Mechanical parts picking through geometric properties determination using deep learning

YJ Lee, SH Lee and DH Kim

Abstract
In this study, a system for automatically picking mechanical parts required in the industrial automation field was proposed. In particular, using deep learning, bolts and nuts were recognized and geometric information of these parts was extracted. By applying YOLOv3 specialized in high recognition rate and fast processing speed, the recognition of target object, location, and postural information were obtained. The geometric information for the bolt can be obtained by creating two bounding boxes and calculating the orientation vector formed by these center values of two bounding boxes after successfully detecting two individual bounding boxes. Moreover, to obtain more precise geometric information on bolts and nuts, image distortion compensation on the detected object was done after detecting the center value of the bolt and nut through YOLOv3. Based on this result, it was proven that an automatic picking of the mechanical parts using a five-axis robot was successfully implemented.

Keywords
Parts picking robot, deep learning, robot vision, YOLO

Date received: 3 March 2021; accepted: 2 January 2022

Topic: Vision Systems
Topic Editor: Alessandro Di Nuovo
Associate Editor: Ning Wang

Introduction
In modern society, factory automation by robots is being conducted extensively. Robots are being used in various process fields such as manufacturing, processing, packaging, and assembly, and processes that involve manual work gradually disappear. In particular, robots have become essential in automated processes that require high load-bearing capacity and accuracy. Recently, intelligent robots combined with rapidly developing artificial intelligence have attracted great attention and many studies are being conducted. However, in most industries, industrial robots are mainly used to grab or move a target object with a fixed position and posture. This is because the level or cost of technology required to build an intelligent robot system for autonomously handling objects in arbitrary positions is high. Additionally, since the existing automation method requires the complete design of the entire process system from the initial process design stage, it is difficult to respond when addition and correction of the intermediate process is required. In particular, if a variable process structure such as a smart factory is applied more in the future in the industry, the existing automation method will become more obsolete, so it is urgent to secure artificial intelligence-based process technology that can be implemented with relatively easy technology and low cost.
Therefore, technologies such as robot vision have been developed to solve this problem, which refers to a technology that combines a visual sensor with a robot and gives the robot the ability to recognize and identify objects through images.

However, to solve complex problems such as bin picking using robot vision, it is necessary to estimate the 3D position and posture of an object, so a high-performance 3D camera sensor is essential. The requirement to use an expensive 3D sensor is a significant obstacle to building an economical automation process, and it is the biggest reason that robot vision technologies are rarely adopted in the actual field even though highly useful robot vision technologies are being developed in various ways. To compensate for this problem, it is necessary to obtain the most accurate object information (class, position, orientation, etc.) using a relatively inexpensive 2D camera. Previously, this work was mainly implemented through the classical image processing technology, but in the 2010s, image processing using deep learning, which is robust to changes in the surrounding environment, has been mainly performed.

Among the numerous deep learning models, the object detection model performs classification and location detection at the same time. The object detection model is a popular technology because it is useful in real life and professional fields, and it is fast and accessible. So far, starting with the first R-CNN series (R-CNN, Fast R-CNN, Faster R-CNN), various models such as YOLO, SSD, and RetinaNet have been developed, and research on the development of models with better performance are in progress.

However, the object detection model is limited in its application to the actual automated process system in that it can give object class and position information but cannot provide orientation information. For this reason, object detection has been mainly used in cases where only approximate information of an object in the screen such as a security camera or a vehicle black box is required. Therefore, it has been known that the sensing system using only the object detection model is so difficult to apply in a process aimed at accurately grasping and picking up the posture of an object.

Generally, the method of obtaining orientation information through point cloud application and additional sensor fusion was considered, and the object detection model was only applied as a supplementary role. The above technologies require specialized technology and computing cost that cannot be compared to the use of a deep learning-based object detection model alone. Therefore, it is time for a simpler and affordable solution.

To solve the shortcomings of the existing deep learning-based object detection model that it cannot determine the orientation of an object, in this study, a new method that each part of the object to be detected is learned as a different object, and the orientation of the object is obtained through the position information of the separated object is presented.

Using object detection model, by giving different labels for each part of an object, which has a non-uniform feature, different bounding boxes for each part are found through deep learning. Consequently, an orientation vector that connects the center positions of each bounding box can be obtained. In this way, the proposed scheme can be used to effectively acquire the center position and orientation information of machine parts such as bolt and nut, through this scheme automatic machine part picking process can be completed. Some works have been reported on bin-picking system using deep learning. These works mainly focused to classify and estimate the size of the target object by creating a bounding box without specifying the detailed geometric information on the object such as posture.

