Proceedings of the 2022 International Symposium on Flexible Automation
ISFA2022
July 3 - 7, 2022, Yokohama, Japan

ISFA2022-025

MACHINE LEARNING-BASED ROBOTIC OBJECT DETECTION AND GRASPING FOR COLLABORATIVE ASSEMBLY

Jianjing Zhang, Chuanping Liu, Joshua Huang, and Robert X. Gao[1]
Department of Mechanical and Aerospace Engineering
Case Western Reserve University
Cleveland, OHIO, USA

ABSTRACT

An integral part of information-centric smart manufacturing is the adaptation of industrial robots to complement human workers in a collaborative manner. While advancement in sensing has enabled real-time monitoring of workspace, understanding the semantic information in the workspace, such as parts and tools, remains a challenge for seamless robot integration. The resulting lack of adaptivity to perform in a dynamic workspace have limited robots to tasks with pre-defined actions. In this paper, a machine learning-based robotic object detection and grasping method is developed to improve the adaptivity of robots. Specifically, object detection based on the concept of single-shot detection (SSD) and convolutional neural network (CNN) is investigated to recognize and localize objects in the workspace. Subsequently, the extracted information from object detection, such as the type, position, and orientation of the object, is fed into a multi-layer perceptron (MLP) to generate the desired joint angles of robotic arm for proper object grasping and handover to the human worker. Network training is guided by forward kinematics of the robotic arm in a self-supervised manner to mitigate issues such as singularity in computation. The effectiveness of the developed method is validated on an eDo robotic arm in a human-robot collaborative assembly case study.

Keywords: Human Robot Collaboration, Deep Learning, Object detection, Self-supervised Learning.

1. INTRODUCTION

A critical building block of smart manufacturing is the incorporation of industrial robots to assist human workers in assembly, where the workers and robots share a workspace and *collaboratively* performs tasks for which direct contact is allowed [1]. Compared to the traditional assembly workspace where the workers and robots are strictly separated for safety reasons and the tasks of each are carried out *sequentially*, human-robot collaboration (HRC) will improve both flexibility and efficiency in assembly [2].

A prerequisite for successful HRC is to allow robot to monitor the workspace, interpret collaboration context, and act accordingly. For this purpose, recognition and prediction of human action have been investigated [2-5]. They provide the basis for the robot to understand the part or tool that is needed to accomplish the subsequent operation. However, research on recognizing, localizing, and grasping the intended part/tool in an appropriate manner is still limited. For example, placement of parts/tools is strictly pre-defined in prior HRC studies [3-5], and dedicated research on robotic object grasping is primarily focused on grasping outcome (success or fail) rather than differentiating the type of objects [6-8]. Considering that the position and orientation of parts/tools cannot be assumed to remain time-invariant, and collaborative operations such as object handover may require specific grasping orientation, research is needed to investigate techniques for: 1) object detection, which recognizes and localizes parts/tools of interest based on sensing images, and 2) robot control, which subsequently determines the joint angles of robotic arm that are needed for the end-effector to arrive at desired position with desired orientation for grasping.

For object detection, two commonly investigated techniques are region-based convolutional neural network, or RCNN [9] (with its variants, Fast RCNN [10] and Faster RCNN [11]) and the method of You Only Look Once, or YOLO [12]. The RCNN-based methods take a two-step approach. First, the regions-of-interest (ROIs) of an image that potentially contain objects of interest is generated by a region-proposal procedure [9-11]. Then, each ROI passes through an object detection network, which predicts the object type within the ROI and the corresponding bounding box position and shape. By contrast, YOLO follows a single-step approach, which splits the image into grid cells and

[1] Corresponding author: Robert.Gao@case.edu

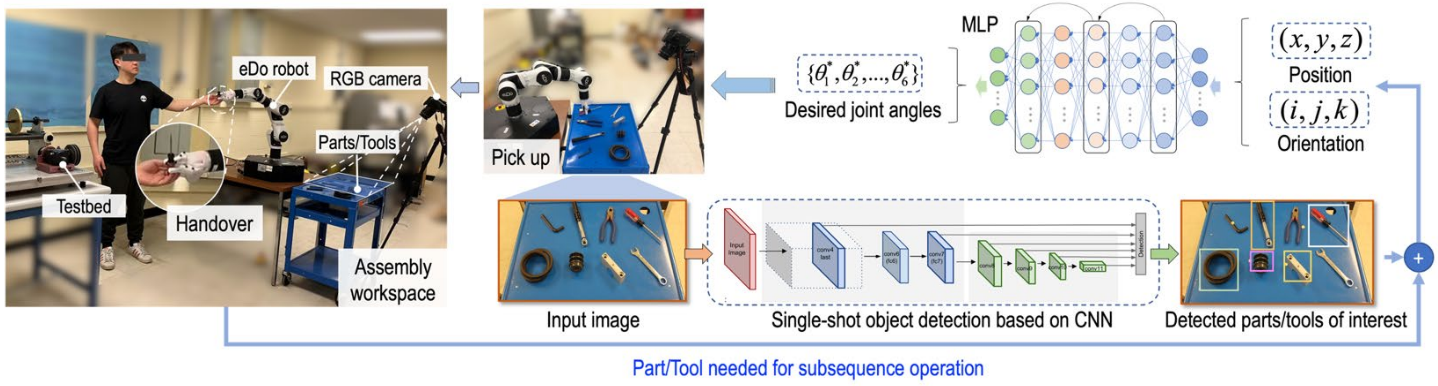
eDo robot
RGB camera
Parts/Tools
Testbed
Handover
Assembly workspace
Pick up
MLP
$\{\theta_1^*, \theta_2^*, ..., \theta_6^*\}$
Desired joint angles
(x, y, z)
Position
(i, j, k)
Orientation
Input image
Single-shot object detection based on CNN
Detected parts/tools of interest
Part/Tool needed for subsequence operation

FIGURE 1: Integrated method for part/tool detection and robot joint angle control in HRC

uses each cell to predict a fixed number of bounding boxes and the corresponding object types [12].

