Simplified autonomous object grasping in material handling process for human–robot collaboration

by Paryanto Paryanto

Submission date: 16-Mar-2025 01:17AM (UTC+0700)
Submission ID: 2615401522
File name: Simplified_autonomous_object_grasping_in_material_handling.pdf (2.54M)
Word count: 15393
Character count: 76532

International Journal of Intelligent Robotics and Applications
https://doi.org/10.1007/s41315-024-00375-6

REGULAR PAPER

Simplified autonomous object grasping in material handling process for human–robot collaboration

Muhammad Farouk Setiawan[1] · P. Paryanto[1] · Joga Dharma Setiawan[1]

Received: 12 July 2024 / Accepted: 9 August 2024

Abstract
The application of Human–Robot Collaboration (HRC) in the manufacturing sector, especially in the material handling process, is aimed at improving productivity through robots actively working alongside humans. In this condition, the robots need to understand how to handle the objects by themselves according to user preferences with an autonomous system. However, there have been challenges in the aspect of teaching robots to autonomously identify object grasp positions only using an RGB camera due to the effect of camera perspective on object visualization for robots. Therefore, this study aimed to propose a simplified method on an RGB camera for autonomous object grasping in the material handling process and implement it for the HRC concept. The method used a prototype robot manipulator with a computer vision system for object detection. During the execution of object grasping, the robot achieved a success rate of 86% for a single object and 76% for multiple objects. In the HRC concept, the robot achieved a success rate of 92% for placing objects one by one and 84% for placing objects continuously. The result also showed fast inference time when the robot in real-time detected the object, which was even just running on the CPU and in the planning process without complexity and requiring additional equipment aside from an RGB camera.

Keywords Human–robot collaboration · Robot manipulator · Computer vision · Material handling · YOLO

1 Introduction

Developments in manufacturing, particularly in automation, coupled with the worldwide effects of the COVID-19 pandemic, have led to changes in the way work is performed. This situation has made industry participants adjust to the new circumstances. A significant outcome of the pandemic is the innovation of a redesigned workspace that changes the way people work together, with workers no longer being physically close to each other (Baratta et al. 2023). As a result, the idea of humans collaborating with robots is increasingly being developed. The collaboration between robots and humans is called Human–Robot Collaboration (HRC) (Castro et al. 2021).

In manufacturing, HRC is applied to various tasks, including material handling (Segura et al. 2021), which includes efficiently distributing materials and products in the manufacturing system (Horňáková et al. 2021). In this situation, robot and human can exchange goods, making it important for robots to understand the intentions of the user when giving or receiving objects (Semeraro et al. 2023). The circumstances present a challenge in providing robots with an understanding of how to handle objects according to user preferences, including motion trajectory planning while the robot grasps the objects using a gripper, which can be applied to many different objects (Zhang et al. 2023). These are intended to increase flexibility in object grasping without requiring extensive information about the desired position of the objects.

Previous studies on HRC with robot manipulators, including those by Akkaladevi et al. (2019) using the UR

P. Paryanto and Joga Dharma Setiawan have contributed equally to this work.

✉ P. Paryanto
paryanto@ft.undip.ac.id

Muhammad Farouk Setiawan
mfarouksetiawan@gmail.com

Joga Dharma Setiawan
joga.setiawan@ft.undip.ac.id

[1] Faculty of Engineering, Department of Mechanical Engineering, Diponegoro University, Semarang 50275, Indonesia

Published online: 28 August 2024

(Universal Robot) 10 robot for the assembly of a steam cooker. Tsamis et al. (2021) used UR 10 for pick-and-place tasks, exchanging components between workstations. Furthermore, (Lotsaris et al. 2021) used a collaborative robot (COBOT) for front suspension assembly in public vehicles, handling heavy components. Vogel et al. (2017) used KUKA iiwa LBR 14 to assist operators in installing screws on plates during component assembly. Paletta et al. (2019) used KUKA iiwa LBR 7 for pick-and-place tasks in a toy problem, arranging a tangram puzzle. Sanchez-Matilla et al. (2020) used KUKA iiwa LBR 7 and UR5 for exchanging cups with various types of cup variations. These explorations consistently use object recognition methods, specifically computer vision, to implement the HRC concept on robot manipulators. The recognition process provides the robot with information to understand and make decisions based on the surrounding environment. Therefore, the computer vision method is considered suitable for addressing the challenges faced by robot manipulators in the HRC concept.

In order to make computer vision work, a camera needs to be integrated with the robot manipulator. The camera can be attached either to the side of the robot or separately but directly to the robot. In addition, this integration allows the robot to understand and perform actions by itself based on the information captured by the camera. In the study by Rakshit et al. (2023), a camera was used to determine the object location, select the object, and figure out the gripper setup for objects in the reach of the robot. Similarly, Rosenberger et al. (2021) used a camera to discover the position and pose of an object using depth information, enabling the robot manipulator to plan and execute movements to grasp the object by avoiding collisions from the environment around the object. Andersen et al. (2016) used a camera to spot car doors, providing information to estimate the position and pose of the component. The information was later projected onto the surface of the component, with details on the edges for worker instructions. Park et al. (2020) used a camera to perform object detection, which involved the capability of the robot to specifically pick up objects in various conditions. The results collectively show the potential for improving the effectiveness and efficiency of robots through computer vision.

Applying computer vision to the material handling process for the HRC concept appears fitting based on the discussed explorations. Most material handling processes include moving goods and still require human participation. Moreover, implementing computer vision only needs a camera, making arrangements simpler. A challenge in running a computer vision system is considering the perspective of the camera, which is how a camera projects a 3D object into a 2D image (Corke 2017). However, defining objects in all parts of the camera frame can be tough, leading to inaccurate determination of the real position of the object. Previous studies assumed the position of the object was at the center of the camera frame and used an RGB-D camera to ignore the perspective problems. Most of the previous studies on the use of an RGB-D camera used the Cornell Grasp Dataset and the Jaquard Dataset to define the position of the object grasp, as well as this conducted by Zhang et al. (2019), Ainetter and Fraundorfer (2021), and Cao et al. (2021). In addition, Rakshit et al. (2023) and Mousavian et al. (2019) used depth parameters generated from RGB-D cameras in image feature extraction. Thus, both methods require an additional step to adjust the algorithms in the datasets to the specific conditions and involve a more intensive computational process for feature extraction. In comparison, explorations by Christen et al. (2023), OpenAI et al. (2021), Xing and Chang (2019), and Zeng et al. (2018) showed using computer vision with objects positioned anywhere in the camera frame. The studies used reinforcement learning, which requires more memory due to storing programs based on robot learning. The process of using RGB-D cameras and reinforcement learning in computer vision is time-consuming due to multiple learning procedures and the need for high-specification devices to handle the computational load. Hence, it is inefficient for use. Thus, this study offers a different perspective on applying a computer vision system to the material handling process for the HRC concept, which proposes a method that modifies in the robot learning process to detect the object position through the camera and defines the correlation of object position in the camera frame and robot position as a simple method to reduce the computational load and addresses concerns, particularly regarding camera perspective, specifically only using an RGB camera.

In order to fully implement the HRC concept, a communication system between robots and humans is needed. One of the methods that can be used in this communication is gestures and motions that can be generated through cameras (Castro et al. 2021). During interaction, hand gestures have meanings that can convey information. For example, by using hand gestures to point at something, this expression indirectly shows sign language. Thus, the use of hand gestures as a communication system, which is then integrated with a computer, can help a person to be able to communicate more intuitively (Indriani and Moh 2021). In addition, the concept of hand gestures through this camera is easier to do because it only uses a camera. Thus, the combination of the two concepts of the automatic grasping system that we propose and this communication system can be an easy method to be carried out in implementing the HRC concept. Regarding its application in previous studies, hand gestures are generally carried out with the same concept as object detection, namely by providing training on a dataset that defines hand gestures, then a hand gesture recognition model can be generated, as done in the studies of Nooruddin et al. (2020), Banarjee et al. (2021), and Mohammadi

et al. (2023). However, these methods are not efficient to use because we need to perform an additional process of dataset retrieval and training related to the desired hand gesture. In order to address the problem, we use MediaPipe, which provides a ready-to-use solution related to hand gesture recognition. In MediaPipe, we do not need to take a dataset or training process because this library provides location points on certain parts of the palm, which can then be interpreted into the desired hand gesture (Zhang et al. 2020).

This study focuses on implementing autonomous object grasping in the material handling process for the HRC concept. At this point, the robot is tasked with picking up objects with various shapes randomly placed in the workspace and delivering the objects to user in the same area. Moreover, the designed robot can independently handle the grasping object process. When passing the objects, the robot stops to await commands, identifying user hand gestures using the same camera used for detecting the objects. This ensures the user is ready to receive the objects. The study uses kinematics and computer vision models to support these processes. The kinematics model of a robot focuses on moving the robot manipulator toward the desired point in motion planning. The model is closely tied to the inverse kinematics method, which adjusts the angle of each robot joint to reach a known coordinate point. Additionally, the inverse kinematics method is applied to the motion of the robot, allowing it to reach the intended object based on a predetermined point. Computer vision is used to provide a robot with the capability to gather information related to objects. The concept includes defining objects based on the position taken in a camera frame, allowing a robot to distinguish between objects in the frame. In review, this study contributes to:

1. Introducing simplified method on an RGB camera for autonomous object grasping in the material handling process and implementing it for the HRC concept.
2. Integrating diverse shapes and positions of objects in the material handling process, making it flexible to use.
3. Presenting the method that can solve camera perspective problems on an RGB camera without complexity and requiring additional equipment.
4. Presenting the simplified method on the communication system between robot and human in the HRC concept.

This study uses a prototype of a 4 DOF (Degree of Freedom) robot manipulator with servo motors as actuators in each robot joint. The end-effector of a robot is equipped with a gripper for picking up and placing objects. In addition, the robot framework is made using a 3D printer with PLA (PolyLactic Acid) material. The robot is controlled by an Arduino Nano microcontroller programmed with the Arduino IDE and the Firmata module, accessible in Python. Moreover, the robot camera of the system is an ESP32-CAM programmed with OpenCV and Python, enabling it to detect objects with the YOLO (You Look Only Once) algorithm as well as execute robot movements.

