**ELKOMIKA: Jurnal Teknik Energi Elektrik, Teknik Telekomunikasi, & Teknik Elektronika**

ISSN(p): 2338-8323 | ISSN(e): 2459-9638 | Vol. 11 | No. 1 | Halaman 128 - 142
DOI : http://dx.doi.org/10.26760/elkomika.v11i1.128 Januari 2023

# Object Detection and Pose Estimation with RGB-D Camera for Supporting Robotic Bin-Picking

**EKO RUDIAWAN JAMZURI, RISKA ANALIA, SUSANTO**

Department of Electrical Engineering, Politeknik Negeri Batam, Indonesia
Email: ekorudiawan@polibatam.ac.id

*Received* 30 November 2022 **|** *Revised* 20 Desember 2022 **|** *Accepted* 3 Januari 2023

## ABSTRAK

*Tujuan dari penelitian ini adalah untuk mendeteksi objek dan mengestimasi pose objek menggunakan kamera RGB-D. Dalam penelitian ini, kami mengusulkan pemrosesan data pada citra RGB dan citra depth saja, tanpa menggunakan point cloud, seperti pada umumnya. Metode yang diusulkan mendeteksi posisi dan orientasi objek menggunakan DRBox-v2 dari Region of Interest (ROI), yang sebelumnya diperoleh dari pendeteksian pada penanda ArUco. Hasil deteksi objek kemudian diskalakan dan digunakan pada citra depth untuk mendapatkan perkiraan posisi dan orientasi objek. Dari sisi pendeteksi objek, usulan metode memperoleh nilai Average Precision (AP) sebesar 0,740. Sedangkan untuk estimator pose, usulan metode menghasilkan kesalahan posisi rata-rata 13,36 mm dan kesalahan orientasi rata-rata 0,75 derajat. Metode yang diusulkan berpotensi menjadi alternatif sistem deteksi objek dan estimasi pose pada kamera RGB-D yang tidak memerlukan pemrosesan point cloud dan tidak memerlukan model referensi objek.*

***Kata kunci****: deteksi objek, estimasi pose, DRBox, ArUco, bin-picking*

## ABSTRACT

*This study aims to detect objects and estimate the object's pose using an RGB-D camera. In this study, we proposed data processing on RGB images and depth images only, without using point clouds, as in general. The proposed method detected the object's position and orientation using the DRBox-v2 from the Region of Interest (ROI), which was previously obtained from detecting ArUco markers. The object detection results were then scaled and used in the depth image to get the object's approximate position and orientation. In object detection, the proposed method obtained an Average Precision (AP) value of 0.740. As for the pose estimator, our method generated an average position error of 13.36 mm and an average orientation error of 0.75 degrees. Therefore, this method can be an alternative object detection and pose estimation system on an RGB-D camera that does not require point cloud processing and an object reference model.*

***Keywords****: object detection, pose estimation, DRBox, ArUco, bin-picking*

## 1. INTRODUCTION

Bin-picking is a classic problem in robotics that aims to pick up a random object from the bin and place it in a specific, predefined location. Bin-picking can also be called dynamic pick-and-place when there is uncertainty about the object location to be picked up. Object detectors and pose estimators are needed to help the robot's perception system. The object detector is used to identify the object class, while the pose estimator predicts the position and orientation of the object concerning a particular coordinate system. The object detector and pose estimator accuracy directly affect the robot's success rate in retrieving objects. Therefore, an accurate object detection system and pose estimator that produces low error is needed in this particular problem.

The types of sensors used in the bin-picking robot perception system can be classified into three types: 1. 2D cameras, 2. RGB-D cameras, and 3. 3D cameras. The 2D cameras are cheaper in terms of price but require complicated algorithms on the computational side. This issue is due to the camera's limitations, which can only capture objects in 2D. The 2D camera has been used by **(Kozák et al., 2021)** in their research on object detection and poses estimation. **(Kozák et al., 2021)** combined feature description with a Convolutional Neural Network (CNN) to process 2D data. The feature description is used to segment objects, while CNN is used as a pose estimator. Unlike a 2D camera, using an RGB-D or 3D camera will produce a point cloud from the captured environment. This point cloud is then matched with a 3D object model to estimate its pose. This type of method is known as CAD-based pose estimation. **(Lee & Lee, 2020)** proposed object detection and CAD-based pose estimation using cascade object detector and Iterative Closest Points (ICP) methods. YOLOv3 is used to detect objects and objects' features sequentially. Then the object's features are matched with the features on the 3D model with ICP to estimate the object's pose. In contrast to **(Lee & Lee, 2020)**, a voting method on the Point Pair Feature (PPF) descriptor is proposed by **(Yan et al., 2020)**. Similar features are grouped into a hash table and will be used as a reference for the pose estimator. Meanwhile, the voting method proposed by **(Zhuang et al., 2021)** used the Semantic Point Pair Feature (SPPF) from the part of the object that was previously detected using MaskRCNN. A different technique is introduced by **(Wong et al., 2022)**, which uses the bounding box output of the YOLOv4 to extract point cloud data. Furthermore, the feature detection and alignment are carried out from the point cloud data using Fast Point Feature Histograms (FPFH). Finally, the pose of the object is estimated by ICP. From the research based on RGB-D and 3D cameras mentioned, the role of object 3D models is crucial for pose estimators. The research which uses an RGB-D camera without a 3D model reference is done by **(Soltan et al., 2020)**. However, the result still has a limitation in estimating object orientation. Currently, it is only capable of predicting object position.

Our study aims to overcome the limitations of estimating object orientation in non-CAD-based pose estimation. In this study, we proposed an RGB-D camera as a sensor, accompanied by ArUco markers and DRBox-v2 as detectors and pose estimators. In contrast to the study by **(Soltan et al., 2020)**, our results can estimate the position of objects in 3D and also the orientation of objects on the $z$-axis.

This research article is organized as follows. Section 2 will introduce the material and method used in this investigation. Then, we will describe how the data set is collected and how the ArUco marker combined with DRBox-v2 detected and estimated the objects. Furthermore, we describe the evaluation method used to compare our research results. Section 3 will elaborate on the research results and discuss some limitations of our research. Finally, the article will be closed with the conclusion statements in Section 4.

## 2. MATERIAL AND METHOD

This section will explain the methods and materials used to support the research. The first subsection will describe the research environment. Furthermore, we will explain the data collection and train a rotatable object detection. The detection method will be described in the third subsection. Then, it will continue with the process of estimating poses. Finally, the last subsection will represent the evaluation of the proposed method.

