End-to-End Learning of Object Grasp Poses in the Amazon Robotics Challenge

Eiichi Matsumoto*, Masaki Saito*, Ayaka Kume and Jethro Tan

***Abstract*— The Amazon Robotics Challenge (ARC) is a robotics competition aimed to advance warehouse automation. One of the engineering challenges is making the system robust to and being able to handle a wide variety of objects, as would be the case in a real warehouse. In this paper, we shortly describe our system used in ARC featuring a method to obtain object grasp poses containing the location of the object as well as orientation for the grasp by using a convolutional neural network with an RGB-D image as input. Through our entry in ARC 2016, we show the effectiveness of our method and the robustness of our network model to a large variety of object types in dense and unstructured environments wherein occlusions are possible.**

I. Introduction

The Amazon Robotics Challenge[1] (ARC)—formerly known as Amazon Picking Challenge, is an annual challenge issued by Amazon that is increasingly difficult. In this challenge, teams from both academia and the industry can enter to create the ultimate robotic system to solve what is considered by Amazon the final engineering challenges found in warehouse automation. Many decisions have to be taken on what approach to take for each subproblem when building such a robotic system for this task, which involves trade offs and compromises given the limited amount of time and the resources each team has after receiving the detailed task specification. The 'pick task' finals this year was thrilling as its winner was chosen by watching replays of their runs for the fastest cycle time after a tied best score between Team Delft and our team (Team PFN).

In this article, we briefly present our robotic system used in ARC 2016 that ended up placing second on the 'pick task' and fourth on the 'stow task'. In particular, we would like to highlight our vision subsystem, which includes the main contribution of this paper: *end-to-end learning of object grasp poses*. We leverage convolutional neural networks (CNN) to not only provide semantic segmentations for each object, but also predict object grasp poses. This allows us to avoid the use of computationally intensive 6D object pose estimation by e.g. point cloud registration and speed up the development process of our system.

*Contributed equally in this work.

Preferred Networks Inc., Tokyo, Japan, {matsumoto, msaito, kume, jettan}@preferred.jp

The authors would like to thank Tobias Pfeiffer, Taizan Yonetsuji, Yasunori Kamiya, Ryosuke Okuta, Keigo Kawaai, Daisuke Okanohara for their contribution and work done in the Amazon Robotics Challenge as part of Team PFN, and FANUC Corporation of Japan for providing the robotic arms and technical support.

[1]https://www.amazonrobotics.com/#/roboticschallenge

More information, source code used to run our system including scripts to train the vision subsystem as well as detailed models and images of the objects present in ARC 2016 can be found via [1].

II. Related Work

A. Amazon Robotics Challenge

In ARC, team entries are evaluated which includes approaches to object grasping in an unstructured environment. The latest ARC victors Team Delft [2] made use of a region-based convolutional neural network (CNN) to provide bounding box proposals for each object before executing a global point cloud registration algorithm to estimate the 6D pose of the objects. Based on this pose estimation, grasp strategies are determined in combination with handcrafted heuristics. Zeng *et al.* [3] developed a self-supervised deep learning system used in Team MIT-Princeton's entry that takes advantage of multi-view RGB-D data of the objects to obtain an initial pose for the iterative closest point algorithm. In [4], Schwarz *et al.* presents evaluation of Team NimbRo's system during ARC 2016, which incorporates a perception system that makes use of CNN for both object detection and segmentation. For an overview of ARC 2015, we refer to [5] in which Correl *et al.* collected lessons learned from all team entries during that year, and to [6] wherein Eppner *et al.* described Team RBO's winning entry.

B. Object Grasp Pose Synthesis

A deep neural network model with two stages to predict and score grasp configurations is proposed in [7]. [8] improves this work by doing this in an end-to-end manner, while Nguyen *et al.* uses a semantic segmentation based model to output object affordances in [9]. However, these methods are only tested in simple environments where objects are neither occluded nor occluding each other. On the other hand, self supervising methods which for picking in unstructured environments have been proposed in [10], [11]. The biggest disadvantage of these methods is the training time they need.

