Motion Generation Based on Reliable Predictability using Self-organized Object Features

Shun Nishide, Tetsuya Ogata, Jun Tani, Toru Takahashi, Kazunori Komatani, and Hiroshi G. Okuno

Abstract— **Predictability is an important factor for determining robot motions. This paper presents a model to generate robot motions based on reliable predictability evaluated through a dynamics learning model which self-organizes object features. The model is composed of a dynamics learning module, namely Recurrent Neural Network with Parametric Bias (RNNPB), and a hierarchical neural network as a feature extraction module. The model inputs raw object images and robot motions. Through bi-directional training of the two models, object features which describe the object motion are self-organized in the output of the hierarchical neural network, which is linked to the input of RNNPB. After training, the model searches for the robot motion with high reliable predictability of object motion. Experiments were performed with the robot's pushing motion with a variety of objects to generate sliding, falling over, bouncing, and rolling motions. For objects with single motion possibility, the robot tended to generate motions that induce the object motion. For objects with two motion possibilities, the robot evenly generated motions that induce the two object motions.**

I. INTRODUCTION

Recently, researches on creating intelligent robots based on KUKANCHI is being actively pursued. KUKANCHI is a Japanese term that represents intelligent space (KUKAN:space, CHI:intelligent) designed for humans and robots to interact. Two main approaches exist for creating robot behaviors [1]. The first is to embed behavior information into the environment for robots to acquire. This approach is often introduced in well-defined environments. Robots would recover embedded information for generating their behaviors. In unknown environments, however, behavior information are required to be obtained from the natural environment, as such information cannot be embedded beforehand. The second approach is to extract behavior information from the natural environment. This approach is based on human perception, affordance theory [2] in particular, to create robot's perception/behavior system. Our method adopts the second approach, using active sensing [3] with the environment to develop the robot's perception/behavior mechanism.

Application of active sensing for motion generation have been conducted, aiming functionalization of affordance. Fitzpatrick, et al. applied active sensing for learning the relation between robot motion and resulting object motion [4].

S. Nishide, T. Ogata, T. Takahashi, K. Komatani, and H. G. Okuno are with the Department of Intelligence Science and Technology, Graduate School of Informatics, Kyoto University, Kyoto, Japan {nishide, ogata, tall, komatani, okuno}@kuis.kyoto-u.ac.jp

J. Tani is with the Brain Science Institute, RIKEN, Saitama, Japan tani@brain.riken.jp

Stoytchev focused on tool affordance to relate the relation between robot, object, and tool [5]. Uğur, et al. evaluated traversability for determining navigation motions of robots [6]. Although these works have shown highly effective results, they contain two issues.

1) Predefinition of object features.
2) Predesign of robot motions.

Therefore, these methods were mostly focused on selecting appropriate robot behaviors to move a target object to a desired position. Our research focuses on resolving these two issues based on the following approaches.

1) Self-organization of object features based on robot's active sensing experiences.
2) Generation of robot motion based on predictability of target object motions.

The contribution of our work is that our model can deal with larger variety of objects and object motions (slide, roll, fall over, and bounce).

In our previous studies, we presented methods for solving each of the two issues. For the first issue, we proposed a bidirectional training method for self-organizing object features [7]. The result has shown that object features representing object motions have been self-organized by the robot's experience. For the second issue, we formulated reliable predictability for generating rolling motions with cylindrical objects [8]. Predictability is said to be one of the important factors for humans to decide their behaviors [9]. The model is created using neural networks for the following reasons.

1) Bidirectional training method requires mutual training of the feature extraction module and dynamics learning module
2) Generalization capability to adapt to unknown environment from few training data

The method requires compatibility between the feature extraction module and dynamics learning module. As neural networks possess both functionalities, we utilize neural networks to create our model.

In this paper, we present our work combining the two works applying the model to general tabletop objects. The method is comprised of three phases. In the first phase, the robot acquires training data by using its motion with various objects. In the second phase, we train our model composed of a dynamics learning module and a feature extraction module using the acquired data through bi-directional training. In the third phase, the robot searches through the trained model based on the predictability of object motion, to generate it's

motion. The results of the experiment show that the method tends to generate robot motions that induce affordable (possible) object motions.