### 2.1 Research Environment

Figure 1 (a) shows details of the research environment. We placed the Universal Robot UR3 manipulator robot on a workbench, and the bin was placed on top of the workbench. This bin is used to put the objects which the robot will take. In this work area, the robot manipulator will take an object randomly, which is located in the bin, and then put the object in a predetermined area. We also mounted the RGB-D camera, Intel RealSense D455, directly above the bin with the face-down position towards the tabletop. This camera, later on, will be utilized to recognize the objects surrounding the bin and estimate the pose. Furthermore, we put the ArUco visual markers to facilitate the Region of Interest (ROI) detection. The ArUco markers are usually used as visual landmarks in indoor robot localization systems **(La Delfa et al., 2016) (Xing et al., 2018) (Yu et al., 2021)**. Additionally, another application is used as a marker for the landing area of an Unmanned Aerial Vehicle (UAV) **(Wubben et al., 2019)**. We chose the ArUco as a visual marker because it is more computationally efficient than other visual markers, such as AprilTag and STag **(Kalaitzakis et al., 2021)**. Figure 1 (b) shows that we attached four ArUco markers with a unique identity, from ID0 to ID3, to the table's surface. Additionally, these markers are installed to form a square with a side distance of 600 mm. These four ArUco markers will be used as a reference for cropping the image to focus only on the ROI area.

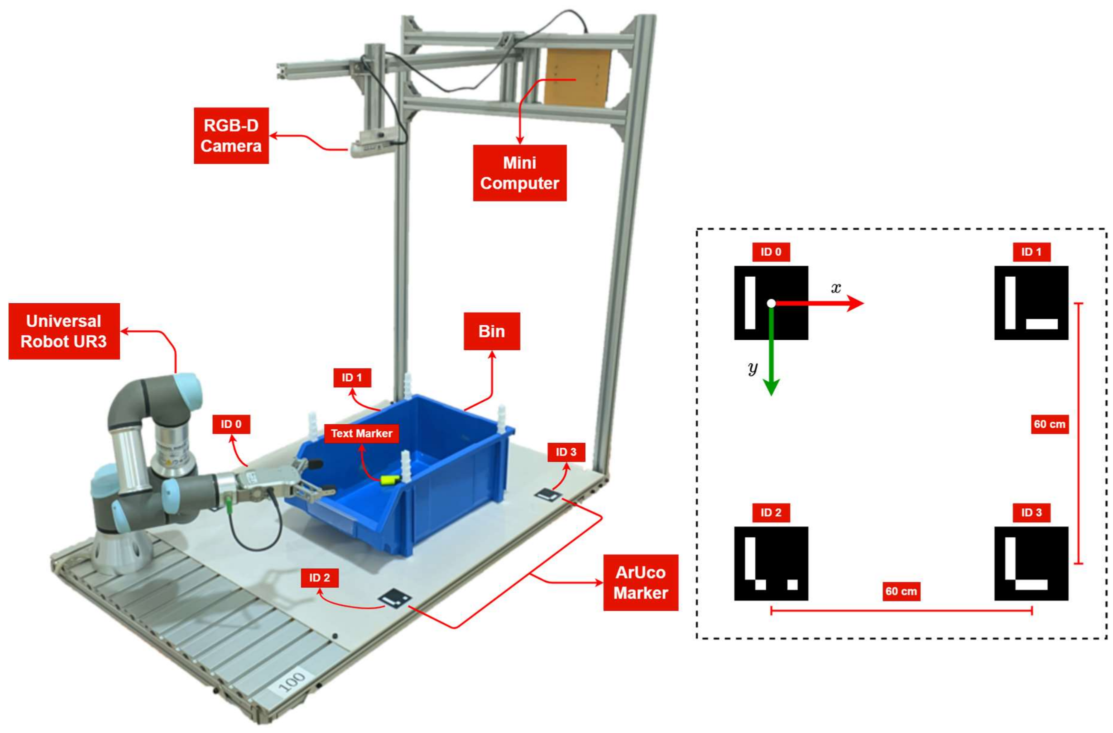


**Figure 1. (a) The Robot Manipulator Research Environment, (b) The ArUco Markers Installation**

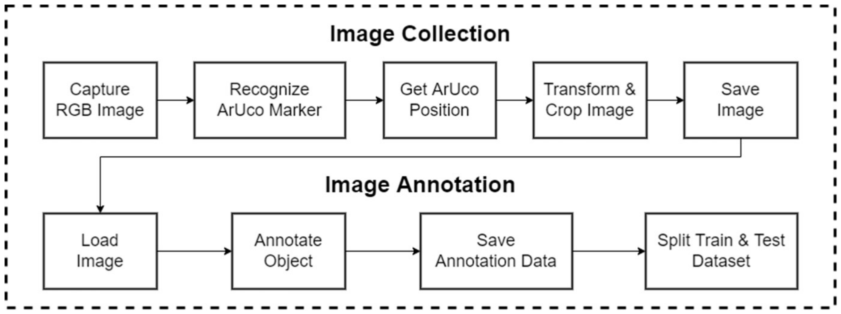


**Figure 2. Dataset Collection Process Flow**

## 2.2 Dataset Collection

The first step was to collect the data set, which contains RGB images focused only on the work area. Then, there are preprocessing and postprocessing stages carried out to collect images. The block diagram in Figure 2 illustrates the collecting data process. The first step was to capture an image from the RGB-D camera. At this stage, we have only taken the RGB image from the camera and have not used the depth image yet. Next, we carried out the identification process on the four ArUco markers. Then, from the four detected ArUco markers, we took the identity and position of the markers. The identity was represented by a decimal number 0-3, and the position was described in terms of the coordinates of a point $(x, y)$ in the image frame. Once all ArUco positions were known, we performed the transformation and cropping process. This process aimed to obtain images focusing only on a predetermined work area. Therefore, if objects were outside the work area, these objects can be removed. In addition, this process aimed to obtain images with a smaller resolution but still described the essential information to detect the object inside the predetermined work area. We used Equations (1) and (2) to perform the transformation and cropping. The $x_i$ and $y_i$ were the midpoint locations of the ArUco markers. In contrast, $x_i'$ and $y_i'$ were the destination coordinate points after the transformation process was performed. The $T$ in Equation (1) was a 3×3 matrix used to transform the point. This transformation was carried out at the four midpoints of the visual markers, as described in Equation (2). In the transformation stage, the point $(x_i, y_i)$ of the four ArUco markers will be transformed into the points $(x_i', y_i')$. We determined the destination coordinates $(x_i', y_i')$ with a fixed parameter. Where for ArUco, ID0 is transformed to (0,0), ID1 is moved to (300,0), ID2 is changed to (0,300), and finally, ID3 is shifted to (300,300). This process will be ultimately formed a square image of 300×300 pixels. Furthermore, this image will be saved for the following processing stage.

