# Real Time Object Position Estimation by Convolutional Neural Networks

Thibault BARBIÉ, Ryodo TANAKA, Ryo KABUTAN, and Takeshi NISHIDA

***Abstract*— We consider the object grasping problem by an industrial robot; and we propose a method where the robot learns how to detect and localize the object with respect to the frame of the robot in real time based on a small data set by using infrared images. In this research, we use a robot handling system that consists of a seven degree of freedom industrial robot and a three-fingered gripper. The target object and the surroundings of the robot are measured by multiple RGB-D cameras. The detection and localization of the object is learned by several convolutional neural networks (CNNs). Consequently, we develop a real time system that successfully localizes the object with a small amount of training data.**

## I. INTRODUCTION

### A. Background

One of the current industrial problems is finding a method to increase the robustness of a picking system, *e.g.*, the Amazon picking challenge [1], which is a contest that aims to overcome this problem. In actual product lines, most of the engineers have adopted the teaching-play-back method to program industrial robots. However, this method is not robust when there is a change in the targets and/or environment. Generally, to construct a robust system many sensors such as vision sensors and ranging sensors are installed in the line system. However, the installation of many sensors causes high economical and computational costs. Moreover, recently, automation systems based on three-dimensional (3-D) computer-aided design (CAD) modeling have been developed and adopted by many companies. However, these systems also require many sensors to adjust with the real environment and the computational and virtual world. To attain a simple, robust, and easy-to-use sensor system, it is essential that industrial robots possess higher intelligence. Therefore, this research is focused on developing an intelligent robot system capable of grasping a target using a unique sensor.

### B. Focus of research

In recent years, convolutional neural networks (CNN) with deep learning and their high recognition ability have been gaining popularity [4]. However, the structure of the best state-of-the-art networks is very complex and huge, *e.g.* the AlexNet network which served as a basis for numerous recent researches has over sixty million parameters [5]. In industrial applications, large training data set and longer training duration results in an increased initial cost for its introduction. In general, when the size of the CNN becomes large, the recognition capability is improved. On the other hand, the amount of data for training the number of connections between the layers, and the computational costs are increased exponentially. In actual industrial applications, it is difficult to collect large number of training data sets. Therefore, there is demand to build a CNN that is as small as possible. In this research, we focus on a part of industrial robot's task and construct a small and reasonable CNN scheme.

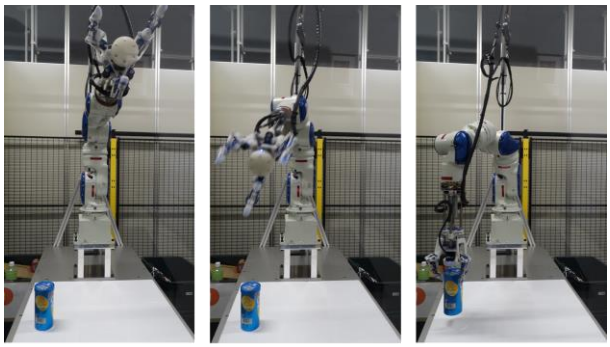


Fig. 1. Picking robot system process. The robot moves above the estimated position of the target, decreases its height, grasps the object and moves up: (a) home position, (b) motion, and (c) grasping of the target.

## II. METHOD

The industrial robotic grasping task can broadly be categorized into four parts [2]: detection and localization of the object, grasping plan generation, trajectory plan generation, and motion to the target. These tasks called "pick and place" are general robotic tasks. The overview of a pick and place system used in this research is shown in Fig. 1. In this research, we focus on the detection and localization of the object, and construct CNNs for the estimation of the position of the target. We chose to estimate only the $x$ and $y$ coordinates, the frame used in this research is shown in Fig. 2. In our developed system, the CNNs are trained by a depth camera and are used for the recognition of the position by using monocular infrared (IR) camera. The depth measurement and IR imaging can be achieved by a Kinect sensor.