The rest of the paper is composed as follows. Section II describes the overview of the technique. Section III describes the setup of the experiment. In Section IV, we present the results of the experiment. In Section V, we discuss the results of the experiment. Conclusions and future work are presented in Section VI.

II. OVERVIEW OF TECHNIQUE

In this section, we describe the overview of the technique. The training model is composed of a dynamics learning module and a feature extraction module. Recurrent Neural Network with Parametric Bias (RNNPB) [10] is used for the dynamics learning module to self-organize robot/object motions. A hierarchical neural network is linked to RNNPB as a feature extraction module. It inputs raw object images while outputting object features. The construction of the model is shown in Fig. 1.

The whole model is trained through bi-directional training of the two modules to self-organize object features. After training, the model searches for the robot motion with high reliable predictability. In the following subsections, we first describe the RNNPB model. Then the bi-directional training method is introduced. Finally, the robot motion searching method is described.

A. RNNPB

RNNPB, shown in the upper half of Fig. 1, is a predictor which inputs the current state $S(t)$ to calculate the next state $S(t + 1)$ as the output. It possesses Parametric Bias (PB) nodes connected to the Jordan-type RNN [11], which is used to learn multiple sequential data in a single model. The values of the PB nodes (PB values) in RNNPB are altered to generate different sequences, while conventional RNN can calculate a unique output sequence from the input and context value. In this RNNPB model, the input/output nodes are divided and assigned into nodes that input/output robot motor values and object feature values.
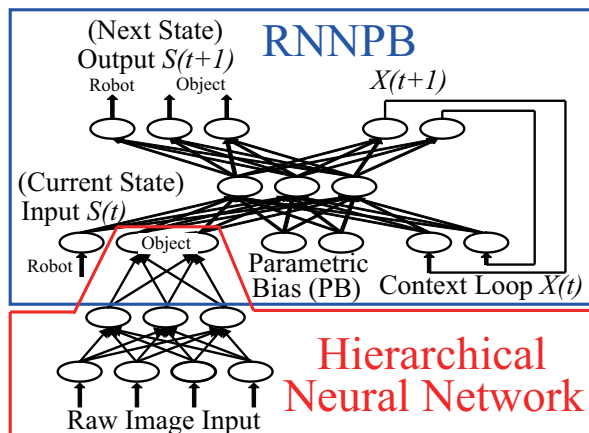

Fig. 1. Construction of the Model

As is the Jordan-type RNN, RNNPB is also a supervised learning system which requires teacher signals. In this paper, we apply the Back Propagation Through Time (BPTT) algorithm [12] for training RNNPB. During training, the back-propagated errors of the weights are accumulated along the sequence to update the PB values. Denoting the step length of a sequence as T, the update equations for PB during the training phase are

$$\Delta\rho = \varepsilon \cdot \sum_{t=1}^{T} \delta_t^{bp}$$

$$p = sigmoid(\rho). \qquad (2)$$

First, the delta force $\Delta\rho$ for updating the internal values of PB p is calculated by (1). The delta error δ_t^{bp} in (1) is calculated by back propagating the output errors from the output nodes to the PB nodes. The new PB value p is calculated by (2) applying the sigmoid function to the internal value ρ which is updated using the delta force. ε is a learning constant.

Training of RNNPB self-organizes the PB space based on the trained sequences. By training RNNPB, each trained sequence is encoded into PB values according to the similarity of the sequences. These PB values form the PB space by creating clusters of similar sequences. The sequences could be reconstructed from the PB values by recursively inputting the output $S(t+1)$ back into the input $S(t)$. This process is called *Closed Loop Calculation*, which calculates the whole sequence from an initial state $S(0)$, initial context $X(0)$, and a PB value.

B. Bi-directional Training

Training of the model requires a bi-directional method using teacher signals from each other module. The original bi-directional training method was proposed by Buessler and Urban [13]. Their idea was to develop a training algorithm for a model decomposed into two models, linked in a sequential composition. This decomposition is often conducted to reduce the dimensionalities of the modules.

Supervised training of a sequential composition of two modules is a difficult task as it requires intermediate variables that are not defined in the response model. Figure 2 shows an example of such model. The model cannot be trained straightforward as the training signal for the intermediate variable, $\hat{z}_1$, is not given. To solve this, Buessler and Urban introduced an inverted model of R_2, R_{2inv}, as shown in Fig. 3. In this model, the intermediate variable $\hat{z}_1$ acts as a teacher signal for R_{2inv}, and the output of R_{2inv}, $\hat{z}_2$ can be used as the teacher signal for R_1.