In this study, using YOLOv3, a commercially available deep learning tool, we propose a method to find the center value and orientation of an object even when the shape is not uniform such as a bolt. It completes the automatic bolt and nut-picking system, which is different from the general bin picking that simply recognizes and picks up a target object. Afterward, the picking and moving operation of the bolts and nuts were directly implemented in a five-axis robot using an inverse kinematics solution. The reliability of the proposed method was verified through repeated experiments after placing the bolts and nuts randomly on the plate. Moreover, the precise center and posture values of the detected target object to accurately pick it up were determined by correcting the distortion of the image that is inevitable in a cheap monocular camera.

### Deep learning and object identification

In this work, we propose an automatic bolt and nut-picking system (Figure 1) that recognizes a target object from bolts and nuts randomly placed on a flat plate, determines the center position and direction of the target object, and then picks up and moves it to the designated position. YOLOv3, a well-known object detection tool, is adopted here but special scheme to find the object and determine the geometric information of the target object is proposed. The input data set comes from the images captured by a camera installed above the robot.

### Object identification

In the training process, M8 bolt and M8 nut images taken by the camera were used. The camera is the oCam-5cro-u-m model of WITHROBOT, a South Korean company, with a resolution of 1280 \( \times \) 720, and the five-axis robot is a low-cost robot driven by Dynamixel servo motors from Robotis company.

PyTorch-based YOLOv3 (eriklindernoren, github) was used for image training and testing. The YOLO series
(YOLO, YOLOv2, YOLOv3, etc.) is a deep learning model for object detection widely used in real-time image processing because it provides high-efficiency results in learning time through an optimized network.

The YOLO\textsuperscript{20} series divides image into $N \times N$ grids and extracts classification and bounding box information for each grid. Naturally, the loss function also reflects both the classification and the bounding box. For more details about the loss function, refer to the study of Redmon and Farhadi.\textsuperscript{20}

YOLOv3\textsuperscript{21} used in this study is further developed from the existing YOLO and performs object detection with three-scale layers. YOLOv3 creates three layers of $13 \times 13$, $26 \times 26$, and $52 \times 52$ grid scale by resizing input image of arbitrary size into $416 \times 416$ and then conducting convolution through Darknet-53 CNN structure. Each of the three layers is responsible for capturing large, medium, and small objects. Finally, the following output of tensor $T$ is derived for each grid as shown in equation (1)

$$T = \begin{bmatrix} t_x & t_y & t_w & t_h & p_o & p_1 & p_2 & \cdots & p_c \end{bmatrix} B$$ (1)

Here, $(t_x, t_y)$ are the center coordinates, $(t_w, t_h)$ are the width and height of the bounding box with respect to the image plane $(x, y)$, and $p_o$ is a confidence score indicating the probability that an object exists in the corresponding bounding box. $p_1 \sim p_c$ is the probability that the corresponding object will be classified into each class for a total of $c$ classes. In the case of YOLOv3, three bounding boxes per grid can be predicted, so $B = 3$. In other words, the output for one grid contains coordinates information and class probability for three bounding boxes.

Finally, only bounding box with a confidence score higher than the threshold specified by the user is displayed on the screen with the highest probability of the class name. Figure 2 shows the process of forming bounding boxes using YOLOv3.

The input image size is $1280 \times 720 \times 3$. The threshold of confidence score was set to 0.85. The threshold for the NMS (non-maximum suppression)\textsuperscript{22} function that controls the overlapping capture of the same bounding box was set to 0.1. Since the target objects of this work are M8 size bolts and nuts, training data sets using images of M8 bolts and nuts were created and used. Therefore, the total number of classes is four, including three classes on the bolt and one class on the nut.

The reason that the number of classes is four is aiming to finding the bolt orientation. To grip the bolt accurately in a robot gripper, information on the orientation is essential, but the YOLOv3 result only informs the center position $(x, y)$, width $(w)$, and height $(h)$ of the object through a bounding box, thus, the orientation of the object is unknown.

In this work, rather than finding the bolt by YOLOv3, it is divided into three classes: Whole bolt, Bolt head, and Bolt tail. On the other hand, the shape of the nut circular, so one class is enough for nut detection and its geometric information.

