International Journal of Mechanical, Industrial and Aerospace Sciences
ISSN: 2517-9950
Vol:14, No:8, 2020

Deep Learning Application for Object Image Recognition and Robot Automatic Grasping

Shiuh-Jer Huang, Chen-Zon Yan, C. K. Huang, Chun-Chien Ting

***Abstract*—**Since the vision system application in industrial environment for autonomous purposes is required intensely, the image recognition technique becomes an important research topic. Here, deep learning algorithm is employed in image system to recognize the industrial object and integrate with a 7A6 Series Manipulator for object automatic gripping task. PC and Graphic Processing Unit (GPU) are chosen to construct the 3D Vision Recognition System. Depth Camera (Intel RealSense SR300) is employed to extract the image for object recognition and coordinate derivation. The YOLOv2 scheme is adopted in Convolution neural network (CNN) structure for object classification and center point prediction. Additionally, image processing strategy is used to find the object contour for calculating the object orientation angle. Then, the specified object location and orientation information are sent to robotic controller. Finally, a six-axis manipulator can grasp the specific object in a random environment based on the user command and the extracted image information**.** The experimental results show that YOLOv2 has been successfully employed to detect the object location and category with confidence near 0.9 and 3D position error less than 0.4 mm. It is useful for future intelligent robotic application in industrial 4.0 environment.

***Keywords*—**Deep learning, image processing, convolution neural network, YOLOv2, 7A6 series manipulator.

I. Introduction

DUE to electronic technology progress, machine learning had been proposed in image, speech recognition and robot automatic operation applications. Cockbum et al. [1] employed unsupervised learning method in robot grasp stability assessment based on tactile sensor information. Yoo and Johansson [2] employed supervised learning scheme to establish a road sign recognition model for mobile robot localization. Deep learning is one type of machine learning with radial basis model (RBM) neural network. This neural network learning structure can improve the converging and accuracy problems. The loss function is an index for training and evaluating the convergence of neural network model. The gradient descent scheme and stochastic gradient descent algorithm were proposed to minimize the loss function during neural network model training process for improving the converging speed [3], [4].

CNN is widely adopted in deep leaning strategy for image classification due to its robustness. It had been applied in object recognition and robotic grasp operation [5] and mobile robot road detection and navigation [6]. Martinson and Yalla [7] employed CNN for human pose detection with depth image and color image sensor fusion CNN learning model. Peng et al. [8] employed the object image contour and object variety pose to train the CNN for identifying the 3D objects. Many CNN structures and algorithms were proposed to improve the model robustness for object image recognition [9]. Shelhamer et al. [10] proposed a fully convolutional network for overlay object image segmentation. Redmon et al. [11] developed a You Only Look Once (YOLO) scheme to integrate with CNN algorithm for establishing a real-time object detection model.

Due to CPU and accompanied GPU computation capability enormous promotion, the machine vision and artificial intelligence applications were widely progresses. For the industrial 4.0 and intelligence manufacturing development, the industrial robot needs to improve the ability for overcoming the random working conditions. Hence, the real-time machine vision system is an important auxiliary function for improving the robot working capability. Here, the deep learning strategy is employed to establish an image CNN model for object real-time recognition and obtaining the object 3D location. This information was sent to PC-based robotic manipulator for executing automatic classification and specified pick-and-place operation. The industrial parts can be identified real-time by placing in random position, orientation and sequence.

II. System Structure

The main hardware structure of this implementation system is vision system, six-degree of freedom (DOF) manipulator and its PC-based control system as shown in Fig. 1. They are communicated with TCP/IP interface. The vision system is adopted Intel RealSense SR300 digital camera, Fig. 2, to catch the image depth information and 2D color image and send to a PC with GTX 1060 GPU through USB 3.0 for deep leaning CNN model training. It has one color image camera with pixel resolution 1920x1080, one IR camera and IR laser projector. The IR depth image resolution is 640x 480. The software is established with Python language based on Intel RealSense software development kit (SDK) library. Since the OpenCv and TensorFlow [12] open source libraries had provided many deep leaning calculation and image processing programs library, they are employed in this research to hasten the overall software development. The integrated robotic manipulator is a six-DOF PC-based control serial joint robot built by ITRI with type 7A6 as shown in Fig. 3. The pneumatic gripper is attached on the robotic wrist for object pick-and-place operation.

Shiuh-Jer Huang is with the National Taiwan University of Science and Technology, Taiwan (e-mail: sjhuang@mail.ntust.edu.tw).