We apply this training architecture to our model. A simplified diagram of Fig. 1 is shown in Fig. 4. In Fig. 4, the intermediate variable, $\hat{z}_{R1,k}$, represents the object features that describe object motions. A major difference between our model and Buessler and Urban's model, is that the inverted model cannot be introduced, since teacher signals for both R_1 and R_2 are not given.
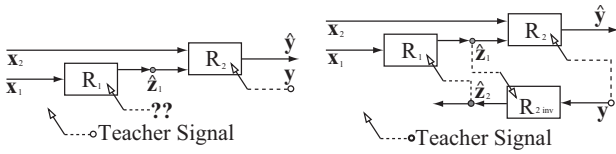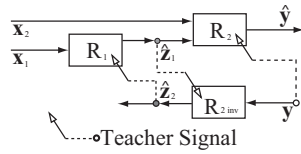
Fig. 2. Basic Architecture of a Sequential Composition

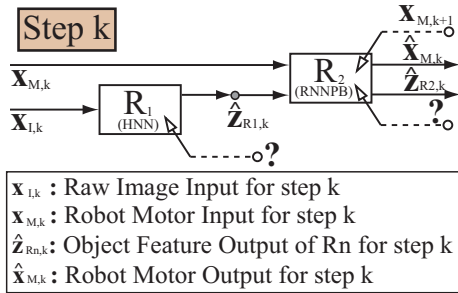Fig. 3. Bi-directional Extensional Training Model Proposed by Buessler and Urban

$x_{I,k}$: Raw Image Input for step k
$x_{M,k}$: Robot Motor Input for step k
$\hat{z}_{Rn,k}$: Object Feature Output of Rn for step k
$\hat{x}_{M,k}$: Robot Motor Output for step k

Fig. 4. Simplified Construction of Training Model

A characteristic of our model is that the output of R_1 for step $k+1$, input of R_2 for step $k+1$, and output of R_2 for step k are composed of the same variant $\hat{z}$ (they all represent the object features for step $k+1$). Considering this characteristic, we use the output of R_1 for step $k+1$ ($\hat{z}_{R1,k+1}$) as teacher signal for training R_2 for step k, and the output of R_2 for step $k(\hat{z}_{R2,k})$ as teacher signal for training R_1 for step $k+1$. This process would be done recursively from the last step to the initial. The training model is shown in Fig. 5. The whole model is trained using the teacher signals from the output of each other module.

C. Searching for Motion with High Reliable Predictability

For motion searching, we apply the method presented in [8]. The method searches through the PB space based on an evaluation function,

$$E(p) = \frac{\delta O^2}{\delta p}, \qquad (3)$$

where O is the object motion calculated by *Closed Loop Calculation* and p is the PB value. This equation evaluates the fluctuation of object dynamics relative to change of PB
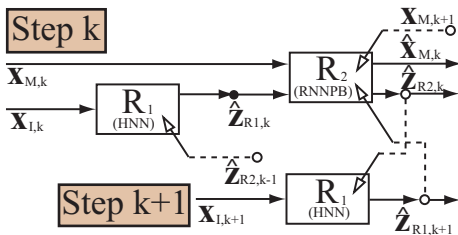

Fig. 5. Bi-directional Training Model for Our System

(representing change of robot motion). Therefore, minimizing (3) would provide the PB value that generates robot motion with high reliable predictability.

The steepest descent method is used to calculate the local minimum of (3), which represents the robot motion with high reliable predictability. Discretizing (3), a similar equation can be derived as

$$E = \frac{1}{\mu} \sum_{i,j,t} (O(p_1, p_2, t) - O(p_1 + i\mu, p_2 + j\mu, t))^2$$

$$(i, j = -1, 0, 1) \qquad (i \cdot j = 0), \qquad (4)$$

where t is the step number in the sequence, $O(p_1, p_2, t)$ is the object sequence calculated from the PB value (p_1, p_2) and t, and μ is the discretization width. Equation (4) is written for the case of two PB nodes, but a similar equation can be derived for a larger number of nodes.

