Determination of Angular Status and Dimensional Properties of Objects for Grasping with Robot Arm

Kürşad Uçar and Hasan Erdinç Koçer

***Abstract*—In any task, robot arms can work more efficiently without human control. With components such as imaging devices, it is possible to program robots to control autonomously. In this study, the calculation of object size and orientation in 2D images was carried out to grasp moving objects with the robot arm. Deep learning-based You Look Only Once (YOLO) recognizes objects moving at unknown speeds on a conveyor belt. Since YOLO does not produce any output about the orientation of objects, we eliminated this deficiency for horizontal objects using YOLO with Principal Component Analysis. For vertical objects, length calculations were made using two images. Then, the velocity, which is the most important property of moving objects, is calculated with template matching. Finally, the robot arm is controlled in the three parts to calculate the angles required for the grip. While two links (one joint)'s angles are calculated with simple image processing, one link is controlled by an Artificial Neural Network as a 3 degree of freedom planar robot arm. According to the obtained speed, the robot arm waits for the object to arrive and then, being held by the grip. In the trials, the robot arm achieved a successful grip of 94.86% and 91.43% for vertical and horizontal objects respectively.**

***Index Terms*—Learning and Adaptive Systems, Grasping, Kinematics, Recognition.**

I. Introduction

Autonomous robot systems have a great demand in recent years in many application areas, such as harvesting [1], garbage separation [2], and urban search and rescue [3]. They attract researchers due to the advantages of working in environments where people cannot work or are tirelessly and sensitively demanding. Serial robots are one of the frequently preferred autonomous systems in recent years. They can naturally imitate human arm movements which is very beneficial for the industry. Despite the advantages mentioned above, they are complex systems that combine data acquisition, image processing, classification tasks, and electronic control. However, the most common problems related to a robot arm can be summarized as follows:

1) The size and shape of target objects and their positions vary according to the application field. This situation makes it difficult to detect and grasp them for robot arms.

2) Some applications ask robot arms to capture moving targets. Different moving speeds and complex backgrounds are also vexed issues.

3) The system needs to work stably to meet requirements such as speed and accuracy for tasks based on target recognition, grasping, and classification.

Kürşad Uçar e-mail:kucar@selcuk.edu.tr.
Hasan Erdinç Koçer e-mail:ekocer@selcuk.edu.tr

Under these problems, robot operation seems to be limited in comparison to human [4] but it has been an important research topic. Generally, robot arms are automated to serve under created stable environmental conditions [4], [5]. However, robot arms are designed to perform smoothly in real-world environments. Changing environmental conditions have been a crucial factor in robot automation. In such difficult situations, the biggest helpers of robots are undoubtedly sensors and cameras. Such devices can provide sufficient information about the circumstances for grasping tasks. However, raw data means nothing to the robot arm. Therefore, the obtained images or sensor data should be made meaningful. The first noteworthy knowledge is to recognize objects in such tasks. The matter can be in different conditions in a real-world problem. It may even be necessary to grasp them from among a group. In a complex environment, it must be known which objects to grasp. In such cases, it is not desirable for the robotic manipulators and algorithms to deal with everything in the working place. So recognizing the relevant object, obtaining its position, and separating it from others are significant steps. Deep learning-based algorithms such as You Only Look Once (YOLO) [6], Single Shot MultiBox Detector (SSD) [7], and Faster Region-Based Convolutional Neural Networks (R-CNN) [8] are available for object recognition and positioning in the recent studies. These algorithms can work in real-time applications because they detect objects with high accuracy and speed [9], [10]. In addition, deep learning provides an opportunity to recognize objects without being affected by distortions in the images. After detecting the objects to be grasped, some variables of interest such as their position and size, grasping point must be obtained correctly.

Image processing algorithms make it easy to do all the mentioned operations. Various approaches have been made to grasp stationary objects by processing this information [4], [5], [11]–[13]. However, when manipulators are required to perform in real world conditions, it is not enough to only grasp fixed objects because objects can move at different speeds, which must be taken into account by the robot at any time [14]. Grasping moving objects can only be possible by predicting their movements, where discovering speed plays a critical role in this task. Since moving objects do not have a fixed position, the object stays in the robot arm's working area for a limited time [14], [15]. For these reasons, the problem of performing all operations in real-time is challenging.

In this study, an image processing and artificial intelligence-based solution to grasping problem in movement is presented. We created a setup as in Fig. 1 by mainly focusing on the following:

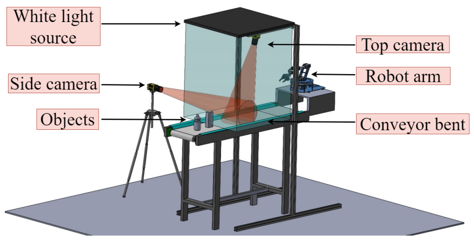


Fig. 1. Experimental setup.

1) We separated robot targets in a complex environment.

2) Target velocity was not given as input. We determined it with image processing. The grip was carried out at different speeds.

3) By using objects with different structural properties, grasping was achieved without being affected by the shapes of the objects.

4) Regardless of whether the objects were vertical or horizontal, all operations have been successfully executed.

Objects move on the conveyor by either standing vertically or horizontally in random placements (position, direction, etc.) on the conveyor along with other things that are not wanted to be grasped. The orientation of these objects is significant for grasping horizontal objects. Although YOLO is very successful in object detection, it does not provide information about the orientation of objects. On the other hand, the Principal Components Analysis (PCA) can calculate the orientations of every object that can be separated from the background [16], but it does not make object classification. By combining these two methods, we have achieved both rapid detection and finding the orientation of objects that is important for grasping. So we used PCA with YOLO to calculate the object orientation of interest. Although vertical objects have no orientation, image processing methods are used to calculate the length of different-size objects in vertical position. The height of the objects were calculated by making mathematical calculations on the same object detected in both side and top images. Because the study aims to realize grasping at different speeds without knowing the speed, correctly calculating the height of objects is very important for calculating velocity using template matching in the next step. After the operations, the manipulator was positioned according to the position, speed, and object direction (vertical, horizontal) information. To control the robot arm quickly, it was divided into three parts and therefore, we eliminated the need for complex inverse kinematic calculations. Two of these parts consist of a single joint. The third part was controlled as a 3 degree of freedom (dof) planar robot using an Artificial Neural Network (ANN) to get fast results.