In learning, as shown in Figure 3, the bolt head was designated as Bolt head, the screw part as Bolt tail, and the
entire bolt as Whole bolt. Unlike general objects, the position and orientation of the bolt are crucial for the robot to pick it up for the following assembly process. The center value of the bolt head is the position at which the robot gripper should move to pick up the bolt, and the orientation of the bolt is necessary for the gripper to pick up the bolt head in the width direction. Here, the orientation can be derived using the vector connecting the bolt tail center and the bolt head center if two bounding boxes of Bolt head and Bolt tail are successfully found through YOLOv3. Lastly, Whole bolt class plays the role in properly matching the bolt head and bolt tail.

When the gripper is placed as shown in Figure 4, the orientation of the gripper can be determined through equations (2) to (4)

\[
\vec{n} = \vec{o} \times \vec{a}
\]

\(\vec{n}\), \(\vec{o}\), and \(\vec{a}\) is the unit vector of the gripper (end-effector) relative to the reference coordinates \((X, Y, Z)\), and \(\vec{v}_h\) is the unit vector for the gripper to take a posture to grasp the bolt, which can be obtained by subtracting the bolt head center position from the bolt tail center. \(\vec{a}\) is perpendicular to the work surface \((XY\) plane\), \(\vec{o}\) is the opposite of \(\vec{v}_h\). When \(\vec{o}\) and \(\vec{a}\) are determined, \(\vec{n}\) is the cross product of two vectors to open and close the gripper. On the other hand, as stated before, the orientation of the nut is unimportant due to its circular shape, so picking of the nut is possible with keeping the initial orientation of the gripper.

In this work, a data set for learning was produced by taking 1000 images of bolts and nuts placed randomly on the floor. Therefore, no other objects were put in the learning data, and all of the data set were taken directly with a camera. Thus, six to eight bolts and nuts were included per image, increasing the learning efficiency compared to the number of image data.

**YOLO Training**

At the first stage, labeling was conducted using YOLOv3 label-master (tzutalin, github).\(^{23}\) Annotation was created by designating the class and size for each bolt and nut in image as shown in Figure 5. As previously explained, there are four classes in the bolt: Whole bolt, Bolt head, Bolt tail, and one class for the nut.

In addition, in the actual robot work, a human hand or a robot gripper could enter the work space. Therefore, to recognize only bolts or nuts, a data set including externally intervened objects was used. If the hand or gripper is not labeled within the data set, as shown in Figure 6, YOLOv3 determines it as an object not to detect and thus learns not to create a bounding box.

After that, data augmentation was performed to secure more input data set. Using the imgaug library (aleju/imgaug),\(^{24}\) five options were applied: Hue value change, brightness change, contrast, blur, and dropout (Figure 7).
Here, hue change was applied in common to other four argumentation. At this time, as shown in Figure 8, the process was repeated for every 100 raw data, and learning time was saved by producing the next data while the previous data were being learned. Finally, the existing 1000 image data set was amplified to 5000 through image augmentation.

**Camera calibration**

Next, in order for the robot to accurately pick up the bolt and nut placed on the floor and move it to the designated location, the 2D coordinates (x, y) for the object, which is obtained by performing object detection and determination of geometric information, should be converted into 3D coordinates based on the reference coordinates (Figure 9). The transformation relationship between image frame and reference frame is summarized in the form of camera equation (5) below.

\[
\begin{bmatrix}
    x \\
    y \\
    1
\end{bmatrix}
= \begin{bmatrix}
    f_s & \text{skew} & c_x & 0 \\
    0 & f_y & c_y & 0 \\
    0 & 0 & 1 & 0
\end{bmatrix}
\begin{bmatrix}
    R_{3 \times 3} & t_{3 \times 1} \\
0_{1 \times 3} & 1
\end{bmatrix}
\begin{bmatrix}
    X \\
    Y \\
    Z \\
    1
\end{bmatrix}
\]

(5)

In equation (5), the values in \( P_{c, \text{avg}} \) is the camera’s intrinsic parameters, representing the transformation between the image frame and the camera frame. The elements of \( M_r \) is the camera’s extrinsic parameters, indicating the transformation between the camera frame and the
reference frame. Each component of $I_{P_{c, \text{avg}}}$ and $^cM_r$ is shown on the last term in equation (5).

