Sensors and Materials, Vol. 36, No. 3 (2024) 1003–1018
MYU Tokyo

S & M 3577

Vision-based Robotic Arm Control for Screwdriver Bit Placement Tasks

Cheng-Jian Lin,[1] Pei-Jung Lin,[2] and Chi-Huang Shih[1*]

[1]Department of Computer Science and Information Engineering, National Chin-Yi University of Technology, Taichung 411, Taiwan
[2]Department of Information Engineering and Computer Science, Feng Chia University, Taichung 407, Taiwan

(Received October 30, 2023; accepted March 8, 2024)

Keywords: robotic arm, object placement, vision detection, deep learning

Robotic arms are widely used in the automation industry to package and deliver classified objects. When the products are small objects with very similar shapes, such as screwdriver bits with slightly different threads, pointed tips, and thicknesses, object selection and assembly often lead to misjudgment. We have developed a practical robotic arm control system based on vision detection techniques for screwdriver bits' placement. In addition to effectiveness, easy deployment and high flexibility in the field are also taken into account. The vision-based system consists of four processing stages in the following order: world coordinate conversion from image pixel coordinates, object detection, edge detection, and object orientation. In the first stage, a manual two-point marking method is proposed to easily configure the coordinate conversion for robot operating system (ROS)-based manipulators. For the following stages, we focus on the fine integration of state-of-the-art methods for the technical feasibility of the screwdriver bit placement. Such integration includes the selection between object detection methods and the data flow control among system stages. The experimental results show that (1) in detecting screwdriver bits, You Only Look Once (YOLO) v4 outperforms YOLOv7 and Single Shot MultiBox Detector at an accuracy rate of 99.51%; (2) in the edge detection, the object detection output can better illustrate the object contour than a whole image, achieving a mean absolute error of 0.86% in estimating the object angle; and (3) a successful real-time replacement rate of 96% is achieved for 12 screwdriver bits randomly scattered on a conveyor belt.

1. Introduction

As Industry 4.0 and IoT accelerate their integration to realize the smart factory concept, modern robots are moving towards a future of fully automated and autonomous operations. In the present day, a large amount of human resources is required for object sorting and operation processes, and the manufacturing industry has been striving to control production costs, including labor salary, machine maintenance cost, and the cost of materials. When robotic arms

*Corresponding author: e-mail: chshih@ncut.edu.tw
https://doi.org/10.18494/SAM4739

ISSN 0914-4935

assist the assembly line, overall productivity can be improved. Although the robotic arm can perform repetitive pick-and-place actions for a long time, what the robotic arm can do is still limited without the assistance of a vision system or sensing techniques. For instance, as multiple objects are scattered on the conveyor belt, obtaining the position, posture, and type of target object for object placement is difficult. Traditional machine vision is regarded as one of the solutions to full automation; problems such as light exposure/shadows/reflections, camera position, complex geometry, and surface roughness of objects can increase the complexity of object recognition on the conveyor belt. Consequently, the general machine vision may not support a precise classification of diverse materials and tools in the assembly line. From the abovementioned difficulties, object recognition and placement for vision-based robotic arm automation remain challenging.

The first step in applying machine vision to robotic-arm-based applications is to deploy a camera in either an eye-in-hand manner or an eye-to-hand manner. In the eye-in-hand deployment method, a camera is installed at the end of the robotic arm, and the object's position can be detected through the movement of the robotic arm. Since the camera is attached to the robotic arm, such deployment is adaptive to the detection area. The eye-in-hand deployment also demands a high-quality camera with depth detection capability to prevent frequent object detection failure.(1) In the eye-to-hand deployment, the camera is fixed on one side or above the working area. Accordingly, the objects' position, angle, and movement in the working area can be observed as a whole. Furthermore, the captured images are less disturbed by light, and the robotic arm can be expected to pick up objects more stably and accurately.(2)

Traditionally, machine vision can achieve object recognition, object positioning, and pose estimation to develop diverse robotic-arm-based applications. In Refs. 3 and 4, an edge detection method, namely, Canny, is adopted to extract the object contour to determine the object's center position. Then, the position of an object in the vision system (typically, images captured by a camera) is converted to the coordinate system of the robotic arm for object grasping. In Ref. 5, a medium filter is utilized first to reduce the noise and enhance the object's features for the processed image. Then, the image segmentation is applied to obtain a specific region covering the object. Finally, the robotic arm grasps the object at the angle calculated by the edge detection method. On the other hand, machine learning techniques can be involved in a vision-based robotic arm grasping system to enhance machine vision performance. In Ref. 6, an object shape classifier using the support vector machine (SVM) was used to effectively classify the targets into four shapes: spherical, cubic, spherical hole, and square hole. The orientation of each square/spherical hole is obtained using an image-based object orientation estimation algorithm for spherical/cubic objects. Although machine learning techniques such as SVM effectively classify objects, the vanishing gradient problem can fail the learning process and limit the sample size and computation capability.

