vecs*, if its corresponding camera id is not in *idx_list*, we will add the corresponding id to *idx_list* and the distance between l and p to *dist_list*. If the camera id corresponding to l is already in *idx_list*, there is a corresponding distance in *dist_list*. If this distance is less than the distance between l and p, we will not process it. Otherwise, we will update the distance between l and p with this distance. After traversing each line in *vecs*, we can get the set N of p's neighborhood lines.

Example: Taking the case in Figure 17 as an example, and assuming that there are two cameras and *dist_table* contains four lines: $A1$, $A2$, $B2$, $B3$, and the distance between $A1$ and $B2$ and $A1$ and $B3$ are greater than *Eps*, we use the improved DBSCAN to cluster *dist_table*. Firstly, we randomly select one line, like $A2$. When we try to find the neighboring lines of $A2$, the closest line between camera B and A is $B2$. We put the line $B2$ into the set N, and $B3$ will not appear in N. Then $A2$ and $B2$ are clustered into one cluster. Then we randomly select one line, like $A1$. According to the assumption, there is no neighboring lines for $A1$, then $A1$ is a new cluster itself. At this time, only $B3$ is not visited. and there is no neighboring lines for $B3$, then $B3$ is classified into a new cluster. In this way, we get the real number of defects.

Algorithm 1: DBSCAN-based defect de-duplication method

```
input          : vecs, cam_points_len, Eps=1.6, MinPts=1
output         : C, k
Initialization : unvisited_list, visited_list, dist_table,N, M
  while len(unvisited_list)>0 do
    p=random.choice(unvisited_list)
    unvisited_list.remove(p)
    visited_list.append(p)
    N=Neighborhood_Object(vecs,dist_table,Eps, p)
    if len(N)>=MinPts then
      C[P]=K
      k=k+1
      for pi in N do
        if pi in unvisited_list then
          unvisited_list.remove(pi)
          visited_list.append(pi)
          M=Neighborhood_Object(vecs,dist_table, Eps, pi)
          for t in M do
            if t not in N then
              N.append(t)
            end if
          end for
        end if
        if C[pi]==-1 then
          C[pi]=k
        end if
      end for
    else
      C[p]=-1
    end if
  end while
```

Algorithm 2: Neighborhood_Object

```
input          : vecs, dist_table, Eps, p
output         : neighbor_list
Initialization : idx_list,dist_list
  for l in vecs do
    cur_dis=dist_table.dist(l,p)
    if cur_dis<=Eps then
      cur_idx=dist_table.cam_points_len_list[l]
      if cur_idx in idx_list then
        old_dis_idx=idx_list.index(cur_idx)
        if cur_dis<dist_list[old_dis_idx]] then
          dist_list[old_dis_idx]=cur_dis
          neighbor_list[old_dis_idx]=l
        end if
      else
        neighbor_list.append[l]
        idx_list.append(cur_idx)
        dist_list.append(cur_dis)
      end if
    end if
  end for
```