$R_{3 \times 3}$ is the rotational matrix, $t_{3 \times 1}$ is the translational vector, $f_x$ and $f_y$ are the focal lengths for the $x$ and $y$ components, respectively, $c_x$ and $c_y$ are the center points for the $x$ and $y$ coordinates, respectively, and skew $\alpha f_x$ is the asymmetry coefficient, which is a value that occurs when the image is tilted due to a precision problem during camera manufacturing, and it is zero in most cases. Finally, $s$ is the scale factor.

Camera calibration\textsuperscript{25,26} is the process of obtaining internal and external parameters, and it was obtained using the Camera Calibrator app of MATLAB. The camera is located 400 mm above the floor, and the calibration was repeatedly performed by taking pictures of 13 checkboards. Through the process of substituting and verifying the parameters obtained through the camera calibration, the most appropriate calibration matrix was confirmed by equation (6)

\[
\begin{bmatrix}
X \\
y \\
z
\end{bmatrix} = \begin{bmatrix}
0 & -1774.9 & -656.9 & 0 \\
-1774.7 & 0 & -325.0 & 0 \\
0 & 0 & 1 & 401
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
z \\
1
\end{bmatrix}
\]

(6)

From the obtained calibration matrix, the internal and external parameters were determined as follows

\[
I_{P_{c, \text{avg}}} = \begin{bmatrix}
1774.9 & 0 & 656.9 & 0 \\
0 & 1774.7 & 325.0 & 0 \\
0 & 0 & 1 & 0
\end{bmatrix}
\]

(7)
\[
\begin{bmatrix}
0 & -1 & 0 & 0 \\
-1 & 0 & 0 & 170 \\
0 & 0 & -1 & 401 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

(8)

Even after the calibration is done successfully, there is no guarantee that the position and orientation of the detected object with respect to the reference frame is correct because the image captured to the camera is likely to be distorted as long as a cheap camera is employed. In particular, the image of the object far from the plate center is more distorted than the image of object placed at the center of the working plate. Here, to correct the position and orientation of the object associated with the distorted image, the correct position and orientation of the object were obtained using the lens distortion coefficient. 27

\[
x_{\text{distorted}} = x(1 + k_1r^2 + k_2r^4) \\
y_{\text{distorted}} = y(1 + k_1r^2 + k_2r^4)
\]

(9)

(10)

where \(k_1\) and \(k_2\) are lens distortion coefficients, which are obtained from internal parameters during the calibration process and are unique values of the lens regardless of resolution. \(x_{\text{distorted}}\) and \(y_{\text{distorted}}\) are distorted values expressed as a pixel location obtained through object recognition, and \(x\) and \(y\) are distortion-corrected values that is also given as a pixel location. \(r\) is the shortest distance from the camera origin to the corresponding pixel, and it is obtained from \(r^2 = x^2 + y^2\). The distortion correction coefficients obtained here are \(k_1 = -0.4328\) and \(k_2 = 0.338\). That is, it can be seen that the distortion is small when an object appears in the center of the image frame, whereas the distortion increases as the distance from the image center increases.

After applying distortion correction, the value of the object’s center position relative to the image coordinate is converted back to the position relative to the reference coordinate through the correction matrix, which becomes the actual position where the robot can pick up the object. Then, the centers between the bolt head and the bolt tail are used to determine the orientation of the object, and details are described in the next section. Finally, the robot arm uses the detected bolt or nut and its geometric information to accurately pick it up and move it to the target position through the robot’s inverse kinematics.

Experimental results in object detection by YOLOv3 and camera calibration

In the learning process to determine the four classes for the bolt, and one class for the nut, and geometric information of the bolt, 2000 epochs were trained for 6000 bolts and nuts data sets through YOLOv3, and the loss was finally reduced to about 0.03. Normally, in YOLOv3, if the loss is less than 0.06, it is considered that the learning is perfectly done. However, this loss is only for the training data set, so the accuracy of the object detection and its geometric information when the actual image is applied may not be guaranteed. To ensure the performance on detection of bolt and nut and orientation angle of the bolt, experiments were performed directly using the finally learned weight values and the success rate for picking up the bolt and nut was measured. The experiments were divided into three areas: Performance on the detection and geometric information for an object through YOLOv3, image correction, and object pick up and movement test.

