Eduvest – Journal of Universal Studies
Volume 5 Number 5, May, 2025
p- ISSN 2775-3735- e-ISSN 2775-3727

COLLABORATION OF 2 ARM ROBOTS IN A PICK AND PLACE TASK WITH A TASK ALLOCATOR METHOD BASED ON COMPUTER VISION

Wahyu Adhie Candra, Pipit Anggreni, David Yizreel Maruli Valentino Pardede[*]
Politeknik Manufaktur Bandung, Indonesia
Email: wahyu@ae.polman-bandung.ac.id, pipit@ae.polman-bandung.ac.id, davidyizreel.p@gmail.com[*]

ABSTRACT

Robotic arms have become integral in the manufacturing sector for automating repetitive tasks such as pick-and-place operations, which were traditionally performed manually. Manual execution is often limited by human fatigue, reduced accuracy, and the risk of injury, prompting a shift toward robotic automation. While single-arm robotic systems have been widely adopted, recent advancements now enable dual-arm collaboration, which introduces new technical challenges related to synchronization, precision, and object tracking. This study addresses these challenges by implementing an object detection system using YOLOv5 combined with HSV filtering to optimize performance under low computational constraints. The system communicates with robotic pole-arms via Python through Arduino serial communication. A perspective transformation technique is employed to ensure accurate mapping between 2D camera input and the robot's 3D operational space. The trained detection model achieved a mean Average Precision (mAP) of 97.7% at 0.5 and 60% at 0.5:0.95. Object detection testing yielded high evaluation metrics, including a precision of 1.00, a recall of 0.977, and an F1 score of 0.96. In real-world testing, the pick-and-place process demonstrated success rates of 95%, 85%, and 85% across three trials. These results indicate that the proposed system is highly effective for industrial applications and lays a foundation for further research into collaborative robot systems in dynamic environments.

KEYWORDS *Computer vision, Task allocator, Pick and place, Robot Interaction.*

Article Info:
Submitted: 06-12-2024 Final Revised: 13-05-2025 Accepted: 15-05-2025 Published: 22-05-2025

How to cite:	Candra, W. A., Anggreni, P,. & Pardede, D. Y. M. V. (2025). Collaboration of 2 Arm Robots in a Pick and Place Task with a Task Allocator Method Based on Computer Vision. Journal Eduvest. *5*(5): 5283-5293.
E-ISSN:	2775-3727
Published by:	https://greenpublisher.id/

INTRODUCTION

Robot arms are automation devices that are commonly used in the manufacturing sector. Its functions include a variety of tasks, including *pick and place* tasks. Pick *and place* tasks involve picking and placing objects from one location to another (Panggaribuan et al., 2021).

Initially, these tasks were performed by humans, but this manual process had a number of drawbacks, such as limited human speed and accuracy, fatigue, and the risk of work accidents. To overcome these obstacles, robots replaced humans in *pick-and-place* tasks. Robots have advantages in speed, accuracy, endurance without fatigue, and a lower risk of accidents (Ramadhan et al., 2021).

Initially, one robot carried out *the task of picking and placing*. However, with the development of technology, the two robots began to work together on the task. The collaboration of the two robots brings a number of advantages, such as increased speed and efficiency, the ability to handle larger and heavier objects, and the ability to work in environments that are dangerous to humans (Imtiaz & Lee, 2023).

However, the collaboration of two robots in *pick-and-place* tasks also poses a number of challenges. Coordination and interaction between the two must be carried out carefully, and the division of tasks must be optimal. One of the problems faced in this collaboration is the problem of suboptimal division of tasks, which can result in an imbalance in workload between robots and reduce overall efficiency and productivity (Prabhu et al., 2016; Wibowo, 2020).

Therefore, to overcome the problems and weaknesses of the use of robots described in the paragraph above, the author conducted a study that discussed the problem of the division and work of robots in carrying out tasks. This is done to complete and provide a clear picture of the right ways and methods that can be used to regulate the division of robot tasks in its work in the industry (Bragança et al., 2019; Kumar et al., 2021; Wan & Goudos, 2020).