When AlexNet is proposed for the deep convolutional neural network (DCNN),(7) deep learning is widely applied to machine vision applications.(8,9) To balance a tradeoff between accuracy and speed in object recognition, deep-learning-based object detection methods can be divided into two categories: two-stage and one-stage methods. In the two-stage method, the region proposals are generated via one convolutional neural network (CNN) module, and a

feature pyramid network (FPN) based on another CNN is adopted to obtain the position for each object category. The well-known two-stage methods include region-based CNN (R-CNN),[10] Fast R-CNN,[11] and Faster R-CNN.[12] Cheng *et al.* presented a densely connected FPN that operates as the feature extractor, and the two-stage detection system is attached to each fused layer to make multiple grasp pose predictions.[13] Generally, the two-stage detection method has a high object detection accuracy with low detection speed. On the other hand, the one-stage detection method classifies and locates each pixel instead of generating region proposals, leading to a higher object recognition speed than the two-stage detection method. Several common one-stage detection methods are You Only Look Once (YOLO)[14] and Single Shot MultiBox Detector (SSD).[15] The work described in Ref. 16 shows the deployment of a two-dimensional (2D) camera and the application of YOLO to detect objects on a running conveyor belt with a constant speed on a frame-by-frame basis. In Ref. 17, Pan *et al.* presented a manipulator-based package sorting and placing system. This system includes two major steps: (1) using YOLOv3 to obtain the object's center point and (2) applying the edge detection technique to calculate the length and width of the target package for manipulator grasping. According to the comparison of related works between two-stage and one-stage detection methods, the one-stage detection method can better preserve the timely requirement for vision-based manipulator applications even though the two-stage detection method typically attains a higher accuracy rate.

In this paper, we present a vision-based object grasping and placing system using robotic arms. Specifically, the proposed system aims to practically apply screwdriver bits with 12 different types in the field. The manual screwdriver bits are characterized by small size and nearly similar shapes. There are slight differences in thread, point tip, and thickness among screwdriver bits. Object recognition, grasping, and placement are challenging for target objects with small sizes for robotic arm control. The experimental settings, including materials, equipment, and operation procedure, mimic the assembly line configurations in the screwdriver bit manufactory. An industrial robotic arm using the robot operating system (ROS) with MoveIt[18] is installed to control the gripper in grasping and placing the target object on a conveyor belt. The eye-to-hand camera deployment is considered by default. In addition to the effectiveness, the proposed system also focuses on easy configuration and high flexibility for such field-based applications. The major components of our system include (1) world coordinate conversion from image pixel coordinates, (2) object detection, (3) edge detection, and (4) object orientation. In (1), a manual two-point marking method conducts the coordinate matching between the pixel coordinates in the image to the world coordinates of the manipulator so as to achieve an easy configuration of coordinate conversion for ROS-based systems. In (2), adapting state-of-the-art object detection methods to the field-based application can provide high flexibility in optimizing the system's performance. In this paper, three one-stage object detection methods, namely, YOLOv4, YOLOv7, and SSD, are conducted to compare their corresponding performance charcteristics in classifying screwdriver bits with slight differences. In (3) and (4), the proposed system focuses on integrating the detection results from (2) with the image processing method to obtain the object contour and then to calculate the orientation (i.e., angle) for grasping purposes. Real-time performance statistics indicate that the presented system can effectively classify and pick up 12 different driver bits scattered on the conveyor belt to the specified area in the storage box.

This paper is organized as follows. In Sect. 2, we describe the architecture and components of the proposed system. The materials and tools used in the paper are also included in this section. In Sect. 3, we present the dataset, a performance comparison between one-stage detection methods, and system validation results. Conclusions and future works are briefly given in Sect. 4.

2. Materials and Methods

2.1 System overview

In this study, we adopt the eye-to-hand method to grasp scattered objects on the conveyor belt. The robotic arm uses a TM robot (type no. TM5-700) with a movable range of 700 mm and a load capacity of 6 kg.[19] The camera used is Logitech C922 with a maximum resolution of 1080p and a maximum frame rate of 30 frames per second (fps). The diagonal field of view (dFoV) is 78° for Logitech C922. The conveyor belt has a speed regulator, LUYANG US425-01, to finely adjust the object transfer speed. Figure 1 presents the experimental setup used in this study. As shown in Fig. 1, the camera is set up above the conveyor belt so that the screen is horizontal to the conveyor belt, and the robotic arm is prevented from hitting the camera during its movement. ROS is considered as an integration tool between hardware and software, while the MoveIt module built in ROS conducts motion planning for robot control.[19] In this study, we aim to manipulate the robotic arm and grasp screwdriver bits provided by Chinshing Industries Co. tools (CICtools) in the presence of RGB images. Figure 1(b) shows the 12 screwdriver bits considered in this paper. In Fig. 1(b), the types of driver bit in the upper row from left to right are 1s, 3s, 2s, 7s, PH2s, PH1s, CIC8, CIC10, PH3, and PH2. The types of driver bit in the bottom row from left to right are PH3L and PH2L.

Figure 2 illustrates the architecture of a vision-based robotic arm grasping system. The system architecture can be divided into three components: (a) the calibration of the robotic arm and camera, (b) object detection and object orientation, and (c) the conversion of coordinates between three-dimensional (3D) space and 2D image pixels and motion planning for vision-based robotic arms.

(a) (b)

Fig. 1. (Color online) Components of robotic arm grasping system: (a) system setup and (b) screwdriver bits.