2 Related works

2.1 Computer vision

In order to develop the autonomous system for the HRC concept, implementing intuitive control for the robot is important, ensuring accurate interpretation of provided actions and commands (Al et al. 2020). The implementation includes using various sensing methods, including the computer vision system used in earlier studies. Most studies use RGB-D cameras, also known as 3-dimensional cameras, that involve learning methods. The learning methods used include separate concepts of object detection with robot motion and a combination of both. The separate learning method uses the Convolutional Neural Network (CNN) theory, which is used in the object detection process along with the object grasp position as well as this conducted in previous studies by Tsamis et al. (2021), Rakshit et al. (2023), Rosenberger et al. (2021), Zhang et al. (2019), and Ratsamee et al. (2019). The grasp position is defined as the position of the object at x, y, and z coordinates, which is correlated to the position of the robot. Thus, the robot motion is then applied using the robot kinematics method. While the combined learning method uses reinforcement learning to define the object and robot motion with a control policy, as well as this conducted by Christen et al. (2023), Xing and Chang (2019), and Zeng et al. (2018). Control policy in the robot learning algorithm is a rule or strategy used by the robot to make decisions or generate actions based on information received from the environment or sensors attached to the robot. In this case, the sensor also comes from the camera used on the robot. While, in the case of only using RGB cameras in the process of defining the object position, the RGB camera is positioned parallel to the work plane, and the object is placed at the center point of the camera frame. Previous studies by Paletta et al. (2019) and Andersen et al. (2016) demonstrated this approach. In our study, we will only use the learning method in object detection, namely by using the YOLO algorithm, which is an algorithm with a very fast ability to detect objects in real-time because it has a smaller computational load (Redmon et al. 2016). In addition, we adapt the study by Rakshit et al. (2023), which used a computer vision system with the object grasp position defined by the centroid of the grasp rectangle based on x and y coordinates. The information was used to estimate the object position

and integrate it with robot kinematics. Similarly, Zhang et al. (2019) implemented a computer vision system to define the object grasp position as a box, with the centroid of the box as the object grasp point. Then, using that point, the robot could plan and execute movements for grasping the object. Based on those two studies, we use the same concept in defining the position of the object, which is based on the centroid of a box showing the object grasp point in the camera frame. Thus, the position of the object at the robot position is estimated with our proposed method, which is then used for robot motion planning. In a separate study, Dairath et al. (2023) applied a computer vision system to a prototype fruit-separating robot, using an Arduino UNO as the microcontroller and an RGB camera to detect fruit quality for subsequent separation. The method included capturing images with the camera, forwarding the information to a computer, and recognizing objects through a computer vision algorithm system using OpenCV and Python. Inspired by the exploration, we implemented a similar concept, using hardware for control, robot actuators, RGB cameras, and programming language to control the robot in this study.

The focus of our proposed method, as mentioned in Section 1, was a simple method to carry out the automatic grasping system that reduced the computational load and addressed concerns, particularly regarding camera perspective, specifically only using an RGB camera. In the previous studies that we have presented, we highlighted the learning method to reduce the computational load. We found in the learning method used by the previous studies, which had the same concept as ours, that they separated the feature extraction process between object detection and grasp detection. Grasp detection in this case was the result of feature extraction from object detection itself in the form of ROI (Region of Interest), which was re-extracted into more specialized parts in certain parts for object grasping positions. In our study, we directly combine object detection and grasp detection into one feature extraction process. Therefore, we can reduce the computational load of the training and executing processes in the learning method. Furthermore, in order to be used for execution on the robot, information from the result grasp detection carried out in the previous studies was combined with depth information obtained from the RGB-D camera. Hence, in this case, the RGB camera is not able to be used. Thus, the method that we propose in this study could become an alternative to this problem.

2.2 Image processing

In the Section 1, we have mentioned that in order to reduce the computational load, we also define the correlation of object position in the camera frame and robot position. This definition is related to the use of the image processing method, which is a computational process used to transform images (Corke 2017). The image transformation process using image processing method can specifically change the shape of the image or the perspective of the image (Bezmaternykh et al. 2023). Thus, in order to overcome the camera perspective problem as mentioned in Section 1, we use one of the operations in image processing, namely warping. Warping is an image transformation process that maps all positions in one image plane to positions in a second plane. This process is often used to remove optical distortions caused by the type of camera used or the perspective of the camera itself (Glasbey and Mardia 1998). An example of the application of the warping method applied to an image can be seen in the study conducted by Arad et al. (1994) and Ruprecht and Muller (1995). In both studies, the warping method was used to change the face shape of an object in the image. Whereas in our study, we use the warping method to be able to position the plane where the object is placed thereby it seems to be parallel to the plane on camera. Thus, we can find out the relationship between the position of the object in the camera and the position of the object in the robot position. The relationship formed is a simple equation that does not require a large computational process to calculate. Thus, in this case we can also simultaneously reduce the computational load.

2.3 YOLO

In order to facilitate the application of the computer vision system in the study, an object detection system relying on the YOLO algorithm is used. YOLO is operated by using an algorithm that identifies objects through a single-step deep learning model, enabling it to directly predict the category and location of an object (Wang et al. 2023). In comparison to two-step deep learning algorithms, YOLO is exceptional for its faster detection speed, allowing real-time application, high efficiency, and flexibility (Li et al. 2023). Through the predictions generated by the algorithm, the system obtains information about the objects in an image, including bounding boxes and labels (Yang et al. 2023). Li et al. (2019) showed the capability of the algorithm to recognize objects by achieving a mean Average Precision (mAP) of 0.928, showing a precision of about 92.8%, using a dataset of 3600 images. Rosenberger et al. (2021) conducted a study including 80 different object categories recognized by robot manipulators using YOLO, leading to a robot success rate of 81.9%. Similarly, Zhang and Xie (2023) used YOLO to recognize and locate the position of chilies that are blocked by branches and leaves as an object retrieval command for a robot. The mAP value obtained from the system was 0.892, which was about 89.2%. The explorations collectively show the exceptional capability of YOLO to effectively detect objects and cover a wide range of object types, making it

highly suitable for material handling processes. In this study, we use the YOLOv8 algorithm to detect the objects and get their position in the camera frame.

YOLO processes the input image by predicting the bounding box and the class of the object contained in the image. The method applied by YOLO is by dividing the input image into sections of size S x S. Each section is called a grid cell. If the center point of an object is in a grid cell, then the grid cell is responsible for detecting the object. Each grid cell predicts the bounding box, the confidence score of the box, and the class probability condition. The confidence score represents how confident the model is that the box contains an object and also how accurate the prediction of the box is. In order to define the confidence score, YOLO considers the Intersection over Union (IoU). IoU measures how well the bounding box predicted by the model overlaps with the ground truth of the bounding box. Meanwhile, the class probability condition is defined on a grid cell containing an object that predicts only one class probability for each grid cell, regardless of the number of boxes. Therefore, the initial process of training the YOLO model requires an annotation process of the object along with its class in order for YOLO to use the data in the training process. Thus, this concept makes YOLO have a very fast detection capability that can be applied in real-time.

2.4 MediaPipe

Hand gesture is one method that can provide a natural way of interaction and communication. One of the methods that can be used to make robots understand hand gestures is hand gesture recognition. Hand gesture recognition aims to interpret human hand gestures that can use various gestures to interact with the robot without touching the robot. In hand gesture recognition, the procedure is started by tracking hand gestures and converting the tracking results into specific commands for the robot. In some previous studies using MediaPipe to implement hand gesture recognition, as conducted in the studies of Alvin et al. (2021), Indriani et al. (2021), and Mishra et al. (2023). MediaPipe is an open-sourced library by Google that can be used for various machine learning implementations, including hand tracking. The MediaPipe framework helps developers to focus on algorithm and model development and supports applications through results that can be processed through different devices or media (Lugaresi et al. 2019). Therefore, in this study we use MediaPipe as a communication system between robots and humans in the HRC concept.

MediaPipe requires input in the form of images or videos that can be generated through an RGB camera or RGB-D camera. In the process, MediaPipe first trains the palm detector. This palm detector works using a model that works like BlazeFace (Bazarevsky et al. 2019). Then, MediaPipe uses a feature extractor to generate small segments of the palm. Lastly, MediaPipe minimizes focal loss during training to support some of the best detection results. The palm detection results are in the form of 21 hand landmarks in x, y, and depth coordinates (can be seen in Fig. 1), a marker for the presence of a hand in the input image, and classification of the hand part as left hand or right hand (Zhang et al. 2020). In our study, we only use these hand landmarks to define the commands given to the robot.

3 Proposed methodology

3.1 Robot manipulator design

The robot used in this study is a prototype with 4 Degrees of Freedom (DOF), consisting of four rotating joints and a gripper as its end-effector. Figure 2 shows the design and motion direction of each joint, along with the robot frame. In addition, the motion direction of each joint was associated with the rotation of the servo motor, where the arrow showed an increase in the angle. The robot frame was defined based on the world frame, covering joints 1, 2, 3, and 4, as well as the end-effector, and each frame determined the position of the respective servo motor.

From a multibody point of view, the robot was composed of a base and four bodies (typically referred to as links) interconnected through a revolute joint. In order to define the robot parameters, Denavit-Hartenberg (DH) convention was usually used (Craig 1994), which attached frames to various parts of the robot and then described the relationships between these frames. The DH parameters of the robot in this study were presented in Table 1.

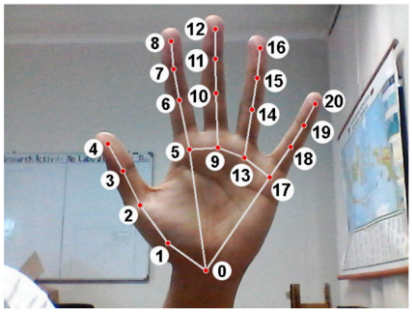


Fig. 1 Hand-landmarks in MediaPipe

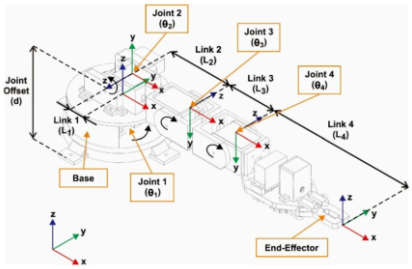


Fig. 2 Design, joint motion direction, and frame of robot manipulator

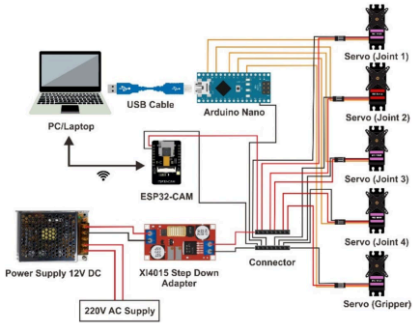


Fig. 3 Wiring diagram of hardware configuration

Table 1 DH parameters of the robot

Joint (i)	Joint angle [θ] (rad)	Joint offset [d] (m)	Link length [L] (m)	Twist angle [α] (rad)
1	θ_1	0.084	0.009	π/2
2	θ_2	0	0.113	π
3	θ_3	0	0.089	0
4	θ_4	0	0.2129	0

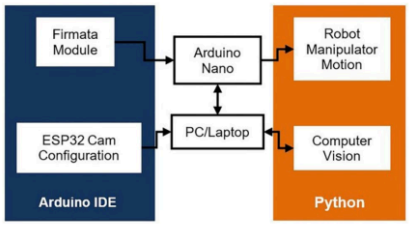


Fig. 4 Software configuration diagram

3.2 Hardware and software configurations

In terms of hardware configuration, the control system integrated an Arduino Nano as the controller, MG996R and DS3225 servo motors as the actuators of the robot, a power supply as the power source that supplied electric power to the robot, an XL4015 Step Down Adapter as the voltage reducer from the power supply to the servo motor, the connector as a terminal connecting electric current, an ESP32-CAM as a camera, and a PC/laptop as an interface. The PC/laptop specifications included an Asus X550VX with an Intel Core i7 6700HQ processor, 8GB of DDR4 RAM, Nvidia GeForce GTX 950M graphics, and a 256GB SSD. Robot motion input was entered from a PC/laptop based on information obtained from the ESP32-CAM, which was interconnected with a Wi-Fi signal and then processed by the Arduino to drive each servo motor. In order to more easily understand the robot hardware configuration, it can be seen in Fig. 3.