To answer the problem formulation that has been described, the limitations of the problem in this study are set as follows: first, the robot arm used is a type of Pole-Arm. Second, the placement of the camera is done in a certain fixed position (Candra et al., 2023; Xiong et al., 2020). Third, the camera used is a webcam type. Fourth, the pick and place scenario will focus on object color detection. Fifth, the microcontroller used in the system is the Arduino Nano. Sixth, the robot's movement will be limited by certain calculations, depending on the robot's specification capacity. Seventh, each robot arm has been determined an object with a certain color to be taken (Mohebbi, 2020; Saeedvand et al., 2019). Eighth, each robot arm has a predetermined number of picking points, namely, robot arm 1 with 81 points and robot arm 2 with 63 points (Putri & R., 2016; Sokop et al., 2016). Ninth, this research will focus on object detection and collaboration between the two robotic arms. Tenth, this study will not discuss the calculation of the robot's arm movement in detail (Sidiq, 2016; Suari, 2017). Finally, the limit of the object's location and camera reading will be limited by a red box border with a length of 40 cm and a width of 40 cm (Mulia et al., 2020).

A study by Kaur et al. (2018) designed an Arduino-based pick and place robot using a color sensor, but it did not involve collaboration between multiple robotic arms. Meanwhile, Yudha et al. (2024) explored the use of the YOLO algorithm for object detection in robotic systems, yet their implementation was limited to a single robot and did not address task division between multiple robots. The novelty of the current research lies in the integration of YOLOv5 and HSV-based object detection for task distribution between two pole-arm robotic arms. This system is supported by serial communication with Arduino and camera perspective transformation to convert object coordinates accurately. The approach introduces a simple yet effective method that is applicable to small-to-medium-scale industrial environments, filling a gap in collaborative robotic automation research (Romzi & Kurniawan, 2020).

This final project aims to develop an efficient system for carrying out pick-and-place tasks using two robot arms collaboratively, utilizing a computer vision-based task division method. In addition, this study aims to analyze and optimize the interaction system between the two robots to ensure the smooth running of tasks performed simultaneously.

RESEARCH METHODS

The system used in this study is a collaboration system of two arm robots for picking and placing tasks, which implements a computer vision-based task division method. The main components in this system involve two industrial arm robots, a camera for object detection, and task-sharing software. The robotic arms used have high precision and strength, allowing them to move objects with the necessary accuracy. The camera functions to detect objects in the work area, providing visual information that is important for decision-making. The task-sharing software then determines the tasks each robot should perform based on the data obtained from the camera.

The work process in this system consists of several integrated stages to carry out picking and placing tasks efficiently. The first stage is Object Detection, where the camera captures an image of an object in the work area, which is then passed to the task-sharing software to determine which tasks should be performed by each robot. The second stage is Movement Planning, which involves algorithms to plan the path and movement of the robot's arms, ensuring that the robot can move efficiently and precisely towards the desired position. The final stage is Robot Control, where control algorithms regulate the robot's movements in real-time.

Position measurements ensure the robot moves according to plan, with movement adjustments made in case of errors or deviations from the desired path. This study adopts VDI 2206 as a research methodology to design and develop a system involving several integrated sub-systems, namely mechanical, electrical, and informatics systems. The selection of VDI 2206 was based on the complexity of the system being developed, which requires close coordination between various disciplines such as mechanics, electronics, software, and controls, as well as to ensure that each component of the system works synergistically.

RESULTS AND DISCUSSION

System Implementation Results

This chapter discusses the results of implementing a two-arm robot collaboration system in pick-and-place tasks using the task allocator method based on computer vision. The stages include system design, hardware and software implementation, and system performance testing.

1. **System Design**

The system's design involves three main components: two robotic arms, a camera for object detection, and task-sharing software. Each component is designed with the needs of the task and the work environment in mind.

- Robot Arm: The two robot arms used have five degrees of freedom (DOF) each, which provides high flexibility in performing *pick and place* tasks. The construction material is aluminum to ensure strength and durability.
- Camera: The camera is fixed and captures work area images. It can detect objects based on color and shape, which is then used to direct the robot arm.
- Task Splitting Software: This software distributes tasks between the two robot arms. The algorithm used is based on visual information obtained from the camera, allowing for efficient and accurate task division.