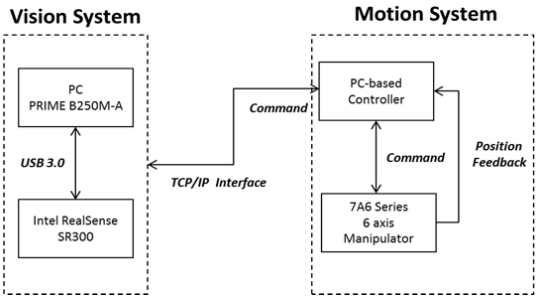

Fig. 1 Overall system signal flow structure

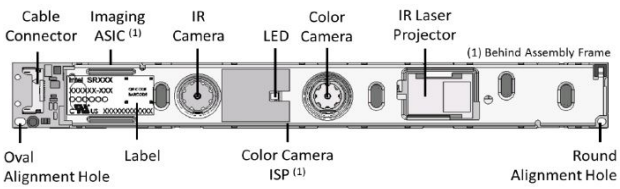

Fig. 2 Frontal face of Intel RealSense SR300 camera

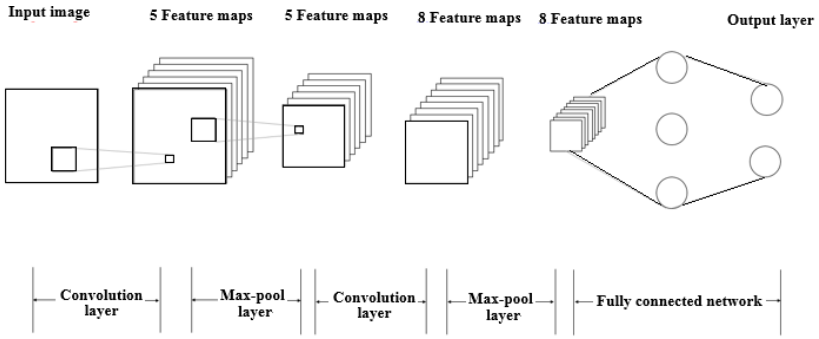

Fig. 3 CNN structure

III. CNN Learning Model

The CNN is modified from back propagation neural network with additional convolutional layer and maxpool layer in head of the neural input layer. Its operational flowchart is shown in Fig. 3. The full image input should go through these two layer operations to pick up the image features for executing neural model training. nxn filters should be selected for the convolutional operation to obtain the features map. Before convolution operation, the padding and striding operations can be introduced for increasing the input data width and setting filter location interval, respectively. Then, the image output matrix was defined. The maxpool operation is to select a matrix frame for extracting the maximum value from those marked matrix element and set a stride value for specifying the frame moving interval for each maximal element pick up to construct a new output matrix. The numbers of convolution layer and maxpool layer, and their connection structure are designed by the user.

IV. YOLOv2 Image Recognition and Classification System

Redmon et al. [11] proposed a YOLO scheme for object detection by using CNN model with robust classification and object position prediction function. Yolov2 is a 2nd version of YOLO with momentum stochastic gradient descent [13] optimization algorithm, weight decay and batch normalization algorithm [14] for improving the model prediction accuracy. YOLO has 24 convolutional layers and 5 maxpool layers with 448x448 pixels image input and 7x7 pixels output followed by 2 fully connected neural network layers. YOLOv2 has 19 convolutional layers and 5 maxpool layers with 224x224 pixels image input and 7x7pixel image output. The operation of convolutional layer will not reduce the pixel number. The operation of maxpool layer will reduce the image pixel number based on specified frame size and strides.

YOLOv2 treats the object detection as a regression problem by separating bounding boxes and associated class probabilities in global image. Hence, the proposed CNN model can directly predict bounding box location and class probability within a full image for YOLO at an image. It can predict multi-bounding boxes location and their corresponding class probabilities simultaneously based on entire image features. The ReLU activation function was chosen for the neural network neuron as

$$\varphi(x) = \begin{cases} x, if x > 0 \\ 0.1x, otherwise \end{cases} \quad (1)$$