The software configuration used two programming languages, C and Python, where C was used to connect the Arduino Nano with the Firmata module and configure the ESP32-CAM through the Arduino IDE, while Python was used for the robot motion program with the Pyfirmata module, which was interconnected to the Firmata module in C by COM port. Additionally, Python also controlled computer vision, using the OpenCV library to access the ESP32-CAM and establish communication with Arduino. Then it commanded the robot based on object detection from the YOLOv8 algorithm through Visual Studio Code. For more details, it can be seen in the software configuration diagram in Fig. 4.

3.3 Robot position and motion conditions

The movement of robots in HRC included three conditions, namely, object detection, grasping objects, and delivering objects. During object detection, the robot stayed stationary in a predetermined position, with the end-effector slope being 60°. After detecting an object in the camera frame, the robot moved to grasp the object based on the centroid location information of the object grasp point. This centroid location defined the position of the object against the

position of the robot. Thus, in this condition, the robot kinematics calculation used was inverse kinematics, which was to find out the angles of each robot joint at the position of the object that had been defined previously applied to be able to run the robot to grasp the object. Note that the object grasping condition in this study was defined with the assumption that the object grasp position was at z = 0. As for the third condition, namely the condition of the robot delivering the object to the user, it involved a certain point that had been defined.

The inverse kinematics calculation used in this study was a geometric approach method where the calculation of the angle of each joint was done by calculating using the concept of trigonometry, as done in the study of Cahyono et al. (2022). In this study, calculating each joint angle of the robot was done by looking at two angles of view of the robot, namely in the XY plane (top view) and the XZ plane (front view) which then calculated each link length relationship and the position of the robot end-effector to the angle of each robot joint. For more details regarding the calculation of the inverse kinematics of the robot, see Fig. 5.

In Fig. 5(a), it can be seen that the equation for calculating the angle of joint 1 (θ_1) seen from the top viewpoint in the XY plane with x_p and y_p as the robot end-effector coordinates is as follows:

$$tan\theta_1 = \frac{y_p}{x_p} \tag{1}$$

$$\theta_1 = tan^{-1}\left(\frac{y_p}{x_p}\right) \tag{2}$$

$$x_a = \sqrt{x_p^2 + y_p^2} \tag{3}$$

In Fig. 5(b), the equations for calculating the angles of joint 2 (θ_2), joint 3 (θ_3), and joint 4 (θ_4) viewed from the front viewpoint in the XZ plane can be found. The relationship between these two points of view was in the x coordinate used, namely x_a. x_a was the coordinate on the x-axis for the robot in the XZ plane. x_a would be equal to x_p (from Fig. 5(a)) if in the XY plane, the y_p was at point 0 or in other words, the angle of joint 1 was 0. Thus, in using this geometry approach method, calculations had to first be made in the XY plane and then in the XZ plane.

As for being able to calculate each joint angle of the robot in this study using the concept as done by Asada (2005), the first thing that needed to be done was to determine the gripper tilt angle (θ_{th}). In this study, we used a tilt angle of 45°. Thus, from this tilt angle, the position of joint 3 of the robot (x_3 and z_3) could be known. Then, from that position, using the concept of the cosine rule, the angles of joints 2 and 3 could be obtained. Meanwhile, joint 4 was obtained by considering the angles of joint 2, joint 3, and the gripper tilt angle. But it was also necessary to know that the condition of the robot gripper position in this study was shifted from the straight-line position of the robot manipulator. The desired position of the robot when picking up the object was at point F (x_F and z_F). Whereas in the application of this formula, the position of the robot end-effector was at point E. Thus, it was necessary to adjust the position of the object to the position of the robot in this study using the equations that can be written as follows:

$$x_r = r\cos\theta_{th} \tag{4}$$

$$z_r = \sqrt{r^2 - x_r^2} \tag{5}$$

The value of x_r and z_r was the value of the position shift that needed to be done to be able to adjust to the position of the robot gripper. Meanwhile, r was the shift of the robot end-effector point based on the formula (point E) to the robot end-effector point in this study (point F), which was 0.028 m. Thus, this position adjustment can be written as follows:

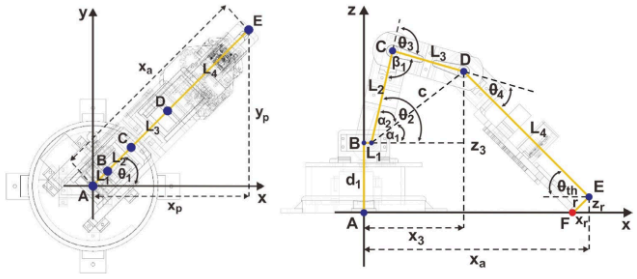


Fig. 5 Geometric approach for inverse kinematics: **a** XY plane; **b** XZ plane

$$x_F = x_a + x_r \tag{6}$$

$$z_F = z + z_r \tag{7}$$

Therefore, the adjustment process meant that the object position and the robot end-effector point in this study were at point F, while the position that was given to the inverse kinematics method in order to get the point F position was point E.

Then, using this new position, the position of joint 3 of the robot can be determined using the following equations:

$$x_3 = x_F - L_4 \cos\theta_{th} \tag{8}$$

$$z_3 = z_F + L_4 \sin\theta_{th} \tag{9}$$

Thus, the equations for the angles of joint 2, joint 3, and joint 4 can be written as follows:

$$c = \sqrt{\left(x_3 - L_1\right)^2 + \left(z_3 - d_1\right)^2} \tag{10}$$

$$\beta_1 = \cos^{-1}\left(\frac{L_2^2 + L_3^2 - c^2}{2L_2L_3}\right) \tag{11}$$

$$\theta_3 = 180 - \beta_1 \tag{12}$$

$$\alpha_1 = tan^{-1}\left(\frac{\left(z_3 - d_1\right)}{\left(x_3 - L_1\right)}\right) \tag{13}$$

$$\alpha_2 = \cos^{-1}\left(\frac{L_2^2 + c^2 - L_3^2}{2L_2c}\right) \tag{14}$$

$$\theta_2 = \alpha_1 + \alpha_2 \tag{15}$$

$$\theta_4 = \theta_2 - \theta_3 + \theta_{th} \tag{16}$$

The position of the robot given in this study when it was positioned to identify objects is shown in 42, 6. These positions were positioned at joint angles of: (1) $\theta_1 = 90°$; (2) $\theta_2 = 90°$; (3) $\theta_3 = 30°$; and (4) $\theta_4 = 120°$. Th51osition of this robot joint paid attention to the direction of rotation of the servo motor towards the desired direction of motion of the robot. While the ESP32 48AM, which was used to detect the objects, was placed on the robot end-effector and in a case. The distance of the camera lens to the rob3 workspace was 0.170 m. As a note in this study, in order to be able to position the angle of the robot joint thereby it could apply to the previous inverse kinematics equation, it was necessary to initialize the robot condition. The initialization condition was a condition where the robot manipulator was positioned in a position parallel to the x-axis. Thus, it could be clearly known the rotation relationship of each servo motor to each part of the robot.

Fig. 6 Initial position of the robot

3.4 Dataset preparation

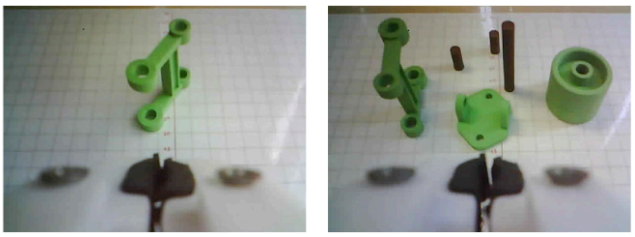
In this study, computer vision, supported by the OpenCV library and the YOLOv8 algorithm integrated into Python, was applied for object detection. Following this result, the aimed objects were components of the belt tightener prototype. Object detection was facilitated by the OpenCV library, which accessed the ESP32-CAM camera, followed by the implementation of the YOLOv8 algorithm. In this study, we used the YOLOv8n model, known for its lightweight nature, to decrease computational load during both robot operation and object detection.

The determination of an object dataset was important for the YOLOv8 training process of the algorithm. This step allowed YOLOv8 to process information related to the object dataset, enabling the subsequent application of the object detection algorithm. A8itionally, the dataset retrieval process included capturing images of objects placed on a measuring mat in various positions, minimizing the potential for errors in object detection. The measuring mat, featuring 0.010 × 0.010 m boxes printed on A4 paper, serves as the

backdrop for photographing a single component and multiple components, as shown in Fig. 7. The dataset for the study comprised a total of 530 images, each with a length of 640 pixels and a width of 480 pixels.

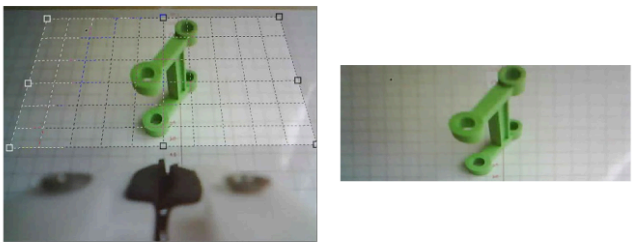
After capturing the dataset image, adjustments to the image frame were necessary to establish a relationship between frame conditions and actual positions. The study by Zeng et al. (2018) had a similar challenge with non-parallel camera positions with the object plane. This issue was from a camera perspective, as mentioned in Section 1. The study said that it was required to change the image position of the frame as though it were parallel to the object plane for a precise object position definition. Therefore, a process called "warping" (as mentioned in Sect. 2.2) was used to achieve this arrangement. In this context, warping was the deformation process used to manipulate images in image processing, which included various methods such as cropping, scaling, rotation, translation, etc. In this study, we used the warping to adjust the position of the frame to create the appearance of a parallel camera position to the work plane, as shown in Fig. 8, and also included the cropping process to exclude the robot tip of the gripper using the OpenCV program called "cv2.warpPerspective". Consequently, the image size in the object dataset after warping was 617 pixels in length and 247 pixels in width. During warping, the placement of points forming the new frame was adjusted the lines on the measuring mat. Furthermore, in order to determine the final image size, we took the largest image pixel difference for each frame coordinate.

The object dataset that had been through the warping process in order to initiate the training process needed to be annotated first. Annotation was the labeling object process in the image frame, conducted using LabelImg in this study. The procedure included drawing a box around the object and the object grasp point, as shown in Fig. 9. The box around the object defined the entire body part of the object inside the camera frame, while the box at the object grasp point was located where the robot should grasp the object. LabelImg automatically provided details about the points that formed these boxes and the object class. In this study, there were 10 classes (Pully, Frame, Base, Shaft, Screw, Pully Grasp, Frame Grasp, Base Grasp, Shaft Grasp, and Screw Grasp), defined by codes 0 through 9. Each class represented a component of the belt tightener and its grasp point. Moreover, the robot used the position of the object grasp point to determine the object grasp position, as explored by Zhang et al. (2019). This position was determined by considering the capability of the gripper to grasp the object and the



(a) (b)

Fig. 7 Object dataset capture positions: **a** Single; **b** Multiple



(a) (b)

Fig. 8 Warping process of object dataset: **a** Before warping; **b** After warping

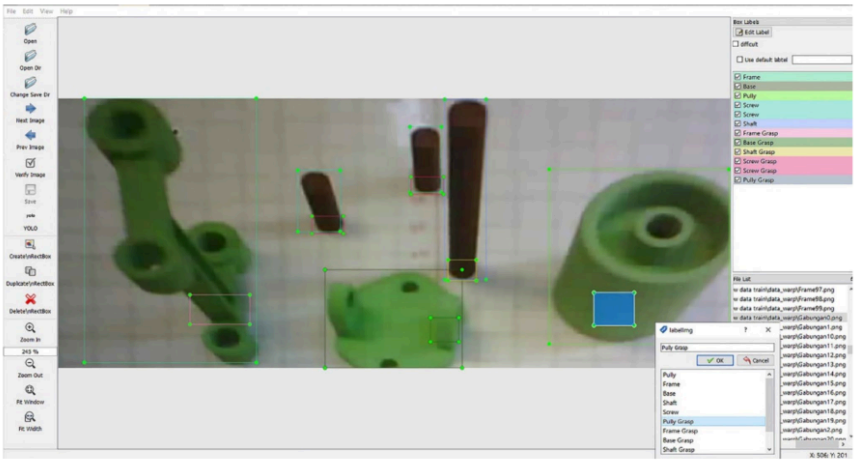


Fig. 9 Dataset annotation process

location on the measuring mat. The measuring mat played an important role, necessitating accurate observation of the position of the object grasp point to avoid errors in the object position definition of the robot. Following the annotation process, the captured image data, along with the labeled data, was consolidated and stored in a folder for training using YOLOv8.