2. **Hardware Implementation**

Hardware implementation involves various electronic components working together to ensure both robot arms' precise and effective operation. Here are the details of the components used:

a. Motor Servo MG995:
   - Function: An MG995 servo motor provides precise movement at each joint of the robot arm. These motors are known for their high torque and good angular control capabilities, which are essential for tasks that require high accuracy, such as pick-and-place.
   - Specifications: The motor has a maximum torque of 11 kg-cm at 6V and a rotation speed of 0.2 seconds/60 degrees at 6V. It uses a PWM (Pulse Width Modulation) signal for position control, allowing it to set the rotation angle precisely.
   - Usage: This servo motor is equipped with each arm robot joint (base, shoulder, elbow, wrist, and gripper) to ensure flexible and accurate movement.

b. Arduino Nano:
   - Function: The Arduino Nano serves as the main microcontroller that controls the servo motor based on PWM signals. It was chosen because of its small size, affordable price, and ease of programming.
   - Specifications: The Arduino Nano uses the ATmega328P as a microcontroller with 14 digital input/output pins (6 of which can be used as PWM outputs), 8 analog input pins, and a clock speed of 16 MHz.
   - Usage: The Arduino Nano is connected to a servo motor via a PWM pin and receives input from the software that controls the robot's movements. The program on the Arduino Nano regulates the PWM signal sent to the servo motor to control the position of each robot joint.

c. Power Adapter:
   - Function: A 220VAC to 5VDC 3A power adapter provides stable electrical power to the whole system. Sufficient and stable power is essential to ensure optimal servo motor performance and prevent damage to electronic components.
   - Specifications: This adapter converts 220V AC voltage from the power source to 5V DC voltage with a maximum current of 3A, enough to run multiple servo motors simultaneously.
   - Usage: This adapter is connected to the system via a female DC jack and controlled by a switch button to regulate the flow of electricity to all components.

d. Breadboard:
   - Function: A breadboard connects various electronic components neatly and flexibly without soldering. This facilitates the assembly and modification of electronic circuits.
   - Specifications: The breadboard used has many connection points (holes) that allow the connection of various electronic components such as resistors, capacitors, and jumper cables.
   - Usage: Breadboards connect Arduino Nanos, servo motors, power adapters, and other components in an orderly circuit and are easy to reset if needed.

3. Software Implementation

Software implementation involves the development of algorithms and technologies that ensure the efficient coordination and operation of a two-arm robotic collaboration system. Here are the details of the software used:

a. Task Allocator

- Function: The *task allocator algorithm* organizes the division of tasks between the two robotic arms based on the visual data received from the camera. This algorithm ensures that each robot arm receives tasks that correspond to its position and the object it must handle.
- Process: *The task allocator* receives input from *the computer vision system* regarding the position and characteristics of the object. Based on this information, the algorithm determines the tasks each robot arm should perform, avoiding conflicts and ensuring a balanced distribution of workloads.
- Usage: *The task allocator* is implemented in software that runs on a computer connected to the system. It controls and monitors the operation of the two robotic arms in real time.

b. Computer vision

- Function: *Computer vision* technology detects and identifies objects in the work area. The system relies on an onboard camera to capture images and process them to recognize objects based on color, shape, and position.
- Process: Images taken by the camera are processed using image processing algorithms to extract important information such as the object's location, size,

and color. Algorithms such as thresholding, edge detection, and template matching are used to recognize objects accurately.

- Usage: The data generated by *the computer vision* system is used as input for the task allocator and control algorithms, ensuring that each robot movement corresponds to the position and characteristics of the identified object.

The first stage in model training is dataset preparation, which includes collecting data in images and annotations. The images can be obtained from existing datasets, such as COCO or PASCAL VOC, or custom datasets you created. Each image must be equipped with an annotation containing a bounding box and a class label for each object in the image. The annotation format used is the YOLO format, where bounding boxes are represented in class_id, x_center, y_center, width, and height, with all values normalized between 0 and 1.

Once the dataset is ready, the next stage is preprocessing. In this process, the images in the dataset are resized to fit the input dimensions of the YOLO model, e.g., 640x640 pixels. In addition, data augmentation is carried out to increase data variety and make the model more robust to changes in real-world data. Augmentation includes rotation, flipping, brightness adjustment, and cropping.
The dataset used in this training consists of 13 object classes, with 27 instances for each class (blue circle, green rectangle, yellow pentagon, red triangle, and others). This even distribution of the number of labels ensures that the model is not biased towards any particular class. The total number of dataset instances is 351.

In the distribution of the position of the bounding box in the image, it can be seen that the x and y coordinates are evenly distributed throughout the image area (0–1). Most bounding boxes are concentrated in the central area, but certain areas have no significant dominance. This ensures that the model can recognize objects throughout the image area.