III. System Overview

A. System Setup

Our system[2], shown in Figure 1, consists of two stationary FANUC M-10iA industrial robot arms, which were each equipped with different end-effectors to handle different objects. Like the previous winner of ARC, we observed that

[2]A video showing our system in action is accessible via *goo.gl/ZavxX1*

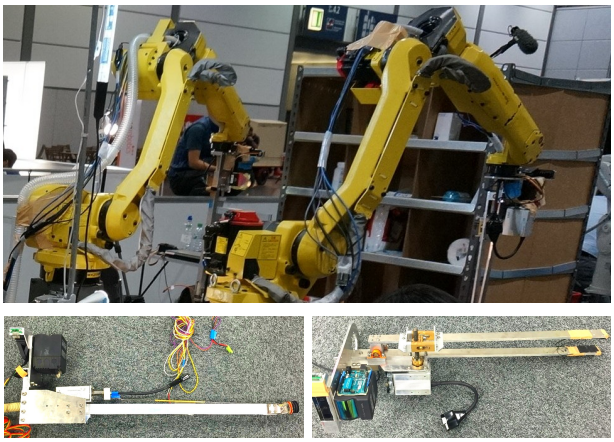


Fig. 1. Team PFN's robotic system entry in the Amazon Robotics Challenge 2016. The right arm had a pinch-gripper, while the left arm was equipped with a vacuum gripper, of which the tip can be actuated to rotate to an angle of 0, 45, or 90 degrees. Both grippers had a length of 60 cm with 40 cm reach and were equipped with an Intel Realsense SR300 RGB-D camera, a Nippon Signal FX-8 3D laser scanner, and a VL6180X proximity sensor.

a vacuum gripper was capable of handling the majority of the items. Therefore, we equipped the left arm with a vacuum gripper connected to a commercially available off-the-shelf Hitachi CV-G1200 vacuum cleaner allowing us to handle 37 out of 39 object types. However, an additional grasp method was needed due the mesh texture of the pencil cup and the weight of the dumbbell. Hence, we deployed our second arm with a pinch-gripper to deal with these items and achieved full type coverage on all the objects with our system.

To control our system and process data, we used a laptop equipped with an Nvidia GeForce GTX 870M GPU and an Intel i7 6700HQ CPU running Ubuntu 14.04 with ROS Indigo. Additionally, two PCs running Microsoft Windows were used to interface the system to the Intel Realsense cameras and the Nippon Signal FX-8 laser scanners on each end-effector. Furthermore, four Arduino Uno boards connected to the VL6180X proximity sensors, a Bosch BME 280 environmental sensor, and the valves and the relay controlling the vacuum were also present in out system. The proximity sensors, which provided redundancy to the cameras and the laser scanners, were used to perform calibration at the very beginning of both the picking and the stowing tasks to calculate the offset and orientation of the shelf relative to the system.

B. Process Flow

To process tasks in our system, we use the 'sense-plan-act' paradigm in combination with a global task planner, which puts the system into a certain state. For motions, we have defined joint space positions for the robots to capture the scene of the tote and the bins. Free space motions from and to these positions, as well as Cartesian space motions are commanded from the laptop to the robot controllers, which are responsible for the motion planning, generation, and execution. Because of the many heuristics present in our system, we will not describe them in this paper and refer to our source code [1].

Depending on the current state of the system and the given task, i.e., pick or stow, our task planner decides a next target using object-specific heuristics, such as size or weight. Afterwards, the robot moves in front of the location containing the target item to capture images of the scene from multiple directions. These images are then forwarded to our CNN—described in Section IV, which in turn provides segmentations for all detected objects and a map with ranked grasp candidates (object type, grasp pose, surface normal, prediction score) in the scene for all images in the local (i.e. the bin or tote with the target item) coordinate system. Grasp candidates are filtered out from this map if either (1) the prediction score is below a threshold, (2) the surface normals of its surrounding pixels are not consistent, or (3) the robot position is not safe because of predicted collision with the shelf or tote. Based on these candidates, our task planner then decides to either (a) pick the target item, (b) request coordinates for a grasp candidate of an item occluding the target item to move the occluding item to another location, or (c) completely give up on picking objects from the current location. To detect whether a grasp of an object has succeeded, we utilize the environment sensor described in Section III-A to measure the pressure of the vacuum. If failure is detected, we retry the grasp up to four times using slightly different positions and orientations. An exception to this process is when the pencil cup or the dumbbell is selected as target item, for which their grasp positions are determined using the iterative closest point algorithm found in PCL (Point Cloud Library).