After creating the object dataset, the training process was initiated. In this study, object dataset training was conducted using Google Colab, a browser owned by Google that facilitated the execution of Python programs. Moreover, Google Colab was exceptional for providing free access to GPUs, making it ideal for training object datasets without congesting the resources of the computer. In order to perform YOLOv8 algorithm training, the dataset needs to be divided into three groups, namely "train", "test", and "valid". The "train" folder included datasets for training, the "test" folder for testing, and the "valid" folder for validation. Consequently, the object dataset acquired earlier had to be distributed into these three groups, with proportions of 70% for train, 10% for test, and 20% for valid, following the methodology shown in Wang et al. (2023). In this study, the optimal number of epochs was set at 200, representing the complete cycle in which the entire training dataset had been used to update model parameters.

3.5 Defining the relationship of object position in camera frame and robot position

The YOLOv8 algorithm provided information about the position of objects in the camera frame, including the points showing the annotated boxes and the corresponding class name code. The object detection processed by the robot in this study required the box centroid of the object grasp point obtained by adding up the positions of points on the same axis from the box and later dividing by 2. Moreover, it was important to observe that the pixel calculations in the image were based on the top-left corner point, serving as the 0-pixel reference for each frame axis. The x-frame showed the horizontal axis, while the y-frame represented the vertical axis of the image.

In the dataset, the position of the object grasp point depended on the location of the object relative to the measuring mat. Therefore, in the camera frame for detection, the object in real-time was also positioned like the dataset, which was obtained through warping. Afterward, we set the center point of the camera frame along with the horizontal and vertical lines that intersect the center point and made sure that the horizontal line was parallel to the horizontal line of the measuring mat, and the same was applied to the vertical line. This process established a reference position connecting the camera frame and the robot position. In this study, the center point of the camera frame was at (in units of meters) $x = 0.250$ and $y = -0.005$ in the position of the

robot. These values could be different in some cases, for example, depending on the robot and the camera used, as well as the camera position on the robot end-effector. Therefore, it could be adapted. The negative y-axis value showed the point was on the left side of the centerline of the robot position, while the positive y-axis was on the right side. Note that in this study, the x-frame of the camera was defined as the y-axis position of the robot, and the y-frame of the camera was the x-axis position of the robot.

The center point of the camera frame was used to define the relationship between the position of the object in the frame and the robot position. We placed some points positioned on the horizontal and vertical lines that intersect the center point of the camera frame using the OpenCV program called "cv2.circle". These points were important for defining the equation connecting each camera frame position to the robot position. The formation of the equation included placing points at 0.010 m intervals against the camera center point of the frame and recording pixel changes in the camera frame for each 0.010 m shift. This process gradually continued until the point was out of the camera frame. In this study, the average pixel change of the horizontal line was 35 pixels, and the vertical line was 24 pixels. These values could also be different, like in the case of the center point of the camera frame in the position of the robot before. Therefore, it could also be adapted. The equations defining the relationship between the position of the object in the frame and the robot position, considering the centroid of object grasp (x_{medium} and y_{medium}) and the camera center point of the frame (x_{cam} and y_{cam}), are as follows:

1. $y_{medium} < y_{cam}$

$$x_{pos} = 0.250 + \left(\left(\frac{y_{cam} - y_{medium}}{24}\right) \times 0.010\right) \tag{17}$$

2. $y_{medium} > y_{cam}$

$$x_{pos} = 0.250 - \left(\left(\frac{y_{medium} - y_{cam}}{24}\right) \times 0.010\right) \tag{18}$$

3. $x_{medium} < x_{cam}$

$$y_{pos} = -0.005 - \left(\left(\frac{x_{cam} - x_{medium}}{35}\right) \times 0.010\right) \tag{19}$$

4. $x_{medium} > x_{cam}$

$$y_{pos} = \left(\left(\frac{x_{medium} - x_{cam}}{35}\right) \times 0.010\right) - 0.005 \tag{20}$$

The location of the object in the robot was determined by the x_{pos} and y_{pos}. The center point of the camera frame was 314 pixels in the x-frame (x_{cam}) and 123 pixels in the y-frame (y_{cam}). A conversion factor of 0.010 was applied to change the value resulting from the consideration of pixel shift into meter units. The values of 35 and 24 were used to determine pixel units from the consideration of the centroid of object grasp and the camera center point of the frame. Whereas the values of 0.250 and −0.005 obtained from the center point of the camera frame in the position of the robot were used to get the specific position of the object in the position of the robot with consideration of the robot axes and the camera axes. In order to easily understand these concepts, for example, the x_{medium} is at 320 pixels and the y_{medium} is at 90 pixels. These positions mean that the equations we will use are Eqs. 17 and 20. Meanwhile, Eqs. 18 and 19, which are also used to determine the position of the object at the robot position, will be skipped. Before, we had mentioned that the x-frame of the camera was defined as the y-axis position of the robot, and the y-frame of the camera was the *x*-axis position of the robot. It means that the larger the x-frame value, the more the position of the point is to the right of the robot, which means that the *y*-axis position of the robot will be larger. Meanwhile, the larger the y-frame value, the closer the point position is to the robot, which means the *x*-axis position of the robot will be smaller. This also applies to the opposite condition on each axis. Therefore, if the x_{medium} is at 320 pixels, it will be known that the object in the position of the robot on the *y*-axis is larger than −0.005. Thus, after the conversion of pixels to meters, it has to be added up by −0.005. While the y_{medium} is at 90 pixels, it will be known that the object in the position of the robot on the *x*-axis is larger than 0.250. Thus, after the conversion of pixels to meters, 0.250 has to be added up by that value.

3.6 Workspace

The workspace of the robot and user was in an unobstructed environment, signifying the realization of the HRC concept in which the user was positioned alongside the robot. The objects designated for grasp by the robot were positioned in front of it, as shown in Fig. 10.

In Fig. 10, the robot was placed on the table directly in front of the user, and the material handling process occurred in the space between the parties. The HRC concept started with the component object placed in front of the robot, representing a real industry scenario where the object originated from another workstation or worker. The placement was independent of the user implementing the HRC concept in this study. Whenever the robot identified the object using its camera, it moved to grasp the object and position it in front of the user. However, instead of immediately delivering the object, the robot awaited a command from the user. The user communicated the command for the robot to give the object through hand gestures. Following the command, the robot delivered the object to the user at a predetermined point, ensuring a safe interaction between the robot and the user.

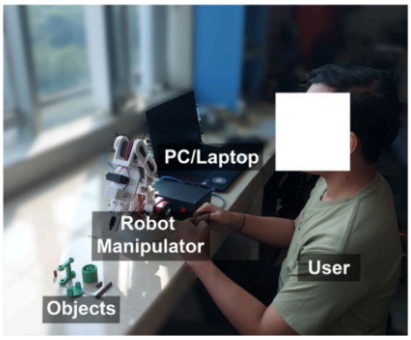


Fig. 10 Robot and human workspace

3.7 Identification of hand gesture

In order to perform the HRC concept in this study, the communication system implemented was a hand gesture given by the user to the robot as an instruction to deliver the object. This hand gesture was given in front of the robot camera which can be seen in Fig. 11.

In Fig. 11, we can see the hand gesture of the user defined using MediaPipe. This study used hand landmarks generated by MediaPipe to access points on the user's fingers. Blue dot and red dot are points that have an important role in giving commands to the robot. If the user positions the palm in an open position, in this case the position of the blue dot is above the red dot, the command given to the robot is to hold the object. Meanwhile, if the user positions the palm in a closed position, in this case the position of the blue dot is below the red dot, the command given to the robot is to give the object.

4 Experiments and results

In this section, we confirm the validity of the methods described in the previous section with the following experiments:

1. Testing the robot when grasping the objects. This experiment aims to validate the previously established equations. Moreover, the test also assessed the performance of the robot in predicting and executing object grasp positions.
2. Testing the robot when implementing the HRC concept. This experiment aims to assess the success rate of the robot and evaluate how it executes the HRC concept to successfully deliver all the objects to the user.
3. Testing the accuracy of hand gesture recognition implementing in the HRC concept. This experiment aims to assess the capabilities of the communication system applied to the HRC concept of this study.

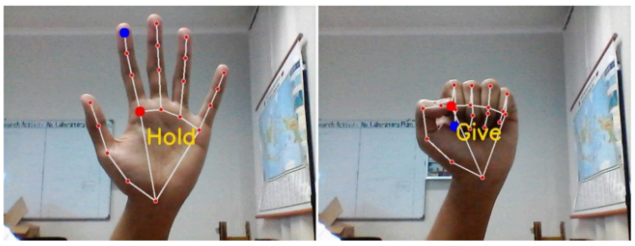
4.1 Performance on grasping objects

4.1.1 Dataset evaluation

The results obtained from the object dataset that had been trained using the YOLOv8 algorithm were in the form of the robot capability to detect objects, as shown in Fig. 12. The main indicators that defined the robot capabilities were mean Average Precision (mAP), precision, and recall. In this study, we didn't make any changes to the YOLOv8's parameters except for the epochs and got the results of those indicators as shown in Table 2.

In Table 2, we can see the mAP, precision, and recall values resulted from the model training processed by the YOLOv8 algorithm. The mAP@0.5 means that the object is classified correctly and the predicted object bounding box had an IoU higher than or equal to 0.5 with the ground truth bounding box. Precision is defined as the ratio of the number of correct predictions to the total predictions that are

Fig. 11 User's hand gestures



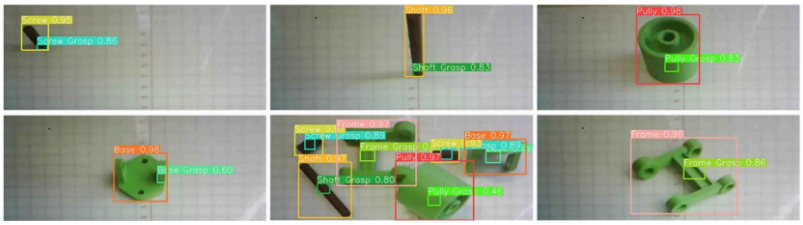


Fig. 12 Example of object detection results

Table 2 Evaluation on dataset

Class	Precision (%)	Recall (%)	mAP@0.5 (%)
All	92.6	92.5	92.5
Pully	98.0	100.0	99.5
Frame	98.7	100.0	99.5
Base	98.3	100.0	99.5
Shaft	98.3	100.0	99.5
Screw	98.9	100.0	99.5
Pully grasp	59.9	59.1	51.7
Frame grasp	100.0	92.3	95.0
Base grasp	80.9	80.0	86.2
Shaft grasp	95.8	96.0	95.4
Screw grasp	97.0	97.6	99.4

predicted as positive values. In addition, Recall is defined as the ratio of the number of correct predictions to the total number of positive actual conditions. The data in Table 2 was obtained from the model prediction of the dataset in the "valid" folder, which was new data outside the training dataset. It can be seen in the overall class that the model produces very good predictions, as seen by the mAP, precision, and recall values of 92.5%, 92.6%, and 92.5%, respectively. However, these values are not solely used as a reference in evaluating the performance in object grasping in this study because specifically the class that is responsible for object grasping is the "grasp" class. It can be seen that the "grasp" class has a tendency to have lower precision and recall values than the "non-grasp" class. This happens because of the different definitions for these classes. For more details, it can be seen in the confusion matrix from the results of the model training process shown in Fig. 13.