Despite the state-of-the-art benchmark performance by RCNN and YOLO, both have limitations in a manufacturing setting. For RCNN, training of a region-proposal network (RPN) is required before object detection can be carried out. However, collecting sufficient training images from the shop floor can be difficult without interrupting normal production schedule. While techniques such as transfer learning may alleviate the data availability issue, dedicated pretrained RPN is currently not available, which can make transfer learning less effective. For YOLO, it relies on a single grid, which makes it less effective in recognizing parts/tools of varying scales.

Considering that convolutional neural network (CNN) with feature maps of different sizes has been widely investigated for image recognition [13-15], which naturally provides multi-scale image analysis, this research investigates the concept of single-shot detection (SSD) [16] that is capable of direct utilization of the exiting feature maps in CNN for object detection. In addition, SSD only incurs a small number of additional network weights and therefore, can be effectively trained using limited training samples.

Once the spatial information of parts/tools are extracted by object detection, the desired position and orientation of the robotic arm end-effector can be obtained, based on which the joint angles that are needed for grasping will be computed. Such joint angles can be computed using inverse kinematics (IK). However, direct inverse solution involves nonlinearity, simplifying assumptions, and singularity [17-18]. Recently, deep learning has been increasingly investigated to bypass the analytical limitations and allow more tolerant and singularity-free robot control. For example, numerical mapping between end-effector spatial information and joint angles has been directly established using supervised learning [18-21]. In [17], generative adversarial network (GAN) has been investigated to solve the inverse kinematics with the generator predicting the joint angles and the discriminator determining whether the predicted angles are valid.

Despite the progress, these methods are generally not related to the physics (i.e., kinematics) that governs the motion of the robotic arm but solely dependent on the large number of labeled training samples. This study presents a novel method for solving inverse kinematics using self-supervised learning. Through self-supervised learning it is feasible to use forward kinematics (FK) to evaluate whether the predicted robot joint angles can guide the end-effector to the desired position and orientation. By replacing the prediction error of the joint angles (computed using labeled training samples) with the error of the end-effector position and orientation (computed using FK) to guide training, training data labeling (i.e., for joint angle) is no longer needed. In addition, the network's physical consistency is enhanced. In Fig. 1, a flowchart of the developed method for object detection and robot joint angle control is shown.

The rest of paper is organized as follows: Section 2 presents the theoretical background of object detection based on SSD and joint angle control using self-supervised learning. In Section 3, experimental setup for evaluating the developed method is described. The results are presented and discussed in Section 4, and conclusions and future work are summarized in Section 5.

2. THEORETICAL BACKGROUND

In the presented study, object detection and robot joint angle control are formulated as learning problems, by extending the capability of the CNN and multi-layer perceptron (MLP), respectively.

2.1 Single-shot object detection of parts and tools

The goal of object detection is to simultaneously: 1) identify the types of parts/tools of interest in sensing images, and 2) determine their positions and orientations by predicting bounding boxes to surround each of them. Different from traditional image recognition using CNN in which each image only contains a single candidate object [13], the task of object detection in this presented study requires the accommodation of varying number of candidate objects as well as changing positions and orientations. This requires enhancement of the standard CNN structure.

The concept of single-shot detector [16] allows to adapt the existing, standard CNN structure as the backbone for object detection, as shown in Fig. 2. The main idea is to: 1) utilize feature maps of size m x n in each layer of the CNN as a *grid* of the same size to cover the input image (in Fig. 2, m is chosen to be equal to n for illustration purpose), and 2) use each *grid cell* to predict a fixed number (i.e., k, in Fig. 2, k=3) of bounding boxes and the types of the objects surrounded by the boxes. In Fig. 2, the predicted bounding boxes from selected grid cells (with shading) are illustrated, and boxes that match the ground truth in the input image are color coordinated.

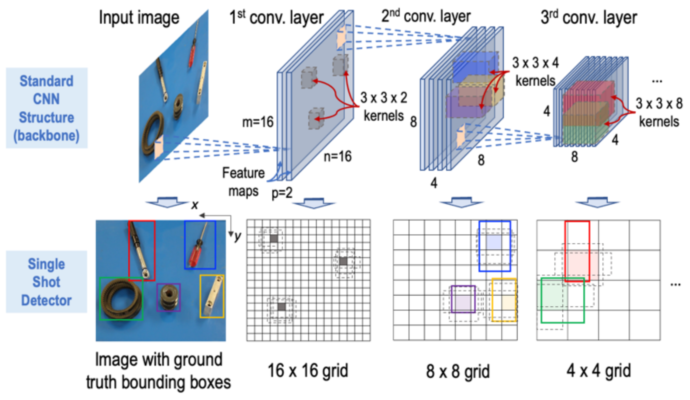

FIGURE 2: Object detection based on SSD

Specifically, for p feature maps of dimension m x n in CNN, each prediction from a grid cell consists of a vector $[\Delta cx, \Delta cy, \Delta w, \ \Delta h, c_1, c_2, \dots, c_N]$, where $\Delta cx, \Delta cy$ represent the relative distances between the center of the predicted bounding box and the center of the grid cell in the x and y coordinates, respectively. Δw, Δh represent changes in the width and height of the bounding box relative to the grid cell dimension, respectively. $c_1, c_2, \dots,$ c_N denotes N scores, each of which indicating how likely there would be an object surrounded by the bounding box that belongs to one of the N candidate object types (e.g., screwdriver, cap, belt, sheave, etc.), including background. $c_1, c_2, \dots,$ c_N then pass through a *softmax* function to obtain the predicted object type. Each element in the vector $[\Delta cx, \Delta cy, \Delta w, \Delta h, c_1, c_2, \dots, c_N]$ is predicted using a 3 x 3 convolutional kernel centered at the corresponding grid cell. Therefore, for feature maps of size m x n, the total number of bounding boxes to be predicted is $m \cdot n \cdot k$ and the total number of kernel weights to be trained is $9 \cdot (N+4) \cdot k$.