For the CNNs implementation, the training phase and the recognition phase are executed sequentially. In general, a huge amount of data containing paired inputs and outputs is required for the training phase. The collection and preparation of this training dataset is a highly time-consuming process for engineers. Therefore, to reduce the training data size, we adopted low resolution input data, *i.e.*, an IR image with 76×76 pixels to detect and estimate the position of target objects. The

T. Barbié is with ISAE-Supaero, Toulouse, France and Kyushu Institute of Technology Japan, R. Tanaka, R. Kabutan, and T. Nishida are with Kyushu Institute of Technology, Kitakyushu, Fukuoka, Japan.

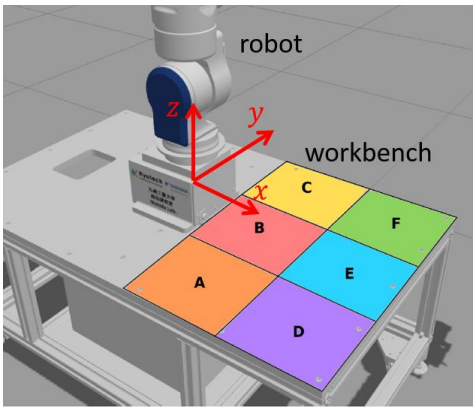


Fig. 2 Division of the picking area on the workbnech into six areas.

IR image is also robust against the change of brightness. Moreover, to reduce the size of the CNNs, we used multiple CNNs for each parameter estimation in parallel, and the outputs of CNNs are integrated and given to the robot controller. We used only a few hundreds of examples to create the data set which is evidently a huge challenge as it forces the CNNs to be able to generalize well.

We focused on the estimation of the position of the object to be grasped with three conditions:

1) Using a single IR image with 76×76 pixels as input.
2) Using a small training data set of 240 examples.
3) Providing a fast pose estimation of at least 100 Hz.

Consequently the proposed method achieved the following abilities:

- Accuracy is less than 21.4 mm including sensor noise.
- Classification rate is 93 %.

We built a robot handling system that is consisted of the industrial robotic arm: *Motoman* which has seven degrees of freedom and a three-fingered gripper called *D-Hand* [2]. The target object and the surroundings of the robot are measured by a *Microsoft Kinect Ver.2* RGB-D camera.

## III. CNN

CNNs are used for the detection and localization of the objects. The picking area on the workbench has been divided into six areas (named as $A, B, C, D, E, F$), and an overview is shown in Fig. 2 We divided the experiment table into six areas instead of training the CNNs on the entire picking area because we obtained better results as observed from experiments. The target is first detected and classified in one of the six area by a CNN classifier; the target position is then estimated by two CNNs (one for *x* and one for *y*). Fig. 3 shows the entire process and display of the inner architecture of the CNNs. For each area two CNNs are built, meaning that we have 12 CNNs to estimate the position. The recognition rates and hold success rates on each area are evaluated.

### A. The training dataset

In our research a training example is constituted of an IR image and a pose estimation measured by the Kinect. The originally captured IR image has 256×256 pixels; it is then cropped and rescaled to a 76×76 pixels image, named as $\boldsymbol{r}_i \in \mathbb{R}^{76\times76}$. In this notation $i$ refers to the index of the image. The

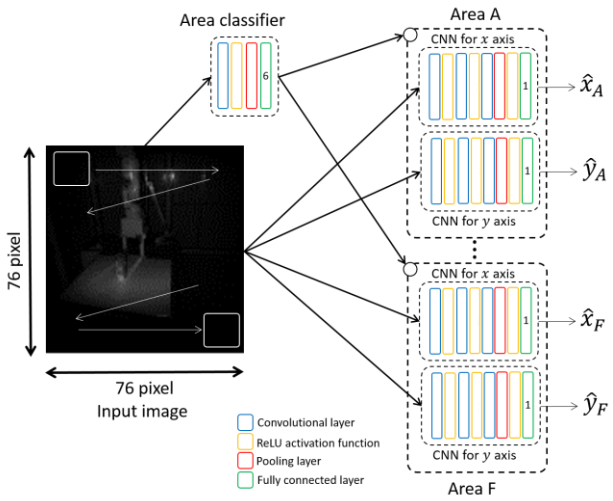


Fig. 3 Structure of the proposed high-throughput CNN.

output is the center position $\boldsymbol{x}_i \triangleq [x_i \quad y_i]^T$ of the target on the workbench with respect to the frame of the robot. A training example is denoted by $\boldsymbol{p}_i \triangleq \{\boldsymbol{r}_i, \boldsymbol{x}_i\}$. Note that the $\boldsymbol{x}_i$ include measurement errors because it was collected by using a merging algorithm on the point cloud data measured by the Kinect.