Detection performance test

To check whether the proposed learning scheme through YOLOv3 to detect the bolt and nut and determine its geometric information was successful, we tried to check whether bounding boxes were created correctly after detecting the bolt. A total of eight objects (four bolts and four nuts) were randomly placed on the plate and the bounding boxes generated from the image were analyzed (Figure 10).

If detection is performed perfectly, 16 bounding boxes should be created for 8 objects, three per bolt (Whole bolt, Bolt head, and Bolt tail) and one per nut (Nut), respectively. Among these bounding boxes, the accuracy was derived by calculating the number of times the bounding box was incorrectly captured. The four types in which the bounding box may be incorrectly caught are as follows.

(a) When an object to be caught is missing
(b) When multiple bounding boxes for one object are captured
(c) When captured where there is no object
(d) When the label is incorrectly classified

Among these, cases (c) and (d) can be solved by creating the test environment similar to the environment of the learning data set. In the actual test, only errors corresponding to cases (a) and (b) appeared.

To obtain the accuracy of detection for each class, the test is repeated 20 times to obtain 320 bounding boxes. After counting the bounding boxes that are detected as missing or duplicate for each class (80 each), the detection accuracy was derived for each class and the results are shown in Table 1.

All four classes showed an accuracy of 90% or more, and the nut was 100% accurate, which states that all

| Table 1. Detection accuracy according to classes. |
|-----------------------------------------------|
| Whole Bolt | Bolt Head | Bolt Tail | Nut |
|------------|-----------|-----------|-----|
| Total      | 80        | 80        | 80  |
| Captured   | 77        | 79        | 74  |
| Error      | 3         | 1         | 6   |
| Accuracy (%) | 96.3      | 98.8      | 92.5|
|            | 100       |           |     |

Lee et al.
bounding boxes for the nuts were detected perfectly without error. Bolt tail’s accuracy was the lowest at 92.50%. This is associated with the fact that the tail shape does not have relatively distinct feature compared to other classes. Since it plays a crucial role in determining the gripper’s posture for pickup, a more data set learning is required.

**Image calibration performance test**

In the case of the monocular camera used in this study, the radial distortion occurred, which resulted in the shift of the image outward like a convex lens. Since the detected bounding box coordinates become also inaccurate by image distortion, image distortion should be corrected for the robot to successfully pick up the object.

Object detection was done by placing one nut in the center of the working plate, where the distortion is least, and four nuts on the edge, where the radial distortion is the most (Figure 11).

By applying distortion correction (9–10) for the center coordinates of each bounding box of the five nuts (denoted by A, B, C, D, and E) the center coordinates for the bounding boxes are corrected. Table 2 shows the results of distortion correction for the center coordinates of five bounding boxes. As can be seen from this result, the more the object moves away from the center of the image, the greater the distortion occurs. After estimating the center value of the object using YOLOv3, it was transformed into the value with respective to the reference coordinates and then compared with the actual measured value. Table 3 shows the comparison results between the two center values. The resolution of the camera used here is 1280 \times 720.

Nut A in the center of the image has a zero error, and the remaining four nuts show errors of approximately 1 mm to 4 mm compared to the actual coordinates. Since B, C, D, and E are at the location where the image distortion is most severe, the relatively large error occurs.

Since the width of the gripper used in this work is 20 mm, the maximum error of 4 mm was judged to be within the allowable range for the gripper used, and it was not a big problem in picking the nut. However, since more precise control is required when assembling the actual nut, it is necessary to perform a more rigorous calibration work and distortion correction.

| Table 2. Bounding box center coordinates by image correction (unit: pixel number). |
|-----------------------------------------------|---------------|---------------|---------------|---------------|---------------|
| A  | B  | C  | D  | E  |
|-----|----|----|----|----|
| Before | (662, 313) | (122, 51) | (1186, 49) | (132, 650) | (1189, 639) |
| After  | (662, 313) | (94, 37)  | (1213, 36) | (103, 668) | (1214, 654) |
| Difference | 0, 0  | 28, 14  | 27, 13  | 29, 18  | 25, 15  |

| Table 3. Bounding box center coordinates by YOLOv3 and actual measurement (unit: mm). |
|-----------------------------------------------|---------------|---------------|---------------|---------------|
| A  | B  | C  | D  | E  |
|-----|----|----|----|----|
| From YOLO | (172, -1)  | (235, 127) | (235, -126) | (93, 125) | (96, -126) |
| Actual  | (172, -1)  | (237, 129) | (237, -129) | (92, 129) | (92, -129) |
| Difference | 0, 0  | 2, 2  | 2, -3  | 1, 4  | 4, 3  |

**Parts picking test**

**System configuration**

Here, experiments were conducted to confirm the reliability of object detection and its geometric information determination. We checked the whole processes after placing several bolts and nuts on the working plate: detecting the bolt and nut, grasping it with right posture, finally moving it to the designated location.