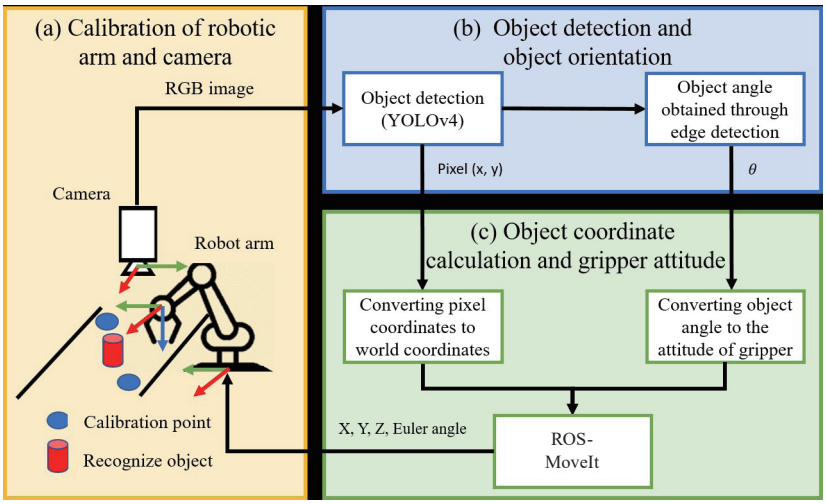


Fig. 2. (Color online) Vision-based robotic arm grasping system.

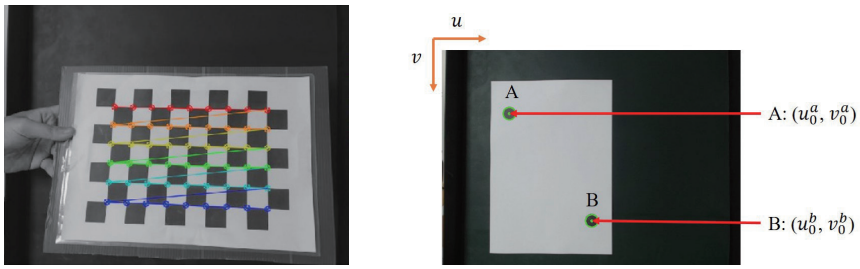
2.2 Calibration of robotic arm and camera

The coordinate relationship between the gripper and the base of the robotic arm is available in ROS. On the other hand, the camera coordinate system can be calculated by hand-eye correction. Before converting between world coordinates defined for robotic arms and image coordinates, camera calibration is necessary for the vision-based system to correct camera distortions while extracting 3D spatial information from 2D images. Typically, the camera calibration can be computed as

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \boldsymbol{A} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \text{ with } \boldsymbol{A} = \begin{bmatrix} f_u & 0 & u_0 \\ 0 & f_v & v_0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{1}$$

where (u,v) represents the pixel coordinates in the image system and (x,y) is the 3D-space coordinates. In Eq. (1), $\boldsymbol{A}$ is the so-called camera intrinsic matrix with (u_0,v_0) as the coordinates of the principal point, and f_u and f_v the scale factors of the u- and v- axes in the image, respectively. To achieve the camera calibration, planar patterns are often utilized to obtain the intrinsic parameters of $\boldsymbol{A}$. Figure 3(a) gives a 9 × 6 checkerboard to calibrate the camera coordinate system by the work presented in Ref. 20.

Since no distance-aware devices, such as a depth camera, are deployed in the experimental setup, a two-point marking method is considered to obtain the coordinate relationship between the camera and the target object [i.e., the coordinates (x,y,z) of the projected point for the object]. In the two-point marking method, each of the two calibration points constructs a circle as the center, and the two constructed circles are located on the diagonal of an image. As shown in Fig. 3(b), two calibration points, A and B, are denoted as (u^a,v^a) and (u^b,v^b), respectively. The two-point marking method comprises two steps as described below.



(a) (b)

Fig. 3. (Color online) Object coordinate correction: (a) checkerboard for camera coordinate calibration and (b) two calibration points in Hough circle detection. In (b), calibration point A is in the left upper corner and calibration point B in the right bottom corner.

(1) The pixel coordinates of the calibration point (i.e., the circle's center) are first calculated through Hough circle detection.[21,22] The equation of a circle is defined as

$$(x-u)^2+(y-v)^2=r^2, \tag{2}$$

where (u, v) is the circle's center coordinates and r is the circle's radius. Given the detected circle, the pixel coordinates of the calibration point can be obtained using Eq. (2).

(2) Then, the front end of the gripper is moved to align the calibration points, and the world coordinates of points A and B can be obtained by the transform (TF) function in ROS-MoveIt. As shown in Fig. 4, the relationship between the pixel coordinates of two calibration points and the world coordinates of the arm base can be expressed by

$$\begin{cases} \overline{x}_W = \dfrac{x_w^b - x_w^a}{u^b - u^a}, \\ \overline{y}_W = \dfrac{y_w^b - y_w^a}{v^b - v^a}, \end{cases} \tag{3}$$

where x_w^a, y_w^a and x_w^b, y_w^b are the arm coordinates of calibration points A and B, respectively. Furthermore, u^a, v^a and u^b, v^b stand for the pixel coordinates of points A and B in the image, respectively. The resultant $\overline{x}_w$, $\overline{y}_w$ can derive the distances x_{range} and y_{range} as the robotic arm moves outside the camera scope:

$$\begin{cases} x_{range} = x_w^b - \overline{x}_w \times u^b, \\ y_{range} = y_w^b - \overline{y}_w \times v^b. \end{cases} \tag{4}$$

The out-of-scope distances $\{x_{range}, y_{range}\}$ are further considered as the important reference for robotic arm control.