IV. Vision Subsystem

The problem definition in ARC is noteworthy in how objects were known beforehand. This not only allowed exploitation of object-specific heuristics, but also simplified the object recognition subproblem.

A. Model Overview

Figure 2 shows the architecture of the neural network used in our vision subsystem. In this network, the input is a four-channel RGB-D image with resolution of 320×240 pixels, while the output consists of two results, one for the semantic segmentation indicating 40 object classes (39 items and the background), and one for a confidence map with grasp poses in the coordinate system of the robots. Although our network was inspired by an existing fully convolutional encoder–decoder network [12] that consists of a "encoder" network with convolutional layers and a "decoder" network with deconvolutional layers, several settings are different. Specifically, we did not use any max pooling and unpooling layers, but instead employed convolutional and deconvolutional layers with stride 2 and kernel size 4. All layers in the encoder have ReLU activation, while all layers in the decoder have Leaky ReLU as activation. As with the original encoder–decoder network, we apply the batch normalization layers after the convolutional layers.

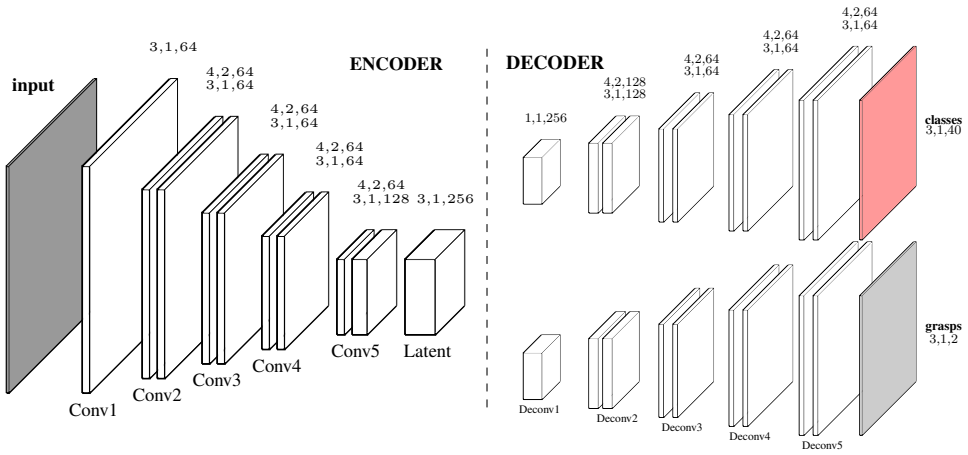


Fig. 2. Layer architecture of our CNN. The output of the encoder is connected to two identical decoders for outputting the semantic segmentation and graspability respectively. The parameters in the convolutional and the deconvolutional layers are denoted as (kernel size), (stride), (output channels).

(a) (b)

Fig. 3. Example of images in dataset used for training. (a) CG image with Blender models of items, and (b) image of an actual scene with real items.

B. Data Collection

Using the robot arm, we have collected about 1,500 images of all the bins and tote containing up to 12 items in a cluttered condition to simulate the competition scene. Although this number is relatively smaller than used in literature about CNN in general image recognition tasks, we found the output of the trained model to be workable for performing semantic segmentation on known objects. Moreover, we manually annotated the ground truth, i.e., per-object semantic segmentations and grasp positions ourselves with an in-house developed annotation tool to equalize the quality of annotation. Data collection and annotation took about two weeks in total.