As steepest descent method is an initial value dependent method possessing multiple local minimums, we evaluate the convergence in the PB space to determine a unique PB. As shown in [8], it is expected that a wider basin would be created in the PB space for reliable motions. We divide the PB space defined as [0, 1] into lattice points, using each lattice point as initial points to converge to a local minimum. The PB with the largest number of initial points to converge is the PB (p^*) encoding robot motion with high reliable predictability. The overview of the method is shown in Fig. 6.

III. SETUP OF EXPERIMENT

We used the humanoid robot Robovie-IIs [14], shown in Fig. 7, for evaluation of the method. Robovie-IIs has three DOF (degrees of freedom) on the neck and four DOF on each arm. It also has two CCD cameras on the head for processing visual information, one of which was used in the experiment. Objects shown in Fig. 8 were used in the experiment.

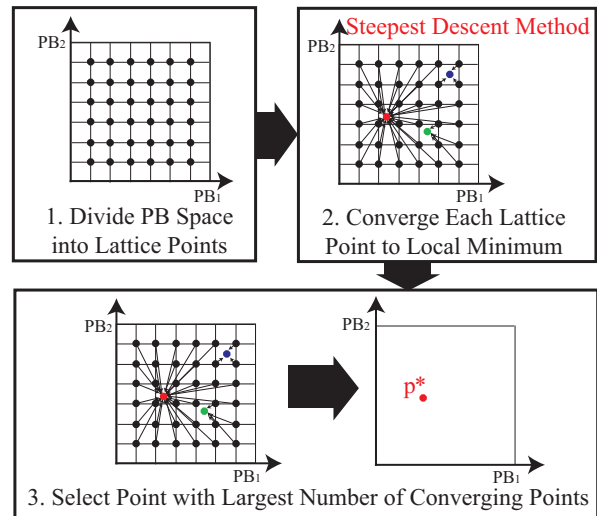The procedure of the experiment is as follows.

1. Divide PB Space into Lattice Points
2. Converge Each Lattice Point to Local Minimum
3. Select Point with Largest Number of Converging Points

Fig. 6. Overview of Motion Searching

Fig. 7.   Humanoid Robot Robovie-IIs

Fig. 8.   Objects used in Experiment

1) Acquire sequences of images and robot motor values while the robot pushes objects.
2) Extract the object from images by deleting background using color information.
3) Train model using motion sequences.
4) For each object, search for the reliably predictable motion for generation.

Procedures 1) - 3) are done for training the model, and 4) is done for evaluation.

A. Motion Sequence Acquisition from Active Sensing

For acquiring training data, we used the pushing motion with Robovie II-s' left arm for each of the objects shown in Fig. 8. Motions were generated at five different heights by altering Robovie II-s' shoulder pitch angle. The pushing motions induced the sliding, falling over, bouncing, and rolling motions of the objects. The balls were put on a cup to create bouncing motions. A total of 59 sequences were acquired, excluding those that object motions could not be extracted due to occlusions, illumination conditions, etc. The breakdown for each motion is shown in Table I.

During the pushing motion of the robot, image and motor sequences were acquired at 10 frames/sec. Acquisition of the sequences were started just before the robot's arm has had contact with the object, and ended after acquiring 10 steps of data, since some objects went out of sight at the eleventh step. Although this experiment was conducted under a fixed neck condition, the model could also be modified to adapt to cases where the robot constantly tracks the object [15].

B. Configuration of the Neural Networks

In this paper, the configuration of the neural networks were decided empirically. The configurations of RNNPB and the hierarchical neural network are shown in Table II and Table III, respectively. The initial weights of the neural networks were decided randomly within a value between [-0.7, 0.7]. The total number of input/output nodes in RNNPB are 5; 1 for the robot shoulder pitch angle normalized to [0,1], and 4 for the dynamic object features to be automatically extracted. The robot shoulder pitch angle is set to 0.5 at the

TABLE I
NUMBER OF EACH OBJECT MOTION

Slide	Fall Over	Bounce	Roll
24	9	13	13

TABLE II
CONFIGURATION OF RNNPB

Number of Motor Input/Output Nodes	1
Number of Object Input/Output Nodes	4
Number of Middle Nodes	20
Number of Context Nodes	20
Number of PB Nodes	2
Learning Constant ε	0.01

TABLE III
CONFIGURATION OF HIERARCHICAL NEURAL NETWORK

Number of Input Nodes	500
Number of Middle Nodes	30
Number of Output Nodes	4
Learning Constant ε'	0.1

initial step to prevent inclusion of contextual information. The input of the hierarchical neural network consists of the grayscale sequential object image, reduced to the resolution 25×20. The output is linked to the object input nodes of RNNPB. The model was trained by iterating the BPTT and BP calculation one million times.