Figure 13 shows the confusion matrix of the training model result, which provides information on the relationship between the detection results performed by the model (predicted) and the actual condition (true). Based on this matrix, we can identify the object detection accuracy generated in each class by considering the True Positive

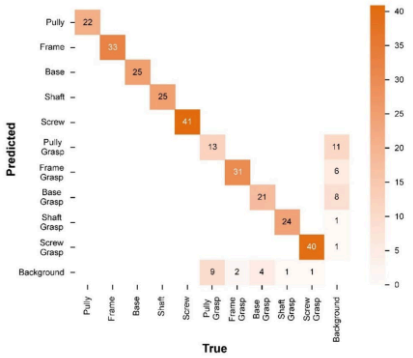


Fig. 13 Confusion matrix of object detection model

(TP), True Negative (TN), False Positive (FP), and False Negative (FN) values, which can be formulated in Eq. 21. TP is positive data that is predicted correctly. TN is negative data that is predicted correctly. FP is negative data but predicted as positive data. While FN is positive data predicted as negative data. In the case of this study, the model error appears because it incorrectly predicts the class defined as background or otherwise. The definition of this model error can be seen in the FP and FN values. In this study, model error only appeared in the "grasp" class. Meanwhile, the "non-grasp" class did not have FP and FN values, which meant that the "non-grasp" class has a detection accuracy of 100%. Whereas the detection accuracy for each class in the "grasp" class, with a total of 292 instances that were supposed to be detected, is shown in Table 3.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \times 100\% \quad (21)$$

Table 3 Accuracy of "grasp" class

Class	TP	TN	FP	FN	Accuracy (%)
Pully Grasp	13	259	11	9	93.15
Frame Grasp	31	253	6	2	97.26
Base Grasp	21	259	8	4	95.89
Shaft Grasp	24	266	1	1	99.32
Screw Grasp	40	250	1	1	99.32
Average accuracy					96.99

Based on Table 3, the accuracy produced by each class in the "grasp" class has very good accuracy, with an average accuracy of 96.99%. However, when looking in more detail at the model error value in the "grasp" class, there was a condition of multiple label classification, which was a condition where one class has several label definitions. In this study, this condition occurred because the definition of the object grasp point was determined based on the position of the object on the measuring mat and the capability of the robot gripper to grasp the object, which caused the object grasp point to vary according to the position and orientation of the object in the robot's workspace. An example of this condition can be seen in Fig. 14.

Figure 14 shows that the grasping point of the Pully varies. This condition was also observed in other components, subsequently affecting the confidence score produced by the model prediction and the model learning process. The confidence score affected the precision and recall values, as shown in Fig. 15. Meanwhile, the model learning process affected the capability of the model to learn, which was defined by the loss value as seen in Fig. 16.

In Fig. 15, it can be seen that on the precision-confidence curve, a low confidence score in the "grasp" class had a much lower precision than the "non-grasp" class. This meant that at low confidence there were many model errors in predicting as a result of the selection process on the IoU value of all label definitions in the class, and if the label did not meet the IoU value, it was defined as negative data. Whereas on the recall-confidence curve, the recall decline in the "grasp" class was much faster than the "non-grasp" class. In this case, it meant that in the "grasp" class, the model lost its positive data faster. Therefore, the model tended to produce fewer detections at high confidence.

In Fig. 16, it can be seen that there are two main loss parameters resulted, namely box_loss, which measures the prediction error in placing the bounding box surrounding the object in the image. This includes errors in predicting the position (centroid coordinates), size (width and height), and shape (aspect ratio) of the bounding box. Meanwhile, the other parameter is cls_loss, which measures the error in classifying objects inside the predicted bounding box. These two parameters are each considered against the epoch value, which is the learning cycle of the model on the dataset. Based on the loss graph, it can be seen that the model had a tendency to be slower in learning. This condition occurred as a result of the model having difficulty in inferring the exact features of the "grasp" class that distinguished it from its surroundings, such as its unique color, shape, size, etc.

In addition to the main indicators, there were other indicators related to the effect of model building and conditions when the model was run on the device PC/laptop. The other indicators were inference time, model size, and training time. The inference time result from the

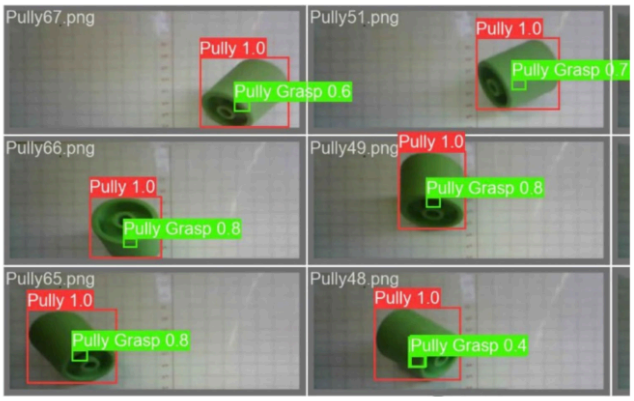


Fig. 14 Variety in Pully's grasp points

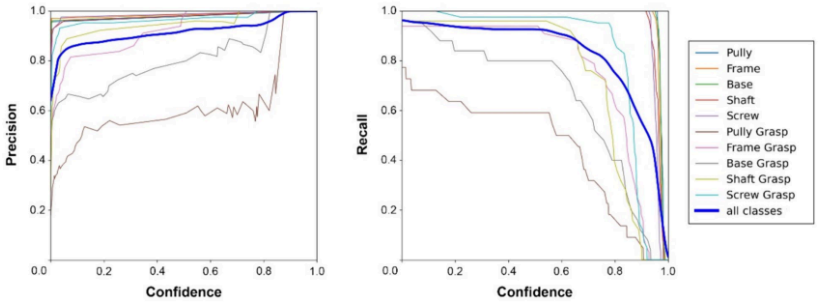
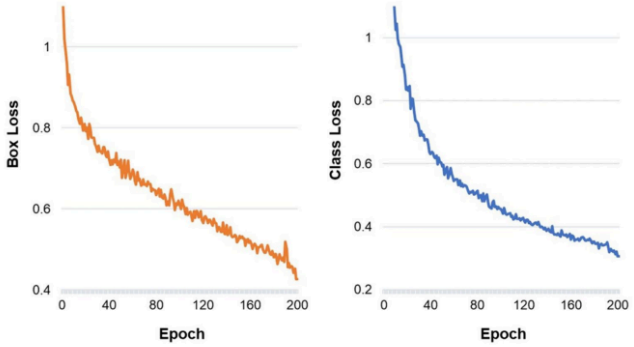


Fig. 15 Precision and recall curves against confidence

Fig. 16 Loss graph of training model



object dataset training process was 4.1 ms. However, the condition would be different if the model were applied in real-time through the camera, as in this study. The inference time obtained when the robot in real-time detected the object and was executed by the CPU was 60–70 ms (16.67–14.28 fps). Meanwhile, the model size generated for the results of the object dataset training process was 5.94 MB. As for the training time required, it was 0.704 hours, or about 42 minutes.

4.1.2 Physical evaluation

The test of the robot when grasping the objects involved the use of a measuring mat. It was conducted 10 times, both with a single object and multiple objects, following the method shown by Rakshit et al. (Rakshit et al. 2023). The object was randomly placed in the robot workspace, each with a predetermined orientation based on the grasping capability of the gripper. The analysis determined the success percentage of the robot in predicting and effectively grasping objects. However, the prediction of object grasp was specifically conducted below the condition of a single object, taking into account the predetermined points (in units of meters), namely: (1) $x = 0.250$, $y = -0.005$; (2) $x = 0.220$, $y = -0.005$; (3) $x = 0.270$, $y = -0.005$; (4) $x = 0.250$, $y = -0.020$; (5) $x = 0.250$, $y = 0.020$; (6) $x = 0.230$, $y = -0.020$; (7) $x = 0.240$, $y = 0.050$; (8) $x = 0.260$, $y = -0.060$; (9) $x = 0.270$, $y = 0.050$; dan (10) $x = 0.270$, $y = -0.040$. Thus, the accuracy of the robot in predicting object positions could be determined. Meanwhile, the

conditions of the multiple objects were also considered with objects that were randomly positioned without following any specific arrangement. In this situation, observing the arrangement in the program, the order of object grasping was defined sequentially, starting from Frame, Shaft, Pully, Screw, and Base.

In testing the robot for grasping a single object, two assessments were performed, namely, success in predicting object grasp and the execution of grasping the object by the robot, considering predetermined points, as shown in Table 4.

In Table 4, the results of the test showed how well the robot performed in grasping a single object placed in different positions. The outcomes covered both the prediction and execution of the robot. In addition, predictions were generated using calculations in the robot program for the x and y axes resulted by Eqs. 17, 18, 19, 20. These calculated positions were later compared with predetermined ones, leading to errors expressed for the x and y axes. Following these values, the objects' average error values were considered to find the total average error. While the success rate in the execution of the robot for grasping objects was evaluated based on the capability to reach the position of the object and lift it.