The loss function for the YOLOv2 is specified as a sum square error as

$$\lambda_{obj}^{coord} \sum_i^{S^2} \sum_j^B l_{ij}^{obj} \left[\left(x_{ij}^{pred} - x_{ij}^{obj}\right)^2 + \left(y_{ij}^{pred} - y_{ij}^{obj}\right)^2 + \left(w_{ij}^{pred} - w_{ij}^{obj}\right)^2 + \left(h_{ij}^{pred} - h_{ij}^{obj}\right)^2 \right] + \lambda_{nonobj}^{coord} \sum_i^{S^2} \sum_j^B l_{ij}^{noobj} \left[\left(x_{ij}^{pred} - x_{ij}^{anchor_{center}}\right)^2 + \left(y_{ij}^{pred} - y_{ij}^{anchor_{center}}\right)^2 + \left(w_{ij}^{pred} - w_{ij}^{anchor_{defalut}}\right)^2 + \left(h_{ij}^{pred} - h_{ij}^{anchor_{defalut}}\right)^2 \right] + \lambda_{obj}^{conf} \sum_i^{S^2} \sum_j^B l_{ij}^{obj} \left[conf_{ij}^{pred} - iou\left(box_{ij}^{pred}, box_{ij}^{truth}\right) \right]^2 + \lambda_{nonobj}^{conf} \sum_{i=0}^{S^2} \sum_j^B l_{ij}^{no_{obj}} \left[conf_{ij}^{pred} - 0 \right]^2 + \sum_{i=0}^{S^2} \sum_j^B l_{ij}^{obj} \left[p_{ij}^{pred}(c) - p_{ij}^{truth}(c) \right]^2 \quad (2)$$

where S is the grid cell number, B is the anchor box number,

$(x_{ij}^{pred}, y_{ij}^{pred})$ are predicted coordinates, $(x_{ij}^{obj}, y_{ij}^{obj})$ are object real coordinates, $(w_{ij}^{pred}, h_{ij}^{pred})$ are predicted object length-width, $(w_{ij}^{obj}, h_{ij}^{obj})$ are object real length-width, $(x_{ij}^{anchor_{center}}, y_{ij}^{anchor_{center}})$ are anchor box center coordinates, $(w_{ij}^{anchor_{default}}, h_{ij}^{anchor_{default}})$ are anchor box length-width, $conf_{ij}^{pred}$ is the predicted confidence, $iou(box_{ij}^{pred}, box_{ij}^{truth})$ are anchor box intersection rate of predicted with respect to real, $(p_{ij}^{pred}(c), p_{ij}^{truth}(c))$ are predicted and truth object classified type c probability, respectively. λ_{obj}^{conf} is the confidence gain of with object, λ_{nonobj}^{conf} is the confidence gain of without object, λ_{obj}^{coord} is the coordinates calculation gain of with object, and λ_{nonobj}^{coord} is the coordinates calculation gain of without object, respectively. l_{ij}^{obj} denotes the jth bounding box predictor with object appears in cell i for object prediction. $l_{ij}^{no_{obj}}$ represents object was not appeared in cell i for jth bounding box predictor. Hence, the CNN model output can be represented as

$$[p_c, b_x, b_y, b_w, b_h, c_1, c_2 \dots \dots \dots c_n]^T \quad (3)$$

where p_c is the confidence of an object existing in a grid cell. (b_x, b_y) and (b_w, b_h) are bounding box coordinates and width and height, respectively. (c_1, c_2, …… c_n) are the probability of object in each class. The overall YOLOve2 model output information can be described as

$$S \times S \times (B \times 5 + C) \quad (4)$$

where S is the number of grid cell, B is the number of bounding box, and C is the object category, respectively.

There may have many bounding boxes and objects categories predicted from the CNN 7x7 image output. The confidence calculation formula is:

$$confidence = P_r(object) \times IOU_{pred}^{truth} \quad (5)$$

where $P_r(object)$ is the probability of grid cell cover object and IOU_{pred}^{truth} is the ratio of the intersection of bounding box real marked area and predicted output area with respect to the union of bounding box real marked area and predicted output area. Then, the confidence of each object classification can be defined as

$$P_r(Class_i|object) \times P_r(object) \times IOU_{pred}^{truth} = P_r(Class_i) \times IOU_{pred}^{truth} \quad (6)$$

Usually, the predicted bounding boxes number is too much for real object number. The non max suppression scheme was employed to filter the redundant bounding boxes by selecting an IOU threshold value to eliminate those bounding boxes with IOU value less than it. In addition, the output information anchor box of YOLOv2 has each box to include the predicted object location and size, confidence and N-classes probability instead of using two boxes to describe object of YOLO. Based on Fig. 4, the object location prediction formula had been modified, too [15].

$$\begin{cases} b_x = \sigma(t_x) + c_x \\ b_y = \sigma(t_y) + c_y \end{cases} \quad (7)$$

$$\begin{aligned} b_w &= p_w e^{t_w} \\ b_h &= p_h e^{t_h} \end{aligned} \quad (8)$$

where (t_x, t_y) are deviation with respect to grid cell, (c_x, c_y) are grid cell length and width (1), $\sigma(x)$ is sigmoid function, (t_w, t_h) are the deviation of Anchor Box length and width, (p_w, p_h) are Anchor Box length and width, respectively.