C. Motion Generation

For evaluation of the experiment, we used the objects shown in Fig. 8. These objects can be divided into four categories based on object motions.

1) Objects that can be slided or fallen over.
2) Objects that can only be slided.
3) Objects that can only be bounced.
4) Objects that can only be rolled.

The number of objects in each category is shown in Table IV. In this paper, we neglect the *Slide* category, since the robot was capable of sliding the object regardless of how the robot pushed the object. Therefore, the experiment was conducted with objects in *Slide or Fall Over*, *Bounce*, and *Roll* categories.

IV. EXPERIMENTAL RESULT

We present the results of the experiment in this section.

The self-organization result for training RNNPB using training data is shown in Fig. 9. As can be seen from Fig. 9, each object motion is self-organized into the PB space, creating clusters of PB values representing each motion. The rhombi, square, triangle, and circle each represent PB values of Slide, Fall Over, Bounce, and Roll motions. The robot motion axis is also formed as the PB_2 axis. The robot pushes high when the value of PB_2 is small, and it pushes low when the value of PB_2 is large.

TABLE IV
NUMBER OF OBJECTS FOR EACH CATEGORY

Slide or Fall Over	23
Slide	10
Bounce	13
Roll	13

For each of the objects for the categories in Table IV, we calculated and plotted the searched PB values into the PB space. The results are shown in Fig. 10, 11, 12 for Categories *Slide or Fall Over*, *Bounce*, and *Roll*, respectively. For each of the PB values in Fig. 10-12, we generated robot motions from the searched PB values. An example of the pushing motion is shown in Fig. 13-18 for the PB values circled in Fig. 10-12. For Fig. 10, we selected the PB values with a clear visual representation of the motion. For Fig. 11 and 12, we selected the PB values near the boundary of success and failure.

PB values in Fig. 10 are categorized based on the actual object motion. Sliding motions are represented as rhombi, while fall over motions are represented as squares. A total of 12 sliding motions and 11 falling over motions were generated. This result shows that the slide and fall over motions are generated at an equal possibility for objects possessing both slide and fall over motion possibilities. An example of the generated motions are shown in Fig. 13 and 14. Fig. 13 represents a trigger sprayer sliding. Fig. 14 represents a plastic yellow bottle falling.

PB values in Fig. 11 are categorized based on whether the object has bounced or not when the robot pushed the object using the searched PB. The triangles represent PB values when bouncing motion was observed, and the x marks represent PB values when they weren't observed. From the result, it is notable that a majority of PB values (9/13) gen-

erated bouncing motion. Examples of the motion generation are shown when the robot succeeded and failed to generate bouncing motions with the plastic red ball and tennis ball placed on a cup in Fig. 15 and Fig. 16, respectively.

PB values in Fig. 12 are categorized based on whether the object has rolled or not when the robot pushed the object using the searched PB. The circles represent PB values when rolling motion was observed, and the x marks represent PB values when they weren't observed. As with bouncing motion, a majority of PB values (9/13) generated rolling motion. Examples of the motion generation are shown when the robot succeeded and failed to generate rolling motions with a money box and plastic container in Fig. 17 and Fig. 18, respectively.

V. Discussions

In this section, we present discussions considering the experimental results.