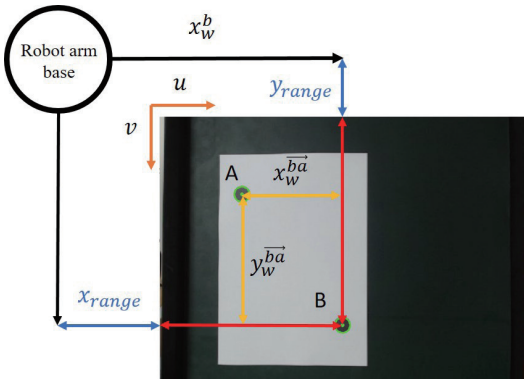


Fig. 4. (Color online) Relationship between pixel and robotic arm coordinates.

2.3 Object detection and orientation

In building the dataset for object detection, the screwdriver bits are scattered randomly on a running conveyor belt. It is assumed that the location and angle are different for all driver bits. Under the constraint that the driver bits can remain static during movement, the camera captures 3584 images. Table 1 presents the statistical data in the dataset. Because only a portion of driver bits can be captured in the images as the conveyor belt operates, the amount of data is inconsistent for all types of driver bit.

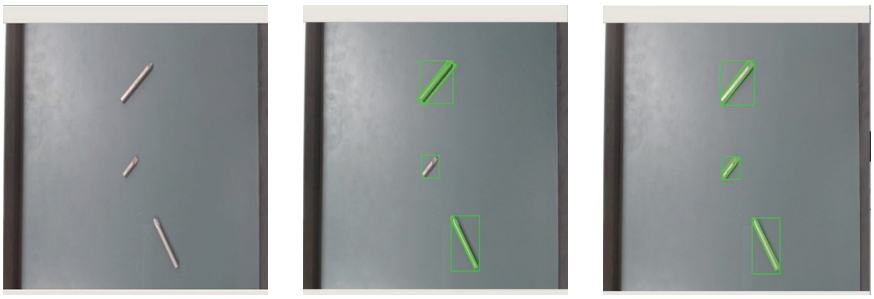
According to the performance comparison conducted in the MS COCO dataset, YOLOv4[23] outperforms SSD and RetinaNet in terms of overall performance and system stability. Although YOLOv7 has a higher detection speed than YOLOv4,[24] we consider YOLOv4 as the object detection method based on the fact provided in Sect. 3 that YOLOv4 can attain a higher detection accuracy than YOLOv7. After the target object is detected by YOLOv4, a Gaussian smoothing method is utilized to reduce noises in the edge detection process. Consequently, the corresponding object angle can be obtained for robotic arm grasping.

In the object detection stage of Fig. 2, YOLOv4 obtains an object label/category and the coordinates for the detected object. The necessary object spatial information, including angles and directions, is absent for robotic arm grasping. Specifically, the length and width of the YOLOv4 bounding box can be used to tell if the target object is located vertically or horizontally. However, the spatial information provided by the bounding box cannot determine a tilted position [refer to Fig. 5(a)]. In this study, we combine Canny, Gaussian smoothing, and gradient detection to conduct object orientation for YOLOv4-detected results. We denote the gray-scale value of a pixel (m,n) as $f(m,n)$. The edge detection stage first applies the Gaussian smoothing to $f(m,n)$ for noise reduction,

$$g_\sigma(m,\ n)=\frac{1}{\sqrt{2\pi\sigma^2}}e^{-\left(\frac{m^2+n^2}{2\sigma^2}\right)}\times f(m,\ n), \tag{5}$$

Table 1
Statistical data for each type of driver bit in dataset.

Category	Amount
1s	193
2s	198
3s	182
7s	357
CIC 8	506
CIC 10	364
PH1s	236
PH2	349
PH2L	333
PH2s	221
PH3	339
PH3L	306
Total	3584



(a) (b) (c)

Fig. 5. (Color online) Edge detection results: (a) original image, (b) applying Canny to whole image, and (c) applying Canny only to bounding box.

where σ is the standard deviation of a Gaussian distribution. Then, gradient estimates can be obtained by continuously conducting a Sobel convolution for image pixels.[25] For the pixel (m,n), the gradient density is given by

$$G(m,\ n)=\sqrt{g_x(m,\ n)^2+g_y(m,\ n)^2}, \tag{6}$$

where $g_x(\cdot)$ and $g_y(\cdot)$ are the gradient variations in the x- and y-axes, respectively. The gradient density can effectively describe the object contour. Finally, a minimum rectangle fitting the object contour is obtained by a contour feature function $CF(\cdot)$.[26] Four vertices of the minimum rectangle can be expressed as

$$\begin{pmatrix} x_1,\ y_1 \\ x_2,\ y_2 \\ x_3,\ y_3 \\ x_4,\ y_4 \end{pmatrix} = CF\big(G(m,\ n)\big), \tag{7}$$

