

# Predicting Object Dynamics from Visual Images through Active Sensing Experiences

Shun Nishide<sup>1</sup>, Tetsuya Ogata<sup>1</sup>, Jun Tani<sup>2</sup>, Kazunori Komatani<sup>1</sup>  
and Hiroshi G. Okuno<sup>1</sup>

<sup>1</sup>*Department of Intelligence Science and Technology,  
Graduate School of Informatics,  
Kyoto University,  
Yoshida-Honmachi, Sakyo, Kyoto 606-8501, JAPAN,  
{nishide, ogata, komatani, okuno}@kuis.kyoto-u.ac.jp*

<sup>2</sup>*Brain Science Institute, RIKEN,  
2-1 Hirosawa, Wako City,  
Saitama 351-0198, JAPAN,  
tani@brain.riken.jp*

## Abstract

Prediction of dynamic features is an important task for determining the manipulation strategies of an object. This paper presents a technique for predicting dynamics of objects relative to the robot's motion from visual images. During the training phase, the authors use *Recurrent Neural Network with Parametric Bias* (RNNPB) to self-organize the dynamics of objects manipulated by the robot into the PB space. The acquired PB values, static images of objects, and robot motor values are input into a hierarchical neural network to link the images to dynamic features (PB values). The neural network extracts prominent features that induce each object dynamics. For prediction of the motion sequence of an unknown object, the static image of the object and robot motor value are input into the neural network to calculate the PB values. By inputting the PB values into the closed loop RNNPB, the predicted movements of the object relative to the robot motion are calculated recursively. Experiments were conducted with the humanoid robot Robovie-IIs pushing objects at different heights. The results of the experiment predicting the dynamics of target objects proved that the technique is efficient for predicting the dynamics of the objects.

*keywords:* Active Sensing, Neural Networks, Dynamics, Humanoid Robot, Object Manipulation

## 1 INTRODUCTION

Understanding human intelligence is essential for creating intelligent robots. Among the abundant functions of the human brain, perception and behavior are the two major key components in robotics. Studies on cognitive systems have revealed that perception and behavior are two sides of a same coin [1] [2]. Further investigation into the neocortex has uncovered that predictability is the key factor in perception [3]. In other words, humans can perceive the environment when it changes as predicted. This prediction capability is referred to as reliable predictability. As perception and behavior are tightly

connected, reliable predictability is also an important factor for generating motions, as humans generate motions so that the environment changes as predicted.

The final goal of our work is to create a robot that can autonomously generate handling motions for objects, based on *reliable predictability*. The work requires means to predict object dynamics relative to the robot’s motion, and to evaluate the prediction for generating motions based on reliable predictability. This paper addresses the first issue to provide the robot the capability to predict the outcome of the object motion provoked by the robot’s pushing motion based on the shape of the object. The work requires modeling of object dynamics, linked with robot motion and object shape. Further on, the prediction would be evaluated to generate robot motions.

Modeling dynamics of objects requires object recognition with the sense of “active sensing” [5]. Noda et al. integrated multiple static features (size, weight, and color images) for object recognition while grasping the object with its hands [6]. The work used a three-layered SOM (Self-Organizing Map [7]) taking into account only the static features, which entails quite an arduous task for applying the results to motion planning where dynamics of objects need to be regarded. Ogata et al. worked on recognizing unknown objects based on the dynamics they bear [8]. Their method classifies unknown objects based on the generalization capability of the Recurrent Neural Network (RNN) trained by the robot’s action with a variety of objects. Works conducted by Takamuku et al. integrate static and dynamic properties of objects for classifying them into predefined labels [9]. These works considering recognition based on dynamics require tactile sensing with unknown objects, which cannot be applied to manipulation strategies.

In this paper, the authors propose a technique to predict the dynamics of objects from an assumed robot motion and static images. Previous works require tactile contact with the objects for extracting dynamics. However, as round objects tend to roll when pushed, there is a tight connection between static (e.g. round) and dynamic features (e.g. roll). The proposed technique self-organizes the dynamic object features observed during active sensing experiences, and links the static object images and robot motion to the self-organized dynamic features. The process leads to prediction of object dynamics from the assumed robot motion and object image without the necessity to have contact with the object. The objective of the technique is not a complete prediction of object dynamics, but an estimate, based on the robot’s experience, of the actual object dynamics resulting from robot motion.

The authors utilize neural networks in order to apply its generalization capability to deal with unknown patterns. Robots are required to deal with various unknown patterns from a small number of training data, since acquisition of many training data would lead to damage of hardware. Neural networks are one such tool that meet this requirement. The recurrent neural network is applied for dealing with sequential data.