It should be noted that the kernels for SSD are different from the kernels in the standard CNN. The purpose of the kernels in the standard CNN is to generate feature maps, whereas the kernels for SSD utilize the generated feature maps for object detection. As the CNN structure goes deeper, the size of the feather maps becomes smaller and the grid changes from "fine" to "coarse", effectively allowing multi-scale image analysis.

The loss function for single-shot object detection training consists of two parts [16]:

$$L(c, \hat{c}, l, \hat{l}\,) = L_{class}(c, \hat{c}) + L_{box}(l, \hat{l}) \qquad (1)$$

In Eq. (1), $L_{class}(c, \hat{c})$ denotes the error term for classification of object types and $L_{box}(l, \hat{l})$ denotes the error term for prediction of position and shape of bounding boxes. During training, each predicted bounding box is first compared to the ground truth boxes to evaluate the degree of matching, which is quantified using the intersection over union (IoU) ratio (Fig. 3). The predicted boxes with an IoU ratio over 0.5 are categorized as "matched". Then, $L_{class}(c, \hat{c})$ is computed as the cross-entropy [17] between the predicted object type $\hat{c}$ and ground truth object type c for all boxes. $L_{box}(l, \hat{l})$ is computed as the squared difference in each of $\Delta cx, \Delta cy, \Delta w, \Delta h$ between the predicted values and ground truth for all *matched* boxes. By minimizing Eq. (1) during training, the capability of both object type recognition and bounding box prediction can be improved.

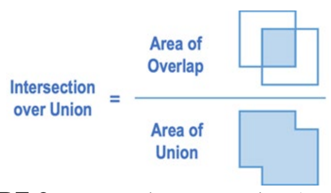

FIGURE 3: Intersection over union (IoU) ratio

To utilize the spatial information extracted from object detection for robot's action, projective transformation is needed to transform the camera coordinates into the robot coordinates. This transformation is shown in Fig. 4, known as the homography transformation [22]. The idea is to transfer a tilted image plane from the perspective view of camera to an orthographic view by using a 3-by-3 homography matrix H. In this study, the objects' z-coordinates are fixed since they are all placed on the same horizontal surface.

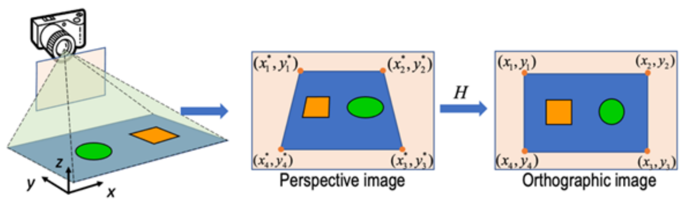

FIGURE 4: Homography transformation

Mathematically, the transformation can be written as [22]:

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix}_R = \lambda \underbrace{\overset{\text{Rotation/Stretching} \quad \text{Translation}}{\underbrace{\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix}}_{\text{Scaling}}}}_{\text{Homography matrix: } H} \begin{pmatrix} x^* \\ y^* \\ 1 \end{pmatrix}_C \qquad (2)$$

where $(x, y, 1)_R^{\mathrm{T}}$ and $(x^*, y^*, 1)_C^{\mathrm{T}}$ are homogeneous vectors of the same physical point in the robot coordinates (R) and camera coordinates (C), respectively. (a, b, d, e) are parameters for image rotation and stretching, (c, f) are parameters for translation, and (g, h) are parameters for scaling. λ is a normalization factor. The parameters can be calibrated using four non-colinear points [22].

2.2 Self-supervised learning for robot joint control

Once the desired end-effector position and orientation is extracted from the object detection step, they are used to generate joint angles of robotic arm such that the end-effector can arrive at the desired position with the desired orientation for part/tool grasping. However, general closed-form IK solution for robot arm to arrive at any spatial point with desired orientation is often infeasible due to geometrical and rotational constraints that are physically associated with the robotic arm [23]. The alternatives that have been reported include: 1) numerical solvers that approach the desirable position iteratively [24], and 2) machine learning methods that minimizes a loss function related to the accuracy required for robot movement [17-19]. Both can find the optimal solution and provide tolerance to the aforementioned issues.

The numerical solvers start with an initial selection of joint angles. At each iteration, FK is applied to obtain the corresponding end-effector position and orientation. Then, the error between the obtained and the desired position and orientation is used to update the joint angles in the next iteration through the inverse Jacobian matrix of the FK until convergence. However, a large number of iterations is often needed for convergence [24]. In addition, the inversion of Jacobian matrix is prone to singularity and the algorithm is affected by the initial selection of joint angles and is prone to local minimum [24].

In this study, machine learning approach is investigated for solving IK problems to avoid initial joint angle selection, iterative computation, and instability in inverting Jacobian matrix. With a loss function that is designed to minimize position and orientation error, machine learning is suited for determining optimal solutions for cases where the precise IK solution does not exist. To achieve this objective, a self-supervised machine learning approach is developed as illustrated in Fig. 5.