Furthermore, data augmentation was performed to reduce overfitting. Instead of cropping the initial image each time in the same region of interest we slide the window over by one pixel up and by one pixel right. This general trick was sufficient to multiply our dataset size by a factor 4.

The objects we used were cylinders of 170 mm in height and 50 mm in diameter. We separated the workbench in six areas and created 40 training examples in each of them. For each area, we randomly placed the object and tried to cover the maximum number of positions. The resulting datasets $P_{\mathrm{m}} \triangleq \{\boldsymbol{p}_i | i = 1, \cdots, 160\}$, where $m = \{A, B, C, D, E, F\}$ are sets containing 160 (40×4) data. We define $\Delta = \cup P_m$ as the whole dataset which contains 960 training examples.

### B. The Classifier

A classifier before the position estimator is adopted to allow specialization of the estimators in each area. The size of the dataset $\Delta$ can then be reduced compared to the one needed if we used one large estimator for the entire workbench. We used the whole dataset $\Delta$ to train the classifier. It was randomly divided into 800 training examples for the training set and 160 training examples for the validation test. The CNN classifier is composed of a single convolution layer with two filters of size 3×3 followed by a pooling layer (to reduce the size of the image to a 38×38 pixels image) and a rectified linear units (ReLU) activation function. The result was feeding a fully connected layer of six neurons giving the softmax probability of the object being in one of the area. For the training, we used stochastic gradient descent with a mini-batch of size 200.

### C. Position estimator

After the area of the target is detected, its exact position is estimated by the CNNs responsible for that area. For this purpose, different CNNs for each area were trained on the suitable $P_m$ dataset of size 160. Each time, the training examples were randomly divided into a training set of size 130 and a validation set of size 30. We have a CNN for each coordinate, *i.e.* one CNN estimates the position on the $x$ coordinate and another one estimates the $y$ coordinate. Thus 12 CNNs are used in this system. All CNNs have the same structure. In order to reduce the training time and the number of training datasets, we chose a relatively shallow CNN architecture. They are comprised of three convolution layers (2, 4, and 2 filters of size 3×3 each), followed by a pooling layer and then a single output neuron. The ReLU activation functions were used after the two first convolution layer and after the pooling layer. For the training, we adopted the Adam optimization method with a mini-batch of size 4.

## IV. Experiments

To test our system we performed three experiments. As our system uses IR picture we used objects of different color. An image of the objects we used could be seen in Fig. 4.

The first experiment was to test the accuracy of the classifier. The picking area on the workbench is divided into six areas, and the classifier should determine the location of the objects. For each area we performed 40 tests. We can observe from Fig. 5 that our classifier has an average success rate of 93%; however the percentages vary between the classified areas. The best classified areas, represented as D and F, which were always correctly classified. The low performing area, E, was wrongly classified most of the time as the object was near the border, which were the areas of F and D.

The objective of the second experiment was to measure the accuracy of our position estimators. We compared the error between the estimation by the CNNs and the estimation by the Kinect for each area. We performed 20 tests for each area. We can observe from Fig. 6 that on an average the maximum error was 21.4 mm and the minimum one was 14.2 mm. It is logical as the localization by Kinect is not precise and is unstable. If one wait to measure the position of an object by the Kinect one would see a range of position varying from about 20 mm. Therefore, it was then expected to not have a perfect accuracy


Fig.4 Object used during the experiments. Because the images taken are infrared images the colors of the objects did not have any effect.

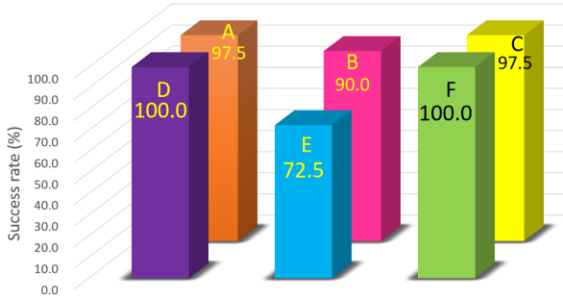


Fig.5 Classification of success rate in percentage with respect to the different areas.