The proposed method consists of two neural networks, each trained independently. The method first trains a recurrent neural network with additional PB nodes in its input layer, namely Recurrent Neural Network with Parametric Bias (RNNPB), using visual sensory data of object features acquired while the robot hits a variety of objects at different heights. The advantage of the RNNPB lies in the

existence of PB nodes in the input layer, which are used for self-organizing the similarity of multiple sequences into the PB space. In other words, PB values of similar sequences are mapped closely in the PB space, while those of differing sequences are mapped distantly. RNNPB is set as a predictor, which inputs the current state of the object feature and outputs the next state. Training the RNNPB results in self-organization of the similarities of object dynamics, relative to robot motion, in the PB space. Next, the relationship of the static image, robot motion, and object dynamics is constructed by training a hierarchical neural network which inputs the static image and robot motion, and outputs the PB, which encodes the self-organized object dynamics. Using the generalization capability of the neural networks, the dynamics of an unknown object is estimated by inputting its static image and robot motion into the hierarchical neural network for calculating its PB, and associating the object dynamics from the PB using RNNPB.

The rest of the paper is composed as follows. Section 2 describes the proposed method with an introduction of the recurrent neural network model. Section 3 describes the robotic configuration and experimental setup. Section 4 describes a dynamics prediction experiment using artificial objects for analysis. Section 5 describes an experiment for evaluation of the method with practical objects. Conclusions and future works are written in Section 6.

## 2 LEARNING ALGORITHM

This section describes the method to link static images and robot motion to dynamic object features through active sensing. The method consists of two learning phases. The first phase applies the transition of object features extracted during active sensing with training objects to self-organize the similarity of object dynamics. The FF-model (forwarding forward model), also known as RNN with Parametric Bias (RNNPB) model, proposed by Tani [10] is used. During this phase, the similarity of dynamics is mapped into the PB space. The second phase uses a hierarchical neural network to link the static images with object dynamics represented by PB. The network configuration of the proposed method with RNNPB is shown in Figure 1.

### 2.1 RNNPB Model

The RNNPB model is an extension to the conventional Jordan-type RNN model [11] with PB nodes in the input layer. In order to deal with sequential data, the RNNPB is set as a predictor which calculates the next state from the current state. In this paper, the input of RNNPB is the current object feature state, and the output is the next object feature state.

The role of the PB nodes in the RNNPB is to learn multiple sequential data in a single model. While RNN calculates a unique output from the input and context value, the RNNPB is capable of altering its output by changing the values of the PB nodes (PB values). This capability provides the RNNPB to learn and generate multiple sequences. Therefore, RNNPB is called a distributed representation of

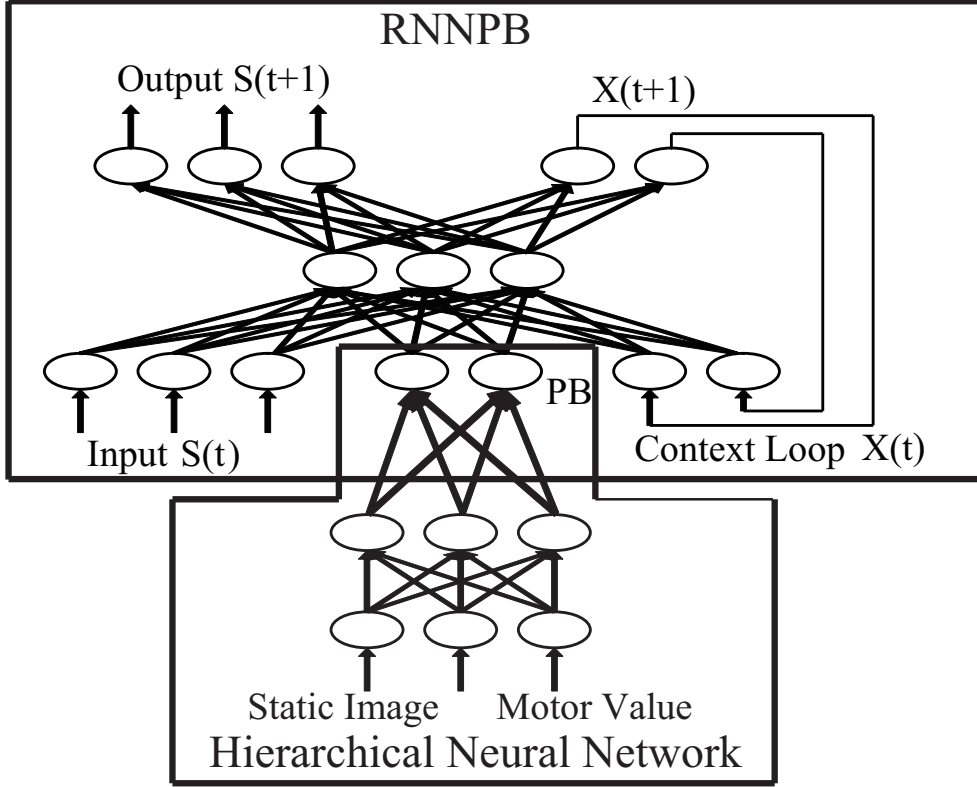


Figure 1: Configuration of RNNPB and the Proposed System

multiple RNNs.