Specifically, a MLP is developed to predict the joint angles of robotic arm $\hat{\boldsymbol{\theta}}$ based on the given position and orientation of the end-effector at the MLP input: position (x, y, z) with the orientation (i, j, k) expressed as a unit vector. Training of the MLP is guided by the FK of the robotic arm, which computes the position and orientation of the end-effector that would be achieved by using the predicted joint angles from the MLP. Errors in the position and orientation between the predicted values from the FK and the given values at the MLP input constitute the loss function. At the end of the training process, errors in the position and orientation of the end-effector are minimized, indicating that the MLP is able to predict the robot's joint angles and control the end-effector to arrive at the desired position and orientation with high accuracy.

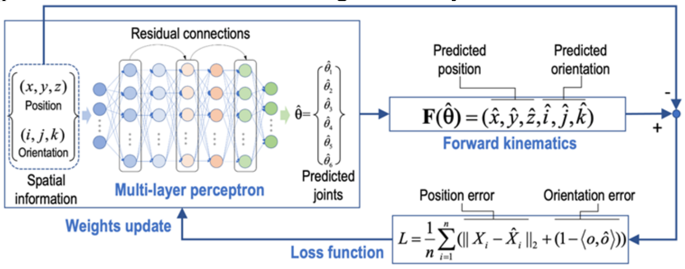

FIGURE 5: Self-supervised learning for joint angle control

The FK investigated in this study is based on the standard notion of Denavit-Hartenberg (D-H convention). In Table 1, the D-H table of the 6-axis robotic arm is presented.

TABLE 1 D-H table for the 6-axis robotic arm

Joint	θ(deg)	d(mm)	a(mm)	α(deg)
1	θ_1^*	337	0	$-90°$
2	$\theta_2^* - 90°$	0	210.5	0
3	$\theta_3^* + 90°$	0	0	$90°$
4	θ_4^*	268	0	$-90°$
5	θ_5^*	0	0	$90°$
6	θ_6^*	306.5	0	0

The parameters in the D-H table describe the relationship between the previous joint coordinate and the current joint coordinate:

- θ(deg): rotation angle from previous x_{i-1} axe to current x_i axe around z_{i-1} axe;
- d(mm): offset of o_i along z_{i-1} axe from o_{i-1};
- a(mm): offset of o_i along x_i axe from o_{i-1};
- α(deg): rotation angle from previous z_{i-1} axe to current z_i axe around x_i axe;
- θ_i^*: joint variables changing with time.

Based on Table 1, the FK of the end effector (x_e, y_e, z_e) from the origin of the robot arm (x_o, y_o, z_o) is computed as:

$$(x_e, y_e, z_e) = \left[\prod_{i=1}^{5} A_{i+1}^{i} \cdot (x_o, y_o, z_o)^{\mathrm{T}}\right]^{\mathrm{T}} \tag{3}$$

where A_{i+1}^{i} represents the transformation matrices computed from D-H table:

$$A_{i+1}^{i} = \begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\sin\alpha_i & a_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\cos\theta_i\sin\alpha_i & a_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4}$$

The structure of the developed MLP for predicting the joint angles consists of residual connections [25] and a sinusoidal activation function [26]. Residual connections are designed to alleviate the limitations of the standard MLP, in which contributions from features extracted by the early layers gradually fade away when processing the late layers. This is achieved by preserving access to these early layer features such that the prediction can utilize all features. Sinusoidal activation function is inspired by the solutions to partial differential equations [26], where improved reconstruction of 1D and 2D data with trigonometric nature is shown. As the purpose of the MLP is essentially to solve equations of IK with trigonometric nature, sinusoidal activation function is considered.

In practice, both the input and output of the MLP are normalized. The activation function at the output layer is the sigmoid function, which constrains the range of the output to [0, 1]. The loss function L for training the MLP is given as:

$$L = \frac{1}{n}\sum_{i=1}^{n}(\| X_i - \hat{X}_i \|_2 + (1 - \langle o, \hat{o} \rangle)) \tag{5}$$

where $\| \cdot \|_2$ denotes the error between the desired end-effector position X_i and the predicted position $\hat{X}_i$ measured using Euclidean distance. $1-\langle o,\hat{o}\rangle \in [0,2]$ measures the error in end-effector orientation. It is 0 when both orientation vectors perfectly align. n is the number of training inputs.

3. EXPERIMENTAL EVALUATION

The developed method is experimentally evaluated in a testbed assembly workspace, as shown in Fig. 6. The testbed is placed to the right of the worker and an eDo robot [27] is installed on the table to his left. Parts that are required for the assembly of a bearing module (bearing cap), a driving sheave module (sheave) and a belt module (belt) of the testbed (see Fig. 7) are placed next to the robot along with the needed tools such as a screwdriver and a ratchet. The parts and tools are monitored by a RGB camera (1280 x 720) installed to the right of the robot.

Human-robot collaboration is achieved by monitoring the human worker using a separate camera (1280 x 720), which is placed outside of Fig. 6. This allows to infer the part/tool that is going to be needed for the subsequent assembly operation and predict the worker's future motion trajectory [4, 14]. The inferred part/tool triggers the object detection algorithm to generate the desired position and orientation of the end-effector for its grasp. The position and orientation information are then fed into the MLP to predict the desired joint angles and grasp the part/tool in an appropriate manner. Once the worker's motion indicates that he/she is ready for handover, the predicted trajectory end-location is again fed into the MLP to predict the joint angles that allow to deliver the part/tool to the handover location.