$$\begin{bmatrix} t_i x_i' \\ t_i y_i' \\ t_i \end{bmatrix} = T \cdot \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \tag{1}$$

$$\mathrm{dst}(i) = (x_i', y_i'), \mathrm{src}(i) = (x_i, y_i), i = 0,1,2,3 \tag{2}$$

We annotated the images after obtaining a collection of images from the preprocessing. This annotation aimed to mark objects with a rotatable bounding box (RBox). We used a custom tool, roLabelImg, to do this process. The results of the data annotation will be saved in a text file in rbox format. This file was then used as a reference to train object detection and become the ground truth for evaluating the detection accuracy. Our data collection process resulted in a total of 1100 annotated images. We then randomly separated the data for the Leave-One-Out Cross-Validation (LOOCV). The LOOCV validation divided the data set into train and test data, where the data used for the training is 1000 images, and the rest are used in the testing.

## 2.3 Rotatable Object Detection

Object detection is mandatory if the objects to be taken by the robot are in a random location. On the other hand, the robot joints can be moved to a constant position if the object is in a fixed location. This method was carried out by **(Sartika et al., 2019)** on a diamond robot pick-and-place system. Meanwhile, the scenario in our research was to use a random object's position inside the bin, so we detected these objects using an RGB-D camera, which produced an output of RGB and depth images. We used the CNN object detection algorithm, with RGB images as input. CNN algorithm was a state-of-the-art method for identifying objects in images. Examples are provided by **(Wahyuni & Hendri, 2019)** to recognize smoke and fire and also **(Dewi et al., 2019)** to recognize various car types in the Intelligence Transportation System (ITS). Generally, object detection produced detection output in classes and bounding boxes. However, the object's orientation was difficult to know because the output was represented in a box. Therefore, we need further processing to identify the object's heading. We propose an object detection system using the DRBox-v2 **(An et al., 2019)** to overcome these limitations.

DRBox-v2 was initially proposed by **(An et al., 2019)** to identify objects in Synthetic Aperture Radar (SAR) images. SAR images are two-dimensional images generated by satellite radar and have a grayscale color. In this study, we used DRBox-v2 to detect text markers in RGB images produced by the RGB-D camera. An overview of the DRBox-v2 architecture is shown in Figure 3. DRBox-v2 uses the VGG16 architecture **(Simonyan & Zisserman, 2014)** to extract the features. Meanwhile, to predict object class and location, DRBox-v2 took the convolutional layer's output from VGG16, which is layer conv3_3 and layer conv4_3.

Detection in DRBox-v2 began with model training, initiated with preprocessing of the annotated training data. This preprocessing aimed to obtain positive and negative samples from annotated images. Preprocessing was done by calculating the $\mathrm{ArIoU}_{180}$ of the RBox and

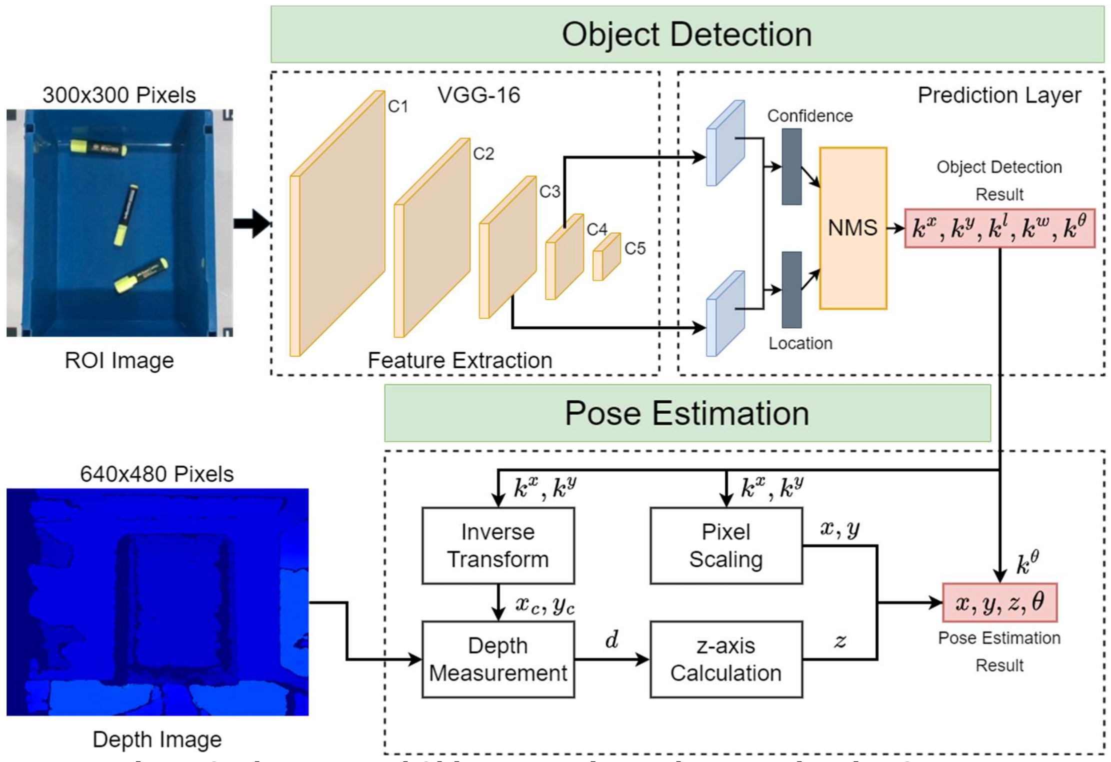


**Figure 3. The Proposed Object Detection and Pose Estimation System**

the prior RBox. The prior RBox was similar to the anchor box in YOLO **(Redmon et al., 2016)** but had an orientation parameter for rotating the box to a certain angle. The prior RBox will be placed at a coordinate point with a specific stride interval during preprocessing. Then, RBoxes were generated on specific parameters of length, width, and orientation, according to the object's size. Furthermore, the $\mathrm{ArIoU}_{180}$ will be calculated using Equation (3), where $A$ was the RBox created and $B$ was the annotated RBox, while $\hat{A}$ denoted to an RBox A with predetermined length, width, and orientation parameters. Equation (3) calculated $\mathrm{ArIoU}_{180}$ by ignoring the orientation of the object's head and tail, the overlap was only calculated at a normalized angle from 0° to 180°. However, if the object's head and tail orientation were essential, then the absolute operator in Equation (3) can be omitted. In this preprocessing, positive samples were selected from the RBox with the $\mathrm{ArIoU}_{180}$ exceeding the $T_{match}$ and from the RBox with the highest $\mathrm{ArIoU}_{180}$.

