2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)
Beijing, China, November 6-9, 2018

Tool-use Model Considering Tool Selection by a Robot using Deep Learning

Namiko Saito, Kitae Kim, Shingo Murata, Tetsuya Ogata, and Shigeki Sugano

***Abstract*—We propose a tool-use model that can select tools that require neither labeling nor modeling of the environment and actions. With this model, a robot can choose a tool by itself and perform the operation that matches a human command and the environmental situation. To realize this, we use deep learning to train sensory motor data recorded during tool selection and tool use as experienced by a robot. The experience includes two types of selection, namely according to function and according to size, thereby allowing the robot to handle both situations. For evaluation, the robot is required to generate motion either in an untrained situation or using an untrained tool. We confirm that the robot can choose and use a tool that is suitable for achieving the target task.**

I. Introduction

Aging societies constitute a major problem nowadays, one that is expected to be solved in part by using robots to help with daily tasks [1] [2]. The space encountered during daily living is a complex environment in which robots are required to perform various tasks. If robots could use tools like people do, their ability to execute tasks and adapt to their environment would improve substantially. Therefore, much research has been conducted on tool-use by robots. However, in most of that research, the tools to be used, the objects to be manipulated, and the actions to be performed were labeled in advance. For example, Stoytchev et al. [3] labeled tools with colors, Tikhanoff et al. [4] labeled tools with the sounds of their names, and Dehban et al. [5] categorized motion by type. Of the studies that did not require labeling in advance, most did not consider both selecting a tool and grasping it. For example, Nishide et al. [6] and Mar et al. [7] focused on learning the features of tools, but the robot was required to grasp a tool in advance before using it. Takahashi et al. [8] focused on where to grip the tools, but there was only one tool set in front of the robot and the robot did not need to consider tool selection. Therefore, there are two problems, namely (i) robots are hampered when using and operating unlabeled objects and tools or conducting unlabeled motions, and (ii) it is difficult for robots to perform a series of operations from tool selection to task execution by themselves with only one model.

To solve these problems, the present research proposes a tool-use model that does not require labeling and considers end-to-end sequential task execution from tool selection to operation. When a person chooses one tool from several, there are usually two ways of doing so. The first way is to select a tool whose function meets the purpose, and the second way is to select a tool of the appropriate length or size. As an example, consider the case in which a large hammer, a small hammer, and a saw are on offer and one must be selected. If we want to drive in a nail, we choose one of the hammers instead of the saw based on function. If we want to drive the nail into a hard material, we choose the large hammer instead of the small one. Herein, we build a model that allows a robot to consider both situations, namely to choose and use a tool whose function and size both match the target.

The present paper is organized as follows. Section I I describes the method of constructing the proposed tool-use model. Section Ⅲ presents the experimental setup, Section Ⅳ presents the experimental results, and Section V discusses them. Finally, Section Ⅵ concludes the paper.

II. Tool-use Model

In this section, we describe the present research methodology and how to construct the tool-use model, which can perform a series of operations from tool selection to task execution. This is based on a study by Takahashi et al. [8]. We begin by guiding a robot to experience selecting some tools and using them to operate on an object. During the experience, we record sensory motor data that include information about the robot's body diagram and the relationships among its body, the tools, and the objects. In the present research, the sensory data are motor angles and images from a camera. The motor data include information about the physical movement of the arm, and the visual data include feedback information about how the object moves according to its own movement and the tools used. Then, by learning all the data simultaneously, the robot can recognize tool features from the relationships. For example, the robot will associate a rake-shaped tool feature with pulling an object and a stick-shaped tool feature with sliding an object. As for tool length, longer tools have features that

N. Saito, K. Kim, and S. Sugano are with Department of Modern Mechanical Engineering, Waseda University, Tokyo, Japan (email: n_saito@sugano.mech.waseda.ac.jp)

S. Murata is with the National Institute of Informatics, Tokyo, Japan and SOKENDAI (the Graduate University for Advanced Studies), Tokyo, Japan

T. Ogata is with Department of Intermedia Art and Science, Waseda University, Tokyo, Japan, and National Institute of Advanced Industrial Science and Technology (AIST), Tokyo, Japan