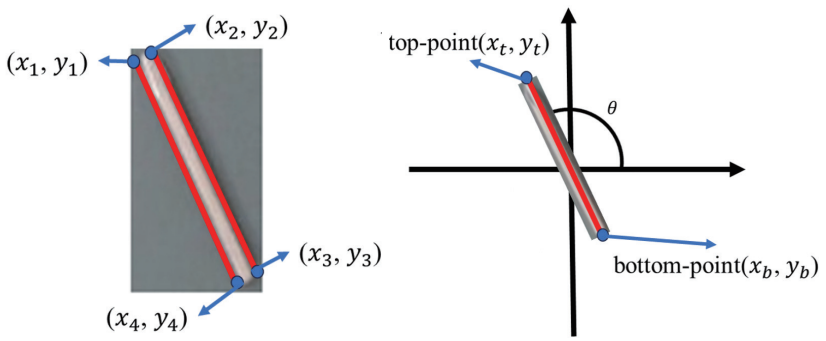
where (x_i,y_i) represents a vertex of the rectangle and $1 \le i \le 4$. As shown in Fig. 5(b), when a whole image is processed, Canny may not generate a minimum rectangular box to fit the target object (e.g., middle driver bit). Note that the rectangle (i.e., bounding box) generated by YOLO only labels the target object without any tilted position information. To fix this problem, only the YOLOv4 bounding box is regarded as the input of Canny. The resultant Canny output in Fig. 5(c) shows that all three driver bits can be fitted to obtain their respective tilted position information (i.e., angle or direction).

As shown in Fig. 6(a), two short sides, namely, $\overline{(x_1,y_1)(x_2,y_2)}$ and $\overline{(x_3,y_3)(x_4,y_4)}$, determine a top-point side and a bottom-point side, respectively. Then, their central points are computed as

$$\begin{cases} (x_t,y_t)=\left(\dfrac{(x_1+x_2)}{2},\dfrac{(y_1+y_2)}{2}\right), \\ (x_b,\ y_b)=\left(\dfrac{(x_3+x_4)}{2},\ \dfrac{(y_3+y_4)}{2}\right). \end{cases} \tag{8}$$

Given these two central points, an angle representing the object direction can be obtained by taking a slope between two points as an input of the arctan function,

$$= \begin{cases} (\arctan\left(\dfrac{y_b-y_t}{x_b-x_t}\right)+\pi)\times\dfrac{180}{\pi},\ \text{if}(x_b-x_t)\times(y_b-y_t)<0, \\ \arctan\left(\dfrac{y_b-y_t}{x_b-x_t}\right)\times\dfrac{180}{\pi},\ \text{if}\,(x_b-x_t)\times(y_b-y_t)>0, \\ 90,\ \text{if}\,(x_b-x_t)=0, \\ 0,\ \text{if}\,(y_b-y_t)=0, \end{cases} \tag{9}$$



(a) (b)

Fig. 6. (Color online) Object-fitted rectangle: (a) four vertices of rectangle and (b) object slope and its corresponding angle.

where the angle unit is degree and $0 \le \theta < 180$. In the first item of Eq. (9), an acute angle with a negative degree would be converted to an obtuse angle with a positive degree. Figure 6(b) illustrates such angle conversion for a negative degree.

3. Results

In the experiment, each image in the dataset contained only one type of screwdriver bit, which was randomly scattered on a running conveyor belt. Therefore, for any driver bit, the direction presented in one image was almost different from others. As shown in Table 1, there were 3584 images for 12 types of screwdriver bit in the dataset. Following a ratio of 8:2 for training and test samples, LabelImg was used as the object labeling tool to generate the object spatial information for subsequent data training and testing. The experimental platform was built on a Linux system with NVIDIA GeForce RTX 2060 GPU.

3.1 Object detection results

The compared methods for object detection in this paper were YOLOv4, YOLOv7, and SSD. Detailed configurations of the compared methods are presented in Table 2. In the experiments, mAP and IoU are the main metrics for evaluating the object detection performance. IoU corresponds to the intersection area between bounding boxes for ground truth and prediction. The larger the intersection area, the higher the IoU score. The IoU score is calculated as

$$IoU = \frac{Area_{pre} \cap Area_{gt}}{Area_{pre} \cup Area_{gt}}. \tag{10}$$

In this study, the threshold of the IoU score is set to 0.5 for mAP calculation. An object is successfully detected as its IoU score exceeds 0.5. On the basis of the IoU threshold of 0.5, we can obtain the following classification metrics for each of the 12 object types: true positive (TP), false positive (FP), false negative (FN), and true negative (TN). Furthermore, the detection validation indexes, namely, precision (P), recall (Rec), accuracy (Acc), and F1 score, can be given in order by

$$Precision = \frac{TP}{TP + FP}, \tag{11}$$

Table 2
Configurations of three compared methods.

Method	SSD	YOLOV7	YOLOV4
Input image	300 × 300	416 × 416	416 × 416
Backbone	VGG16	E-ELAN	CSPDarknet53
Learning rate	0.001	0.001	0.001
Momentum	0.9	0.9	0.9
Iteration	60000	6000	6000

$$Recall = \frac{TP}{TP + FN}, \tag{12}$$

$$Accuracy = \frac{TP + TN}{\left(TN + TP + FN + FP\right)}, \text{ and} \tag{13}$$