$$\mathrm{ArIoU}_{180}(A,B) = \frac{\mathrm{area}(\hat{A} \cap B)}{\mathrm{area}(\hat{A} \cup B)} |\cos(\theta A - \theta B| \tag{3}$$

After preprocessing, the model will be trained to minimize two loss functions: confidence loss ($L_{conf}$) and location loss ($L_{loc}$). Confidence loss is a function of minimizing object classification errors, while location loss is a function of calculating the regression error of the predicted RBox. These loss functions are then combined to obtain the total loss from Equation (4), where $N_1$ represented the number of positive samples, and $N_2$ denoted the number of negative samples. Meanwhile, the $L_{conf}$ and $L_{loc}$ are obtained from Equations (5) and (6). The $L_{conf}$ combines Hard Negative Meaning (HNM) **(Xuan et al., 2020)** and Focal Loss (FL) **(Lin et al., 2020)** functions to achieve good results. In Equation (5), variable $I$ defined a matrix indicating that the prior RBox matches the ground truth RBox, and $c$ is the probability vector of the predicted class. In Equation (6), $K$ represents the prediction, and $G$ is the ground truth. There is a $\mathrm{smooth}_{L1}$ function, which is an extension of the $L1_{norm}$ in $L_{loc}$. Function $\mathrm{smooth}_{L1}$ can be calculated using Equation (7).

$$L(I,c,r,g) = \frac{1}{N_1 + N_2} L_{conf}(I,c) + \frac{1}{N_2} L_{loc}(I,K,G) \tag{4}$$

$$L_{conf} = -\sum_{i \in \mathrm{Pos}} (1-c_i)^{\gamma} \log(c_i) - \sum_{j \in \mathrm{Hard\_Neg}} c_j^{y} \log(1-c_j) \tag{5}$$

$$L_{loc} = \sum_{i \in \mathrm{Pos}} \sum_{j} \sum_{m \in \{x,y,w,l,\theta\}} I_{ij} \mathrm{smooth}_{L1}(k_j^m - g_j^m) \tag{6}$$

$$\mathrm{smooth}_{L1}(s) = \begin{cases} 0.5s^2, & \text{if } |s| < 1 \\ |s| - 0.5, & \text{else} \end{cases} \tag{7}$$

After going through the training, DRBox-v2 can be used for inference to process images in a feedforward manner with the weights and biases obtained. The prediction results will be post-processed by selecting an RBox with a particular probability and choosing one RBox if there are stacked RBoxes with the same predicted class. The predicted RBox will be selected if the probability value exceeds $T_{conf}$. In addition, if there are stacked RBoxes with the same class, a Non-Maximum Suppression (NMS) **(Hosang et al., 2017)** will be performed to take the one with the highest probability. The final result will generate a vector $[k^x \quad k^y \quad k^l \quad k^w \quad k^\theta]^T$, where $(k^x, k^y)$ defines the RBox's location in the camera frame, $k^l$ and $k^w$ define RBox length and width, respectively, and $k^\theta$ defines the RBox heading angle.

### 2.4 Pose Estimation
In the case study of bin-picking, the object's pose must be represented against the world coordinate to facilitate the calculation of the inverse kinematics **(Du et al., 2021)**. Therefore, we defined the world coordinate at the midpoint of the ArUco marker ID0 so that the object's location in a three-dimensional coordinate will be represented concerning the origin point of this frame coordinate. As seen in Figure 1 (b), the $x$-axis points to the right, the $y$-axis points down, and the $z$-axis points out of the image. Moreover, the orientation is represented according to the object's rotational direction on a particular axis.

We combined distance measurement from depth image with the detection result of the object and ArUco to estimate the object's pose. As Figure 1 (b) explained, we placed ArUco markers 600 mm apart in each square corner. Moreover, we processed the image to take a 300×300 pixels ROI. From the known distance between ArUco and ROI dimensions, the scale factor from pixels to millimeters ($s$) can be obtained. We used $s$ for converting the object's location in the camera frame to the object's position ($x, y$) in the world coordinate. Meanwhile, to obtain the object's position on $z$-axis, we performed further data processing on the depth image. The depth image represented the distance in terms of gray intensity in millimeters. So, the object's distance from the camera can be determined by taking the gray intensity at the object's center point. However, because we detected objects from a 300×300 ROI resulting from transformation, the object's coordinate must be transformed back to the 640×480 depth image coordinate. This inverse transformation was described by Equation (8), where $k^x$ and $k^y$ are the object's midpoints in a 300×300 ROI. Meanwhile, $T$ is the transformation matrix previously described in Equation (1). This inverse transformation produces $x_c$ and $y_c$, which defined the object location of the object in the 640×480 image. After getting $x_c$ and $y_c$, we took the gray value of the depth image at those coordinate pixels. We symbolized this value by $d$, defined by estimating the distance from the camera to the object's surface. The final result of the object's position was represented by vector $p$ in Equation (9). There is a constant parameter that defines the distance between the table surface to the camera lens ($h_c$). On the other hand, for determining object orientation, the DRBox-v2 output $k^\theta$ can be taken directly as object orientation. The object's orientation representation in a rotation matrix is described by Equation (10), where $R_z$ defines the object's orientation in the $z$-axis.

$$\begin{bmatrix} x_c \\ y_c \\ 1 \end{bmatrix} = T^{-1} \cdot \begin{bmatrix} k^x \\ k^y \\ 1 \end{bmatrix} \tag{8}$$

$$p = \begin{bmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} k^x \\ k^y \\ h_c - d \end{bmatrix} \tag{9}$$

$$R_z = \begin{bmatrix} \cos(k^\theta) & -\sin(k^\theta) & 0 \\ \sin(k^\theta) & \cos(k^\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{10}$$