The rest of the paper is as follows. Previous works are described in Section 2. The methods are explained in Section 3. Trials are mentioned in Section 4. The results are given in Section 5. And the conclusion is presented in Section 6.

II. Related Works

In the pioneering work on grasping moving objects, imaging system was examined [17]. A grasping strategy was applied by simulating human arm movements. The algorithm realized toy train grasping.

In the industry where robot arms are frequently used, conveyor belts are widely used for moving objects. It is possible to pick up moving objects on the conveyor belt with a vacuum gripper without difficulty. In [18], four different objects on the conveyor belt were also separated with a vacuum gripper using a 3D camera.

Image processing algorithms can distinguish objects based on their representative features, such as shape and color. In [19], the color of the object was detected with an infrared sensor placed next to a conveyor belt. The position of the sensor and gripper and the speed of the conveyor were constant. For this reason, the time elapsed until the moment of grasping after the sensor detected an object was simply calculated. At the end of this period, the grasper reached the object according to its color.

Objects can have different colors and different shapes. In [20], the colors, shapes, and positions of some objects were obtained by image processing for a separation task.

Various objects and environments create complex scenes in images, making them difficult to distinguish. In such cases, deep learning plays a key role in distinguishing them. In [2], deep learning was used for object recognition and pose estimation. They recognized the plastic bottles moving on the conveyor belt in complex scenes and grasped them with the manipulator.

The movement made by the robot arm to grasp the objects moving on the conveyor belt is similar to each other. Using this similarity, they presented constant-time motion algorithms in [14] based on the use of similar paths to grasp different objects moving on the conveyor belt. They have shown that their algorithm is highly effective in performing similar repetitive tasks.

Objects can also move without the conveyor belt. Objects can move in fixed and unstable routes, on the ground, or in the air. In [21], Kim et al. presented an approach for capturing thrown objects with a robotic arm. The system learned object flight dynamics by observing flying object samples. The grasping distributions of each object were learned by human demonstration. The system tried to find the appropriate catching motion by searching an accessible area for trajectory intersection of the launched object. The proposed approach allows the robot to take advantage of potential grips on any part of the object, regardless of the direction of motion, and reach trajectories that are constantly replanned to grasp while avoiding self-movement. They also utilized a learning-based grasping planner that could be generalized from a small number of samples to new objects with unseen shapes.

In [22], a search-based kinodynamic motion planning algorithm is presented. It can carefully select the object at the earliest possible point in its trajectory and takes into account time to create trajectories for both the arm and the end effector. The proposed approach tried to produce solutions against the high dimensionality of the time parameter kinodynamic planning problem by using informative heuristics and adaptive dynamic motion primitives. In [23], Marturi et al. created a set of offline trajectories for tracking and grasping selected objects and then, random moving ones. They made iterative inverse kinematics calculations with exposure tracking. Considering the collisions and kinematic constraints, the coupling trajectory with the smallest mission field error was selected, and achieved successfully.

In [24], a singularity-robust dual quad-based visual servo integrated with an efficient grasping algorithm is presented to grasp randomly moving objects in the robot's task area. The proposed system simultaneously analyzed the quality of multiple clutch configurations and their distance from the gripper in real-time. The object path was planed inside joint limits.

In [25], an aerial manipulator system is presented for grasping moving objects. They proposed a control strategy based on separate control of the unmanned aerial vehicle (UAV) and a manipulator. Experimental trials showed that moving objects can be grasped with an aircraft. In another UAV gripping study [26], a new passive manipulator was integrated on an autonomous drone to capture a moving target. In their work, they aimed to separate the moving target from a flying UAV. In [15], the authors provided a deep reinforcement learning-based control for a manipulator by recognizing randomly moving objects with YOLO. By keeping the relative position between the gripper and the moving object position stable, they adjusted the reward function. Thus performing the approach-tracking-grasping integrated operation for the moving objects.

III. Methods

To grasp the moving objects, firstly, YOLO detected objects moving on the conveyor. If the object was vertical, its height was calculated by image processing. When the object was horizontal, its orientation on the conveyor was found by PCA. Template matching was used to calculate the speed of the objects. Finally, the robot arm was positioned for gripping. Details of the methods are given in this section.

A. Object Recognition with YOLO

The process needs to be executed quickly for the smooth execution of real-time applications. Since deep network requires high processing power, deep learning models run slowly. For this reason, it is significant to fast and correct classification in object detection tasks. YOLO can directly give classification results and location coordinates of class members, which is very suitable for real-time object tracking applications [27]. YOLO also offers a trade-off between speed and accuracy.

Depending on the object's orientation and camera's position, object appears differently in 2D cameras. For example, a standing upright glass looks like a circle on top images but looks like a glass when viewed from the side. Since object recognition is difficult under these conditions, three different YOLO were trained to detect horizontally and vertically objects. These three networks were trained according to various images and properties of objects. Imaging was carried out with 2D cameras with a resolution of 480 pixel x 360 pixel. Top images of horizontal objects were used as the training data for the first deep network. The second deep network was trained with side images of vertical objects. And the third deep network was trained to recognize vertical objects in images taken from the top.