C. Pre-training by CG images

In order to improve the accuracy, we first pre-trained our model with an artificially generated CG dataset containing 100,000 images representing scenes from the tote. For the items in these images, we created 39 textured models with Blender, which allowed us to obtain the annotation of each CG scene. Despite of these images being generated, we found that the quality of these images are comparable to images taken from the actual scene with real items, see Figure 3. However, we empirically observed that the accuracy increase of the model is negligible, and therefore conclude that the performance gained by pre-training on CG images is not worth its effort. We conjecture that this is caused by the discrepancy of information in the depth channel between the CG images and images acquired from sensors in our system. We note that further analysis and experiments should be done to confirm this discrepancy and alleviate it by e.g. adding noise to the depth channel of the CG images.

D. Training

For training, we initialized the weights in the encoder with HeNormal [13] and the decoder with zero centered normal distribution with $\sigma^2 - 0.02$. As optimizer, we used Adam with learning rate $\alpha = 0.001$. As loss function, we employed the sum of the two loss functions defined by $\mathcal{L} = \mathcal{L}_{\mathrm{cls}} + \mathcal{L}_{\mathrm{pos}}$, where $\mathcal{L}_{\mathrm{cls}}$ denotes the softmax cross entropy loss representing the object class for each pixel, and $\mathcal{L}_{\mathrm{pos}}$ denotes the binary cross entropy loss representing the grasp pose. During training, we observed that segmentation by the model trained with the default softmax cross entropy loss tends to ignore small objects such as the scissors. Therefore, we balanced the weight by multiplying it proportional to the reciprocal of the area: $\mathcal{L}_{\mathrm{cls}} = \frac{N}{C} \sum A_i^{-1} \mathcal{L}_{\mathrm{cls}}^i$, where N indicates the total number of pixels in the dataset, C the number of classes, A_i the number of pixels registered to class i in the dataset, and $\mathcal{L}_{\mathrm{cls}}^i$ the softmax cross entropy loss for the class i.

Because of the inconsistency in image resolutions in our dataset, we resized all images to 320×240 pixels before performing random cropping to 224×224 pixels for training. The entire training process took two days on an Nvidia Titan X (Maxwell) GPU.

E. Inference

Similar to the training process, we resize retrieved RGB-D images to 320×240 pixels when inferring our network. Moreover, because the objects types in all locations are known, channels for object types that are not in a specific location are ignored.

V. Results

In preparation for ARC, we have evaluated the effect of the weight balancing described in Section IV-D as well as using the depth channel in our network model. Results of

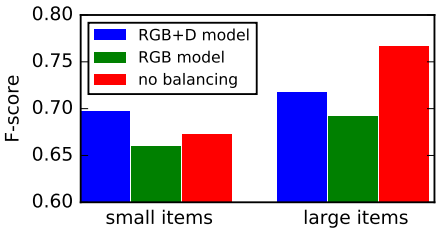
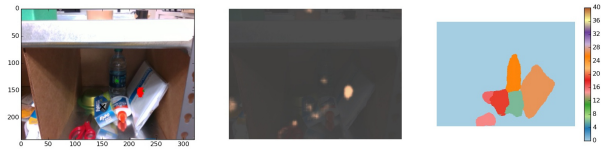
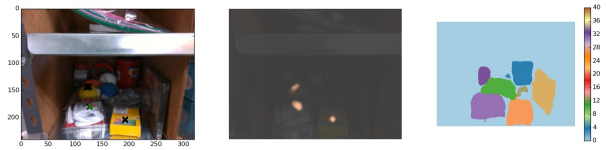


Fig. 4. Evaluation of semantic segmentation for variants of our model. The 'RGB model' was trained without depth channel information, while the 'no balancing model' did not use weight balancing for the object area size. Small items and large items are categorized according to an object's volume being lower or higher than $400\,\mathrm{cm}^3$. F-scores are evaluated with a test dataset containing over 100 images in total.



(a)



(b)

Fig. 5. Example of outputs produced by our network model containing (from left to right): (i) a color image with the selected grasp, (ii) depth image with all available grasp candidates, and (iii) semantic segmentation of objects in the scene.

our experiments can be found Figure 4. We observe that using depth information in addition to the RGB channels significantly improved the overall estimation accuracy of our model. Furthermore, our final model trained with weight balancing enabled outperformed our trained model without balancing in usefulness during the competition, because it contributed towards more reliable semantic segmentation for the small objects in ARC, albeit lowering the F-score for the large objects. The results of our final trained model are shown in Figure 5, demonstrating accurate semantic segmentation of objects in dense, unstructured scenes. We notice that the grasp candidates are scored with consideration to the manipulation complexity of objects. For example, although the plastic coffee jar is correctly detected in the semantic segmentation results of Figure 5, no grasp candidates are proposed for this item in the grasp candidates map. Taking 200 milliseconds to output these results, our vision subsystem was able to give us competitive advantage in data processing compared to the top scoring teams, as shown in Table I.