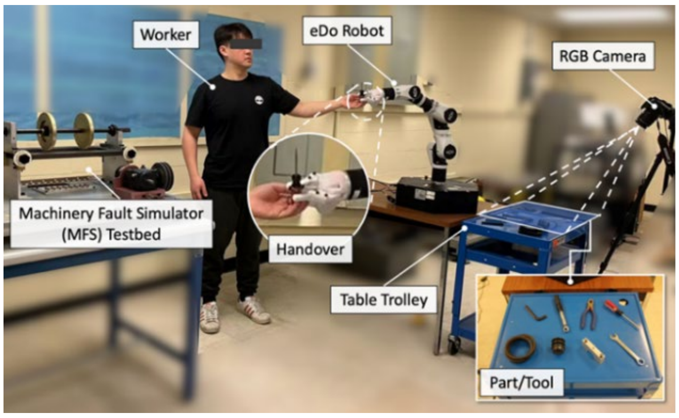

FIGURE 6: Collaborative testbed assembly workspace

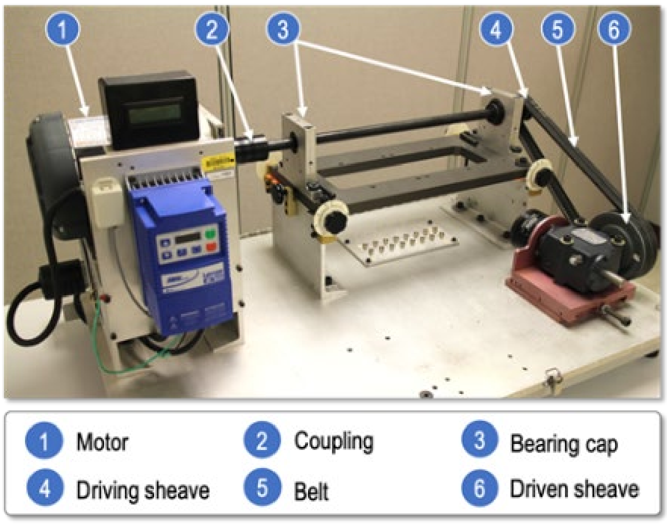

FIGURE 7: Modules in testbed assembly

To train the object detection algorithm, a total of 180 images of parts/tools were collected and split into 130 images for training and 50 for testing. Each image contains a different combination of the five parts and tools that are of interest: bearing cap, sheave, belt, screwdriver, and ratchet. The position and orientation of each part/tool is varied from image to image. To facilitate the identification of desired end-effector position and orientation for grasping the screwdriver and ratchet, the handles of both tools are labeled separately from the shank. The center of the handle bounding box determines the end-effector position, while the direction perpendicular to the line connecting the centers of the handle and the shank bounding boxes determines the orientation.

For the backbone CNN, a MobileNet [28] pretrained using the "common objects in context", or the COCO dataset [29] is selected. MobileNet is characterized by a series of modifications to the standard CNN to reduce the computational complexity such as decomposing the convolution operation into depth-wise and point-wise operations [28]. The network structure of is illustrated in Table 2. The COCO dataset contains many daily objects, such as fork, knife, and spoon, which share visual similarities to the parts/tools commonly seen in a manufacturing setting. Each image of dimension 1280 x 720 is first cropped to fit the table trolley area before resized to 640 x 640 to serve as the network input. For SSD, the number of predictions made by each grid cell is empirically set to k=4 [16]. Object detection training is carried out using a Tesla P100 GPU in Google Colab and learning rate is set to 0.05.

TABLE 2 Network structure of pretained MobileNet

Input Size	# Output Feature Maps	Kernel Stride
640 x 640 x 3	32	2
320 x 320 x 32	16	1
320 x 320 x 16	24	2
160 x 160 x 24	32	2
80 x 80 x 32	64	2
40 x 40 x 64	96	1
40 x 40 x 96	160	2
20 x 20 x 160	320	1
20 x 20 x 320	1280	1
20 x 20 x 1280	-	-

For robot joint angle control, joints 4 and 6 of the eDo robot (see image in Table 1) are constrained. Specifically, joint 4, which controls the rotation of end-effector with respect to the operation plane of joints 2 and 3, is set to 0 given that the robotic arm only grasps objects vertically in this study. Joint 6, which controls the opening direction of the gripper, is constrained by the orientation information from the object detection and the rotating angle of joint 1. Therefore, the prediction of joint angles for part/tool grasping is reduced to joints 1, 2, 3, and 5.

The structure of the MLP consists of five hidden layers with 81 neurons each. During training, no ground truth data for joint angles is required. Instead, 8,000 end-effector position and orientation samples, which are the inputs to the MLP, are randomly generated in a 1x1x1m cube within the workspace, where 60% of the samples are used as training data and the rest 40% used as testing data. To ensure that the sampled position and orientation can be reached by the robotic arm, the sampling process follows the limits of joints to cover the workspace cube.

4. RESULTS AND DISCUSSION

The outcome of the experimental evaluation is presented and discussed in this section. First, the outcome of algorithm training is evaluated using the collected training and testing data as described in Section 3. Next, the algorithms are evaluated in a collaborative assembly scenario for which additional variations are added, such as parts/tools not seen during training, to evaluate the robustness of the developed methods.

4.1 Algorithm training outcome

The training and testing curves for SSD-based object detection is shown in Fig. 8. The testing curve is obtained by feeding the object detection algorithm with testing data once every 100 epochs. Convergence is observed after approximately 1500 epochs. It is noted that at the end of the training process, both curves are closely aligned, indicating no overfitting.