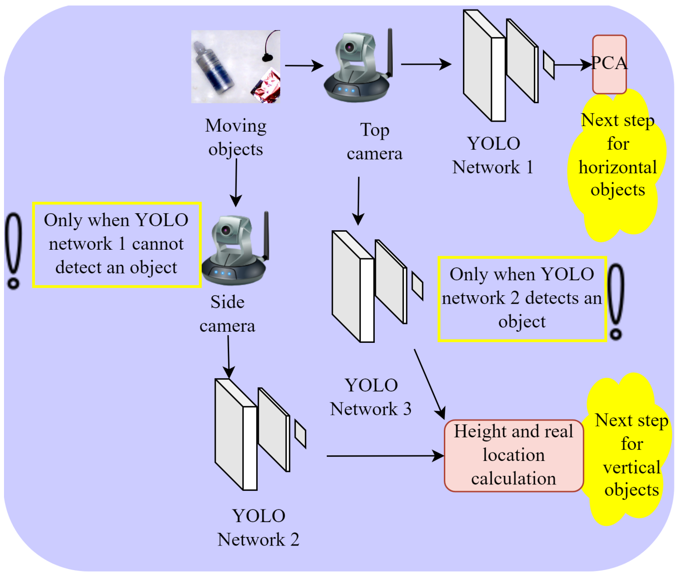

Moving objects
Top camera
YOLO Network 1
PCA
Next step for horizontal objects
Only when YOLO network 1 cannot detect an object
Side camera
Only when YOLO network 2 detects an object
YOLO Network 3
Height and real location calculation
Next step for vertical objects
YOLO Network 2

Fig. 2. Object detection system.

Fig. 2 visualizes the object detection system. Firstly, a frame is taken from the top, and it is evaluated whether there is a horizontal object with first YOLO. If it is the case, the necessary process for gripping is to start without running other networks. If the first deep network cannot detect a horizontal object, a frame is taken from the side, and it is evaluated whether there is a vertical one. If the vertical object is also not found, the system returns to the beginning, and this process is repeated until the object is detected. The third deep network is operated only when a vertical object is detected after taking a side view.

B. Calculation of the Height of Vertical Objects

Due to the height differences of objects, as in Fig. 3, tall objects travel less distance than shorter objects in the top images. The distance these objects must travel is the same as the number of pixels of the camera in that direction. In other words, the two lines in Fig. 3 are shown with the same pixel number in the image. For this reason, even if the objects move at the same speed, they appear to move at different speeds in the images because the distance they will travel in images is not equal. Therefore, we must know the height of the object correctly to overcome this problem. We offered to calculate the object height by processing the images taken from the side. However, the height can be seen differently depending on the distance from the camera. For this reason, it was necessary to find the space of the object from the camera.

To find the height of vertical objects, first, the object is searched with the deep network in top images. The approximate distance from the point where the object is detected to the edge of the conveyor belt is called $DisApp_{(p)}$ (where $_{(p)}$ denotes the height in pixels) as shown in Fig. 4(a). It can be also provided an approximation of the height of the objects

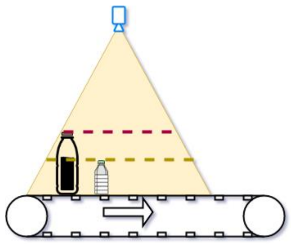
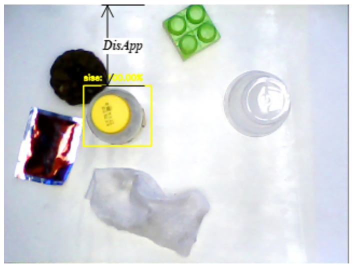
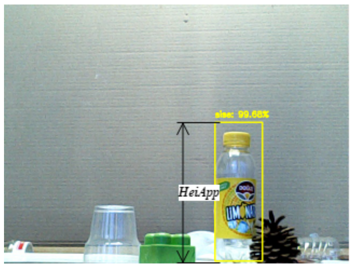


Fig. 3. From side view, the tip of higher objects travel shorter distances (red line) than shorter objects in longer distances (yellow line)



(a) *DisApp*$_{(p)}$



(b) *HeiApp*$_{(p)}$

Fig. 4. Position and height of the object.

with the second deep network. The height of the object in pixels *HeiApp*$_{(p)}$ is shown in Fig. 4(b). *HeiApp*$_{(p)}$ is in the side view as the upper limit of the box drawn by the second YOLO and is calculated as in Equation (1);

$$HeiApp_{(p)} = 480 - BoxTopY_{(p)} \tag{1}$$

where 480 and *BoxTopY*$_{(p)}$ are the number of camera pixels and the upper bound of the box drawn by YOLO in pixels, respectively. To calculate the approximate height *HeiApp*$_{(cm)}$, the Equation (2) was used according to the Fig. 5:

$$HeiApp_{(cm)} = \frac{HeiApp_{(p)}.(65 + 30.(DisApp_{(p)}/480))}{480.65} \tag{2}$$

To calculate the distance *CamDisY* seen by the camera at the approximate height of the object in Fig. 6, the similarity of triangles for the top camera is used as in Equation (3):

$$CamDisY_{(cm)} = \frac{30.(72 - HeiApp_{(cm)})}{72} \tag{3}$$

where 30, 72 and, *HeiApp* are the length seen by the camera on the conveyor belt and the height of the camera from the conveyor belt, respectively.

Since the distance from the camera to the object tip is known, the actual distance of the object to the edge of the

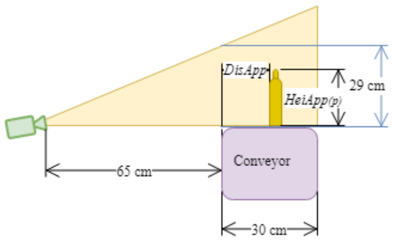


Fig. 5. Similarity of triangles used to find the approximate height of objects.

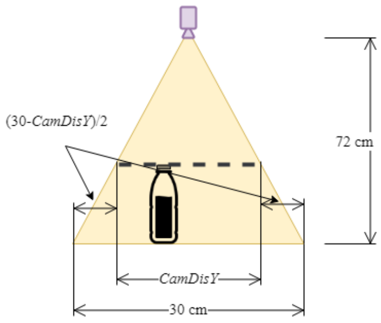


Fig. 6. The distance (black line) that the top camera sees at the approximate height of the object (*CamDisY*)

conveyor belt, *RealDis*$_{(cm)}$, can be calculated as in Equation (4):

$$RealDis_{(cm)} = \begin{cases} L + \dfrac{30 - CamDisY}{2}, \\ \text{if } DisApp \leq 240 \\ 30 - L - \dfrac{30 - CamDisY}{2}, \\ Otherwise \end{cases} \tag{4}$$

where *L* = *CamDisY.DisApp*/480. Using real distance *RealDis*, the real height of the objects *RealHei*$_{(cm)}$ can be calculated as in Equation (5):

$$RealHei_{(cm)} = \frac{HeiApp_{(p)}.(29.(65 + RealDis_{(cm)}))}{480.65} \tag{5}$$

C. Direction Calculation with Principal Component Analysis

PCA is a very effective method for revealing necessary information from data. PCA finds the projection of data in multidimensional space to a lower-dimensional space [28]. By calculating PCA, two perpendicular lines are drawn in a set of points in space. To create the first of these lines, one with the shortest average distance to all points is selected. Then, another most suitable line is drawn among the lines perpendicular to the first line drawn. These operations repeat until the variance of a new dimension falls below a certain threshold. Therefore, the obtained two lines formed the bases of a space.