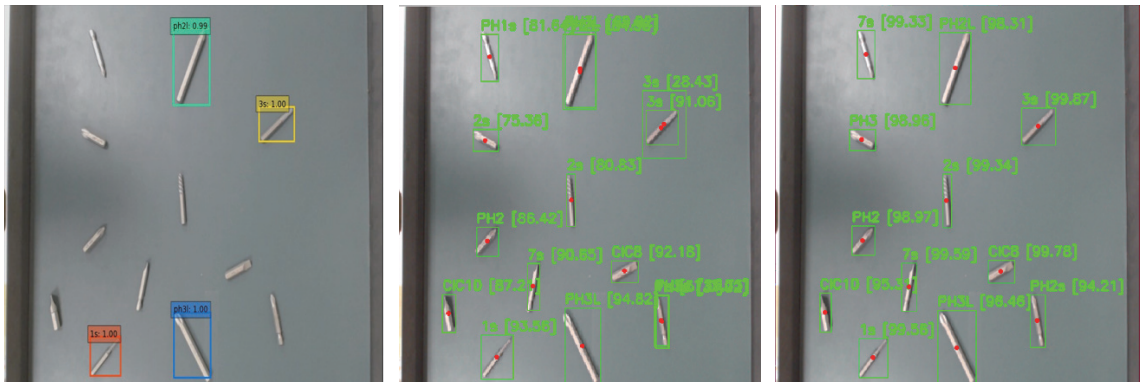
$$F1 = \frac{2 \times precision \times recall}{precision + recall}. \tag{14}$$

From Eqs. (11) and (12), a common validation index, average precision (AP), is the area under the precision–recall curve. Then, mAP can be computed as the average of AP over all detected classes.

The performance results of the object detection stage are presented in Table 3. From Table 3, YOLOv4 outperforms SSD and YOLOv7 in detecting randomly scattered screwdriver bits. As shown in Fig. 7, we can observe that (1) SSD can correctly identify the driver bits PH2L, PH3L, 2s, and 3s, and fails to detect the other eight driver bits; (2) YOLOv7 is capable of labeling all driver bits with misclassification errors and multiple boxes for the same object; and (3) YOLOv4 can successfully identify all driver bits with their corresponding type and induces a slightly lower frame rate per second than YOLOv7 (referring to Table 3). Consequently, YOLOv4 is adopted in this paper as the object detection method for subsequent robotic arm grasping.

Table 3
Object detection performance comparison among three compared methods.

Method	*mAP* (%)	*IoU* (%)	*Rec.*	*P*	*Acc.*	*F*1	*FPS*
SSD	88.1	74.02	75.3	81.0	88.3	75.9	21
YOLOV7	97.8	74.75	94.6	84.7	97.9	89.4	31
YOLOV4	99.4	83.07	98.8	94.3	99.5	96.5	27



(a) (b) (c)

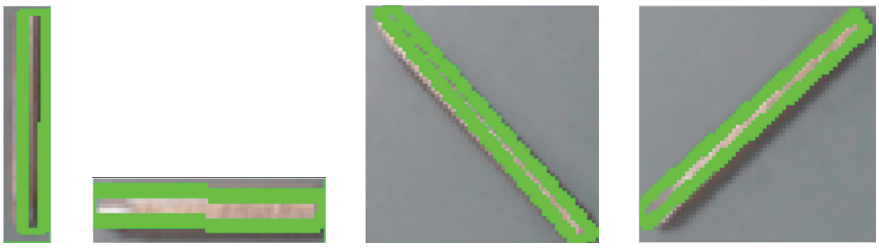
Fig. 7. (Color online) Object detection results for three methods: (a) SSD, (b) YOLOv7, and (c) YOLOv4.

3.2 Object orientation results

Figure 8 shows the edge detection results as a target driver bit is placed at angles of 0, 45, 90, and 125°. As shown in Fig. 8, the edge detection method can fit the object contour with a minimum rectangle. The resultant rectangle provides sufficient spatial information for robotic arm grasping in terms of precise coordinate location, shape, and angle. Table 4 presents the results of angle estimation errors using mean absolute error (*MAE*) and root mean squared error (*RMSE*) to evaluate the object orientation performance. *MAE* and *RMSE* are defined as

$$\begin{cases} MAE = \dfrac{\sum_{i=1}^{N} \left| \theta_i' - \theta_i \right|}{N}, \\ RMSE = \sqrt{\dfrac{\sum_{i=1}^{N} \left(\theta_i' - \theta_i \right)^2}{N}}, \end{cases} \tag{15}$$

where N is the number of samples, θ_i represents the target angle of the i-th sample, and θ_i' stands for the estimated angle of the i-th sample. Currently, 30 samples are collected for each degree and N is 30. Among the four degrees, 0° has the smallest *MAE* and *RMSE* values. As the target degree increases to 90°, the angle estimation errors increase accordingly. Both the *MAE* and *RMSE* values decrease beyond 90°. In Table 4, an average *MAE* of 1.54 is obtained, and the angle error rate can be further calculated as (1.54/180) × 100% = 0.86%, where the target angle ranges from 0 to 180°.



(a) (b) (c) (d)

Fig. 8. (Color online) Object orientation results: (a) 90, (b) 0, (c) 125, and (d) 45°.

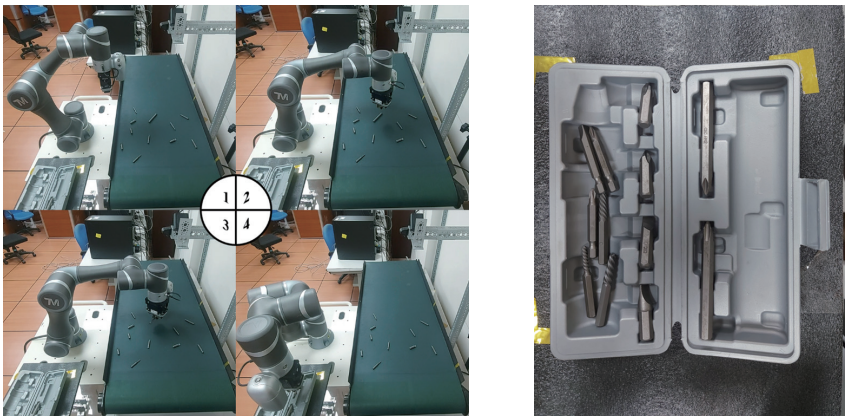
Table 4
Angle error statistics.