TABLE I

COMPARISON OF OBJECT GRASP POSE GENERATION TIMES

Team	Delft[1]	NimbRo Picking[2]	MIT-Princeton	PFN
Time (sec.)	5 – 40	0.9 – 3.3	10 – 15	0.2

[1] Derived from [2] and analyzing the video linked in [2].
[2] Time is excluding ICP for 6D pose estimation.

VI. FUTURE WORK

With the lessons learned during the competition, we note that many improvements can still be made to our system in order for it to be applicable in a real warehouse. In order to use our vision subsystem to its full extent, we have spent about two weeks to collect and annotate data. By applying techniques such as semi-supervised or weakly supervised learning, we can improve the scalability of our system to more object types as well as its robustness to unknown items. Furthermore, while our model was able to predict grasp poses for our vacuum gripper, we argue that a new method is needed for data collection and training in order for it to become applicable to more complex (e.g. higher DOF) end-effectors. Other functionalities requiring consideration include addition of visuomotor capabilities to the system in order to deal with complex object manipulation.

VII. CONCLUSION

In this paper, we briefly described our system entry to ARC 2016 and proposed a vision subsystem using CNN to enable end-to-end learning to predict object grasp poses from RGB-D images. Thanks to the generalization abilities of deep learning models, our trained model was able to successfully produce semantic segmentation of objects in unstructured environments where items can be occluded, while also outputting grasp poses for these objects within 200 ms. The effectiveness of our method was demonstrated in the competition, where our entry placed among the top scoring teams.

REFERENCES

[1] (2016) Source code of Team PFN's entry in ARC 2016. [Online]. Available: https://github.com/amazon-picking-challenge/team_pfn
[2] C. Hernandez, *et al.*, "Team Delft's Robot Winner of the Amazon Picking Challenge 2016," 2016. [Online]. Available: https://arxiv.org/abs/1610.05514
[3] A. Zeng, *et al.*, "Multi-view Self-supervised Deep Learning for 6D Pose Estimation in the Amazon Picking Challenge," in *Proc. IEEE ICRA*, 2017.
[4] M. Schwarz, *et al.*, "NimbRo Picking: Versatile Part Handling for Warehouse Automation," in *Proc. IEEE ICRA*, 2017.
[5] N. Correll, *et al.*, "Lessons from the Amazon Picking Challenge," 2016. [Online]. Available: http://arxiv.org/abs/1601.05484
[6] C. Eppner, *et al.*, "Lessons from the Amazon Picking Challenge: Four Aspects of Building Robotic Systems," in *Proc. Robotics: Science and Systems*, 2016.
[7] I. Lenz, *et al.*, "Deep learning for detecting robotic grasps," *The International Journal of Robotics Research (IJRR)*, vol. 34, no. 4-5, pp. 705–724, 2015.
[8] D. Guo, *et al.*, "Deep Vision Networks for Real-time Robotic Grasp Detection," *International Journal of Advanced Robotic Systems*, vol. 14, no. 1, 2016.
[9] A. Nguyen, *et al.*, "Detecting object affordances with convolutional neural networks," in *Proc. IEEE IROS*, Oct 2016.
[10] L. Pinto and A. Gupta, "Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours," in *Proc. IEEE ICRA*, 2016.
[11] S. Levine, *et al.*, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *arXiv preprint arXiv:1603.02199*, 2016.
[12] J. Yang, *et al.*, "Object Contour Detection with a Fully Convolutional Encoder-Decoder Network," 2016. [Online]. Available: https://arxiv.org/abs/1603.04530
[13] K. He, *et al.*, "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification," 2015. [Online]. Available: http://arxiv.org/abs/1502.01852