Images consist of a set of points called pixels. Each point has an intensity value. Anything can be distinguished in an image according to the differences in the intensity values. By applying PCA to these point clusters, two lines are drawn in the direction of the object. In addition, the center of the object can be considered the intersection point of these lines.

First, a frame is taken when YOLO detected the object. Since a red green blue (RGB) image is a three-dimensional

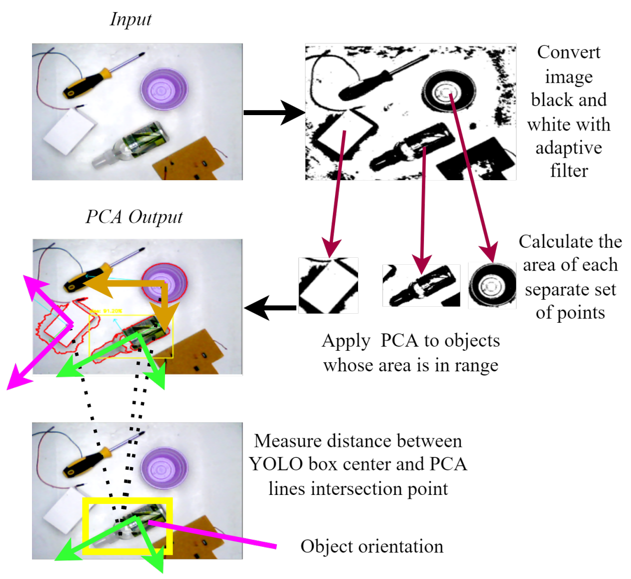

Input
Convert image black and white with adaptive filter
PCA Output
Calculate the area of each separate set of points
Apply PCA to objects whose area is in range
Measure distance between YOLO box center and PCA lines intersection point
Object orientation

Fig. 7. Object orientation calculation steps.

matrix, this image is converted to a two-dimensional gray-level image. A black-white image is obtained by applying an adaptive filter to the gray-level image. This image generates the point sets required for PCA. As seen in Fig. 7, the directions of all objects occupying a specific range are shown. The best direction line is the closest one to the box (yellow box) drawn by YOLO. Thus, information about the other objects is not processed.

D. Template Matching and Speed Calculation

After calculating the height and position of the object, the next step was to calculate the velocity. Since the objects are in motion, there is a limited time for grasping. Therefore, accurate calculation of speed is a critical step. If the robot arm cannot grip it on time, it may not have another chance to try again.

Template matching is the process of finding and positioning a template within an image [29]. A template is selected from one image and scanned on the other image. Thus, it tries to find the position of the template. The similarity score is calculated by sliding the template on the image. If the similarity score is more than a threshold value, the template is considered to be found.

When the deep network detects an object for the first time, it means the object is in the viewport. A template is taken from the center point of the object. The intersection point of the PCA lines is accepted as the center point for horizontal objects. The YOLO box center point is the center point for vertical objects. Because these points are the most unique point candidates in the images. Then, the camera takes three more images at certain time intervals. Template matching operated on these images with the generated template. By comparing the template matching results, the position difference of the object was calculated. The motion of the object was obtained in pixels. Thus, three speeds were obtained.

E. Robot Arm, Control and Object Grip

The Tinkerkit Braccio robot arm in Fig. 8 performed the gripping task. The robot arm, whose kinematic model is shown in Fig. 8, has six joints. The working area of the robot arm is a circle with a radius of 40 cm. The gripper width and maximum openings are 9 cm and 8 cm, respectively. The shoulder joint, which connects the robot with the ground, provides orientation according to the objects. A motor rotates the gripper, and a motor opens and closes the gripper. It is possible to control the remaining 3- wrist part as a 3 dof robot arm. The control of the 3 dof planar robot wrist was made with ANN.

The robot arm grasps at a certain distance to constrain the infinite point in 3D space. This distance (*GripPointX*) is 15 cm for horizontal objects and 36 cm for vertical objects. The reason for gripping at different distances for horizontal and vertical objects is that the grip is suitable for the operating range of the manipulator. The robot arm can grasp horizontal and vertical objects from above and from the front at these distances, respectively. The object location on the conveyor belt *GripPointY* is needed to calculate the angle of the shoulder point. *GripPointY* is the y coordinate (*CenterY*) of the center point found by PCA horizontal objects. For vertical objects, it is the *RealDis* value. The center of the robot arm is at the midpoint of the conveyor belt. Therefore, the angle of the shoulder motor θ_1 is calculated using *GripPointY* and *GripPointX* as in Equation (6):

$$\theta_1 = \begin{cases} \arctan \dfrac{15 - GripPointY}{GripPointX}, \\ \text{if } GripPointY \leq 15 \\ -\arctan \dfrac{15 - GripPointY}{GripPointX}, \\ Otherwise \end{cases} \tag{6}$$

For the planar robot wrist of the robot arm, an ANN was trained and controlled without kinematic calculations as shown in Fig. 9. ANN had 40 nodes in the first hidden layer and 20 nodes in the second hidden layer. Activation functions were ReLu in the hidden layers and Linear in the output layer. Training was carried out according to accuracy and Adam was chosen as the optimizer. The required distance (*GripX*), height (*GripY*), and sum of angle (θ_{sum}) of the gripper are entered into the input of the ANN. At first, 3000 data separately for vertical and horizontal objects were produced for training and testing as in Equation (7).