Figure 12 is the overall work flow of picking up, transferring, and dropping off bolts and nuts after identifying the target. Using the proposed method that divides several parts from one object by creating bounding boxes through YOLOv3 the center and orientation information of each bolt and nut are identified, and these values are transformed relative to the reference coordinates to inform the robot gripper to pick it up. Then, the robot determines the picking order for identified objects, and then the geometrical information on the bolts and nuts and picking order is delivered to the robot arm controller.

Once the position and orientation of the target object are identified through the proposed deep learning algorithm, the inverse kinematic solution to control the robot is adopted to work for picking up, transferring to the designated location, and dropping off the target object. Generally, since kinematic decoupling may not be designed satisfactorily in a degree-lacking robot system, it is difficult to solve the inverse kinematics problem using a geometric solution or an algebraic solution. Therefore, a numerical solution for solving the inverse kinematics was developed and then applied. The design of the inverse kinematics solution of the robot is similar to described in detail.18
Now, the pseudo code of the whole process is summarized in the following box.

### Pseudo code

**Input**
- \( I \): Current Image Matrix of Camera censer

**Locals**
- \( x \): The x-center of the object’s bounding box
- \( y \): The y-center of the object’s bounding box
- \( w \): Width of the object’s bounding box
- \( h \): Height of the object’s bounding box
- \( \text{conf} \): Confidence score of object’s bounding box

**Outputs**
- \( X \): Target x coordinate of Gripper
- \( Y \): Target y coordinate of Gripper
- \( Z \): Target z coordinate of Gripper
- \( R \): Target Orientation Matrix of Gripper (\( R = [r_{ij}]_{3 \times 3} \))

**Algorithm**

#### Procedure Object Detection(I)

```
repeat
    Yolo_result = []
    if number of Object is not 0
        for number of Object do
            \([x, y, w, h, \text{conf}] = \text{YOLOv3}(I)\)
            if conf is above the threshold
                Yolo_result[i] = \([x, y, w, h]\)
            end if
        end for
    end if
    Yolo_result = Sort_for_Ordering(Yolo_result)
    X, Y, Z, R = Pixel_to_reference(Yolo_result[i])
    return X, Y, Z, R
```

Then, the return value is sent to the control part.

---

### Bolt orientation test

In this part, experiments were conducted to check whether the bolt orientation obtained by the proposed method on object recognition and determination of geometric information was correct. The orientation angle of the bolt \( \phi \) (Figure 13) that comes by connecting the two center values of the bolt head and bolt tail once two center values are successfully identified is calculated as follows

\[
\phi = \arctan\left(\frac{y_{ct} - y_{ch}}{x_{ct} - x_{ch}}\right) \tag{11}
\]

where \((x_{ch}, y_{ch})\) is the bolt head center values and \((x_{ct}, y_{ct})\) is the tail center values that are determined by YOLOv3 training and test.

Then, the orientation angle was compared with the directly measured angle. Figure 14 shows the bounding boxes of bolt head and tail for each bolt and the corresponding orientation angles for the bolts. Table 4 shows the comparison between the determined orientation angles of five bolts and the measured angles.

In this experiment, an insignificant orientation angle error of 0.3 or less was found for all five bolts. To increase the reliability of the method for determining the bolt posture, six additional experiments were conducted. Table 5 shows the results, similar to Table 4, and it can be seen that the average errors are less than 0.6° for all bolts.
Picking test

After placing two bolts and two nuts in the work space, the robot is controlled to pick them up and move them to specific positions one by one. Figure 15 shows the entire picking process when performing the task. Figure 16 shows that even with the gripper and human hand moving in the work-space during the operation, the bounding boxes are captured only for bolts and nuts on the plate by training the data set shown in Figure 6. It can be confirmed that object recognition of bolts and nuts proceeds smoothly even if such external intervention occurs.

Repeated picking task test

Table 6 summarizes the results of picking task for 20 times, 50 times, and 100 times each for bolt and nut, respectively.