Degree	*MAE*	*RMSE*
0	0.64	0.82
45	1.86	2.91
90	2.21	2.33
125	1.46	1.99
Total	1.54	2.01

3.3 Real-time grasping results

In this study, we aim to detect screwdriver bits on the conveyor belt correctly and to pick up the selected driver bits using a ROS-controlled robotic arm to the storage box. In grasping multiple driver bits on a stationary conveyor belt, YOLOv4 first detects the object to obtain the predefined object category and its pixel coordinates in the image. Then, the edge detection of Canny is applied to the YOLOv4 bounding box to calculate the object's angle. By combining the pixel coordinates and object angle, the ROS system can estimate the arm grasping parameters, such as the target's relative position and the gripper's posture, to pick up the selected driver bit. Finally, the picked object would be put in a storage box. Figure 9(a) shows this grasping process, and each sub-image is labeled with the grasping order ranging from 1 to 4. In Fig. 9(b), the robotic arm can precisely put any of the six selected driver bits, namely, PH2, PH2L, PH3, PH3L, CIC8, and CIC10, into their corresponding specified locations. The robotic arm arbitrarily puts the other six driver bits in an unspecified storage box area after picking them. The procedure mentioned above is conducted in order following the object detection sequence determined by YOLOv4.

The experiment procedure mentioned above was repeated ten times to evaluate the arm grasping performance, and the corresponding grasping results were recorded. Grasping succeeded when the robotic arm picked and placed the target driver bit in the specified location. Table 5 presents the successful grasping rates for each of the ten experiments. The grasping fraction is calculated as the number of successful grasping items over the number of total items (i.e., 12). It is seen in Table 5 that (1) seven of the ten experiments can achieve a successful grasping rate of 100%, and (2) in the cases where the successful grasping rates are lower than 100%, the number of missed items ranges from 1 to 2. By inspecting the experiment records, the misclassification induces the grasping misses; therefore, the object is placed in the wrong location. A successful grasping rate of 96% can be attained from all ten experiments.



(a) (b)

Fig. 9. (Color online) Arm grasping setup: (a) grasping sequence (1–4) and (b) object location in storage box after picking.

Table 5
Arm grasping statistics.

Serial number	Grasping fraction	Successful grasping rate (%)
1	11/12	91
2	12/12	100
3	12/12	100
4	12/12	100
5	12/12	100
6	10/12	83
7	12/12	100
8	11/12	91
9	12/12	100
10	12/12	100
Total	116/120	96

4. Conclusions

In this paper, we propose a vision-based robotic arm control system for the object placement task. Specifically, the proposed vision-based robotic arm control system can automatically grasp 12 different screwdriver bits and put them into the specified area in the storage box. There are four major stages in the proposed system, namely, world coordinate conversion, object detection, edge detection, and object orientation. In the object detection stage, it is found that YOLOv4 performs better than SSD and YOLOv7 in the CICtools dataset. YOLOv4 can attain 99.47, 83.07, and 99.51% for mAP, IoU, and accuracy, respectively. In addition to the object class and position given by YOLOv4, Canny is mainly adopted in the edge detection stage to further obtain the object's angle. According to the experimental results, the object orientation stage receives an *MAE* rate of 0.86% and an *RMSE* rate of 1.12% in the 180^{o} system. To control the robotic arm based on RGB images, the two-point marking method converts the pixel coordinates in the image to the world coordinates used for the robotic arm and accordingly conducts the calibration between two different coordinate systems. Consequently, the ROS-driven robotic arm can effectively grasp objects on the basis of spatial information, including object class, position, and angle. From a total of ten experiments, our robotic arm control system can achieve a successful grasping rate of 96%.

Although the current system is effective, assuming the conveyor belt remains stationary, failed object detection may occur as the conveyor belt is running/moving. The future work aims to design a predictive grasping method on the running conveyor belt. The difficulties may originate from two aspects: (1) the current object detection and edge detection need to be improved for running the conveyor belt, and (2) the position and angle of the object may change frame by frame. The state-of-the-art machine learning and semantic segmentation techniques can be considered to overcome these potential difficulties.

Acknowledgments

The authors thank Chinshing Industries Co. for providing the experiment materials.