TABLE 3 Joint angle prediction results comparison

Network structure	Activation function	Robot arm system	Position error	Orientation error
MLP with residual connections	Sinusoidal, Sigmoid	4 axes in 3D space	10mm	2.1°
MLP without residual connections	Sinusoidal, Sigmoid	4 axes in 3D space	24mm	2.3°
MLP [21]	Tanh, ReLU	3 axes in 2D space	100mm	N/A
MLP [18]	Tanh, ReLU	6 axes in 3D space	78mm	N/A

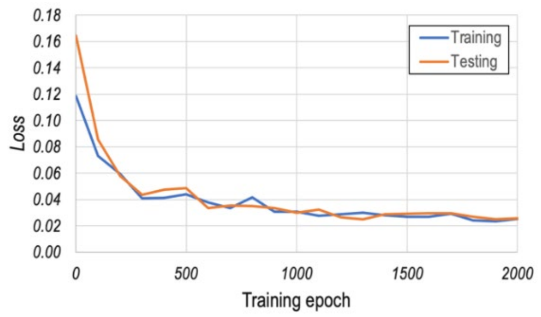

FIGURE 8: Training and testing results for object detection

Four representative testing images are shown in Fig. 9 to demonstrate the performance of the object detection algorithm. Images (a) and (d) do not contain the full set of the five parts/tools of interest. This is common in assembly as different parts/tools can be used in different orders. In addition, for images (b) and (c) where the full set of five parts/tools are available, their positions and orientations have also been varied. It is seen in Fig. 9 that the object detection algorithm has successfully recognized and localized the parts/tools in these scenarios, confirming the effectiveness and robustness of the algorithm in spite of the variations introduced for the trained objects.

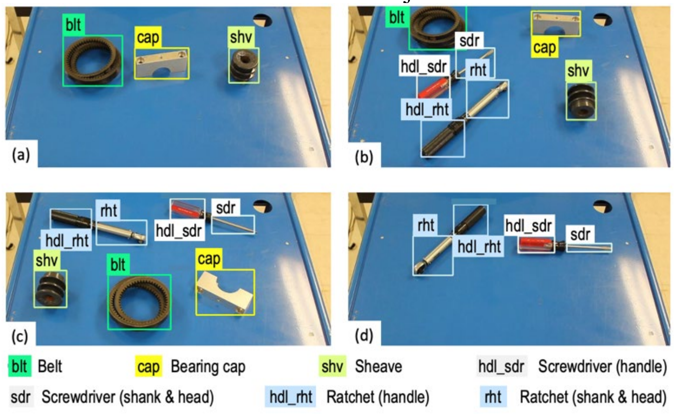

FIGURE 9: Experimental results for SSD-based object detection

For training the self-supervised MLP, convergence is observed at round 500 epochs, as shown in Fig. 10. Table 3 illustrates the effectiveness of the residual connections and sinusoidal activation function for improving object prediction accuracy. Specifically, the developed MLP with residual connections and sinusoidal activation is compared to: 1) a MLP with only sinusoidal activation, 2) two MLPs without residual connections and sinusoidal activation from [18, 21]. From Table 3, the developed algorithm has shown to achieve the highest accuracy in the predictions of both position and orientation.

4.2 Experimental test

In this section, the performance of the developed methods is evaluated in a collaborative assembly scenario. Of direct interest is the sequence from the robot's receiving a command on the needed part/tool to the handover to human worker. To simulate a realistic shop floor environment, additional parts/tools that are not used during the object detection training such as an Allen key, wrench, and pliers, are placed along with the five parts/tools of direct interest to evaluate whether the object detection algorithm can reliably detect the correct ones. This scenario is common since different workspaces may share a central location for parts/tools. Object detection is also continuously carried out during the whole grasping action to evaluate the impact of the gripper on detection performance. For simplicity, the robotic arm is also constrained to 4-axis during handover, with the head of screwdriver or ratchet pointing perpendicularly away from rotation plane shared by joints 2, 3, and 5.

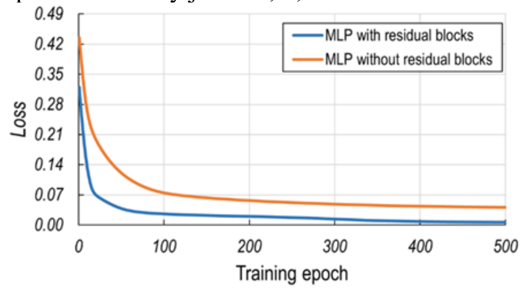

FIGURE 10: MLP Training with and without residual connections

To avoid collision with the parts/tools, each robot grasping is preceded by a "reaching" stage during which the end-effector arrives at the position with the desired *x-y* coordinate and a *z*-coordinate of 100 mm before moving downwards and closing the gripper. A representative grasping scenario is illustrated in Fig. 11, in which a screwdriver is of interest. The three frames (a)-(c) indicate when the robot is at standby position ("standby", in which object detection is carried out and robot is triggered), when the gripper moves to the position above the object ("reaching"), and when the gripper moves downwards towards the object, and arrives at the position for grasping ("grasping"), respectively.