### 2.5 Performance Evaluation
We evaluated two critical performances during the test: object detector accuracy and pose estimator accuracy. The testing on object detector performance adopts evaluation metrics on the PASCAL VOC dataset **(Everingham et al., 2010)**, which measured the Mean Average Precision ($mAP$). Because of in this study only one class object exists, so the measured metric is the Average Precision ($AP$). The calculation of $AP$ begins by calculating the True Positive ($TP$), True Negative ($TN$), and False Positive ($FP$) results from the object detector. The calculation is carried out on images in the test data set. In addition, this calculation is carried

out when the object detector is set by a particular value of $T_{matc}$ . After knowing the $TP$, $TN$, and $FP$, the precision ($P_d$) and recall ($R_d$) rates can be calculated using Equations (11) and (12). Furthermore, the precision and recall rates are used to calculate $AP$ by using the 11-point interpolation, which refers to the **(Everingham et al., 2010)** method.

$$P_d = \frac{TP}{TP + FP} \tag{11}$$

$$R_d = \frac{TP}{TP + FN} \tag{12}$$

The pose estimator performance is measured by calculating how much pose error was generated. This pose error can be divided into two, namely: 1. position error and 2. orientation error. The position error is obtained by finding the Euclidean distance from the actual objects to the predicted object. Since the object's position is represented in three-dimensional coordinates, the position error ($d_{err}$) could be calculated by Equation (13), where the vector $p$ defines the actual position and the vector $\hat{p}$ is the predicted position. Meanwhile, the orientation error can be measured by calculating the difference between the actual and predicted object orientation on the $z$-axis. The measurement takes place on the $z$-axis only because our method is limited in predicting orientations on the $x$-axis and $y$-axis. This orientation error ($\theta_{err}$) can be calculated using Equation (14), while $\theta$ defines the actual object's heading and $\hat{\theta}$ denotes the predicted object's orientation.

$$d_{err}(p, \hat{p}) = \sqrt{\sum_{i=1}^{n} (p_i - \hat{p_i})^2} \tag{13}$$

$$\theta_{err} = \operatorname{asin}(\sin(|\hat{\theta} - \theta|)) \tag{14}$$

## 3. RESULT AND DISCUSSION

This section will explain the obtained results of this research. First, we will describe the parameter settings for the experiment. Next, the results of starting with the model training will be explained in detail. Then we explain the level of accuracy of the object detector and the pose estimator.

### 3.1 Parameter Setting for Experiment

During the experiment, we set parameters in the prior RBox configuration. The baseline model sets these parameters to various lengths and widths. While the orientation interval is set to a constant value of 30°. These settings will generate many prior RBoxes, which take a long preprocessing time. Then, we finetuned these configurations by setting the constant prior RBox length and width. This determination is based on the object type we detect, one type with a constant dimension, so the object's length and width in the camera view will not change too much. We set the prior RBox length and width to 15 and 90 pixels. These values were determined by analyzing the average prior RBox size that has been annotated, and then the parameters are set by adding 5 pixels to each average. Additionally, we varied the configuration of RBox orientation from 10° to 30°, with 10° intervals. Furthermore, we varied the IoU threshold to determine which configuration yielded the best accuracy.

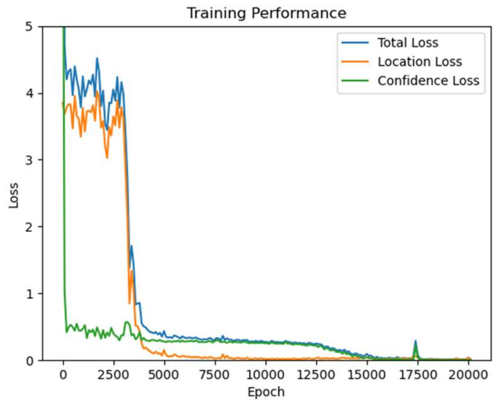
## 3.2 Object Detection Training Performance

The results of the DRBox-v2 training with the settings described in the prior subsection are illustrated below. First, we described the effect of the orientation interval ($\theta_{step}$) to the preprocessing time and the preprocessing time in Table 1. Through the data listed in Table 1, it can be concluded that the smaller the $\theta_{step}$ interval generated the more prior RBox. In effect, there is an increase in preprocessing time. As noted in Table 1, the preprocessing time when $\theta_{step}$ is set at 10° is approximately 68 minutes. Meanwhile, when the configuration is set to 30°, the preprocessing time becomes shorter, around 23 minutes.

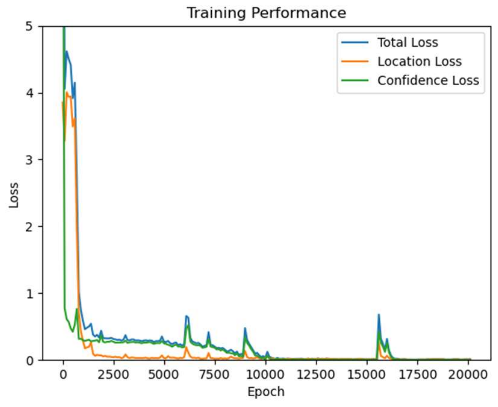
We then analyze the $\theta_{step}$ effect on the training performance. The analysis is carried out by storing historical loss functions during training. We store the total loss, confidence loss, and location loss, then plot these values into a graph to analyze their correlation to the number of epochs. As seen in Figure 4, the entire graph showed a significant decrease in the loss value

**Table 1. Effect $\theta_{step}$ to Total Prior RBoxes Generated and Preprocessing Time**

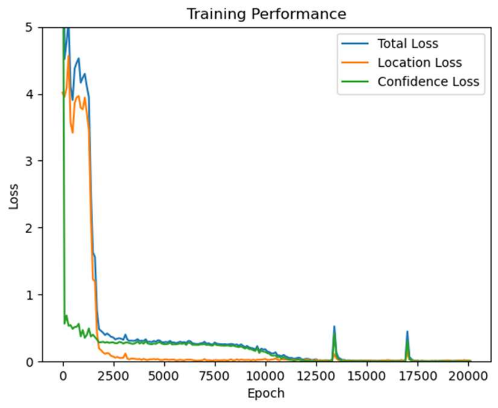
| $\theta_{step}$ | Total Prior RBoxes | Preprocessing Time |
|---|---|---|
| 10° | 254484 RBoxes | 68 Minutes |
| 20° | 127242 RBoxes | 34 Minutes |
| 30° | 84828 RBoxes | 23 Minutes |