Based on Table 4, the error values varied for each object, and the total average error was approximately 0.004 for the x-axis and 0.004 for the y-axis. The error was influenced by the condition of placing the center point of the camera frame on the measuring mat, the condition when annotating the object, the equation in defining the position of the camera frame to the robot position, and the precision of object detection. The condition of placing the center point of the camera frame on the measuring mat concerned the repeatability of the robot, and in this case, the robot used in this study has very poor repeatability due to the condition of the actuator used. Meanwhile, in this study, to be able to carry out the proposed concept, it required conditions when the robot would grasp the object, the same as the dataset used. The consideration against the conditions when annotating objects, this was related to the influence of the camera perspective, which required maintaining consistency in placing the object grasp point. The equation in defining the camera frame against the robot position was related to the possibility of inconsistent changes every 0.010 m shift in the work plane against the camera frame plane as a result of the camera position shifting from the condition when defining the frame position against the robot position. Meanwhile, when considering the precision of object detection by the YOLO algorithm, this was related to the difficulty of the model to be able to infer the actual position of the object. Therefore, these things become very important considerations to be able to reduce the error value. While for the execution of the robot, the percentage of robot success in grasping objects

Table 4 Results of grasping a single object

Object / Component	Robot prediction (meters)		Error		Robot execution
	x	y	x	y	
Pully	0.248	−0.006	0.002	0.001	●
	0.220	−0.007	0.000	0.002	●
	0.266	−0.004	0.004	0.001	●
	0.245	−0.021	0.005	0.001	–
	0.250	0.012	0.000	0.008	●
	0.230	−0.059	0.000	0.009	●
	0.248	0.050	0.008	0.000	●
	0.265	−0.065	0.005	0.005	–
	0.274	0.056	0.004	0.006	●
	0.276	−0.047	0.006	0.007	●
Pully's average error			0.003	0.004	Total 8/10
Frame	0.264	−0.006	0.014	0.001	●
	0.224	−0.009	0.004	0.004	●
	0.278	−0.008	0.008	0.003	●
	0.258	−0.023	0.008	0.003	●
	0.256	0.024	0.006	0.004	●
	0.229	−0.057	0.001	0.007	●
	0.246	0.051	0.006	0.001	●
	0.268	−0.067	0.008	0.007	●
	0.277	0.047	0.007	0.003	●
	0.275	−0.043	0.005	0.003	●
Frame's average error			0.006	0.004	Total 10/10
Base	0.254	−0.008	0.004	0.003	●
	0.220	−0.003	0.000	0.002	●
	0.275	−0.005	0.005	0.000	●
	0.253	−0.023	0.003	0.003	●
	0.253	0.011	0.003	0.009	●
	0.235	−0.061	0.005	0.011	●
	0.244	0.051	0.004	0.001	●
	0.262	−0.064	0.002	0.004	●
	0.278	0.052	0.008	0.002	●
	0.273	−0.051	0.003	0.011	●
Base's average error			0.004	0.005	Total 10/10
Shaft	0.251	−0.003	0.001	0.002	●
	0.227	−0.007	0.007	0.002	●
	0.266	−0.004	0.004	0.001	–
	0.251	−0.021	0.001	0.001	●
	0.253	0.016	0.003	0.004	●
	0.235	−0.053	0.005	0.003	●
	0.248	0.046	0.008	0.004	●
	0.268	−0.064	0.008	0.004	●
	0.278	0.044	0.008	0.006	●
	0.274	−0.046	0.004	0.006	●
Shaft's average error			0.005	0.003	Total 9/10

Table 4 (continued)

Object / Component	Robot prediction (meters)		Error		Robot execution
	x	y	x	y	
Screw	0.251	−0.010	0.001	0.005	●
	0.225	−0.005	0.005	0.000	●
	0.265	−0.008	0.005	0.003	–
	0.249	−0.022	0.001	0.002	●
	0.248	0.020	0.002	0.000	●
	0.228	−0.058	0.002	0.008	●
	0.235	0.043	0.005	0.007	–
	0.265	−0.066	0.005	0.006	●
	0.264	0.047	0.006	0.003	–
	0.269	−0.045	0.001	0.005	–
Screw's average error			0.003	0.004	Total 6/10
Total average error			0.004	0.004	Success rate 86%

Notes: (1) ● = successful
(2) – = fail

was 86%. This showed that the robot had a good capability to apply the concepts used in this study.

Table 5 Results of grasping the multiple objects

Object/Component	Robot execution
Pully	9/10
Frame	8/10
Base	7/10
Shaft	8/10
Screw	6/10
Success rate	76%

In testing the robot for grasping multiple objects, only the success of the execution of grasping objects by the robot was considered and evaluated based on the capability of the robot to reach the position of the object and lift it, as shown in Table 5.

In Table 5, it was evident that the success rate of the robot in grasping multiple objects during the execution was 76%, showing a decrease in the capability compared to the single object. This decline occurred because the robot initially defined the positions of all objects, and when grasping one object, the robot inadvertently caused other objects to shift, affecting its capability to grasp the intended object. The robot determined object positions based on the initial placement, leading to inaccuracies during grasping in the multiple object condition. Additionally, the order of object grasp was influenced by the object detection algorithm. Despite a predefined order, the robot grasped objects based on the sequence in which the objects were detected by the YOLO algorithm. This supported the interpreter properties of Python, executing programs sequentially from top to bottom.

4.2 Performance on HRC concept

The test of the robot when implementing the HRC concept was explored to assess the capability of applying HRC to a robot manipulator in the material handling process. Before exploring, comprehending the process of implementing the HRC concept was important, as shown in Fig. 17.

In the execution of the HRC concept, the robot was initially positioned in the defined workspace, as shown in Fig. 17. The robot moved upon identifying an object through its camera, delivering it to the position of the user. In step 4, the user commanded the robot using hand gestures detected by MediaPipe, capturing the palm of the user. Furthermore, the pose point on the finger of the user served as a command input for the robot. An open-finger position instructed the robot to hold the object, while a closed-finger position instructed the robot to give the object to the user. After giving out the object, the robot returned to its initial position and identified the object.

The test conducted on the HRC concept assessed the success rate of the robot, evaluating how it executed the HRC concept from initiation to completion in conditions of random object positions and without a measuring mat. This test was conducted 10 times that involved five tests in which the objects were placed one by one, waiting for the robot to be in a position to detect each object. The remaining five tests included placing objects continuously, regardless of whether the robot was still in motion or grasping objects. The results of the test in executing the HRC concept are shown in Table 6.

Table 6 shows the success rate of the robot in grasping objects during the execution of the HRC concept was 92% for placing objects one by one and 84% for placing objects continuously. Similar to the issue encountered with single-object grasp, the success of the robot in completing the HRC concept was dependent on successfully grasping objects. Ensuring the capability of the robot to grasp objects accurately was important for the execution of the HRC concept in this study. However, it was important to observe that, for the robot to receive instructions effectively, precautions had to be taken to avoid objects obstructing the camera, allowing the robot to interpret hand gestures correctly.

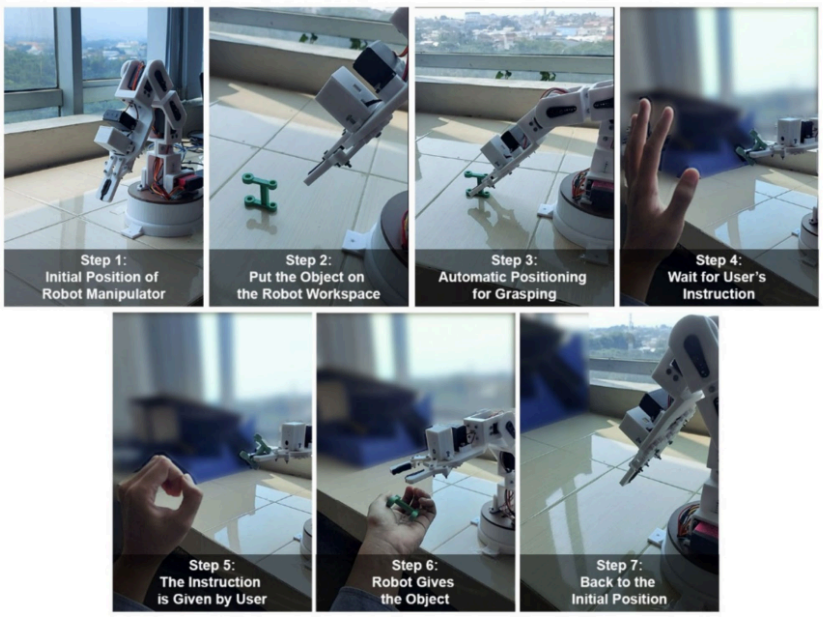


Fig. 17 HRC steps of concept

Table 6 Performance on the HRC concept

Object/Component	Robot execution	
	One by one	Continuously
Pully	5/5	5/5
Frame	5/5	4/5
Base	5/5	5/5
Shaft	4/5	4/5
Screw	4/5	3/5
Success rate	92%	84%

4.3 Performance of hand gesture recognition

In testing the performance of hand gesture recognition in this study, we used a dataset that we defined with various hand positions and poses (can be seen in Fig. 18). This dataset contained 154 images, with each image measuring 640 × 480 pixels. In order to produce more robust accuracy, we augmented the dataset using Albumentation, which was an open-source Python library for fast and flexible image augmentation (Buslaev et al. 2020). The augmentation that we performed

Fig. 18 Sample of hand gesture dataset

Fig. 19 Sample of data augmentation of hand gesture dataset

Table 7 Performance of hand gesture recognition

Dataset	Number of true predictions	Accuracy (%)
Original dataset	143	92.86
Horizontal flip	139	90.26
Vertical flip	140	90.91
Gaussian blur	143	92.86
Random brightness contrast	142	92.21
Average accuracy		91.82

was with the Horizontal Flip, Vertical Flip, Gaussian Blur, and Random Brightness Contrast pipelines, each applied to the original dataset (can be seen in Fig. 19). Thus, each of these pipelines produced 154 images as well, and then we compared the augmentation results of each pipeline with the original dataset to get the accuracy in each of these conditions. Furthermore, we averaged the accuracy results of each condition to get the overall accuracy of the hand gesture recognition used in this study. The parameters that we used to calculate the accuracy of hand gesture recognition were the number of true predictions (meaning all hand landmarks appear) and the total images, which could be formulated as follows:

$$\text{Accuracy} = \frac{\text{Number of True Predictions}}{\text{Total Image}} \times 100\% \quad (22)$$

Based on Eq. 22, the accuracy of hand gesture recognition in this study can be presented in Table 7. The average accuracy of the entire dataset had an accuracy rate of 91.82%. The difference in accuracy produced by each dataset also has a value that is not much. In this case, MediaPipe has a high accuracy rate even though it is applied to a variety of different conditions. Therefore, the use of MediaPipe in developing a communication system on the HRC concept can be the best option for ease of use and setup. However, in practice, as mentioned in Sect. 4.2, this concept can only be applied if the object does not obstruct the camera, meaning that the landmark points used as references must be visible to the camera.

5 Discussion

In the previous tests, the prototype of robot manipulator successfully executed the HRC concept in material handling process. However, some issues require consideration and evaluation. In Section 1, we concentrated on introducing a simplified method of autonomous object grasping in the material handling process for HRC. This method was related not only to the robot used but also to the other aspects used, including cameras, detection positions, and others. When viewed from the robot used, it affected the accuracy of the robot in grasping objects and its repeatability. The use of an RGB camera in this method limits object position definition to the x-axis and y-axis. This is because adjustments made using the warping method only apply to the work plane itself. Meanwhile, the object will still be affected by the camera perspective. Therefore, the z-axis has to be fixed in a certain position. However, object detection using the YOLO algorithm combined with the warping method could overcome the influence of camera perspective in grasping objects. Thus, the robot was still able to determine the position of the object without additional parameters, as was done in previous studies involving depth parameters. Additionally, based on the initial position for detecting an object, in this method only one condition was used. If there is more than one object detection position, then this method can't be used because it will affect the object detection algorithm, which can possibly cause incorrect detection of objects. On the other side, in the communication system method, the robot can only receive instructions if the command of the user can be seen by the camera. In the case where the grasped object covers the entire camera frame, this method can't be used.