It is noted from Fig. 11(d)-(f) that the object detection algorithm exhibits robustness when objects that are not of interest (i.e., Allen key, the pilers and the wrench) are placed along with the objects of interest. For example, none of the Allen key, pliers and wrench is mistakenly detected. In addition, the object detection algorithm also demonstrates robustness when the objects are partially blocked by the gripper, as in Fig. 11(e), without being specifically trained for such scenario. However, it is noted that, the handle of the screwdriver is mis-recognized as bearing cap (Fig. 11(f)) when the gripper arrives at the grasping position, indicating that there are still scenarios for which the object detection algorithm needs to be improved as a subject of future research. The overall object detection accuracy is illustrated in Fig. 12. Quantitative evaluation of the MLP performance in robotic grasping indicates that the mean errors in the position and orientation induced by the joint angle prediction

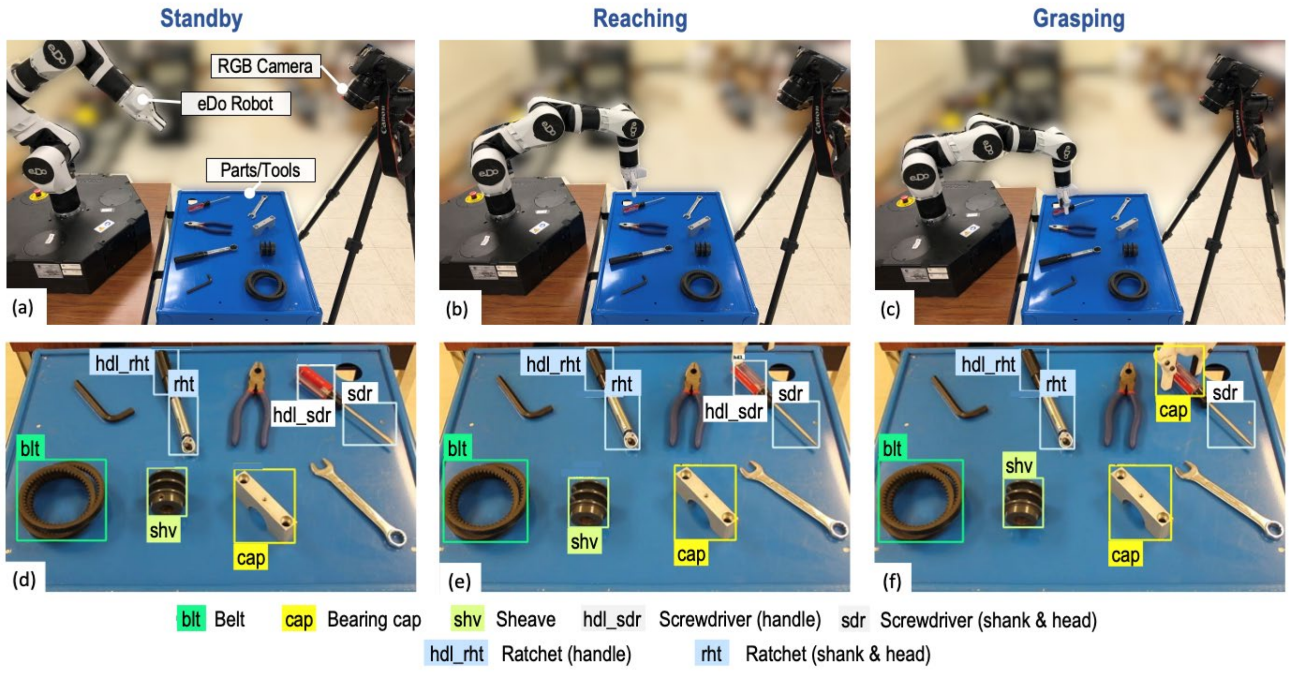

FIGURE 11: Object detection performance as end-effector descends for grasping screwdriver

error from MLP are 15 mm and 1.5 degree with respect to the desired values, respectively, which are on the same order as the simulation results shown in Section 4.1 (10 mm and 2.1 degree).

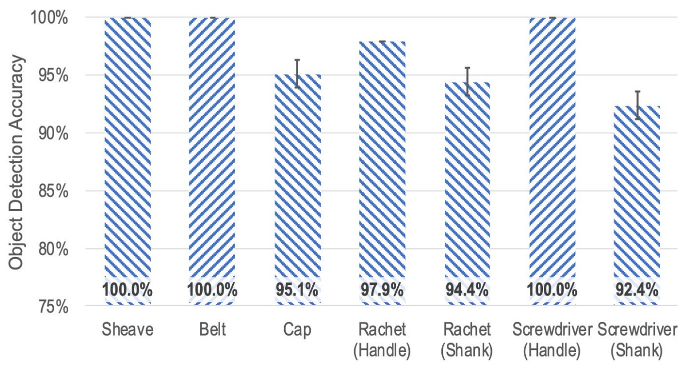

FIGURE 12: Object detection accuracy

5. CONCLUSIONS

In an effort to improve the adaptivity of robot in dynamic workspace shared with human workers and realize human-robot collaborative assembly, an integrated method based on object detection and robot joint angle control has been developed. The work fills an existing gap in recognizing, localizing, and properly grasping the needed parts/tools for collaborative operations. Object detection is based on the concept of single-shot detection, which allows for the utilization of existing, pretrained CNN as the computation backbone. The desired robotic arm end-effector position and orientation for object grasping, which are extracted from the object detection algorithm, are then fed into a MLP to predict the required robotic arm joint angles. A self-supervised training method is developed for the MLP, which relies on the forward kinematics of the robotic arm and does not require labeled training data. Evaluated in a collaborative testbed assembly case study, the object detection method has shown predominantly reliable recognition and localization of parts/tools of interest, without making false identifications on previously unseen objects. The MLP with self-supervised learning capability has also achieved joint angle prediction with good accuracy, with the mean end-effector position error being 15mm and mean orientation error being 1.5 degree only. Future research will systematically investigate scenarios where parts/tools are partially blocked, to further enhance the object detection capability, and extend the self-supervised learning method to full 6-axis robotic arm configuration.