A. Motion Generation based on Reliable Predictability

The largest characteristic of the result is that the majority of searched PB values generated robot motions that induce possible object motion (bounce or roll) as shown in Fig. 11 and Fig. 12. Seventy percent of the objects in the "Bounce"
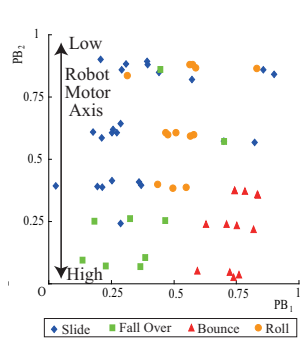

Fig. 13. Sliding Motion Generation
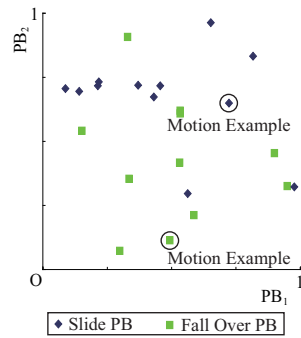

Fig. 14. Falling Over Motion Generation

Fig. 9. Self-organized PB Space

Fig. 10. PB Distribution for *Slide or Fall Over* Category

Fig. 15. Success for Bouncing Motion Generation

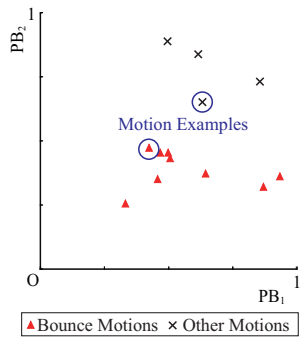Fig. 16. Failure for Bouncing Motion Generation

Fig. 11. PB Distribution for *Bounce* Category
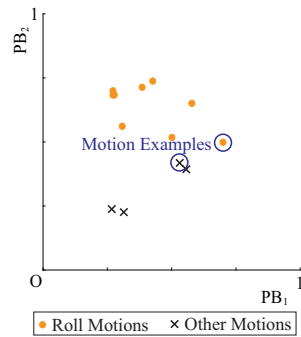
Fig. 12. PB Distribution for *Roll* Category

Fig. 17. Success for Rolling Motion Generation
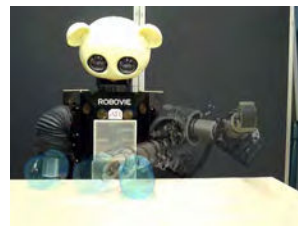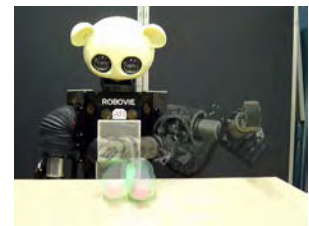

Fig. 18. Failure for Rolling Motion Generation

and "Roll" categories generated the appropriate motions, while an equal number of sliding and falling over motions were generated for the objects in the "Slide or Fall Over" category. This result implies that the evaluation of reliable predictability would narrow down the robot's motion to the affordance the object possesses.

An issue of the motion searching technique is that the motions are all trained using a single RNNPB model. Therefore, every data was related to each other during the training process. Since the training data were composed of a number of sliding motions, the PB space created a wide basin for the sliding motion. This resulted as a bias in the PB space towards the sliding motion, affecting the motion searching result. Although generalization is indispensible for relating training data (such as in generating medial motions), differentiation is also required when the data should not be related. The model is required to be improved to perform generalization and differentiation simultaneously.

B. Relation to Affordance Theory

As described in Gibson's proposal of affordance, the environment (object) possesses information about human actions [2]. Human's perceptual/behavior mechanisms are said to be divided into inherent factors and experiential factors. Our approach mainly focused on experiential factor. The perceptual model, where RNNPB and the hierarchical neural network were trained through bi-directional training, is based on transformational invariants which describe what the motions are. In the experiment conducted in this paper, the object features self-organized as the input of RNNPB and output of hierarchical neural network correspond to transformational invariants, since they are used to self-organize the PB space creating clusters of the four object motions.

For motion generation, we focused on predictability of the object as a criterion to search for the affordance the object possesses. Using the pushing motion, the robot tended to generate motions that would induce possible object motions (affordance of the object). The results of the experiment show the capability of using predictability as a criterion to search for the affordance of the object. In order to functionalize the ability to perceive affordance, the robot is required to integrate these experiences with other motions at a higher level. This would require a local representation of the model, such as the MOSAIC model [16], to determine which motion to generate (e.g. push or grasp) at a higher level, and how to move (e.g. push from left or push from front) at a lower level. These works are still left as future works.