978-1-5386-7282-2/18/$31.00 ©2018 IEEE

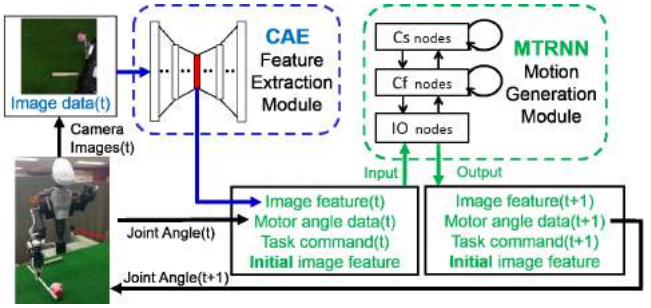

Fig. 1. Proposed model for tool selection and use, considering image data, motor data and task command. First, the feature-extraction module extracts the image features. Then, we integrate the image features, initial image features, motor angle, and task commands. Finally, the motion-generation module trains them to output the data for the next step.

allow them to tackle objects that are farther away and shorter tools have features that allow them to tackle objects that are nearer. In addition, to select a proper tool by itself, the robot must be able to understand what its human operator is telling it to do. To realize this, we issue a task command to the robot. Therefore, the robot must also consider the relationship between the sensory motor data and the task command. Finally, the model generates motion and allows the robot to move properly based on that relationship.

A. Overview of Proposed Model

Figure 1 shows an overview of the proposed learning model. The model comprises a feature-extraction module and a motion-generation module. The former module compresses the number of dimensions, which for image data is considerably larger than it is for other types of data. Therefore, to learn all the data in good balance and with low calculation cost, we use this module to compress the image data. The motion-generation module learns the relationship between the sensory motor data and the task command and generates motion according to that relationship. Simultaneously, the time series of image features, initial image features, motor angle data, and task command are integrated and learned with the module. The reason for using the initial image features is that by continuing to input information about the initial image, we can prevent lack of information about the environment even if the tool or object cannot be seen because it is occluded by the robot arm. That is similar to remembering what kinds of tools are set and where the object is set. Levine et al. [9] also used this method of providing initial image information to grasp an object when equipped with only an RGB camera.

B. Feature-extraction Module

The feature-extraction module compresses the raw image data and extracts low-dimensional image features. For the module, we use a convolutional auto-encoder (CAE) [10]. The CAE is a multi-layered neural network that has convolutional layers and fully connected layers and can be viewed as being a combination of a convolutional neural network (CNN) [11] and an auto-encoder (AE) [12]. Because CNNs have shown high performance in the field of image recognition, we also use them in the present study. The input data pass through the middle layer with the fewest nodes, which then outputs the data with the original number of dimensions again. By training the module so that the output restores the input image, the image features can be extracted low-dimensionally from the middle-layer nodes. We use a sigmoid function as the activation function for the middle layer only; for all other layers we use the ReLU function. Its high performance on general tasks allows the features of an unknown environment to be represented.

C. Motion-generation Module

For the motion-generation module, we use a multiple-timescale recurrent neural network (MTRNN) [13], a type of recurrent neural network (RNN) that can predict and generate the next step given the current state. Unlike a conventional RNN, the MTRNN comprises three types of node that differ according to their time constants, namely slow-context (Cs) nodes, fast-context (Cf) nodes, and input–output (IO) nodes. With their large time constant, Cs nodes are expected to learn the data sequence, whereas Cf nodes with their small time constant are expected to learn the detailed motion primitives. This allows learning the long-term dynamics of the time-series data associated with the primitives.