Fig.6 Average error in milimeters between the estimation by the Kinect and the CNNs with respect to the different areas.

in the position estimation. A visualization on the software *Rviz* is shown in Fig. 7.

Therefore, to verify that our CNNs were well trained we performed a third experiment that involved randomly placing the objects anywhere on the picking area and check whether the robot could grasp it. To grasp the objects the robot was first asked to move above the CNNs' estimated position, then decrease its height, close its fingers and ultimately increase its height. If the object did not fall, it was considered as a

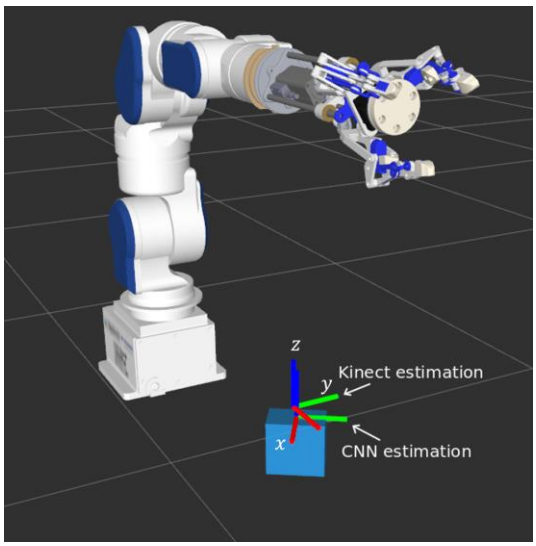


Fig.7 Visualization on *Rviz* of the robot pose, the Kinect position estimation of the object, and the CNN estimation. Since the object is a cylindrical shape , the directions of the $x$ axis and $y$ axis are optional .

successful grasp. We performed this experiment forty times, and all the results were successful grasps, except one area misclassification and two bad position estimation.

## V. Conclusion and Future Work

Our method showed that it is possible with CNNs to estimate the position of an object with respect to the frame of the robot. It allows real time detection and estimation of the object to be grasped. Moreover, we used a considerably small number of images to train the CNNs, which indicates good generalization property of the networks, and allow the creation of a dataset in small duration. As the resulting CNNs are considerably small, the speed of estimation is extremely high. We did not perform precise measurements but in practice we were doing estimation as a 100 Hz frequency showing that our system is suitable for real time applications.

However, we always used the same shape for the target objects. We tried to estimate the position of other objects with different shapes and the results were acceptable (approximately 50mm errors) given that the networks had never seen them before. We are planning to extend this work to check whether it can be generalized to multiple shape object detection and localization.

Furthermore, instead of manually creating the dataset we will investigate the possibility to perform automatic dataset generation using the robot itself. Indeed, if there is a precise but slow localization method then the robot could grasp the object, randomly place it on the workbench and hence create a new training example for the CNN. The resulting neural networks will then replace the localization method to increase the speed of the localization system.

In addition, we were detecting and estimating the position of only one object at a time. It is a difficult problem to create a training dataset containing multiples objects. Another interesting problem that arises is the management of network outputs, considering that it cannot be changed.

Finally, we focused only on the estimation of $x$ and $y$, we did not estimate the height and orientation of the object. We want to continue this work further to allow the networks to perform it.

## References


[1] 2016 Amazon Picking Challenge Official Rules, " http://amazonpickingchallenge.org/APC_2016_Official_Rules.pdf", 20016.

[2] J Bohg A. Morales,T. Asfour, D. Kragic, "Data-driven grasp synthesis—a survey," in *IEEE Transactions on Robotics*, vol. 30, 2014, pp. 289-209.

[3] R. Kabutan and T. Nishida, "Development of robotic intelligent space using multiple rgb-d cameras for industrial robots" *Unpublished*, 2016

[4] Levine, Sergey, et al. "Learning Hand-Eye Coordination for Robotic Grasping with Deep Learning and Large-Scale Data Collection." *arXiv preprint arXiv:1603.02199* (2016).

[5] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." *Advances in neural information processing systems*. 2012.