**(a)**



**(b)**



**(c)**

**Figure 4. Learning Curve with (a) $\theta_{step}$ = 10°, (b) $\theta_{step}$ = 20°, (c) $\theta_{step}$ = 30°**

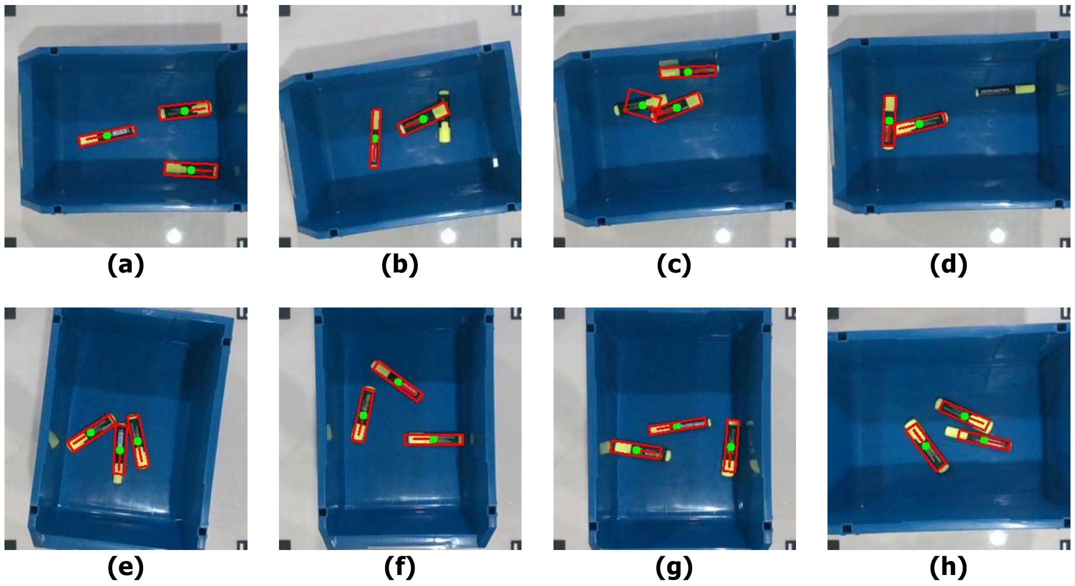


**Figure 5. Object Detection Result from Region of Interest**

after several training epochs. The whole graph showed that when the training reaches its maximum epochs, the overall loss drops to 0. However, the decrement speed was slightly different for each $\theta_{step}$. It can be seen in Figure 4 (b) when the $\theta_{step}$ is set to 20°, the loss decreased very quickly to 0.461 from its initial value of 12.716 before the epoch reached 2500. Compared to the training curve in Figure 4 (a), when the $\theta_{step}$ is set to 30°, the loss value decreases slightly. A significant decrease occurred in the 4100th epoch, with a total recorded loss value of 0.490, while at the beginning iteration, this value was 12,634. From the graph comparison, it can be concluded that $\theta_{step}$ = 20° is the fastest model training configuration compared to other $\theta_{step}$ values.

## 3.3 Object Detection Evaluation Performance

The predicted objects from test images can be seen in Figure 5. This test is performed with the parameter $T_{conf}$ = 0.1 and $T_{match}$ = 0.3. Furthermore, we used a trained model with a configuration $\theta_{step}$ = 20° and visualized the predicted objects with a red RBox. The green dot in the middle RBox indicated the object's center point or location concerning the camera frame. Generally, the detector recognizes the objects, which is indicated by the RBox that marks the text markers in the image. In addition, the RBoxes faced precisely according to the text marker's orientation, indicating that text marker orientation has been correctly predicted. However, the system still failed when detecting overlapping objects, as shown in Figure 5 (b). In addition, to detect the overlapping objects, several objects failed to be detected, as shown in Figure 5 (d).

After testing the object detector functionality, then we evaluated the $AP$ metrics with several $T_{match}$ configurations. We tested the $T_{matc}$ in the range of 0.1 to 0.9, with a change interval of 0.1. Additionally, we calculated the $AP$ with different $\theta_{step}$ parameters, and described the results in Table 2. The symbol $AP^{\mathrm{IoU}=.1}$ in Table 2 describes that we measured the $AP$ using $T_{match}$ = 0.1 or using the $\mathrm{ArIoU}_{180}$ threshold equal to 0.1, as well as other symbols.

**Table 2. Resulted Average Precision Over Different ArIoU Threshold**

| $\theta_{step}$ | $AP^{\text{IoU}=.1}$ | $AP^{\text{IoU}=.2}$ | $AP^{\text{IoU}=.3}$ | $AP^{\text{IoU}=.4}$ | $AP^{\text{IoU}=.5}$ | $AP^{\text{IoU}=.6}$ | $AP^{\text{IoU}=.7}$ | $AP^{\text{IoU}=.8}$ | $AP^{\text{IoU}=.9}$ |
|---|---|---|---|---|---|---|---|---|---|
| 10° | 0.604 | 0.504 | 0.504 | 0.504 | 0.504 | 0.492 | 0.468 | 0.305 | 0.014 |
| 20° | **0.740** | 0.590 | 0.590 | 0.590 | 0.586 | 0.583 | 0.571 | 0.331 | 0.024 |
| 30° | 0.687 | 0.544 | 0.544 | 0.544 | 0.535 | 0.522 | 0.508 | 0.281 | 0.010 |