In forward calculation of the MTRNN, the output values are computed. First, the internal value u_i of neuron i at step t is calculated as

$$u_i(t) = \left(1 - \frac{1}{\tau_i}\right) u_i(t-1) + \frac{1}{\tau_i}\left[\sum_{j \in N} w_{ij} x_j(t)\right], \quad (1)$$

where N is the number of neurons connected to neuron i, τ_i is the time constant of neuron i, w_{ij} is the weight value from neuron j to neuron i, and $x_j(t)$ is the input value of neuron i from neuron j. Then, the output value is calculated by

$$y_i(t) = tanh\left(u_i(t)\right). \quad (2)$$

The value of $y_i(t)$ is used as the next input value as

$$x_i(t) = \begin{cases} \alpha \times y_i(t-1) + (1-\alpha) \times T_i(t-1) & i \in IO \\ y_i(t-1) & otherwise \end{cases} \quad (3)$$

where $0 \leq \alpha \leq 1$ is the feedback rate and $T_i(t)$ is input datum i given as part of the teaching data. If neuron i is an IO node, the input value $x_i(t)$ is calculated by multiplying the output of the preceding step $y_i(t-1)$ and the teaching datum $T_i(t)$ by the feedback rate α. By contrast, for Cs and Cf nodes, the previous output value is used as input. We can adjust the input data by means of the feedback rate α. If we set the feedback rate to unity, the model predicts the next step from the current step,

which is known as closed-loop prediction. With closed-loop prediction, given only initial information, the robot can associate the following values and generate a series of actions. It is as if the robot, provided with only the initial data, can "close its eyes" and imagine what will happen next without needing to see any movement. By contrast, if we set the feedback rate to zero, the model predicts the next step from the current step of the teacher data, which is known as open-loop prediction. In addition, with open-loop prediction, the robot can also generate motion if provided with current input data other than the teaching data. Therefore, with this method, the robot repeatedly predicts images and generates the motion of the next step from the current situation.

In backward calculation, we use the back propagation through time (BPTT) algorithm [14] to minimize the training error given by (4), and the weight is updated as (5):

$$E = \sum_{i} \sum_{i \in IO} \left(y_i(t-1) - T_i(t) \right)^2, \tag{4}$$

$$w_{ij}^{n+1} = w_{ij}^{n} - \eta \frac{\partial E}{\partial w_{ij}^{n}}, \tag{5}$$

where η is the learning rate and n is the number of iterations.

III. Experimental Setup

A. Task Design

We used the NEXTAGE humanoid robot developed by Kawada Robotics [15], guiding it to try some tasks with its right arm, which has six degrees of freedom and one gripper. For the model to consider two types of selection, namely according to function and according to length, we prepared two stick-shaped tools and two rake-shaped tools for training as shown in Fig. 2 (left). We also prepared a stick-type tool for testing, a screwdriver, as shown in Fig. 2 (right). All tasks were conducted using the pig-shaped object shown in Fig. 2 (left); the stick-shaped tools were used to slide the object whereas the rake-shaped tools were used to pull the object in front. As shown in Fig. 3, in each task two tools were placed in front of the robot and the robot had to select one of them. As shown in Fig. 4, there were three possible initial tool positions and two possible initial object positions. At this time, of the two tools to be placed, the one that did not match the task execution was placed in the middle position. We set a total of 72 tasks (12×6; combination patterns of tool setting $\times$ positions of tools), of which we used 48 for training and 24 for testing. As for the task command, we provided information to the robot via two parameters: the first parameter expressed the "pull" command and the second parameter expressed the "slide" command. Before step 10, one of the two command parameters was set to 0.9 according to the demanding task. After step 10, we returned both parameters to zero. In this manner, we set the system to give a command only

Fig. 2. Left: stick-shaped and rake-shaped tools and the object for training. Right: stick-shaped tool for testing.

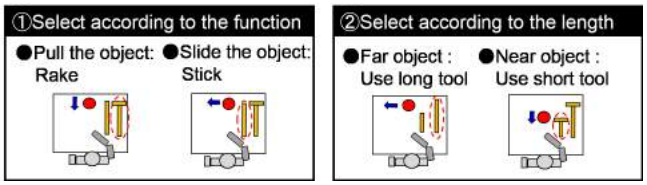

Fig. 3. Experimental setting. Two tools and the object are placed in front of the robot in each task, and we guide the robot to do sliding or pulling actions. With this setting, we allow the robot to consider two types of selection, namely according to function and according to length.

at the beginning of the tasks. Finally, the robot began moving from a fixed initial position in every task.