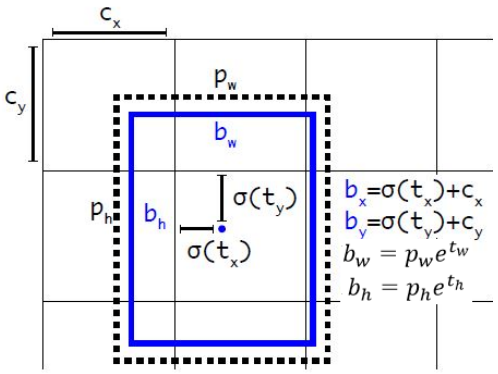

Fig. 4 Object center position direct prediction [15]

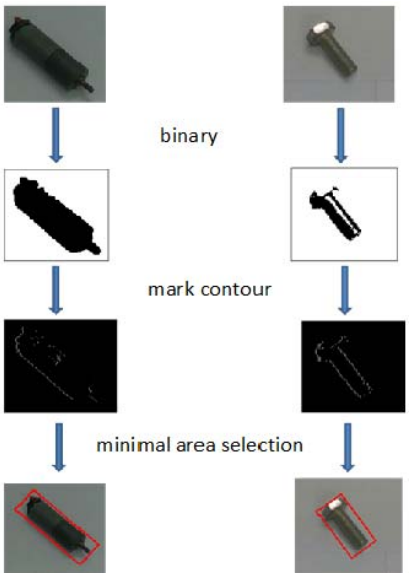

Fig. 5 Processing flowchart of object orientation searching

YOLOv2 employed Coco dataset for model training. However, it is difficult or impossible to prepare so much big datasets for model training in industrial application. If the CNN model is trained with several hundred data set only, the converged model may have overfitting or accuracy problems. Hence, the fine-tuning strategy [16] was adopted for CNN model training by using previous trained YOLOv2 model based on Coco data set as the initial CNN model.

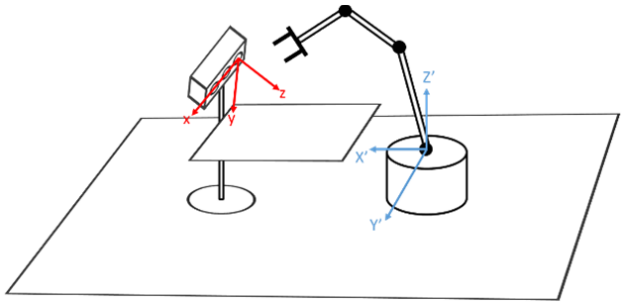

Fig. 6 The coordinate system's correlation between image system and robotic system

V. Object 3D Coordinates and Orientation Calculation

Based on deep learning CNN model, the object categories and central position can be obtained. However, the object orientation and 3D coordinates still need other image processing scheme or optical principle to derive. The captured image was binarized first and followed by morphology image processing. Then, the findConyours function of OpenCV function library was applied to find the existing objects contour based on Suzuki and Abe [17] algorithm from binary image. Since, the industrial parts have regular exterior, the polygon was employed to approach the finding object contour for reducing the coner points. Here, approxPolyDp function of OpenCV library was used to executing the polygon approaching based on Ramer-Douglas-Peucker algorithm. After that, the cvMinAreaRect2() function of OpenCv liobrary can be adopted to mark the minimal area of object contour for object orientation angle calculation. The processing flowchart is shown in Fig. 5.