VI. CONCLUSIONS

In this paper, we described a method to generate predictable motions for general tabletop objects based on the robot's experience. The model is constructed by a dynamics learning module, namely RNNPB, and a feature extraction module which are trained by bi-directional training. Steepest descent method is applied to search for robot motion with high reliable predictability. Experiments were conducted with general objects with four object motions: sliding, falling over,

bouncing, and rolling. The results have shown that the model tends to generate robot motions that induce possible object motions (object affordance).

As future work, we plan to evaluate the model with more complex robot motions such as grasping motions. This will require improvement of the motion searching method and learning model. We plan to integrate our model with other works to create an efficient model for more practical cases. We believe that our work would lead to acquisition of robot behavior through it's own experience, leading to functionalization of affordance to the robot's ability.

VII. ACKNOWLEDGMENTS

REFERENCES

[1] Y. Fukusato, E. S. Shimokawara, T. Yamaguchi, and M. Mizukawa, "A Service System Adapted to Changing Environments Using "Kukanchi"," *Journal of Robotics and Mechatronics*, Vol. 21, No. 4, pp. 443-452, 2009.

[2] J. J. Gibson, "The Ecological Approach to Visual Perception," *Houghton Mifflin*, ISBN: 0898599598, 1979.

[3] R. Bajcsy, "Active Perception," *IEEE Proceedings, Special issue on Computer Vision*, Vol. 76, No. 8, pp. 996-1005, 1988.

[4] P. Fitzpatrick, G. Metta, L. Natale, S. Rao, and G. Sandini, "Learning About Objects Through Action - Initial Steps Towards Artificial Cognition," *Proc. ICRA*, pp. 3140-3145, 2003.

[5] A. Stoytchev, "Learning the Affordances of Tools Using a Behavior-Grounded Approach," *Springer Lecture Notes in Artificial Intelligence*, pp. 140-158, 2008.

[6] E. Uğur, M. R. Doğar, M. Çakmak, and E. Şahin, "The learning and use of traversability affordance using range images on a mobile robot," *Proc. ICRA*, pp. 1721-1726, 2007.

[7] S. Nishide, T. Ogata, J. Tani, K. Komatani, and H. G. Okuno, "Self-Organization of Dynamic Object Features based on Bi-Directional Training," *Advanced Robotics*, Vol. 23, pp. 2035-2057, 2009.

[8] S. Nishide, T. Ogata, J. Tani, K. Komatani, and H. G. Okuno, "Autonomous Motion Generation based on Reliable Predictability," *Journal of Robotics and Mechatronics*, Vol. 21, No. 4, pp. 478-488, 2009.

[9] J. Hawkins and S. Blakeslee, "On Intelligence," *Times Books*, ISBN: 0805078533, 2004.

[10] J. Tani and M. Ito, "Self-Organization of Behavioral Primitives as Multiple Attractor Dynamics: A Robot Experiment," *IEEE Trans. on SMC Part A*, Vol. 33, No. 4, pp. 481-488. 2003.

[11] M. Jordan, "Attractor dynamics and parallelism in a connectionist sequential machine," *Eighth Annual Conf. of the Cognitive Science Society* (Erlbaum, Hillsdale, NJ), pp. 513-546, 1986.

[12] D. Rumelhart, G. Hinton, and R. Williams, "Learning internal representation by error propagation," in *D. E. Rumelhart and J. L. McLelland, editors Parallel Distributed Processing (Cambridge, MA: MIT Press)*, 1986.

[13] J. L. Buessler and J. P. Urban, "Neurobiology Suggests the Design of Modular Architectures for Neural Control," *Advanced Robotics*, Vol. 16, No. 3, pp. 297-307, 2002.

[14] H. Ishiguro, T. Ono, M. Imai, T. Maeda, T. Kanda, R. Nakatsu, "Robovie: an interactive humanoid robot," *Int. Journal of Industrial Robotics*, Vol. 28, No. 6, pp. 498-503, 2001.

[15] R. Yokoya, T. Ogata, J. Tani, K. Komatani, and H. G. Okuno, "Experience Based Imitation Using RNNPB," *Advanced Robotics*, Vol. 21, No. 12, pp. 1351-1367, 2007.

[16] M. Haruno, D. M. Wolpert, and M. Kawato, "MOSAIC Model for Sensorimotor Learning and Control," *Neural Computation*, Vol. 13, pp. 2201-2220, 2001.