B. Training Setting

Each of the 72 tasks took between 11.0 s and 18.5 s to complete. By holding the robot in its final position for a while after task completion, we recorded each set of sensory motor data for the common time of 19.5 s. The sensory motor data of the 72 tasks were recorded every 0.1 s, thus we sampled 195 steps each. At this time, we moved the robot remotely and took the data. The image data taken by a camera mounted on NEXTAGE had 12,288 dimensions ($64 \times 64 \times 3$; width $\times$ height $\times$ channel). This high-dimensional visual information was compressed to 10 dimensions by the CAE. The CAE structure is shown in Fig. 5. The extracted 10-dimensional image features and initial image features, the seven-dimensional joint-angle data, and the two-dimensional task-command data were put into the MTRNN. The values of all the data were normalized to $[-0.9, 0.9]$. Table I details

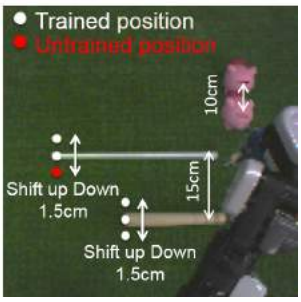

Fig. 4. Initial positions of tools and object. There are three possible initial positions for each tool, all 1.5 cm apart, and two possible initial positions for the object, both 10 cm apart.

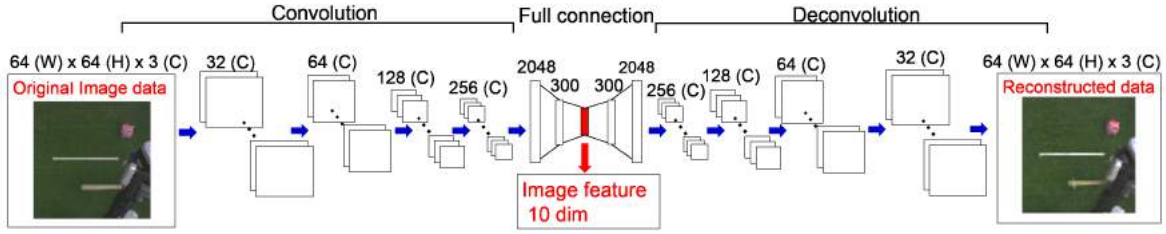

Fig. 5. Structure of convolutional auto-encoder (CAE) feature-extraction module. This module compresses 12,288-dimensional data and extracts 10-dimensional image features.

TABLE I
STRUCTURE OF MTRNN

Node name	Number of nodes	Time constant
IO nodes	29	1
Cf nodes	120	10
Cs nodes	10	40

TABLE II
SETTINGS OF MTRNN INPUT DATA

Input	Feedback rate
Motor angle	1.0
Task command	0.0
Initial image feature	0.0
Image feature	0.2

the MTRNN structure. For the input, we modified the closed- or open-loop feedback rate α as given in Table II. Regarding the motor angle, the next position had to be decided with reference to the previous position; we set the rate to generate motion with closed-loop prediction. Because the task command and the initial image features were all fixed values, the prediction was set to open loop and the fixed values were input each time. To make it possible to adapt to the circumstances on each occasion, image features were predicted with an open loop 80% of the time. To make it possible to predict the next image based on the previous one, 20% of predictions were closed loop.

C. Experimental Evaluation

We conducted the following three experiments to evaluate the generalization ability of the proposed model:

(A) the robot generated motion offline using the 24 untrained data;
(B) the robot generated motion online using tools that were set in the untrained position;
(C) the robot generated motion online using the untrained tool.

The offline method involves using data that are prepared in advance, whereas the online method involves moving the robot while taking current data and using them for immediate prediction. In experiment A, we used the data of 12 tasks in which the tools were untrained combinations and 12 tasks in which the tools were set in the untrained position shown in Fig. 4 with a red dot. In addition, we analyzed the internal values of the motion-generation module to reveal what is represented therein. In experiment B, we assessed whether the robot could handle the tools being in the unlearned position while in motion. Finally, in experiment C, we provided the untrained tool shown in Fig. 2 (right), issued the "pull" task command, and assessed whether the robot could handle this situation while in motion.

IV. RESULTS

We trained the MTRNN with the data of 48 training tasks and performed this 194,000 times, that being the epoch whose error was the least in the 200,000 times of learning. We then conducted the three evaluation experiments using the trained model.