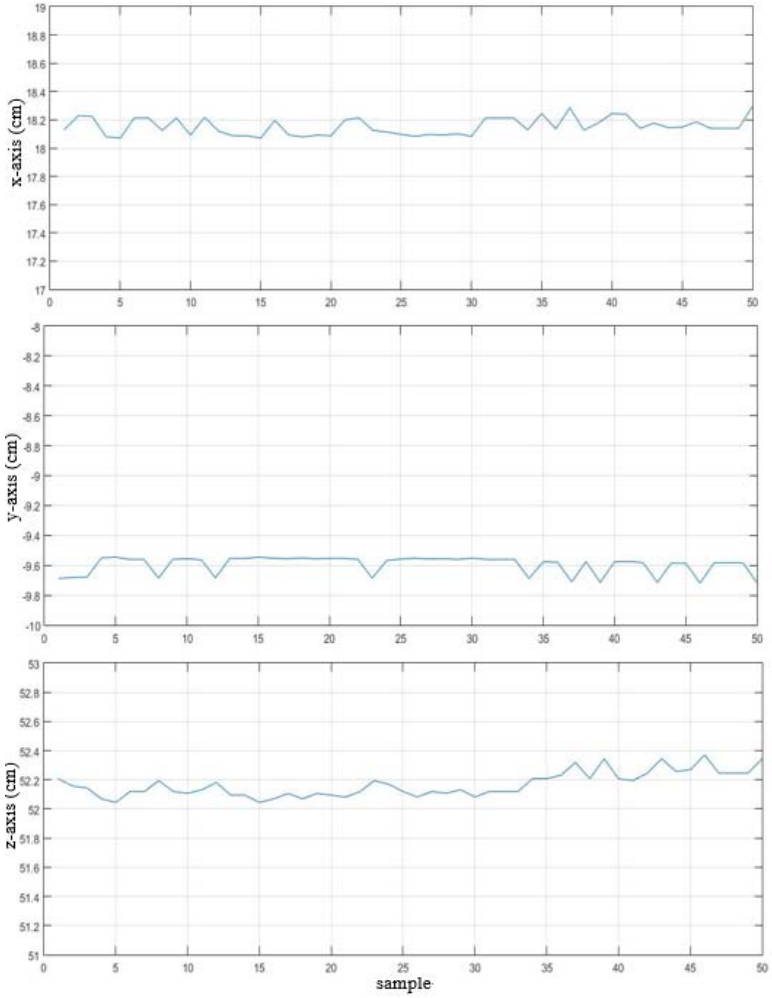

Fig. 7 Golden color part central position estimation

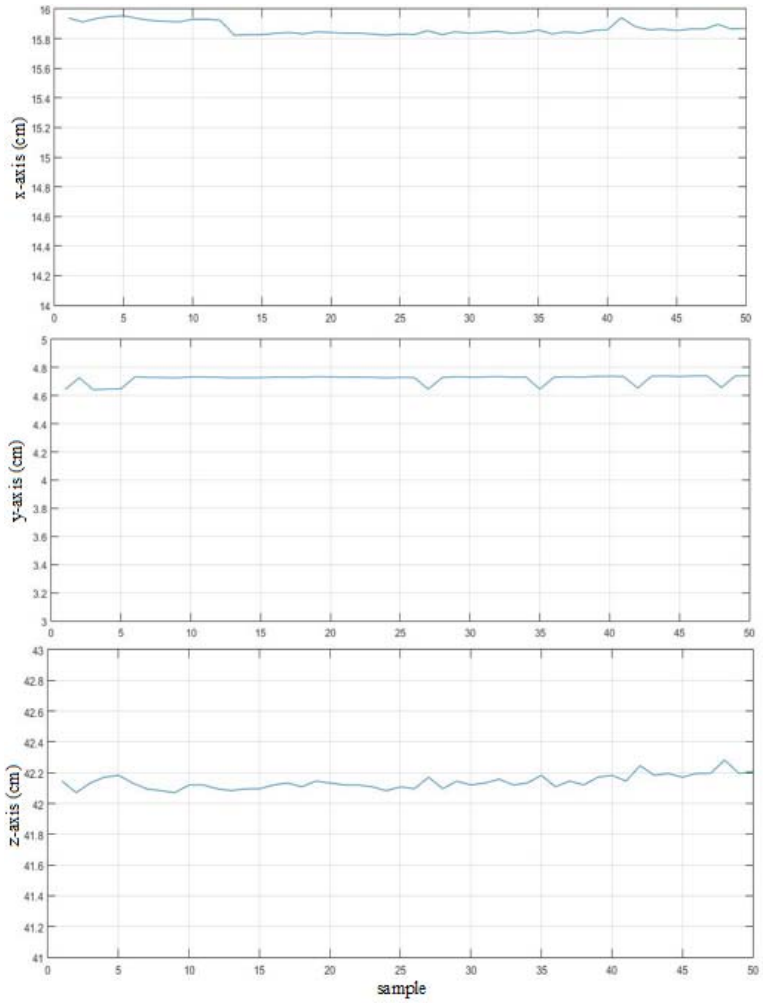

Fig. 8 Variation of screw central position estimation

The detected object 3D coordinates with respect to image coordinate system and robotic coordinate system need to be calculated individually for robotic automatic pick-and-place application. Here, the embedded Intel RealSense SR300 camera provides IR depth information for 3D coordinates calculation.

$$X = \frac{u*Z}{f}, Y = \frac{v*Z}{f} \quad (9)$$

where u, v are the pixels locations of object image center in full image, f is the focus length of camera and Z is the depth coordinates.