The size of the bounding box's width distribution is 0.05–0.12, while the height is in a similar range, 0.06–0.15. This distribution shows that the objects in the dataset are small to medium in size, reflecting diverse real-world conditions.
Furthermore, the YOLO model was chosen as the architecture used. This study used a version of YOLOv5, which offers high efficiency for real-time object detection. The model configuration file is organized according to the number of classes to be detected and anchor boxes relevant to the size of the objects in the dataset. Hyperparameters such as learning rate, batch size, and number of epochs are also determined at this stage to optimize the training process.

The model training process begins by dividing the dataset into three main parts: training, validation, and testing datasets. The model is trained using a training dataset, where optimization is carried out by minimizing losses calculated based on three main components: box loss for errors in bounding box predictions, class losses for errors in class predictions, and objectness loss for errors in detecting the presence or absence of objects in a grid. The training process is carried out iteratively in several epochs, with the model storing a "checkpoint" at the end of each epoch to record its performance progress.

After training, model performance is evaluated using validation datasets. Evaluation is carried out by calculating metrics such as precision, which measures the proportion of correct predictions of all predictions; recall, which measures the proportion of objects that are successfully detected; and mean Average Precision (mAP), which is the average of precision at various Intersection over Union (IoU) thresholds. In addition, F1-score, a harmonization between precision and recall, is also used to give an overview of the model's overall performance.

During the YOLO model training process, several important metrics are displayed in graphs to evaluate the model's performance. The graph shows the model's performance from start to finish training over 100 epochs. Loss values and evaluation metrics such as precision, recall, and mAP are analyzed to ensure the model can detect and classify objects optimally.

The model is trained using a carefully prepared dataset, ensuring each image has accurate bounding boxes and labels. At the beginning of the training, the train/box loss value was at 1.9, which then decreased steadily to 1.2 in the last epoch, indicating an increase in the accuracy of the bounding box prediction. The same thing happened with val/box_loss, which decreased to close to the same number.

The loss values in the object classification (train/cls_loss and val/cls_loss) decrease from 4.0 to 1.0, indicating that the model is increasingly capable of correctly classifying objects. Meanwhile, train/dfl_loss and val/dfl_loss were stable at around 1.0 at the end of the training, indicating an increasingly accurate prediction of the distribution of bounding boxes.

In terms of evaluation metrics, precision and recall improved sharply during training. Precision reached almost 1.0, while recall was close to the maximum value in the last epoch. This shows that the model can detect objects with high accuracy without significant errors. In addition, the mAP@50 metric reaches 0.95, while mAP@50-95 reaches 0.6, which indicates the model's success in detecting objects at various levels of IoU difficulty.

After training, model performance is evaluated using validation datasets. Evaluation is carried out by calculating metrics such as precision, which measures the proportion of correct predictions of all predictions; recall, which measures the proportion of objects that are successfully detected; and mean Average Precision (mAP), which is the average of precision at various Intersection over Union (IoU) thresholds. In addition, F1-score, a harmonization between precision and recall, is also used to give an overview of the model's overall performance.

2. F1-Confidence Curve (Grafik 1):

This graph illustrates the relationship between the F1-score and the confidence threshold. At confidence 0.298, all classes have an average F1-score of 0.96. This shows a fairly stable performance in detecting all classes of objects. However, there is variation between classes where some classes, such as the Green Triangle, have a decrease in F1-score at high confidence, which reflects the potential difficulty in detecting certain objects with high accuracy.

3. Precision-Confidence Curve (Grafik 2):

This graph shows how the precision changes against the confidence threshold. The precision for all classes reached 1.0 at a confidence threshold of 0.806, indicating that the model can accurately predict without generating many wrong predictions at high confidence. Some classes, such as the Green triangle, show a decrease in low confidence, reflecting the potential for higher false positives at the low confidence threshold.

4. Precision-Recall Curve (Grafik 3):

This graph illustrates the relationship between precision and recall. The average mAP@0.5 for all classes is 0.977, indicating that the model performs well in detecting a variety of objects. However, the Green pentagonal class has lower precision and recall values (precision 0.769) than others, indicating difficulty detecting these objects.

5. Recall-Confidence Curve (Grafik 4):

This graph shows the change in recall against the confidence threshold. The average recall for all classes is 0.99 at the confidence threshold of 0.0, indicating that the model can capture almost any object at a low threshold. However, recall decreased significantly for some classes, such as the red triangle, when the confidence threshold was raised.