A. Offline Motion Generation

Figure 6 shows the trajectories of the motor angles of (i) the target data prepared in advance and (ii) the generated results. As can be seen, the motor data can be generated along with the target data. The 10 other experiments gave similar results. Hence, all of the tool selections and actions attempted by the robot were correct for the object position and task command. We therefore reason that the robot can learn data and generate motor angle data properly.

We also analyzed the internal values of the model. Figure 7 shows the results of principal component analysis of the values of the Cs nodes at step 10 (Cs(10)) in the MTRNN. Step 10 is when the task commands are turned to zero and comes before grasping a tool, which is around step 50. We can reveal the focus of the robot in a series of tasks by analyzing the Cs node space, which learns the features of sequential data. Consequently, as shown in Fig. 7, the Cs node space expresses the features of the tools to be used. In the figure, solid circles show trained data and hollow circles show test data, changing color according to the target-tool features. The horizontal PC1 axis represents tool shape, while the vertical PC3 axis represents tool length. Different tool features are separated and clustered. We confirmed that the motion-generation module self organizes the features of the target tools.

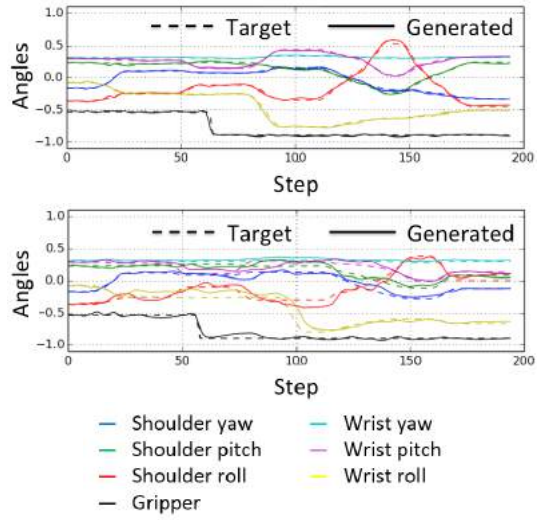

Fig. 6. Trajectories of the six joint angles and one gripper angle of the robot arm. The upper graph shows a set of experimental results for an untrained tool combination. The lower graph shows a set of experimental results for the untrained tool position.

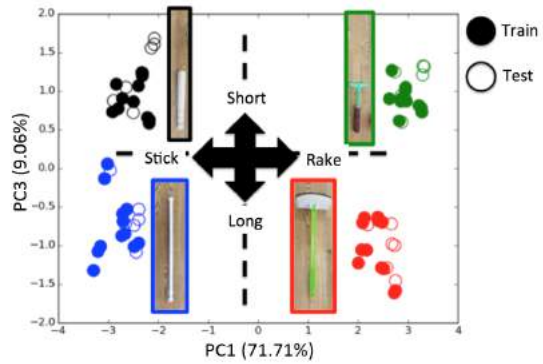

Fig. 7. Results of principal component analysis of internal Cs(10) values of motion-generation module. The color depends on the target tool to be selected for each task. Different features of the target tool are separated and clustered.

B. Online Motion Generation: Untrained Position

We experimented with four task settings shown in Table III to evaluate whether the robot can select according to both function and length. All trials succeeded in online tool selection and task execution. We show one of these trials in Fig. 8. In this experiment, we set the object in the farther-away position and gave the robot the "pull" command. The robot then succeeded in choosing the longer rake and pulling the object as expected.

C. Online Motion Generation: Untrained Tool

Here, we set the object in the nearer position and gave the robot the "slide" command. The results are shown in Fig. 9.The robot chose the shorter stick that was untrained, but the height of the arm was slightly mismatched with respect to the object, and the arm swung out above the object. However, although the height did not match, the selection and the attempted action by the robot were correct. We explain in Section V the reason for this failure.