ACKNOWLEDGEMENTS

The authors acknowledge the support for this research by the National Science Foundation under award CMMI-1830295.

REFERENCES

[1] Lu, Y., Xu, X. and Wang, L., 2020, "Smart manufacturing process and system automation–a critical review of the standards and envisioned scenarios. Journal of Manufacturing Systems," 56, pp.312-325.
[2] Wang, L., Gao, R., Váncza, J., Krüger, J., Wang, X.V., Makris, S. and Chryssolouris, G., 2019, "Symbiotic human-robot collaborative assembly," CIRP annals, 68(2), pp.701-726.
[3] Wang, P., Liu, H., Wang, L. and Gao, R., 2018, "Deep learning-based human motion recognition for predictive context-aware human-robot collaboration," CIRP annals, 67(1), pp.17-20.
[4] Zhang, J., Liu, H., Chang, Q., Wang, L. and Gao, R., 2020, "Recurrent neural network for motion trajectory prediction in human-robot collaborative assembly," CIRP annals, 69(1), pp.9-12.
[5] Wang, L., Liu, S., Cooper, C., Wang, X.V. and Gao, R., 2021, "Function block-based human-robot collaborative assembly driven by brainwaves," CIRP annals, 70(1), pp.5-8.
[6] Levine, S., Pastor, P., Krizhevsky, A., Ibarz, J. and Quillen, D., 2018, "Learning hand-eye coordination for robotic grasping

with deep learning and large-scale data collection," International journal of robotics research, 37(4-5), pp.421-436.

[7] Zeng, A., Song, S., Lee, J., Rodriguez, A. and Funkhouser, T., 2020, "Tossingbot: Learning to throw arbitrary objects with residual physics," IEEE Transactions on Robotics, 36(4), pp.1307-1319.

[8] Rao, K., Harris, C., Irpan, A., Levine, S., Ibarz, J. and Khansari, M., 2020, "Rl-cyclegan: Reinforcement learning aware simulation-to-real," Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition pp. 11157-11166.

[9] Girshick, R., Donahue, J., Darrell, T. and Malik, J., 2014, "Rich feature hierarchies for accurate object detection and semantic segmentation," Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 580-587.

[10] Girshick, R., 2015, "Fast r-cnn," Proceedings of the IEEE international conference on computer vision, pp. 1440-1448.

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

[12] Redmon, J., Divvala, S., Girshick, R. and Farhadi, A., 2016, "You only look once: Unified, real-time object detection," Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 779-788.

[13] Krueger, J., Lehr, J., Schlueter, M. and Bischoff, N., 2019, "Deep learning for part identification based on inherent features," CIRP Annals, 68(1), pp. 9-12.

[14] Zhang, J., Wang, P. and Gao, R., 2021, "Hybrid machine learning for human action recognition and prediction in assembly," Robotics and Computer-Integrated Manufacturing, 72, p.102184.

[15] Xiong, Q., Zhang, J., Wang, P., Liu, D. and Gao, R., 2020, "Transferable two-stream convolutional neural network for human action recognition," Journal of Manufacturing Systems, 56, pp.605-614.

[16] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y. and Berg, A.C., 2016, "Ssd: Single shot multibox detector," European conference on computer vision, pp. 21-37.

[17] Ren, H. and Ben-Tzvi, P., 2020, "Learning inverse kinematics and dynamics of a robotic manipulator using generative adversarial networks," Robotics and Autonomous Systems, 124, p.103386.

[18] Aggarwal, L., Aggarwal, K. and Urbanic, R.J., 2014, "Use of artificial neural networks for the development of an inverse kinematic solution and visual identification of singularity zone (s)," Procedia CIRP, 17, pp.812-817.

[19] Karlik, B. and Aydin, S., 2000, "An improved approach to the solution of inverse kinematics problems for robot manipulators," Engineering applications of artificial intelligence, 13(2), pp.159-164.

[20] Chiddarwar, S.S. and Babu, N.R., 2010, "Comparison of RBF and MLP neural networks to solve inverse kinematic problem for 6R serial robot by a fusion approach," Engineering applications of artificial intelligence, 23(7), pp.1083-1092.

[21] Duka, A.V., 2014, "Neural network based inverse kinematics solution for trajectory tracking of a robotic arm," Procedia Technology, 12, pp.20-27.

[22] Szeliski, R., 2010, "Computer vision: algorithms and applications," Springer Science & Business Media.

[23] Ho, T., Kang, C.G., Lee, S., 2012, "Efficient closed-form solution of inverse kinematics for a specific six-DOF arm", International Journal of Control, Automation and Systems, 10(3), pp.567-573.

[24] Beeson, P., Ames, B., 2015, "TRAC-IK: An open-source library for improved solving of generic inverse kinematics", IEEE-RAS 15th International Conference on Humanoid Robots, pp. 928-935.

[25] He, K., Zhang, X., Ren, S. and Sun, J., 2016, "Identity mappings in deep residual networks," In European conference on computer vision, pp. 630-645.

[26] Sitzmann, V., Martel, J., Bergman, A., Lindell, D. and Wetzstein, G., 2020, "Implicit neural representations with periodic activation functions," Advances in Neural Information Processing Systems, 33, pp.7462-7473.

[27] https://www.comau.com/

[28] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A. and Chen, L.C., 2018, "Mobilenetv2: Inverted residuals and linear bottlenecks," Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 4510-4520.

[29] Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P. and Zitnick, C.L., 2014, "Microsoft coco: Common objects in context," European conference on computer vision, pp. 740-755.