Conclusion: The model performs well with an average F1-score of 0.96, an average precision of 1.0 at a high threshold, and an mAP of 0.977. However, certain classes, such as green triangles, have lower precision and recall performance, indicating the need for further evaluation of datasets and augmentation strategies to improve difficult class representations. This model is particularly effective at high confidence but requires special attention to overcome the trade-off between recall and precision at low confidence thresholds.

The trained model is then tested on a test dataset to measure its generalization on new data. The test results are visualized through a confusion matrix, which shows the accuracy of the model's predictions in each class, as well as precision-recall and F1-score graphs to analyze the model's performance at various confidence thresholds.

The confusion matrix evaluates the model's performance in classifying various objects. In the first confusion matrix image, the model shows the classification results for 14 categories of objects. Examples:

- The "Blue circle" category was detected as true 7 times, but 2 times were considered "background".
- The category "segilimaHijau" is incorrectly classified as "segilimaBiru" 1 time.

The total classification results in this confusion matrix show that some categories have perfect accuracy. For example, the categories "Blue Triangle," "Green Triangle," and "Yellow Triangle" have 9 correct predictions from 9 data points.

The second confusion matrix is a normalized version, in which the numbers in the matrix are converted to proportions (scale 0–1). Examples:

- The "Blue circle" category has an accuracy level of 0.78, while the "Green pentagon" is only 0.11 because of a classification error.
- The background is also incorrectly classified as a "blue circle" with a proportion of 0.22.

From these two matrices, it can be concluded that the model performs well in most categories and is near-perfectly accurate in 10 categories, but it needs improvement in distinguishing some similar categories, such as "green triangle" and "blue triangle."

As a final step, the best training process model is saved for deployment purposes. This model can be used in real-time applications, such as object detection from cameras or video surveillance.

Result Analysis

From HSV calibration, data training, testing, and YOLO implementation, these three methods are combined to improve the accuracy of object readings. In Figure IV.26, the results of the mask display obtained through green border detection are displayed. Figure IV.27 shows the display results of masks without objects, while Figure IV.28 shows masks with detected objects. Figure IV.29 shows the results of the frame display covering the entire image detection and processing process. In this process, the detect_green_border function is used to detect green borders by processing the image using the HSV model, generating a binary mask based on the green color range. Then, the get_contours() function is used to find the contour on the mask, and the largest contour is used to define the square area, which is further processed. After detecting the green border, the four_point_transform function is applied to perform a perspective transformation on the square area. Figure 2 shows the result of a perspective transformation that aligns the work plane to a specific resolution. This aims to make detecting objects and mapping coordinates in the work area easier. Furthermore, in Figure 3, object detection is carried out using YOLO, where the object detection results are marked with a bounding box drawn on the detected object. The associated code to detect this object using the YOLO model is results=model(warped, verbose=False) and to draw a bounding box on the detected object is used the cv2.rectangle() and cv2.putText() commands.

CONCLUSION

Based on the research that has been conducted on the collaboration of two robot arms in pick and place tasks using the computer vision-based task allocator method, it can be concluded that the implementation of this collaboration system is proven to increase the efficiency of pick and place tasks, with a faster completion time than each robot separately, which is a time difference of 29 seconds. The developed system demonstrates object recognition capabilities with excellent accuracy, with an average F1-score of 0.96, an average precision of 1.0 at high thresholds, and an mAP of 0.977. The addition of HSV also makes it easier for robots to recognize the limits of picking up objects. The test results on the pick and place task showed success in three tests, with the first result reaching 95%, and the second and third tests reaching 85%. Based on the results of this study, some suggestions that can be considered for further research include further development of object recognition algorithms to improve detection accuracy and speed, and additional sensors such as depth sensors or 3D sensors to improve object detection capabilities. In addition, although laboratory tests have shown satisfactory results, further testing in real industrial environmental conditions needs to be carried out to ensure that the system can operate properly in various situations and conditions, and there is a need for software optimization to improve the overall performance of the system.