$$\begin{aligned} GripX &= l_1 cos(\theta_2) + l_2 cos(\theta_2 + \theta_3) + l_3 cos(\theta_2 + \theta_3 + \theta_4) \\ GripY &= l_1 sin(\theta_2) + l_2 sin(\theta_2 + \theta_3) + l_3 sin(\theta_2 + \theta_3 + \theta_4) \end{aligned} \tag{7}$$

Where l_1 = 12.5 cm (Link 2), l_2 = 12.5 cm (Link 3) and l_3 = 15 cm (Link4 + Gripper) as shown in Fig. 8. Since these objects are grasped at different positions and heights, two ANNs are trained with the same characteristics according to their grasping positions. The training and test data of ANN are *GripX* ∈ [12 cm, 15 cm], *GripY* ∈ [-11 cm, -9 cm] for horizontal objects, *GripX* ∈ [35 cm, 40 cm], *GripY* ∈ [-5 cm, 0 cm] for vertical objects. Training success for vertical and horizontal objects was 91.63% and 99.99%, respectively.

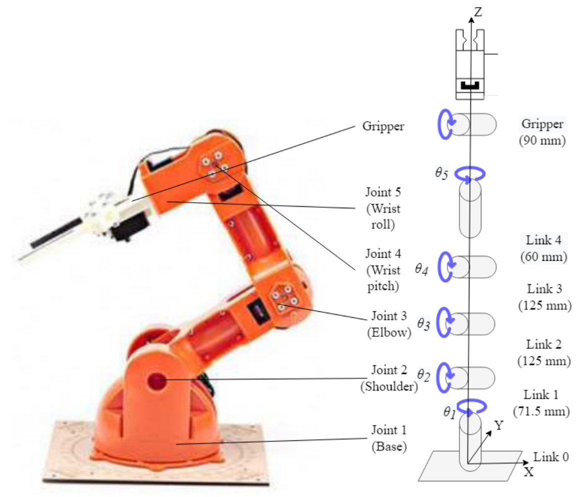


Fig. 8. Tinkerkit Braccio Robot arm and its kinematic model.

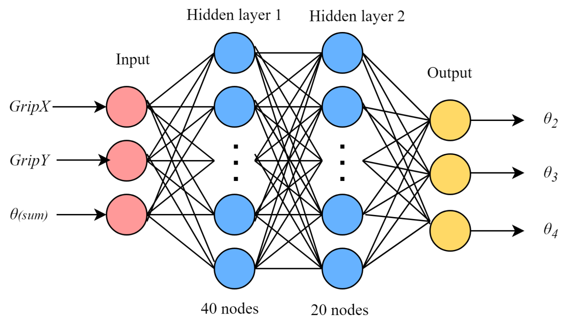


Fig. 9. ANN structure.

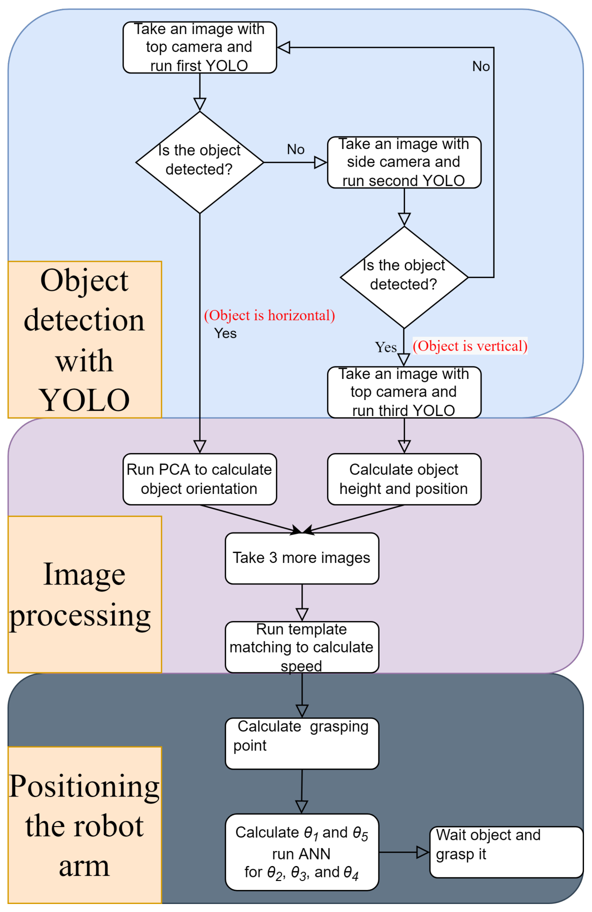

Take an image with top camera and run first YOLO
Is the object detected?
No
Take an image with side camera and run second YOLO
Is the object detected?
No
(Object is horizontal)
Yes
Yes
(Object is vertical)
Take an image with top camera and run third YOLO
Object detection with YOLO
Run PCA to calculate object orientation
Calculate object height and position
Take 3 more images
Image processing
Run template matching to calculate speed
Calculate grasping point
Positioning the robo arm
Calculate θ_1 and θ_5 run ANN for θ_2, θ_3, and θ_4
Wait object and grasp it

Fig. 10. System flowchart.

The data of the ANN created for vertical objects covers an area of 5 cm x 5 cm, while it covers an area of 2 cm x 3 cm for horizontal objects. Due to this coverage difference, the training success was lower as the datasets created for vertical objects differed more than for horizontal objects. While *GripY* is 1.5 cm above the conveyor belt for horizontal objects, it is set to half of *RealDis* for vertical objects. *GripX* is calculated according to *GripPointY* and *GripPointX* as in Equation (8);

$$GripX = \begin{cases} \sqrt{GripPointX^2 + (15 - GripPointY)^2}, \\ \text{if } GripPointY \leq 15 \\ \sqrt{GripPointX^2 + (GripPointY - 15)^2}, \\ Otherwise \end{cases} \tag{8}$$

In the calculation of the angle of the gripper, no extra processing was performed for vertical objects because the grip was made in the direction of movement. However, for horizontal objects, the angle of the object on the conveyor belt is calculated as in Equation (9) according to α and θ_1.

$$\theta_5 = \begin{cases} \alpha - (-\theta_1), & \text{if } \alpha - (-\theta_1) \leq 90 \\ \alpha - (-\theta_1) - 180, & Otherwise \end{cases} \tag{9}$$

Finally, the angles and grip time are sent to the robot arm control card via serial.