References

1 T. Nakamura: Proc. Int. Conf. Robotics and Biomimetics (IEEE, 2011) 784–788.
2 G. Flandin, F. Chaumette, and E. Marchand: Proc. IEEE Int. Conf. Robotics and Automation (IEEE, 2000) 2741–2746.
3 J. M. Cho and K. Kim: Proc. 2017 14th Int. Conf. Ubiquitous Robots and Ambient Intelligence (IEEE, 2017) 497–499.
4 Z. Zhang, Z. Li, and Y. Wang: Proc. 2017 Chinese Automation Congress (IEEE, 2017) 6464–6469.
5 K. Zhou, Z. Meng, M. He, J. Hou, and T. Li: IEEE Access **8** (2020) 27178. https://doi.org/10.1109/ACCESS.2020.2971349
6 C. Y. Tsai, C. C. Wong, C. J. Yu, C. C. Liu, and T. Y. Liu: IEEE Systems J. **9** (2015) 1. https://doi.org/10.1109/JSYST.2014.2358876
7 A. Krizhevsky, I. Sutskever, and G. E. Hinton: Commun. ACM **60** (2017) 6. https://doi.org/10.1145/3065386
8 F. Tokuda, S. Arai, and K. Kosuge: IEEE Access **9** (2021) 91820. https://doi.org/10.1109/ACCESS.2021.3091737
9 Y. Yu, Z. Cao, Z. Liu, W. Geng, J. Yu, and W. Zhang: IEEE Trans. Systems, Man, and Cybernetics: Systems **52** (2022) 2. https://doi.org/10.1109/TSMC.2020.3018757
10 R. Girshick, J. Donahue, T. Darrell, and J. Malik: Proc. 2014 IEEE Conf. Computer Vision and Pattern Recognition (IEEE, 2014) 580–587.
11 R. Girshick: Proc. 2015 IEEE Int. Conf. on Computer Vision (IEEE, 2015) 1440–1448.
12 S. Ren, K. He, R. Girshick, and J. Sun: Proc. Neural Inf. Process. Syst. (MIT Press, 2015) 91–99.
13 H. Cheng, Y. Wang, and M. Q. H. Meng: IEEE Sens. J. **22** (2022) 10. https://doi.org/10.1109/JSEN.2022.3163730
14 J. Redmon, S. Divvala, R. Girshick, and A. Farhadi: Proc. 2016 IEEE Conf. Computer Vision and Pattern Recognition (IEEE, 2016) 779–788.
15 W. Liu, D. Anguelov, D. Erhan, C. Szegedy, and S. Reed: Proc. Eur. Conf. Comput. Vis. (Springer, 2016) 21–37.
16 K. Uçar and H. E. Koçer: IEEE Latin Am. Trans. **21** (2023) 2. https://doi.org/10.1109/TLA.2023.10015227
17 Z. Pan, Z. Jia, K. Jing, Y. Ding, and Q. Liang: Proc. 2020 Chinese Control And Decision Conf. (IEEE, 2020) 409–414.
18 H. Deng, J. Xiong, and Z. Xia: Proc. 2017 IEEE Int. Conf. Real-time Computing and Robotics (IEEE, 2017) 612–616.
19 K. T. Song, Y. H. Chang, and J. H. Chen: Proc. 2019 IEEE/ASME Int. Conf. Advanced Intelligent Mechatronics (IEEE, 2019) 254–258.
20 Z. Zhang: IEEE Trans. Pattern Analysis and Machine Intelligence **22** (2000) 1330. https://doi.org/10.1109/34.888718
21 P. V. C. Hough: Proc. Int. Conf. High Energy Accelerators and Instrumentation (CERN, 1959) 554–558.
22 D. H. Ballard: Pattern Recognition **13** (1982) 2. https://doi.org/10.1016/0031-3203(81)90009-1
23 A. Bochkovskiy, C. Y. Wang, and H. Y. M. Liao: arXiv preprint (2020). https://doi.org/10.48550/arXiv.2004.10934
24 C. Y. Wang, A. Bochkovskiy, and H. Y. M. Liao: arXiv preprint (2022). https://doi.org/10.48550/arXiv.2207.02696
25 Research Gate: https://doi.org/10.13140/RG.2.1.1912.4965 (accessed August 2015).
26 Sakshi and V. Kukreja: Proc. 2022 Int. Conf. Decision Aid Sciences and Applications (IEEE, 2022) 305–310.

About the Authors

Cheng-Jian Lin received his B.S. degree in electrical engineering from Ta Tung Institute of Technology, Taipei, Taiwan, R.O.C., in 1986, and his M.S. and Ph.D. degrees in electrical and control engineering from National Chiao-Tung University, Taiwan, R.O.C., in 1991 and 1996, respectively. Currently, he is a chair professor of the Computer Science and Information Engineering Department, National Chin-Yi University of Technology, Taichung, Taiwan, R.O.C., and the dean of Intelligence College, National Taichung University of Science and Technology, Taichung, Taiwan, R.O.C. His current research interests are in machine learning, pattern recognition, intelligent control, image processing, intelligent manufacturing, and evolutionary robots. (cjlin@ncut.edu.tw)

Pei-Jung Lin received her Ph.D. degree in computer science from Feng Chia University, Taiwan. Currently, she is an associate professor of the Department of Information Engineering and Computer Science, Feng Chia University, Taichung, Taiwan. Her research interests include extended reality, brain computer interface, artificial intelligence, and wireless mesh networks. She has distinguished herself with 11 patents and numerous gold medals from esteemed domestic and international invention competitions. Her dedication to fostering collaboration between industry and academia is evident through her proactive involvement in various projects, resulting in multiple instances of technology transfers. (linpj@fcu.edu.tw)

Chi-Huang Shih received his Ph.D. degree in electrical engineering from National Cheng Kung University, Taiwan, R.O.C., in 2008. Currently, he is an associate professor of the Computer Science and Information Engineering Department, National Chin-Yi University of Technology, Taichung, Taiwan, R.O.C. His current research interests are in machine learning, multimedia network communication, biomedical worn devices, embedded systems, and IoT applications. (chshih@ncut.edu.tw)