As a result of the tests, it was confirmed that the picking and subsequent transporting of the target object were performed very well. There was no significant difference in the success rate when picking bolts and nuts repeatedly 20 times, 50 times, and 100 times in the picking experiments. However, in the case of repeating 100 times, there were three times of bolt-gripping failures. Some failures belong to the second case described in section “Image calibration performance test,” which was caused by an incorrect postural command because two bounding boxes for one bolt tail were caught for one bolt, and this can be resolved by more appropriately adjusting the YOLO v3’s NMS value. Another failure factor is the fourth case described in section “Image calibration performance test,” where the bolt head and tail are recognized as the same class. In other words, when the surrounding environment changes, the tail of the bolt is not recognized correctly, and the three bounding boxes are not clearly distinguished. This can be overcome by properly adjusting the threshold value of each corresponding bounding box.

Performance comparison with general YOLOv3 based on COCO data set

The performance of the proposed YOLOv3-based object detection process was compared with YOLOv3 (named Original) performed on the basis of the existing COCO data set. For mAP, the original referenced the results of YOLOv3-416 shown in Levine et al. and the performance results of this study were obtained from the detection rate shown in Table 7. Frame per second (FPS) was set as the average value directly measured for 1 min.

The mAP was 96.25, which was significantly improved compared to the original case of 55.3. It is regarded as a result of learning by applying various image options in a limited workspace. For the actual application process, the goal was to achieve mAP of 90 or higher, and although it is not perfect, it is understood that it has reached a sufficiently applicable value.

In the case of FPS, it was reported as 34.48 in the original case, but in the actual execution, it was shown to be 12.07. This seems to be a difference due to computing power. The FPS of this study was measured to be 11.72, which was similar to the previous value.

As a result, this study succeeded in achieving sufficient mAP at a level that can be applied to the process while acquiring object orientation information that was
Figure 15. Execution process of picking bolt and nut (top: nut, bottom: bolt, video supplemented).

Figure 16. Bounding boxes generation overcoming hand and gripper intervention.
Table 6. Picking task success rate for bolts and nuts.

|       | Bolt | Nut | Bolt | Nut | Bolt | Nut |
|-------|------|-----|------|-----|------|-----|
| Try   | 20   | 20  | 50   | 50  | 100  | 100 |
| Success | 20   | 20  | 49   | 50  | 97   | 98  |
| Fail  | 0    | 0   | 1    | 0   | 3    | 1   |
| Success rate (%) | 100 | 100 | 98 | 100 | 97 | 99 |

Table 7. Performance comparison with YOLOv3 based on COCO data set.

|                         | mAP | FPS |
|-------------------------|-----|-----|
| Original (COCO data set)| 55.3| 12.07 |
| This study              | 96.25| 11.72 |

previously impossible through YOLOv3 without reducing FPS.

Conclusions

In this work, an automatic bolt and nut-picking system that recognizes bolts and nuts and extracts geometric information at the same time by applying YOLOv3 architecture was introduced, and the effectiveness of this system was confirmed through actual tasks. In the case of bolt, by creating multiple bounding boxes for one bolt, the picking position was accurately determined by the center of the bolt head, and a vector connecting the two centers of the bounding boxes of the bolt head and bolt tail was found to determine the posture to pick up the bolt. Also, even if an object other than the target object intervened in the middle of object recognition, only the target object was detected by excluding the intervened objects in the training process. As a result, using a basic YOLOv3 architecture, it was confirmed that automatic pickup of target object from bolts and nuts randomly placed on the plate can be achieved with sophisticated object detection algorithm and its geometric information extraction.

In this work, since object detection was performed with a low-cost monocular camera, the center value of the bounding box was different from the actual value due to the camera distortion. To solve this problem, the image correction is performed to find the correct object center and then send the information to the robot controller. Due to the limitation of the monocular camera, automatic picking was performed only for bolts and nuts placed on the flat working plate, which has the fixed Z axis value. By further expanding the work, it is expected to be able to perform automatic pickup of objects on a curved surface by introducing stereovision using a binocular camera system or an additional distance measurement sensor.

On the other hand, deep learning algorithm is advancing very rapidly, and the YOLO model applied to this system has been upgraded from YOLOv3 to a higher version such as YOLOv4 and YOLOv5. If the latest high-performance YOLO model for object detection along with appropriate sensors is employed, it is expected that an automatically picking a target arbitrary placed on 3D surface with higher reliability can be developed.