IV. Experiments

The operation of the system is given in Fig. 10. Making experiments, two different object classes were tried to recognize and grasp. One class of objects was 7 juice boxes of fixed sizes but different colors. The other class consisted of 7 plastic bottles in different colors and sizes. The maximum and minimum widths (diameter) of the pet bottles were 6 and 4.5 cm, respectively. The short and long sides of the rectangular juice boxes were approximately 3.8 and 4.8 cm, respectively, and a diagonal of about 6 cm. Table I presents the height and width of the objects. During the trials, the speed of the conveyor belt was kept constant at three different speeds (these speeds were unknown). For each vertical object, 25 griping attempts were made at each speed. A total of 350 trials were conducted at all three speeds. A total of 1050 griping attempts were made, with an equal number of vertical juice boxes and bottles.

On the other hand, thirty trials were made for each horizontal bottle at each speed of the conveyor. A total of 105 trials were conducted with juice boxes at all speeds. Thus, a total of 630 and 315 trials were conducted with bottles and juice boxes, respectively.

The gripper could open its fingers up to 9 cm. For this reason, there was a maximum tolerance of 2.5 cm and a minimum tolerance of 1.5 cm for *RealDis*. Since the finger length of the gripper was 8 cm, there was a tolerance of 8 cm

TABLE I
HEIGHT AND WIDTH OF OBJECTS.

Object	Height(cm)	Width(cm)
Bottle 1	16.5	5
Bottle 2	15.5	4.5
Bottle 3	18	5
Bottle 4	17	5
Bottle 5	19.5	6
Bottle 6	20	5.5
Bottle 7	13.5	4.5
Juice Boxes	12.5	3.8 and 4.8

in the direction of movement. In other words, grasping objects should be in an area of 8 cm x 9 cm.

The computer characteristics were a 8 GB of RAM and an Intel(R) Core(TM) i5-3610ME CPU @2.70 GHz. The system started to take images on average 0.4368 seconds after it had started working for the first time. If the relevant object was in the image, it took an average of 1.9856 seconds from the first image acquisition until the angles were sent to the robot arm. Computing in terms of memory, the usage was 29.5% at the beginning, and then became up to 31.3% at the end of the process. Whereas the CPU usage was initially [0, 1.5, 0, 1.5] for each CPU, then it became [20.3, 0, 0, 7.8] during the first image acquisition. CPU usage at the end of the process was recorded as [18.7, 10.8, 28.1, 0].

V. RESULTS

A. Vertical Object Grip Results

The calculated average height, speeds, and successful grip rates of the objects in the trials from the lowest speed to the highest speed are presented in Table II.

Of the 350 trials at speed 1 (the lowest speed) 332 were successful. Eighteen attempts failed due to eleven high seven low-speed calculations. When the speed was calculated as high, the gripper failed because the gripper performed gripping before the object reached the gripper. On the other hand, when the speed was calculated as lower than the actual speed, the object touches the end of the gripper. The object that could not go further due to the gripper also fell.

Of the 350 attempts at speed 2, only 14 failed. Eight and six of the failed attempts were due to speed was calculated as high and low, respectively.

The most unsuccessful speed in three different speed trials was the highest speed (speed 3) with 19 unsuccessful attempts. Twelve attempts failed due to high speed, and seven attempts failed due to low speed. The robot arm removed the objects in 998 of the 1050 trials. As a result, the grasping success rate was 95.05% for vertical objects. The biggest reason was that the height of bottles 3 and 7 were shorter difficulty its detection since the bottle lids were transparent. Since the camera's side view is not perpendicular but angled, tall objects appear to be taller. This incorrect display caused the speed of the objects to be calculated differently.

Velocities were calculated on average as 4.28, 5.1, and 6.02 cm/sec for speed 1, speed 2 and speed 3, respectively. In speed calculations, it is seen that as the height of the object increases, its velocity was calculated to be lower. However, these differences are 3.51, 3.32, and 5.2 pixels/sec for speed 1, speed 2 and speed 3, respectively. Considering that 16 pixels is 1 cm, the speed difference was calculated as 0.325 cm/sec at most. This value is also suitable for the tolerance range of the gripper for the conveyor belt length used.

The lowest success in vertical objects was 80% in speed 3 for bottle 7. In general, the lowest number of grasping was again in bottle 7. The biggest reason for this might be that the speed was calculated lower than that of other objects. At the same time, since bottle 7 was the shortest, it had the least area for gripping. Even though this bottle was longer than the juice box, the area where it can be gripped was narrower due to the curvature in the top area.

Grasping objects with different success rates at each speed showed that grasping was performed independently of the object. As a result of the trials, no variable other than speed's effect was observed in the failed trials. Variables such as object sizes, object class, and location were not very effective in grasping failure.

B. Horizontal Object Grip Results

A total of 630 and 315 attempts were made with bottles and juice boxes, and the numbers of successful grasps were 583 and 281, respectively. In Table III, number and rate of successful grip of bottles and fruit juice boxes by speed for horizontal object is given.

The height of the bottles were important, but since even the shortest bottle used was almost equal to the size of the juice boxes, it was possible to grasp the short bottles. However, in short bottles, it is more difficult to grasp since the distance between the cap area and the center point is shorter. In addition, the fact that the cap areas of the bottles have different characteristics prevents working according to the bottles of only certain criteria. Since the aim of the study was to grasp objects with different properties, the grip success rates of bottles with such criteria were not presented separately.

These results showed that the effect of speed was very small and that the grip failures were caused by the errors given in Table IV.

For bottles on horizontal objects, speed 1 led to the most unsuccessful grip. For juice boxes, success was closed to each other at every speed. Our success in grasping horizontal objects was 93% and 90% for bottles and juice boxes, respectively. The overall success was 91.53%. The main reason for failure due to the miscalculation of the speed was seen as timing.