Based on the coordinated system geometric correlation between image system and robotic system as shown in Fig. 6, the 4x4 DH transformation matrix between both systems can be derived with rotation with respect to image x axis, following a rotation with respect to image z axis, and then a 3D coordinates translation. Hence, object central position and orientation with respect to robotic coordinated system can be calculated by using the derived DH transformation matrix and the object central point 3D coordinates with respect to IR camera system as:

$$\boldsymbol{O}_{\text{robot}} = \boldsymbol{H} * \boldsymbol{O}_{\text{image}} \quad (10)$$

$$\begin{bmatrix} X' \\ Y' \\ Z' \\ 1 \end{bmatrix} = \begin{bmatrix} cos\theta_Z & sin\theta_Z & 0 & T_x \\ -sin\theta_Z cos\theta_x & cos\theta_Z cos\theta_x & sin\theta_x & T_y \\ sin\theta_Z sin\theta_x & -cos\theta_Z sin\theta_x & cos\theta_x & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \quad (11)$$

VI. Experiment Planning and Results Discussion

In this study, the deep learning CNN model was employed to classify the object category and find the central point position. It still needs certain image processing for finding object 3D position and orientation for integrating with industrial robot for whole system automatic finding object and executing pick-and-place or assembly operation. Five different industrial parts are selected for this experiment.

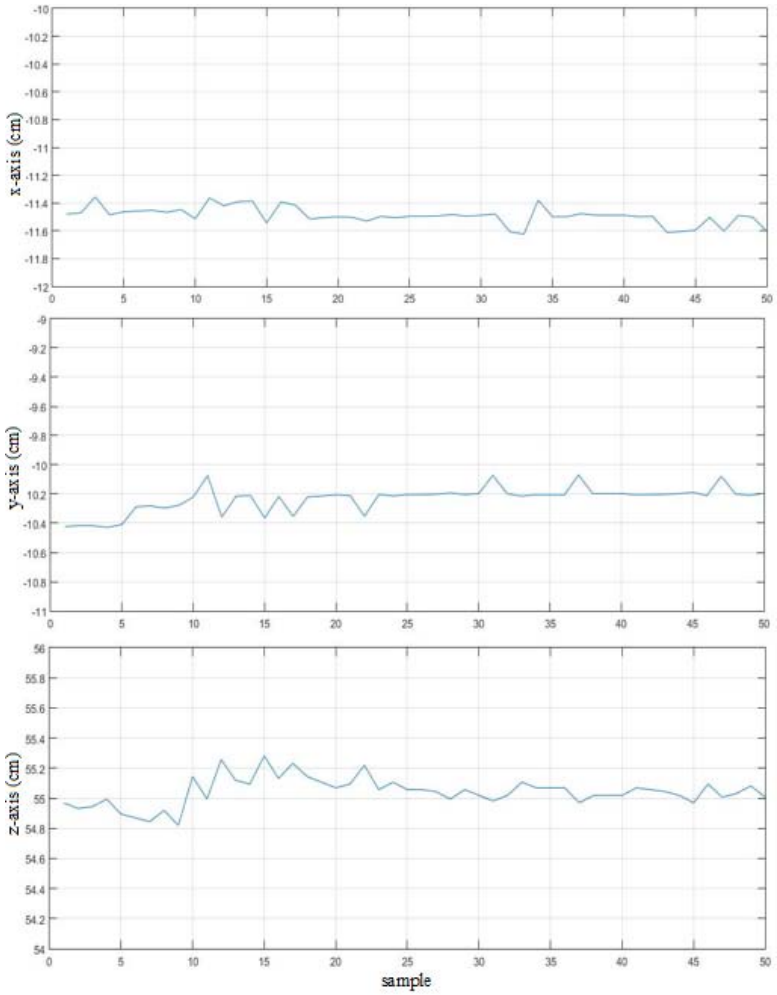

Fig. 9 Variation of motor central position estimation

(A) Deep Learning CNN Model and Parameters Setting

Here, a CNN model with nine convolutional layers and six maxpool layers neural network structure was constructed with 416x416 pixels image input and 12x12 pixels size output for neural network. The 640x480 RGB image data were resized to 416x416 for CNN model training. 500 datasets are prepared for CNN training with five different objects (circle, motor, screw and gold color part, iron square) located at different location, orientation and luminance. The CNN model parameters are shown in Table I. The Darknet 19 [18] was chosen as the pre-trained model and the Pascal VOC [19] was selected as the training dataset. Finally, the 500 industrial parts datasets are used to execute fine-tuning with 0.471 converged loss function value. The object detection confidence at different location, orientation and luminance are shown in Table II. It can be observed that the golden color part recognition at different orientation has smaller confidence due to its irregular shape.