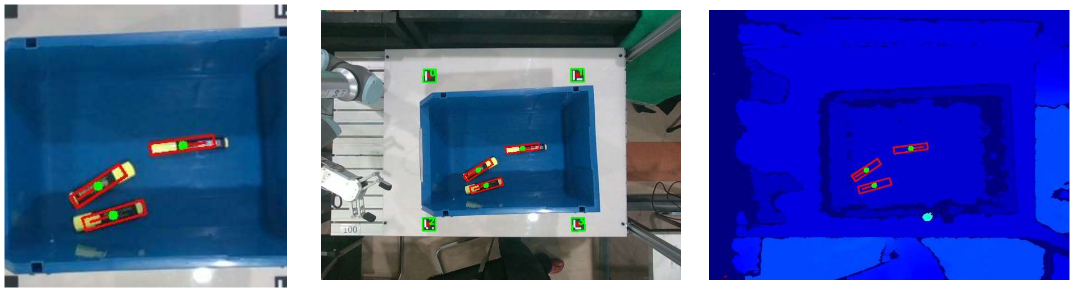

**(a)** **(b)** **(c)**

**Figure 6. Resulted Inverse Transformation on The Depth Image**

As shown in Table 2, we obtained the maximum $AP$ while using the $\theta_{step}$ = 20°. We found that the $AP$ in the overall $T_{match}$ were higher than using the $\theta_{step}$ = 10° or $\theta_{step}$ = 30°. The $\theta_{step}$ which has the worst performance is $\theta_{step}$ = 10°, and resulted in a smaller $AP$ in the overall $T_{mat}$ threshold compared to the other $\theta_{step}$ parameters. The data in Table 2 shows that the greater $T_{match}$ will be generated a higher $AP$. This condition happened in all $\theta_{step}$ configurations. A significant decrement will occur if the $T_{match}$ is greater than or equal to 0.8. Conversely, if the $T_{match}$ is set to high, it will cause the decrement in $AP$. We obtained the best performance using the $\theta_{\text{step}}$ = 20° and $T_{match}$ = 0.1, with an $AP$ score of 0.740.

## 3.4 Pose Estimation Accuracy

In testing pose estimation, first, we verify the pose estimation approach presented in Section 2.3. Then, we visualized the verification results in Figure 6. Figure 6 (a) shows the object detection results in the ROI, which is a 300×300 image transformed from the ArUco marker coordinates. After getting the object's center point and size in the image, the object's coordinates are transformed back into 640×480 pixels depth and original color image. Figure 6 (b) shows that the markers still precisely mark objects after the inverse transformation, even in the raw images. Therefore, we also use the inverse transformation to mark the object's location in the depth image. Next, we use the object's center point in the depth image to measure the distance from the camera to the object. The measurement results are then used to estimate the object's position concerning the $z$-axis.

We randomly vary the object's pose to test the pose estimator's accuracy by providing a wedge under the bin to vary the object's height so that the object's height to the table varies. The variation object height samples are 19 mm, 55 mm, 65 mm, 83 mm, and 139 mm. We then calculated the position and orientation error values for all detected objects from the 100 sample images. As a result, we got an average error of 13.36 mm for positional error, while for orientation error, the average error of 0.75°. Meanwhile, the maximum error we got is as follows; the maximum position error is 158.86 mm, while the maximum orientation error is 1.56°. The most significant factor contributing to the positional error is the measurement result on the $z$-axis. The average error on the $z$-axis is 10.80 mm, while on the $x$-axis it is 3.55 mm, and on the $y$-axis it is 3.14 mm.

**Table 3. Pose Estimation Result from 5 Sample Images**

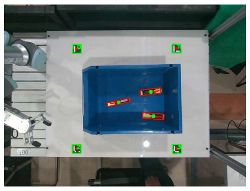
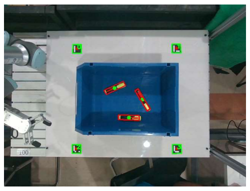
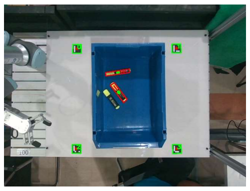
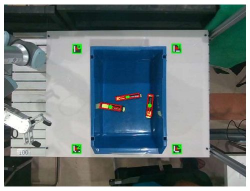
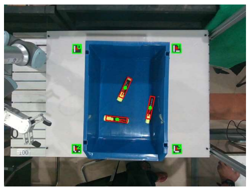
| Image | Actual Object Pose | | | | Estimated Object Pose | | | |
|---|---|---|---|---|---|---|---|---|
| | $x$ (mm) | $y$ (mm) | $z$ (mm) | $\theta$ (deg) | $x$ (mm) | $y$ (mm) | $z$ (mm) | $\theta$ (deg) |
|  | 253 | 323 | 19 | 167.39 | 253 | 323 | 13 | 168.08 |
| | 448 | 261 | 19 | 171.98 | 445 | 262 | 13 | -6.72 |
| | 462 | 409 | 19 | 4.01 | 461 | 409 | 15 | 3.28 |
|  | 320 | 418 | 55 | 1.72 | 316 | 418 | 47 | 1.55 |
| | 398 | 310 | 55 | 149.06 | 397 | 309 | 46 | 238.65 |
| | 229 | 240 | 55 | 159.95 | 228 | 239 | 43 | 154.75 |
|  | 253 | 264 | 65 | 141.04 | 254 | 265 | 72 | 232.66 |
| | 251 | 144 | 65 | 170.26 | 248 | 146 | 59 | -8.25 |
| | 209 | 306 | 65 | 143.33 | X | X | X | X |
|  | 438 | 349 | 83 | 6.30 | 440 | 344 | 75 | 94.82 |
| | 221 | 355 | 83 | 9.74 | 204 | 353 | 241 | 8.33 |
| | 314 | 294 | 83 | 173.70 | 305 | 293 | 74 | 170.52 |
|  | 286 | 249 | 139 | 24.64 | 291 | 238 | 128 | 114.18 |
| | 439 | 369 | 139 | 10.31 | 441 | 358 | 135 | 101.53 |
| | 239 | 425 | 139 | 6.88 | 244 | 428 | 135 | 5.80 |

Table 3 presents the pose estimator results for ten different images. In this test, we found a system failure to detect objects in the third image, which caused the object's pose not to be estimated. Moreover, we found that the most significant difference between the predicted and actual positions occurs on the $z$-axis. For example, in the first image, when the object is placed at position $z$-axis = 19 mm, the estimated object's positions are 13 mm and 15 mm. This difference is extensive when compared to the $x$-axis and the $y$-axis. We suspect that the factor of RGB-D depth measurement accuracy is causing this problem.

Regarding orientation, sometimes, the estimator generates a flipped angle. For example, in the second row of the first picture, while the actual object orientation is 171.98°, an estimator shows the angle of -6.72°. Although the values differ, the final result will be the same after normalizing the angle. This condition is evidenced by the first image visualization that shows that RBox correctly marks the object orientation in the image even if the predicted orientation is flipped. This condition also appears in the third image of the second predicted object.