Although the success in detecting target objects varies according to YOLO's training success, it is also possible to work with similar objects that have never been seen. Thus, gripping was realized without the need for costant object properties. It has been seen that it was possible to calculate some angles required for positioning the manipulator with ANN. Thus, there was no need to search for a solution among many possible solutions. Unlike 3D camera images, which require high processing power, the necessary properties of objects have been obtained with a 2D camera. Thus, it is ensured that all these operations were realized quickly, which was

TABLE II
CALCULATED HEIGHT DIFFERENCES, SPEEDS, AND SUCCESSFUL GRASPING RATES FOR VERTICAL OBJECT.

	Speed 1			Speed 2			Speed 3		
Objects	Height difference (cm)	Speed (pixels/sec)	Success Rate	Height difference (cm)	Speed (pixels/sec)	Success Rate	Height difference (cm)	Speed (pixels/sec)	Success Rate
Bottle 1	+0.73	67.11	0.84	+0.44	80.62	1	+0.54	94.42	0.96
Bottle 2	+0.26	67.06	1	+0.30	81.21	0.88	+0.11	95.24	0.96
Bottle 3	-1.83	67.85	0.92	-1.81	80.68	0.96	-1.95	96.43	0.88
Bottle 4	+0.62	67.48	0.88	+0.43	81.13	0.92	+0.45	96.11	0.88
Bottle 5	+1.52	67.14	1	+1.41	80.64	0.96	+1.54	94.48	0.96
Bottle 6	+2.46	66.42	1	+2.44	80.09	0.92	+2.48	93.63	0.96
Bottle 7	-0.05	66.65	0.92	-0.07	79.43	0.92	-0.21	92.72	0.80
Juice Boxes	-0.23	69.93	0.96	-0.20	82.75	0.98	-0.31	97.92	0.98
Mean		68.51	0.95		81.65	0.96		96.32	0.95

TABLE III
NUMBER AND RATE OF SUCCESSFUL GRIP OF BOTTLES AND FRUIT JUICE BOXES BY SPEED FOR HORIZONTAL OBJECT.

	Bottle		Fruit juice boxes	
Speeds	Number of successful grips	Rate	Number of successful grips	Rate
Speed 1	185	0.88	96	0.91
Speed 2	198	0.94	93	0.89
Speed 3	200	0.95	93	0.89
Total	583	0.93	282	0.90

TABLE IV
REASONS FOR FAILED GRASPING.

Reason	Number of attempts	Explanation
Height incorrectly calculated	10	The object could not be grasped because the ANN calculated the height incorrectly.
Timing	65	The time when the object was grasped was calculated incorrectly.
Gripper made contact with the bottle.	9	The position of the bottle changed from the grip as the gripper touched the moving bottle.
The location was calculated incorrectly.	6	Grip did not occur because the gripping location was calculated incorrectly.

significant for real-time applications. As a result, horizontal or vertical objects with different speeds and properties could be grasped with the robot arm.

VI. CONCLUSION

In this study, an application of a robot arm to grip moving objects was carried out. Deep network-based YOLO recognized and detected objects. Image processing methods were used to calculate the speed and height of objects viewed with 2D cameras. The gripping was performed according to the unknown speed and position of the objects. The proposed algorithm for speed calculation, which is one of the most significant parameters in the grasping of moving objects, was quite effective. The algorithm could not calculate with sufficient precision for gripping up to 4.95% of the trials made without the speed information given to the system. In general, a successful grasping rate was achieved at different speeds.

When we looked at the errors we encounter in the system, we found that there are four different error states. The main reason for the problem that the gripper cannot hold the object (especially cylindrical objects such as bottles), as a result of the wrong determination of the landing distance was because of the margin of error in the ANN. To overcome this, we suggest improvements to the training of the ANN. By giving the ANN an extra object class input, the class information from YOLO, can be added to the training. When we look at the errors encountered in timing, we see that there is a miscalculation of the speed. One of the main reasons for this is miscalculation of boundary lines in pattern matching. We recommend increasing the camera resolution to fix this error. We see that the main reason for the problem of the gripper touching the object during the holding process is the low sensitivity of the robotic manipulator. In addition, the wrong calculation of the speed and the processing time delays affect the formation of this error. We can say that the positioning error of the holder is due to template matching. Incorrect results in template matching are dependent on camera and lighting. Therefore, it is of great importance to install a high-resolution camera and the right lighting setup (perhaps a closed image capture unit can solve this problem) in applications for object grasping with the robot arm.

ACKNOWLEDGMENT

This research is supported by Selçuk University Coordinatorship of Faculty Member Training Program (Project number: 2019- ÖYP- 008).

REFERENCES

[1] E. J. Van Henten, J. Hemming, B. Van Tuijl, J. Kornet, J. Meuleman, J. Bontsema, and E. Van Os, “An autonomous robot for harvesting cucumbers in greenhouses,” *Autonomous robots*, vol. 13, no. 3, pp. 241–258, 2002.

[2] C. Zhihong, Z. Hebin, W. Yanbo, L. Binyan, and L. Yu, “A vision-based robotic grasping system using deep learning for garbage sorting,” in *2017 36th Chinese control conference (CCC)*, pp. 11223–11226, IEEE, 2017.

[3] Y. Liu, Y. Zhong, X. Chen, P. Wang, H. Lu, J. Xiao, and H. Zhang, “The design of a fully autonomous robot system for urban search and rescue,” in *2016 IEEE International Conference on Information and Automation (ICIA)*, pp. 1206–1211, IEEE, 2016.

[4] B. Wang, L. Jiang, J. Li, H. Cai, and H. Liu, “Grasping unknown objects based on 3d model reconstruction,” in *Proceedings, 2005 IEEE/ASME International Conference on Advanced Intelligent Mechatronics.*, pp. 461–466, IEEE, 2005.

[5] A. Saxena, J. Driemeyer, and A. Y. Ng, “Robotic grasping of novel objects using vision,” *The International Journal of Robotics Research*, vol. 27, no. 2, pp. 157–173, 2008.

[6] M. J. Shafiee, B. Chywl, F. Li, and A. Wong, “Fast yolo: A fast you only look once system for real-time embedded object detection in video,” *arXiv preprint arXiv:1709.05943*, 2017.