Declaration of conflicting interests

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

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This research was supported by Korea Institute for Advancement of Technology (KIAT) grant funded by the Korea Government(MOTIE) (P0008473, HRD Program for Industrial Innovation).

ORCID iD

DH Kim https://orcid.org/0000-0002-7977-6981

Supplemental material

Supplemental material for this article is available online.

References

1. Report. Global robotics market-growth, trends, Covid-19 impact, and forecast (2021-2016). October 2021, Mordor Intelligence. 2021.
2. Report. Industrial robotics market, March 2020. IndustryARC. 2020.
3. Zhao Y. Current status and industrialization development of industrial robot technology. 2021 International conference on applications and techniques in cyber intelligence, 2021; 804–808.
4. Robinson A. Industrial automation: a brief history of manufacturing application & the current state and future outlook. Cerasis. 2020.
5. Brogårdh T. Present and future robot control development—an industrial perspective. Ann Rev Control 2007; 31(1): 69–79.
6. Camilleri D and Prescott TJ. Analyzing the limitations of deep learning for developmental robotics. Lecture Notes Comput Sci 2017; 10384: 86–94.
7. Alex OH. Robot vision vs computer vision: what’s the difference? Posted on July 07, 2016.
8. Wikipedia. Bin picking. https://en.wikipedia.org/wiki/Bin_picking (2021, accessed on November 2, 2021).
9. Murase H and Nayar SK. Visual learning and recognition of 3-d objects from appearance. Int J Comput Vision 1995; 14: 5–24.
10. Wu H, Su H, Liu Yet. and al. Object detection and localization using stereo cameras. 5th ICARM, 2020; 628–633.
11. Wilson A. Deep learning brings a new dimension to machine vision. Vision System Design. Posted on May 17th. 2019.
12. Mahony NO, Campbell S, Carvalho A, et al. Deep learning vs. traditional computer vision. Computer Vision Conference (CVC) 2019; DOI: 10.1007/978-3-030-17795-9_10.
13. Malhotra P and Garg E. Object detection techniques: a comparison. 7th ICSSS. 2020; 1–4.

14. Nguyen TA and Lee S. 3D orientation ad object classification from partial model point Cloud based on PointNet. In 2018 IEEE International Conference on Image Processing, Applications and Systems (IPAS), 12–14 December 2018, pp. 192–197. IEEE.

15. Contreras-Rodriguez LA, Muñoz-Guerrero R and Barraza-Madrigal JA. Algorithm for estimating the orientation of an object in 3D space, through the optimal fusion of gyroscope and accelerometer information. In 2017 14th International conference on electrical engineering, computing science and automatic control (CCE), 20–22 October 2017, pp. 1–5. IEEE.

16. Li TT and Lin CY. Bin-picking for planar objects based on a deep learning network: a case study of USB packs. Sensors. 2019;1–31.

17. Levine S, Pastor P, Krizhevsky A, et al. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. Int J Res 2018; 37: 421–436.

18. Sugihara T. Solvability-unconcerned inverse kinematics by the Levenberg-Marquardt method. IEEE Trans Robot 2011; 27(5): 984–991.

19. Eriklin dernoren. PyTorch-YOLOv3.github. https://github.com/eriklindernoren/PyTorch-YOLOv3 (2021, accessed on October 20, 2021).

20. Redmon J and Farhadi A. YOLOv3: an incremental improvement. arXiv:1804.02767. 2018.

21. Redmon J, Divvala S, Girshick R, et al. You only look once: unified, real-time object detection. arXiv:1506.02640. 2016.

22. Bodla N, Singh B, Chellappa R, et al. Improving object detection with one line of code. arXiv:1704.04503. 2017.

23. tzutalin. labelImg, github. https://github.com/tzutalin/labelImg (2020, accessed on December 10, 2020).

24. aleju. “imgaug”, github. https://github.com/aleju/imgaug (2020, accessed on November 12, 2020).

25. Semeniuta O. Analysis of camera calibration with respect to measurement accuracy. Procedia CIRP 2015; 766–768.

26. Sturm P and Maybank S. On plane-based camera calibration: a general algorithm, singularities, applications. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Fort Collins, CO, USA, 23–25 June 1999, pp. 432–437. IEEE.

27. Park J, Byun SC and Lee BU. Lens distortion correction using ideal image coordinates. IEEE Tran Consumer Electr 2009; 55(3): 987–991.