## 4. CONCLUSION

We proposed a deep learning-based object detector DRBox-v2 combined with ArUco markers to detect and estimate object poses in a bin using an RGB-D camera. The results of this study can be concluded that the orientation interval parameter in DRBox-v2 ($\theta_{step}$) and also the IoU threshold ($T_{match}$) has a significant influence on the Average Precision ($AP$) value of the detection results. Furthermore, the $\theta_{step}$ parameter also affects the convergence speed during model training. The use of $\theta_{step}$ = 20° results in faster convergence during the training compared to other $\theta_{step}$ parameters. Meanwhile, in terms of inference, the experiments we carried out showed that the best parameters that produced the highest $AP$ were $\theta_{step}$ = 20° and $T_{match}$ = 0.1. At the model inference, the configurations resulted in $AP$ = 0.740. Meanwhile, from the pose estimator, we got an average position error of 13.36 mm. As for orientation, we obtained an average orientation error of 0.75°. From the experimental results, there was a very significant error in the position estimation on the $z$-axis, which was the result of depth image processing. Furthermore, we found that the object detector sometimes fails to detect objects, especially in stacked objects. In future works, we will investigate these issues so that the pose estimation can estimate the object's position more precisely.

## ACKNOWLEDGEMENT

This research is part of the Research and Community Service for Vocational Higher Education research grant. We thank the Direktorat Jenderal Pendidikan Vokasi, Politeknik Negeri Batam, and Barelang Robotics and Artificial Lab (BRAIL) for providing the equipment and facilities for this research.

## REFERENCES

An, Q., Pan, Z., Liu, L., & You, H. (2019). DRBox-v2: An Improved Detector With Rotatable Boxes for Target Detection in SAR Images. *IEEE Transactions on Geoscience and Remote Sensing*, *57*(11), 8333–8349.

Dewi, I. A., Kristiana, L., Darlis, A. R., & Dwiputra, R. F. (2019). Deep Learning RetinaNet based Car Detection for Smart Transportation Network. *ELKOMIKA: Jurnal Teknik Energi Elektrik, Teknik Telekomunikasi, & Teknik Elektronika*, *7*(3), 570.

Du, G., Wang, K., Lian, S., & Zhao, K. (2021). Vision-based robotic grasping from object localization, object pose estimation to grasp estimation for parallel grippers: a review. *Artificial Intelligence Review*, *54*(3), 1677–1734.

Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., & Zisserman, A. (2010). The Pascal Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision*, *88*(2), 303–338.

Hosang, J., Benenson, R., & Schiele, B. (2017). Learning Non-maximum Suppression. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, *2017-Janua*, 6469–6477.

Kalaitzakis, M., Cain, B., Carroll, S., Ambrosi, A., Whitehead, C., & Vitzilaios, N. (2021). Fiducial Markers for Pose Estimation. *Journal of Intelligent & Robotic Systems*, *101*(4), 71.

Kozák, V., Sushkov, R., Kulich, M., & Přeučil, L. (2021). Data-Driven Object Pose Estimation in a Practical Bin-Picking Application. *Sensors*, *21*(18), 6093.

La Delfa, G. C., Monteleone, S., Catania, V., De Paz, J. F., & Bajo, J. (2016). Performance analysis of visualmarkers for indoor navigation systems. *Frontiers of Information Technology & Electronic Engineering*, *17*(8), 730–740.

Lee, S., & Lee, Y. (2020). Real-Time Industrial Bin-Picking with a Hybrid Deep Learning-Engineering Approach. *2020 IEEE International Conference on Big Data and Smart Computing (BigComp)*, 584–588.

Lin, T.-Y., Goyal, P., Girshick, R., He, K., & Dollar, P. (2020). Focal Loss for Dense Object Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *42*(2), 318–327.

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 779–788.

Sartika, E. M., Sarjono, R., & Chrisophras, H. X. (2019). Sistem Pick and Place Dua Derajat Kebebasan menggunakan Metoda Regresi. *ELKOMIKA: Jurnal Teknik Energi Elektrik, Teknik Telekomunikasi, & Teknik Elektronika*, *7*(3), 521.

Simonyan, K., & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*.

Soltan, S., Oleinikov, A., Demirci, M. F., & Shintemirov, A. (2020). Deep Learning-Based Object Classification and Position Estimation Pipeline for Potential Use in Robotized Pick-and-Place Operations. *Robotics*, *9*(3), 63.

Wahyuni, E. S., & Hendri, M. (2019). Smoke and Fire Detection Base on Convolutional Neural Network. *ELKOMIKA: Jurnal Teknik Energi Elektrik, Teknik Telekomunikasi, & Teknik Elektronika*, *7*(3), 455.

Wong, C.-C., Tsai, C.-Y., Chen, R.-J., Chien, S.-Y., Yang, Y.-H., Wong, S.-W., & Yeh, C.-A. (2022). Generic Development of Bin Pick-and-Place System Based on Robot Operating System. *IEEE Access*, *10*, 65257–65270.

Wubben, J., Fabra, F., Calafate, C. T., Krzeszowski, T., Marquez-Barja, J. M., Cano, J.-C., & Manzoni, P. (2019). Accurate Landing of Unmanned Aerial Vehicles Using Ground Pattern Recognition. *Electronics*, *8*(12), 1532.

Xing, B., Zhu, Q., Pan, F., & Feng, X. (2018). Marker-Based Multi-Sensor Fusion Indoor Localization System for Micro Air Vehicles. *Sensors*, *18*(6), 1706.

Xuan, H., Stylianou, A., Liu, X., & Pless, R. (2020). Hard Negative Examples are Hard, but Useful. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics): Vol. 12359 LNCS* (pp. 126–142).

Yan, W., Xu, Z., Zhou, X., Su, Q., Li, S., & Wu, H. (2020). Fast Object Pose Estimation Using Adaptive Threshold for Bin-Picking. *IEEE Access*, *8*, 63055–63064.

Yu, J., Jiang, W., Luo, Z., & Yang, L. (2021). Application of a Vision-Based Single Target on Robot Positioning System. *Sensors*, *21*(5), 1829.

Zhuang, C., Wang, Z., Zhao, H., & Ding, H. (2021). Semantic part segmentation method based 3D object pose estimation with RGB-D images for bin-picking. *Robotics and Computer-Integrated Manufacturing*, *68*, 102086.