[7] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, “Ssd: Single shot multibox detector,” in *European conference on computer vision*, pp. 21–37, Springer, 2016.

[8] S. Ren, K. He, R. Girshick, and J. Sun, “Faster r-cnn: Towards real-time object detection with region proposal networks,” *Advances in neural information processing systems*, vol. 28, 2015.

[9] I. V. Zoev, A. P. Beresnev, and N. G. Markov, “Convolutional neural networks of the yolo class in computer vision systems for mobile robotic complexes,” in *2019 International Siberian Conference on Control and Communications (SIBCON)*, pp. 1–5, IEEE, 2019.

[10] A. Ćorović, V. Ilić, S. Ðurić, M. Marijan, and B. Pavković, “The real-time detection of traffic participants using yolo algorithm,” in *2018 26th Telecommunications Forum (TELFOR)*, pp. 1–4, IEEE, 2018.

[11] J.-W. Chang, R.-J. Wang, W.-J. Wang, and C.-H. Huang, “Implementation of an object-grasping robot arm using stereo vision measurement and fuzzy control,” *International Journal of Fuzzy Systems*, vol. 17, no. 2, pp. 193–205, 2015.

[12] K.-T. Song and S.-C. Tsai, “Vision-based adaptive grasping of a humanoid robot arm,” in *2012 IEEE International Conference on Automation and Logistics*, pp. 155–160, IEEE, 2012.

[13] A. Bicchi and V. Kumar, “Robotic grasping and contact: A review,” in *Proceedings 2000 ICRA. Millennium conference. IEEE international conference on robotics and automation. Symposia proceedings (Cat. No. 00CH37065)*, vol. 1, pp. 348–353, IEEE, 2000.

[14] O.-L. Ouabi, P. Pomarede, N. Declercq, N. Zeghidour, M. Geist, and C. Pradalier, “Learning the propagation properties of plate-like structures for lamb wave-based mapping,” 2021.

[15] P. Chen and W. Lu, “Deep reinforcement learning based moving object grasping,” *Information Sciences*, vol. 565, pp. 62–76, 2021.

[16] X. Feng and P. Milanfar, “Multiscale principal components analysis for image local orientation estimation,” in *Conference Record of the Thirty-Sixth Asilomar Conference on Signals, Systems and Computers, 2002.*, vol. 1, pp. 478–482 vol.1, 2002.

[17] P. K. Allen, A. Timcenko, B. Yoshimi, and P. Michelman, “Automated tracking and grasping of a moving object with a robotic hand-eye system,” *IEEE Transactions on Robotics and Automation*, vol. 9, no. 2, pp. 152–165, 1993.

[18] Y. Zhang, L. Li, M. Ripperger, J. Nicho, M. Veeraraghavan, and A. Fumagalli, “Gilbreth: A conveyor-belt based pick-and-sort industrial robotics application,” in *2018 Second IEEE International Conference on Robotic Computing (IRC)*, pp. 17–24, IEEE, 2018.

[19] D. K. Reddy *et al.*, “Sorting of objects based on colour by pick and place robotic arm and with conveyor belt arrangement,” *Int. J. Mech. Eng. & Rob. Res*, vol. 3, no. 1, p. 3, 2014.

[20] W. T. Abbood, O. I. Abdullah, and E. A. Khalid, “A real-time automated sorting of robotic vision system based on the interactive design approach,” *International Journal on Interactive Design and Manufacturing (IJIDeM)*, vol. 14, no. 1, pp. 201–209, 2020.

[21] S. Kim, A. Shukla, and A. Billard, “Catching objects in flight,” *IEEE Transactions on Robotics*, vol. 30, no. 5, pp. 1049–1065, 2014.

[22] A. Menon, B. Cohen, and M. Likhachev, “Motion planning for smooth pickup of moving objects,” in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 453–460, IEEE, 2014.

[23] N. Marturi, M. Kopicki, A. Rastegarpanah, V. Rajasekaran, M. Adjigble, R. Stolkin, A. Leonardis, and Y. Bekiroglu, “Dynamic grasp and trajectory planning for moving objects,” *Autonomous Robots*, vol. 43, no. 5, pp. 1241–1256, 2019.

[24] C. De Farias, M. Adjigble, B. Tamadazte, R. Stolkin, and N. Marturi, “Dual quaternion-based visual servoing for grasping moving objects,” in *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*, pp. 151–158, IEEE, 2021.

[25] G. Zhang, Y. He, B. Dai, F. Gu, L. Yang, J. Han, G. Liu, and J. Qi, “Grasp a moving target from the air: System & control of an aerial manipulator,” in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1681–1687, IEEE, 2018.

[26] B. Vidyadhara, L. A. Tony, M. S. Gadde, S. Jana, V. Varun, A. A. Bhise, S. Sundaram, and D. Ghose, “Design and integration of a drone based passive manipulator for capturing flying targets,” *Robotica*, vol. 40, no. 7, pp. 2349–2364, 2022.

[27] D. Wu, S. Lv, M. Jiang, and H. Song, “Using channel pruning-based yolo v4 deep learning algorithm for the real-time and accurate detection of apple flowers in natural environments,” *Computers and Electronics in Agriculture*, vol. 178, p. 105742, 2020.

[28] A. Daffertshofer, C. J. Lamoth, O. G. Meijer, and P. J. Beek, “Pca in studying coordination and variability: a tutorial,” *Clinical biomechanics*, vol. 19, no. 4, pp. 415–428, 2004.

[29] M. A. Hossain and S. Afrin, “Optical character recognition based on template matching,” *Global Journal of Computer Science and Technology*, 2019.

Kürşad UÇAR received the Master degree from Department of Electric –Electronic Engineering, Institute of Science, Selçuk University, Konya, Turkey (2018). He continues his Ph. at same university. He is a research assistant at Selçuk University since 2016. His research interests are in image processing, robotic, automation and control, and machine learning.

Hasan Erdinç KOÇER received the Ph.D. degree from Department of Electric–Electronic Engineering, Institute of Science, Selçuk University, Konya, Turkey (2007), where he is an associate professor since 2018. His research interests are in machine vision and pattern recognition, industrial automation on camera, medical image processing and, automation and control.