REFERENCES

Bragança, S., Costa, E., Castellucci, I., & Arezes, P. M. (2019). A brief overview of the use of collaborative robots in industry 4.0: Human role and safety. *Stud. Syst. Decis. Control*, *202*, 641–650. https://doi.org/10.1007/978-3-030-14730-3_68

Candra, W. A., Sunarya, A. S., & Saraswati, W. S. (2023). Computer Vision Implementation in Scratch Inspection and Color Detection on the Car Roof Surface. *Motiv. J. Mech. Electr. Ind. Eng.*, *5*(2), 317–328. https://doi.org/10.46574/motivection.v5i2.230

Imtiaz, M. B., & Lee, B. (2023). *Deep Reinforcement Learning-based Industrial Robotic Manipulations*.

Kaur, R., Kaur, M., & Singh, J. (2018). Endothelial dysfunction and platelet hyperactivity in type 2 diabetes mellitus: molecular insights and therapeutic strategies. *Cardiovascular diabetology*, *17*, 1–17.

Kumar, S., Savur, C., & Sahin, F. (2021). Survey of Human-Robot Collaboration in Industrial Settings: Awareness, Intelligence, and Compliance. *IEEE Trans. Syst. Man, Cybern. Syst.*, *51*(1), 280–297. https://doi.org/10.1109/TSMC.2020.3041231

Mohebbi, A. (2020). Human-Robot Interaction in Rehabilitation and Assistance: a Review. *Curr. Robot. Reports*, *1*(3), 131–144. https://doi.org/10.1007/s43154-020-00015-4

Mulia, S. B., Candra, W. A., Faisal, M., & Bandung, P. M. (2020). *Rancang Bangun Prototipe Automated Guided Vehicle*.

Panggaribuan, T., Hutauruk, S., & Sihombing, J. (2021). *Prototype Design of a One-Armed Robot with Three Levels of Freedom of Movement Based on Arduino with Distance Sensor on Bluetooth Smartphone*. *4*(1), 46–53.

Prabhu, N., Anand, M. D., Satish, S., & Akhil, C. K. (2016). Dynamic Modeling of Scorbot-ER Vu Plus Industrial Robot Manipulator using LabVIEW. *IJST*, *9*(April). https://doi.org/10.17485/ijst/2016/v9i13/90581

Putri, D., & R., A. (2016). Image Processing Using Web Cam on Moving Vehicles on the Highway. *JIPI (Scientific Journal of Informatics Research and Learning)*, *1*(01), 1–6.

Ramadhan, D. R., Rafi, A., Tahtawi, A., Wijayanto, K., & Kunci, K. (2021). *Position Control of the Robot Arm on Mission Pick and place with the Fuzzy Logic Control Method*. 4–5.

Romzi, M., & Kurniawan, B. (2020). Python Programming Learning with an Algorithmic Logic Approach. *JTIM: Journal of Masterpiece Informatics Engineering*, *3*(2), 37–44.

Saeedvand, S., Jafari, M., Aghdasi, H. S., & Baltes, J. (2019). A comprehensive survey on humanoid robot development. *Knowl. Eng. Rev.*, *34*. https://doi.org/10.1017/S0269888919000158

Sidiq, S. A. (2016). Image processing for egg identification based on size. *Elinvo*

(Electronics, Informatics, and Vocational Education), *1*(3), 151–156.

Sokop, S. J., Mamahit, D. J., & Sompie, S. R. U. A. (2016). Peripheral trainer interface based on arduino uno microcontroller. In *Journal of Electrical and Computer Engineering* (Vol. 5, Nomor 3, hal. 13–23).

Suari, M. (2017). The use of arduino nano in the design of physics learning media. *Natural Science*, *3*(2), 474–480.

Wan, S., & Goudos, S. (2020). Faster R-CNN for multi-class fruit detection using a robotic vision system. *Computer Networks*, *168*, 107036.

Wibowo, A. (2020). *Prototype of Arduino Uno-Based Arm Joint Manipulator Robot in Goods Sorting System*.

Xiong, Y., Ge, Y., Grimstad, L., & From, P. J. (2020). An autonomous strawberry-harvesting robot: Design, development, integration, and field evaluation. *J. F. Robot.*, *37*(2), 202–224. https://doi.org/10.1002/rob.21889

Yudha, Y. F., Aditya, A. A. Y., Rasyid, R. A. Y., Indra, N. I. A., & Melati, M. W. W. (2024). Perancangan Sistem Deteksi Objek Pada Robot Transporter Menggunakan Metode Darknet YOLOv8. *Electrician: Jurnal Rekayasa dan Teknologi Elektro*, *18*(2), 161–170.