TABLE I
CNN NEURAL MODEL PARAMETERS

Parameters	value	parameters	value
Input image pixel	416x416	Optimization scheme	SGD
Output image pixel	12x12	Learning rate	0.001
Convolutional layers	9	momentum	0.9
Maxpool layers	6	Weight-decay	0.005
Activation function	Leaky Relu	Batch-normalization	1
Loss function	Sum square error	Anchor Box	5
$\lambda_{coordinate}$	1	Object category	4
$\lambda_{non-object}$	1	Output data length	45
Object scale	5	Batch size	64
Class-scale	1	epoch	300

TABLE II
OBJECT DETECTION CONFIDENCE AT DIFFERENT CONDITIONS

Object	Golden color part	Circle	Motor	Screw
Confidence (location variation)	90.3	85.2	83.6	88.1
Confidence (orientation change)	78.8	94.9	82.7	86.1
Confidence (luminance variation)	81.3	94.2	83.2	86.7

(B) Object 3D Central Position Accuracy Evaluation

The 3D coordinates of detected object can be calculated with YOLOv2 CNN trained model and the IR camera depth information. 50 pictures of each static industrial object are selected to execute 3D coordinate error analysis. The experimental results of the central position estimation variation of golden color part, screw and motor are shown in Figs. 7-9, respectively. It can be observed that the position variation of each estimation has a little increasing with respect to object size. However, the error of each object center 3D component is less than 4 mm. It is good enough for robotic pick-and-place or assembly application.

(C) The Overall Integration Application of Object Image Detection and Robotic On-Line Automatic Operation

The PC-based CNN object detection and classification system is integrated with a 7A6 6 DOF PC-based control robot for object random position/orientation placing operation. The overall system control flowchart is shown in Fig. 10. The operation sequence is planned as CNN object detection system finding the screw first, robot move to pick the screw and move to other location for put down it. Then, the second robot pick-and-place object was specified as motor. Finally, a top view similar to motor iron block is randomly placed on the table and whole system is commanded to execute the object classification/location searching and monitor robot to do the pick-and place operation. 12 pictures are listed to show the specified robot manipulator on-line object finding from working table and pick up operation as shown in Fig. 11.

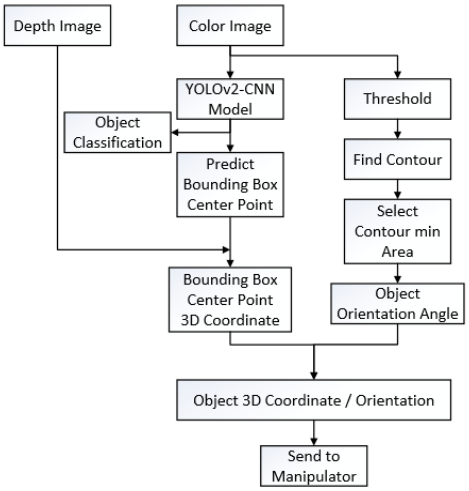

Fig. 10 The object detection/location finding and robot integration overall system control flowchart

VII. Conclusions

The PC-based CNN object image detection system and 6 DOF robotic integration application system was constructed. YOLOv2 based CNN has been successfully employed to detect the object location and category with confidence near 0.9 and 3D position error less than 0.4 mm. The developed control software can execute the specified object image searching and its location calculation, and transform the object 3D coordinates and orientation into robotic working space for real-time planning the manipulator moving path to pick-and-place the specified part or follow a specified assembly sequence. It is useful for future intelligent robotic implementation of object random placing or input from conveyer/carriage. This is an antecedently study for intelligent robotic application in industrial 4.0 environment.