TABLE III
SETTINGS FOR EXPERIMENT B

Tools (set as upper/lower)	Object position	Command
short stick / long rake	**near**	slide
short rake / long stick	**near**	**pull**
long stick / long rake	**far**	**slide**
short rake / long rake	**far**	**pull**

V. Discussion

With experiment A, we showed that the model was trained correctly and could generate motion. From analyzing the internal values of the motion-generation module, we reason that the robot acquired the features of the target tool from the relationship between the sensory motor data and the task command. In addition, the robot recognized a suitable tool (i.e., one whose features matched the task command and the distance to the object) from the two tools in front of it before grasping one. With experiment B, we showed that motion could be generated even online, and with experiment C we showed that unlearned tools could also be selected and used. All tasks were done in series without using labels, and we showed that the proposed model could solve two problems associated with previous research, namely (i) robots are hampered when using and operating unlabeled objects and tools or conducting unlabeled motions, and (ii) it is difficult for robots to perform a series of operations from tool selection to task execution by themselves with only one model.

However, in experiment C, the robot failed to move the object because of the height of the arm. That problem occurred because it is difficult for the robot to consider three-dimensional space with only one RGB camera. We could solve this problem by installing another camera to consider the height.

VI. Conclusion

Unlike previous research, we constructed a model that does not use labels or models of the environment and actions and that can perform a series of operations from tool selection to task execution. With the model, the robot could acquire the features of tools and select one according to its function or length to match the given command and the environmental situation. To realize the model, we allowed the robot to experience selecting a tool and using it, and we recorded sensory motor data during that time. In addition, we gave the robot task commands, which were integrated with sensory motor data and learned by the model. For evaluation, we conducted three experiments, (i) offline motion generation, (ii) online motion generation with an untrained tool position, and (iii) online generation using an untrained tool. We confirmed the generalization performance of the model and showed that the robot could handle the

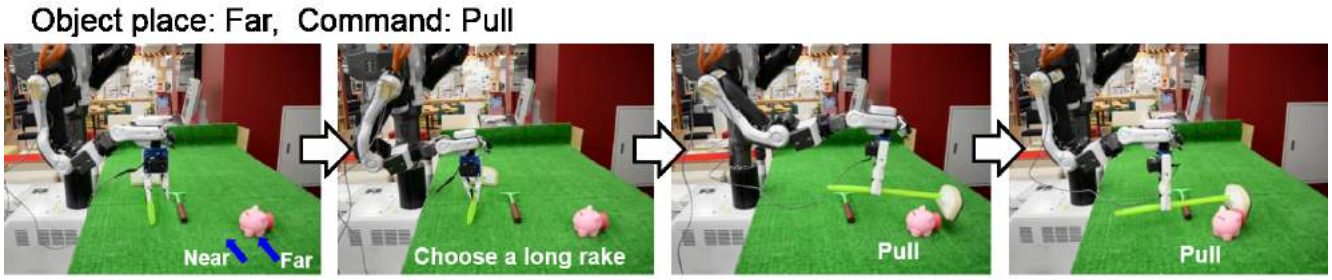

Fig. 8. Sequence of robot movements in experiment with tool placed in untrained position. The robot succeeded in using a long rake-shaped tool to pull the object placed in the farther-away position.

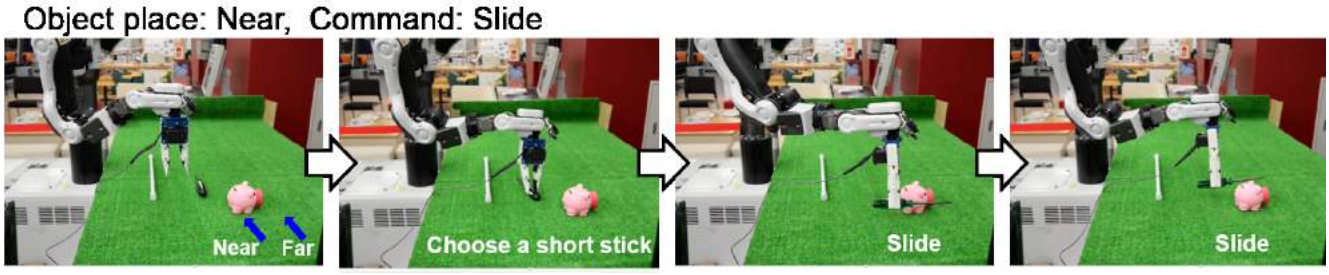

Fig. 9. Sequence of robot movements in experiment with untrained tool. The robot succeeded in select the shorter stick-shaped tool as expected. However, it could not slide the object because the height of the arm with respect to the object was slightly mismatched, and the arm swung out above it. However, the selection and the attempted action were correct.

unlearned tool and tool position while in motion.