Nevertheless, our method offered faster inference time when the robot in real-time detected the object, which even just running on the CPU obtained 60–70 ms (16.67–14.28

fps) than the study by Rakshit et al. (2023), which running on the GPU obtained 11.1 fps. This shows that with our method, we can reduce the computational load. Thus, the robot can perform the tasks more quickly. In addition to the test results in terms of performance on grasping objects, our method had a percentage that was close to previous studies, including that conducted by Rakshit et al. (2023), which had a percentage of robot success in grasping a single object, namely 97.8%, and multiple objects, namely 96.4%. The study conducted by Zhang et al. (2019) was 92.5% for a single object and 83.75% for multiple objects, as was the study conducted by Mousavian et al. (2019), which had a percentage of robot success in grasping objects, namely 88%. Furthermore, in the test of the robot when implementing the autonomous object grasping for the HRC concept, our method may be said to be the simplest method when compared with methods conducted in previous studies, including those by Mousavian et al. (2019) and Christen et al. (2022), which used the reinforcement learning method, as well as studies conducted by Rakshit et al. (2023), Rosenberger et al. (2021), and Breyer et al. (2021), which used depth parameter in feature extraction to make the planning process more complex. In addition, the studies conducted by Sanchez-Matilla et al. (2020) used a method that involved considering dimensions, transparencies, shapes, and fillings with two RGB cameras, which made the planning process require additional equipment aside from a camera.

The test results of the robot on the HRC concept showed a significant error in the scale of the robot used in this study. The error was attributed to the precision of the robot manipulator, the placement of the center point of the camera frame against the measuring mat, the conditions during object annotation, the equation in defining the position of the camera frame to the position of the robot, and the precision of object detection. In addition to addressing precision and camera frame placement issues, it is recommended to use a robot with higher precision or an improved actuator. This will improve accuracy in grasping objects and prevent shifts in the camera frame when the robot returns to the position for detecting the object. Addressing object annotation conditions includes creating an object dataset with predetermined placement positions, allowing precise annotation of the object grasp point at predetermined locations on the measuring mat. Furthermore, possible errors resulting from the equation defining the position of the camera frame concerning the position of the robot might have appeared due to inconsistent changes in point shifts in the frame. As a result, the equations used in this study might not have sufficed. Exploring machine learning concepts, including decision trees or linear regression, can be considered to make an equation accurately define the frame position. Meanwhile, the problem of object detection precision can be overcome by providing more specific information regarding the position of objects that have certain features to make it easier for the object detection algorithm to learn the position of the object.

Thus, the consequences of the simplified method in this study are: (1) The annotation process of the object dataset has to be selective and carried out carefully; (2) The more varied the object grasp points are, the more the precision of object detection decreases and the learning process of the model is slower; (3) If the condition of object detection is quite different from the condition in the training dataset, it may lead to a fault in predicting the position of the object; (4) The limitation of the Field of View (FOV) of the camera makes this method can only be used at the camera frame boundary; and (5) The use of an RGB camera in this method makes the position of the object on the z-axis coordinate can't be determined. If there is a need for consideration of this coordinate, it can be achieved by adding manually to the robot program for each component.

6 Conclusion

This study presents the robot performance on autonomous object grasping in the material handling process for the HRC concept with a simplified method, where the specially designed robot independently grasping objects with random positions in the workspace achieved a success rate of 92% for placing objects one by one and 84% for placing objects continuously. Additionally, in the performance of grasping objects without the HRC concept, the robot achieved a success rate of 86% for a single object and 76% for multiple objects, with the average error (in units of meters) being approximately 0.004 for the x-axis and 0.004 for the y-axis. The error was influenced by factors including the precision of the robot manipulator, the placement of the center point of the camera frame against the measuring mat, the conditions during object annotation, the equation in defining the position of the camera frame to the position of the robot, and the precision of object detection. Nevertheless, our method offered faster inference time when the robot in real-time detected the object, which was even just running on the CPU. In addition, our method also offered the simplest method without complexity and required additional equipment aside from an RGB camera for the planning process.

Additionally, the important consideration in our proposed simplified method was the use of RGB cameras, limiting the definition of the position of the object to the x-axis and y-axis, with the z-axis remaining fixed in a certain position.

In addition, based on the initial position for detecting an object, in this study only one condition was used. If there is more than one object detection position, then this method can't be used because it will later affect the object detection algorithm, which can possibly cause incorrect detection of objects. On the other side, in the communication system method, the robot can only receive instructions if the command of the user can be seen by the camera. In the case where the grasped object covers the entire camera frame, this method can't be used.

Acknowledgements The authors thank UPT Laboratorium Terpadu, Diponegoro University, and Robotics & Automation Laboratory (RAL) for supporting this study.

Author contributions Conceptualization: Muhammad Farouk Setiawan; Methodology: All authors; Formal analysis: All authors; Investigation: Muhammad Farouk Setiawan; Software: Muhammad Farouk Setiawan; Writing—original draft preparation: All authors; Writing—review and editing: All authors; Validation: P. Paryanto and Joga Dharma Setiawan; Supervision: P. Paryanto and Joga Dharma Setiawan. All authors read and approved the final manuscript.

Data availability Data will be made available on request.

Declarations

Competing Interests The authors declare no competing interests.

References

Ainetter, S., Fraundorfer, F.: End-to-end Trainable Deep Neural Network for Robotic Grasp Detection and Semantic Segmentation from RGB. In: 2021 IEEE International Conference on Robotics and Automation (ICRA). pp. 13452–13458. IEEE, Xi'an, China (2021)

Akkaladevi, S.C., Plasch, M., Pichler, A., Ikeda, M.: Towards reinforcement based learning of an assembly process for human robot collaboration. Procedia Manuf. **38**, 1491–1498 (2019). https://doi.org/10.1016/j.promfg.2020.01.138

Al, G.A., Estrela, P., Martinez-Hernandez, U.: Towards an intuitive human-robot interaction based on hand gesture recognition and proximity sensors. In: 2020 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI). pp. 330–335. IEEE, Karlsruhe, Germany (2020)

Alvin, A., Shabrina, N.H., Ryo, A., Christian, E.: Hand Gesture Detection for Sign Language using Neural Network with mediapipe. Ultima Comput. J. Sist. Komput. 13, 57–62 (2021). https://doi.org/10.31937/sk.v13i2.2109

Andersen, R.S., Madsen, O., Moeslund, T.B., Amor, H.B.: Projecting robot intentions into human environments. In: 2016 25th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN). pp. 294–301. IEEE, New York, NY, USA (2016)

Arad, N., Dyn, N., Reisfeld, D., Yeshurun, Y.: Image warping by radial basis functions: application to facial expressions. CVGIP Graph. Models Image Process. **56**, 161–172 (1994). https://doi.org/10.1006/cgip.1994.1015

Asada, H.H.: Introduction to Robotics. In: 2.12 Lecture Notes. MIT, USA (2005)

Banerjee, T., Srikar, K.V.P., Reddy, S.A., Biradar, K.S., Koripally, R.R., Varshith, Gummadi.: Hand Sign Recognition using Infrared Imagery Provided by Leap Motion Controller and Computer Vision. In: 2021 International Conference on Innovative Practices in Technology and Management (ICIPTM). pp. 20–25. IEEE, Noida, India (2021)

Baratta, A., Cimino, A., Gnoni, M.G., Longo, F.: Human Robot Collaboration in Industry 4.0: a literature review. Procedia Comput. Sci. **217**, 1887–1895 (2023). https://doi.org/10.1016/j.procs.2022.12.389

Bazarevsky, V., Kartynnik, Y., Vakunov, A., Raveendran, K., Grundmann, M.: BlazeFace: Sub-millisecond Neural Face Detection on Mobile GPUs, http://arxiv.org/abs/1907.05047, (2019)

Bezmaternykh, P.V., Nikolaev, D.P., Arlazarov, V.L.: High-performance digital image processing. Pattern Recognit Image Anal. **33**, 743–755 (2023). https://doi.org/10.1134/S1054661823040090

Breyer, M., Chung, J.J., Ott, L., Siegwart, R., Nieto, J.: Volumetric Grasping Network: Real-time 6 DOF Grasp Detection in Clutter, http://arxiv.org/abs/2101.01132, (2021)

Buslaev, A., Iglovikov, V.I., Khvedchenya, E., Parinov, A., Druzhinin, M., Kalinin, A.A.: Albumentations: fast and flexible image augmentations. Information **11**, 125 (2020). https://doi.org/10.3390/info11020125

Cahyono, G.R., Nurmahaludin, Setiawan, M.F., Rizal, Y., Riadi, J.: Comparison of 4 DOF Arm Robot for Trajectory Planning with 3rd and 5th Polynomial Orders. In: 2022 11th Electrical Power, Electronics, Communications, Controls and Informatics Seminar (EECCIS). pp. 281–286. IEEE, Malang, Indonesia (2022)

Cao, H., Chen, G., Li, Z., Lin, J., Knoll, A.: Lightweight Convolutional Neural Network with Gaussian-based Grasping Representation for Robotic Grasping Detection, http://arxiv.org/abs/2101.10226, (2021)

Castro, A., Silva, F., Santos, V.: Trends of human-robot collaboration in industry contexts: handover, learning, and metrics. Sensors. **21**, 4113 (2021). https://doi.org/10.3390/s21124113

Christen, S., Yang, W., Pérez-D'Arpino, C., Hilliges, O., Fox, D., Chao, Y.-W.: Learning Human-to-Robot Handovers from Point Clouds, http://arxiv.org/abs/2303.17592, (2023)

Corke, P.: Robotics. Vision and Control. Springer International Publishing, Cham (2017)

Craig, J.J.: Introduction to robotics: mechanics and control. Addison-Wesley, Reading, Mass (1994)

Dairath, M.H., Akram, M.W., Mehmood, M.A., Sarwar, H.U., Akram, M.Z., Omar, M.M., Faheem, M.: Computer vision-based prototype robotic picking cum grading system for fruits. Smart Agric. Technol. **4**, 100210 (2023). https://doi.org/10.1016/j.atech.2023.100210

Glasbey, C.A., Mardia, K.V.: A review of image-warping methods. J. Appl. Stat. **25**, 155–171 (1998). https://doi.org/10.1080/02664769823151

Horňáková, N., Jurík, L., Hrablik Chovanová, H., Cagáňová, D., Babčanová, D.: AHP method application in selection of appropriate material handling equipment in selected industrial enterprise. Wirel. Netw. **27**, 1683–1691 (2021). https://doi.org/10.1007/s11276-019-02050-2

Indriani, Harris, Moh., Agoes, A.S.: Applying Hand Gesture Recognition for User Guide Application Using MediaPipe: Presented at the 2nd International Seminar of Science and Applied Technology (ISSAT 2021) , Bandung, Indonesia (2021)

Li, X., Chen, J., He, Y., Yang, G., Li, Z., Tao, Y., Li, Y., Li, Y., Huang, L., Feng, X.: High-through counting of Chinese cabbage trichomes based on deep learning and trinocular stereo

microscope. Comput. Electron. Agric. **212**, 108134 (2023). https://doi.org/10.1016/j.compag.2023.108134

Li, X., Wang, J., Xu, F., Song, J.: Improvement of YOLOv3 Algorithm in Workpiece Detection. In: 2019 IEEE 9th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER). pp. 1063–1068. IEEE, Suzhou, China (2019)

Lotsaris, K., Fousekis, N., Koukas, S., Aivaliotis, S., Kousi, N., Michalos, G., Makris, S.: Augmented Reality (AR) based framework for supporting human workers in flexible manufacturing. Procedia CIRP. **96**, 301–306 (2021). https://doi.org/10.1016/j.procir.2021.01.091