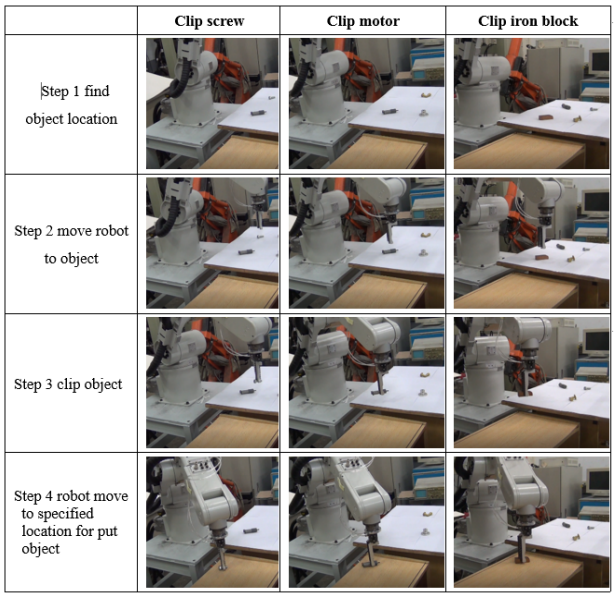

Fig. 11 The serial object finding and robotic pick-and-place operation pictures

Acknowledgment

Authors would like to thank the funding support from Industrial Technology Research Institute under the contract number 107AL859 of NTUST.

References

[1] Deen Cockbum, Jean-Philippe Roberge, Thuy-Hong-Loan Le, Alexis Maslyczyk and Vincent Duchaine, "Grasp stability assessment through unsupervised feature learning of tactile images," IEEE International Conference on Robotics and Automation (ICRA), May 29 ~ June 3, pp. 2238-2244, Singapore, 2017.

[2] Jaehyun Yoo and Karl H. Johansson, "Semi-supervised learning for mobile robot localization using wireless signal strengths," International 12Conference on Indoor Positioning and Indoor Navigation (IPIN), Sapporo, Japan, Sept 18-21, 2017.

[3] S. K. Lenka and A. G. Mohapatra, "Gradient Descent with momentum based neural network pattern classification for the prediction of soil moisture content in precision agriculture," IEEE International Symposium on Nanoelectronic and Information Systems, Indore, India, October 21-23, 2015, pp. 63-66.

[4] D. Soudry, D. Di Castro, A. Gal, A. Kolodny and S. Kvatinsky, "Memristor-based multilayer neureal network with online gradient descent training," IEEE Transactions on Neural Networks and Learning Systems, 26 (10), pp. 2408-2421, 2015.

[5] Andy Zeng, Kuan-Ting Yu, Shuran Song, Daniel Suo, Ed Walker, Alberto Rodriguez and Jianxiong Xiao, "Multi-view self-supervised deep learning for 6D pose estimation in the Amazon Picking Challenge," IEEE International Conference on Robotics and Automation (ICRA), May 29 ~ June 3, Singapore, pp. 1386-1393, 2017.

[6] G. E. Pazienza, P. Giangrossi, S. Tortella, M. Balsi and X. Vilasis-

Cardona, "Tracking for a CNN guided robot," Proceedings of the 2005 European Conference on Circuit Theory and Design, 2005.
[7] E. Martinson and V. Yalla, "Real-time human detection for robots using CNN with a feature-based layered pre-filter," 25th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN), New York, USA,pp. 1120-1125, 2016.
[8] X. Peng, B. Sun, K. Ali and K. Saenko, "Learning deep object detectors from 3D models," IEEE International Conference on Computer Vision (ICCV), Santiago, pp. 1278-1286, 2015.
[9] S. Ren, K. He, R. Girshick and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal mnetworks," IEEE Transactions on Pattern Analysis and Machine Intelligence, 39 (6), pp.1137-1149, 2017.
[10] E. Shelhamer, J. Long and T. Darrell, "Fully convolutional networks for semantic segmentation," IEEE Transactions on Pattern Analysis and Machine Intelligence, 39 (4), pp. 640-651, 2017.
[11] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You only look once: Unified, real-time object detection," IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 779-788, 2016.
[12] Intel® RealSense™ Technology, "Intel® RealSense™ SDK", Revised Jun 2016.
[13] Ning Qian, "On the momentum term in gradient descent learning algorithms," Neural networks, 12(1), pp. 145–151, 1999.
[14] Sergey Ioffe and Christian Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," arXiv preprint arXiv:1502.03167v3, 2015.
[15] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, pp. 6517-6525, 2017.
[16] Li, Zhizhong, and Derek Hoiem, "Learning without forgetting," IEEE Transactions on Pattern Analysis and Machine Intelligence, 40(12), pp. 2935-2947, 2017.
[17] S. Suzuki and K. Abe, "Topological structural analysis of Digitized binaryimages by border following," Computer vision, graphics, and image processing 30, pp. 32-46, 1985.
[18] J. Redmon. Darknet: Open source neural networks in c. http://pjreddie.com/darknet/, 2013–2016
[19] Mark Everingham, Luc Gool, Christopher K. Williams, John Winn and Andrew Zisserman, "The Pascal visual object classes (VOC) Challenge," Int. J. Comput. Vision 88(2), pp. 303-308, 2010.