The original motivation for using tools was to understand the limitations of the robot body. In future work, we will focus on the body restrictions of robots and will construct a model that can consider not only which tool to use but also whether to use tools.

ACKNOWLEDGMENTS

This work was supported in part by a JSPS Grant-in-Aid for Scientific Research (S) (No.25220005), JST CREST Grant Number: JPMJCR15E3, and the "Fundamental Study for Intelligent Machine to Coexist with Nature" program of Research Institute for Science and Engineering, Waseda University, Japan.

References

[1] K. Yamazaki, R. Ueda, S. Nozawa, M. Kojima, K. Odaka, K. Matsumoto, M. Ishikawa, I. Shimoyama, and M. Inaba, "Home-assistant robot for an aging society," *Proc. IEEE*, vol. 100, no. 8, pp. 2429–2441, 2012.

[2] K. Nagahama, K. Yamazaki, K. Okada, and M. Inaba, " Manipulation of multiple objects in close proximity based on visual hierarchical relationships," in *IEEE Int. Conf. Robot. Autom.*, pp. 1303–1310, 2013.

[3] A. Stoytchev, "Behavior-grounded representation of tool affordance," in *IEEE Int. Conf. Robot. Autom.*, pp. 3060–3065, 2005.

[4] V. Tikhanoff, U. Pattacini, L. Natale, and G. Metta, "Exploring affordances and tool use on the iCub," in *2013 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, Atlanta, GA, 2013, pp. 130–137.

[5] A. Dehban, L. Jamone, A. R. Kampff, and J. Santos-Vistor, "Denoising auto-encoders for learning of objects and tools affordances in continuous space," in *IEEE Int. Conf. Robot. Autom.*, pp. 3200–3206, 2015.

[6] S. Nishide, J. Tani, T. Takahashi, H. G. Okuno, and T. Ogata, "Tool-body assimilation of humanoid robot using a neurodynamical system," *IEEE Trans. Auton. Mental Develop.*, vol. 4, no. 2, pp. 139–149, 2012.

[7] T. Mar, V. Tikhanoff, G. Metta, and L. Natale, "Self-supervised learning of grasp dependent tool affordances on the iCub Humanoid robot," in *IEEE Int. Conf. Robot. Autom.*, pp. 3200–3206, 2015.

[8] K. Takahashi, K. Kim, T. Ogata, and S. Sugano, "Tool-body assimilation model considering grasping motion through deep learning," *Robot. Auton. Syst.*, pp. 115–127, 2017.

[9] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *Int. J. Robot. Res.*, vol. 37, pp. 421–436, 2017.

[10] J. Masci, U. Meier, D. Ciresan, and J. Schmidhuber, "Stacked convolutional auto-encoders for hierarchical feature extraction," in *LNCS 6791*, pp. 52–59, 2011.

[11] A. Krizhevsky, I. Sutskever, and G. H. Hinton, "ImageNet classification with deep convolutional neural networks," in *NIPS'12 Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, pp. 1097–1105, 2012.

[12] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimentionality of data with neural networks," *Science*, vol. 313, pp. 504–507, 2016.

[13] Y. Yamashita and J. Tani, "Emergence of functional hierarchy in a multiple timescales recurrent neural network: A humanoid robot experiment," *PloS Comput. Biol.*, vol. 4, no. 11, p. e1000220, 2008.

[14] D. Rumelhart, G. Hinton, and R.Williams, "Learning internal representation by error propagations by error propagation," in *Parallel Distributed Processing: Explorations in the Mmicrostructure of Cognition*, D. Rumelhart and D. McClelland, Eds. Cambridge, MA: MIT Press, 1986, pp. 318–362.

[15] Kawada Robotics, "Next generation industrial robot Nextage," http://nextage.kawada.jp/