Lugaresi, C., Tang, J., Nash, H., McClanahan, C., Uboweja, E., Hays, M., Zhang, F., Chang, C.-L., Yong, M.G., Lee, J., Chang, W.-T., Hua, W., Georg, M., Grundmann, M.: MediaPipe: A Framework for Building Perception Pipelines, http://arxiv.org/abs/1906.08172, (2019)

Mishra, O., Suryawanshi, P., Singh, Y., Deokar, S.: A Mediapipe-Based Hand Gesture Recognition Home Automation System. In: 2023 2nd International Conference on Futuristic Technologies (INCOFT). pp. 1–6. IEEE, Belagavi, Karnataka, India (2023)

Mohammadi, Z., Akhavanpour, A., Rastgoo, R., Sabokrou, M.: Diverse hand gesture recognition dataset. Multimed. Tools Appl. **83**, 50245–50267 (2023). https://doi.org/10.1007/s11042-023-17268-8

Mousavian, A., Eppner, C., Fox, D.: 6-DOF GraspNet: Variational Grasp Generation for Object Manipulation, http://arxiv.org/abs/1905.10520, (2019)

Nooruddin, N., Dembani, R., Maitlo, N.: HGR: Hand-Gesture-Recognition Based Text Input Method for AR/VR Wearable Devices. In: 2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC). pp. 744–751. IEEE, Toronto, ON, Canada (2020)

OpenAI, O., Plappert, M., Sampedro, R., Xu, T., Akkaya, I., Kosaraju, V., Welinder, P., D'Sa, R., Petron, A., Pinto, H.P. d O., Paino, A., Noh, H., Weng, L., Yuan, Q., Chu, C., Zaremba, W.: Asymmetric self-play for automatic goal discovery in robotic manipulation, http://arxiv.org/abs/2101.04882, (2021)

Paletta, L., Brijacak, I., Reiterer, B., Pszeida, M., Ganster, H., Fuhrmann, F., Weiss, W., Ladstatter, S., Dini, A., Murg, S., Mayer, H.: Gaze-based Human Factors Measurements for the Evaluation of Intuitive Human-Robot Collaboration in Real-time. In: 2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA). pp. 1528–1531. IEEE, Zaragoza, Spain (2019)

Park, D., Seo, Y., Shin, D., Choi, J., Chun, S.Y.: A Single Multi-Task Deep Neural Network with Post-Processing for Object Detection with Reasoning and Robotic Grasp Detection. In: 2020 IEEE International Conference on Robotics and Automation (ICRA). pp. 7300–7306. IEEE, Paris, France (2020)

Rakhimkul, S., Kim, A., Pazylbekov, A., Shintemirov, A.: Autonomous Object Detection and Grasping Using Deep Learning for Design of an Intelligent Assistive Robot Manipulation System. In: 2019 IEEE International Conference on Systems, Man and Cybernetics (SMC). pp. 3962–3968. IEEE, Bari, Italy (2019)

Rakshit, A., Pramanick, S., Bagchi, A., Bhattacharyya, S.: Autonomous grasping of 3-D objects by a vision-actuated robot arm using Brain-Computer Interface. Biomed. Signal Process. Control **84**, 104765 (2023). https://doi.org/10.1016/j.bspc.2023.104765

Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You Only Look Once: Unified, Real-Time Object Detection, http://arxiv.org/abs/1506.02640, (2016)

Rosenberger, P., Cosgun, A., Newbury, R., Kwan, J., Ortenzi, V., Corke, P., Grafinger, M.: Object-Independent Human-to-Robot Handovers Using Real Time Robotic Vision. IEEE Robot. Autom. Lett. **6**, 17–23 (2021). https://doi.org/10.1109/LRA.2020.3026970

Ruprecht, D., Muller, H.: Image warping with scattered data interpolation. IEEE Comput. Graph. Appl. **15**, 37–43 (1995). https://doi.org/10.1109/38.365004

Sanchez-Matilla, R., Chatzilygeroudis, K., Modas, A., Duarte, N.F., Xompero, A., Frossard, P., Billard, A., Cavallaro, A.: Benchmark for human-to-robot handovers of unseen containers with unknown filling. IEEE Robot. Autom. Lett. **5**, 1642–1649 (2020). https://doi.org/10.1109/LRA.2020.2969200

Segura, P., Lobato-Calleros, O., Ramírez-Serrano, A., Soria, I.: Human-robot collaborative systems: Structural components for current manufacturing applications. Adv. Ind. Manuf. Eng. **3**, 100060 (2021). https://doi.org/10.1016/j.aime.2021.100060

Semeraro, F., Griffiths, A., Cangelosi, A.: Human–robot collaboration and machine learning: a systematic review of recent research. Robot. Comput.-Integr. Manuf. **79**, 102432 (2023). https://doi.org/10.1016/j.rcim.2022.102432

Tsamis, G., Chantziaras, G., Giakoumis, D., Kostavelis, I., Kargakos, A., Tsakiris, A., Tzovaras, D.: Intuitive and Safe Interaction in Multi-User Human Robot Collaboration Environments through Augmented Reality Displays. In: 2021 30th IEEE International Conference on Robot & Human Interactive Communication (RO-MAN). pp. 520–526. IEEE, Vancouver, BC, Canada (2021)

Vogel, C., Walter, C., Elkmann, N.: Safeguarding and supporting future human-robot cooperative manufacturing processes by a projection- and camera-based technology. Procedia Manuf. **11**, 39–46 (2017). https://doi.org/10.1016/j.promfg.2017.07.127

Wang, Z., Liu, Y., Duan, S., Pan, H.: An efficient detection of non-standard miner behavior using improved YOLOv8. Comput. Electr. Eng. **112**, 109021 (2023). https://doi.org/10.1016/j.compeleceng.2023.109021

Xing, X., Chang, D.E.: Deep Reinforcement Learning Based Robot Arm Manipulation with Efficient Training Data through Simulation, http://arxiv.org/abs/1907.06884, (2019)

Yang, W., Wu, J., Zhang, J., Gao, K., Du, R., Wu, Z., Firkat, E., Li, D.: Deformable convolution and coordinate attention for fast cattle detection. Comput. Electron. Agric. **211**, 108006 (2023). https://doi.org/10.1016/j.compag.2023.108006

Zeng, A., Song, S., Welker, S., Lee, J., Rodriguez, A., Funkhouser, T.: Learning Synergies between Pushing and Grasping with Self-supervised Deep Reinforcement Learning, http://arxiv.org/abs/1803.09956, (2018)

Zhang, H., Lan, X., Bai, S., Zhou, X., Tian, Z., Zheng, N.: ROI-based Robotic Grasp Detection for Object Overlapping Scenes. In: 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 4768–4775. IEEE, Macau, China (2019)

Zhang, F., Bazarevsky, V., Vakunov, A., Tkachenka, A., Sung, G., Chang, C.-L., Grundmann, M.: MediaPipe Hands: On-device Real-time Hand Tracking, http://arxiv.org/abs/2006.10214, (2020)

Zhang, H., Kebria, P.M., Mohamed, S., Yu, S., Nahavandi, S.: A Review on Robot Manipulation Methods in Human-Robot Interactions, http://arxiv.org/abs/2309.04687, (2023)

Zhang, S., Xie, M.: Real-time recognition and localization based on improved YOLOv5s for Robot's picking clustered fruits of chilies. Sensors. **23**, 3408 (2023). https://doi.org/10.3390/s23073408

Muhammad Farouk Setiawan is a Graduate Student in Mechanical Engineering Department at Diponegoro University, Indonesia. He received his B.Eng. degree in Mechanical Engineering from Lambung Mangkurat University, Indonesia. His research interests include Robotics, Computer Vision, and Mechanical Engineering.

P. Paryanto received his Dr.-Ing. degree in Automation and Production System from Friedrich-Alexander-Universitaet (FAU) Erlangen-Nuremberg, Germany. He is currently an Assistant Professor in Mechanical Engineering Department at Diponegoro University, Indonesia. His research interests include Digital Transformation and Industry 4.0, Factory Planning and Automation, Manufacturing, and Production System.

Joga Dharma Setiawan received his B.Sc. degree in Mechanical Engineering from Northeastern State University, USA and M.Sc. degree in Aeronautics and Astronautics from Massachusetts Institute of Technology, USA. He earned his Ph.D. degree in Mechanical Engineering from Michigan State University, USA. He is currently an Associate Professor in Mechanical Engineering Department at Diponegoro University, Indonesia. His research interests include Mechatronics and Robotics, Vehicle Dynamics, Aerospace Engineering, Machine Vision and AI, IoT and Hybrid Energy System.

Simplified autonomous object grasping in material handling process for human–robot collaboration

ORIGINALITY REPORT

9%	9%	%	2%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.mdpi.com Internet Source	1%
2	ebin.pub Internet Source	1%
3	patents.justia.com Internet Source	1%
4	repo-dosen.ulm.ac.id Internet Source	<1%
5	dokumen.pub Internet Source	<1%
6	www.frontiersin.org Internet Source	<1%
7	insightsociety.org Internet Source	<1%
8	arxiv.org Internet Source	<1%
9	doi.org Internet Source	<1%
10	ipfs.io Internet Source	<1%
11	www.researchgate.net Internet Source	<1%
12	www.scitepress.org Internet Source	<1%
13	vdocuments.mx Internet Source	<1%

14	orca.cardiff.ac.uk Internet Source	<1%
15	Submitted to King's College Student Paper	<1%
16	Submitted to University of Witwatersrand Student Paper	<1%
17	cibworld.org Internet Source	<1%
18	Submitted to National Taipei University of Technology Student Paper	<1%
19	export.arxiv.org Internet Source	<1%
20	rodin.uca.es Internet Source	<1%
21	www.hindawi.com Internet Source	<1%
22	1library.org Internet Source	<1%
23	Submitted to Modern Education Society's College of Engineering, Pune Student Paper	<1%
24	Submitted to The ISF Academy Student Paper	<1%
25	Submitted to Southern Cross University Student Paper	<1%
26	docsbay.net Internet Source	<1%
27	www.manuelgarcia.designrshub.com Internet Source	<1%
28	www.yahoo.com Internet Source	<1%

29	harbinengineeringjournal.com Internet Source	<1%
30	pubmed.ncbi.nlm.nih.gov Internet Source	<1%
31	www.medrxiv.org Internet Source	<1%
32	wscg.zcu.cz Internet Source	<1%
33	www2.mdpi.com Internet Source	<1%
34	ejournal.undip.ac.id Internet Source	<1%
35	jise.btu.edu.tr Internet Source	<1%
36	www.ecva.net Internet Source	<1%
37	e-journal.usd.ac.id Internet Source	<1%
38	ebook.unimma.ac.id Internet Source	<1%
39	openreview.net Internet Source	<1%
40	www.rc.is.ritsumei.ac.jp Internet Source	<1%
41	docplayer.net Internet Source	<1%
42	doczz.es Internet Source	<1%
43	epdf.pub Internet Source	<1%
44	eprints.whiterose.ac.uk Internet Source	<1%

45	harmony-eu.org Internet Source	<1%
46	ia803107.us.archive.org Internet Source	<1%
47	iris.polito.it Internet Source	<1%
48	link.springer.com Internet Source	<1%
49	peerj.com Internet Source	<1%
50	repository.upiyptk.ac.id Internet Source	<1%
51	vdoc.pub Internet Source	<1%
52	www.fil.ion.ucl.ac.uk Internet Source	<1%

Exclude quotes	On	Exclude matches	< 5 words
Exclude bibliography	On		