RNNPB is a supervised learning system requiring teaching signals as is the Jordan-type RNN model. The learning phase consists of weight optimization and self-organization of the PB values using the back propagation through time (BPTT) algorithm [12]. For updating the PB values, the back-propagated errors of the weights are accumulated along the sequences. Denoting the step length of a sequence as  $T$ , the update equations for the parametric bias are

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

$$p = \text{sigmoid}(\rho). \quad (2)$$

First, the  $\delta$  force for updating the internal values of PB  $p$  is calculated by (1). The delta error  $\delta^{bp}$  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  $p_t$  which is updated using the delta force.

The training of RNNPB self-organizes the PB according to the similarity of the sequences, forming the PB space which creates clusters of similar sequences. In other words, the training of RNNPB encodes the sequences into the PB values. Therefore, the sequences could also be reconstructed from the PB values by recursively inputting the output  $S(t+1)$  back into the input  $S(t)$ . This process called

*association* calculates the whole sequence from an initial object state  $S(0)$  and a PB value. *Association* is used in this paper for predicting the dynamics of objects.

The number of PB nodes determines the learning capability of RNNPB. Currently, there are works for evaluating the relationship, though there are no concrete criteria [13]. In this paper, we focus more on the prediction capability using the neural networks. Therefore, the number of PB nodes are determined heuristically in the experiments.

## 2.2 Two Phase Training

The proposed method contains an additional phase to the conventional RNNPB to link the static images and robot motion to object dynamics. Using the advantage that RNNPB self-organizes the similarities of each sequence with numerical values, we attach a hierarchical neural network to the PB nodes.

The training phase of each neural network is conducted independently. First the PB values are self-organized using the method described in the previous subsection. Using the PB values as teaching signals, the hierarchical neural network is trained with static images and motor values as its input signals. The system, as a result, extracts the static features and movements of the robot that attract most the dynamics of the object, and links them with the PB values of RNNPB, which resemble the dynamics of the object.

## 3 Active Sensing Experiments

The authors have used the humanoid robot Robovie-IIs [14], which is a refined model of Robovie-II developed at ATR [15], 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.

The procedure of the experiment is as follows.

1. Acquire motion sequences of object features (center position of object and inclination of the principal axis of inertia) from visual images while the robot pushes training objects.
2. Train RNNPB using the motion sequences.
3. Train the hierarchical neural network using the self-organized PB values, static images of training objects, and shoulder pitch angle, which determines the height of the arm.
4. Input static images of target objects and motor value into the hierarchical neural network to calculate PB values relative to the object and motion.
5. Input the PB values into the RNNPB for prediction.

### 3.1 Robot Motion and Target Objects

The shape of an object and robot motion are two large factors that affect the object dynamics. For instance, an upright box would tend to fall if pushed on the top, where it would just slide when pushed on the bottom. A cylindrical object would roll when pushed by its side.

The authors have focused on an active sensing motion that the robot pushes an object on the table with its arm. The pushing motion is conducted at different heights, with a constant pitch angle, where the object and arm could take contact. Consequently, the objects are compelled to roll, fall over, slide, or bounce depending on their shape and motion of the robot. Figure 2 shows the scene of an actual experiment where Robovie-IIs pushes and moves the object by rotating the shoulder motor (roll axis).

The authors have conducted two experiments: one for evaluation and analysis of the model, and the other for evaluation with practical objects. The object is placed at the same initial position.

The first experiment consists mainly of artificial objects. Seven types of objects were used for training the neural networks as shown in Figure 3: *three triangle poles, three cylinders, and one cuboid*. The robot pushed the objects at two different heights to acquire a total of seventeen motion sequences (each consisting of seven steps acquired at 2.5 frames/sec), placing the objects in several orientations for manipulation. These motion sequences were used for training the RNNPB. Prediction of motion sequences was conducted using four target objects shown in Figure 4: *a box, a shampoo container, a cylinder, and a packing tape*. The cylinder, also used for training, was put upright for prediction, where it was laid down during training. The results of preliminary experiments with target objects considering the relationship between robot motion and consequence of the objects are shown in Table



Figure 2: Robovie-IIs and Scene of Experiment

Table 1: Robot Motion and Consequence of Object

Height of Robot Motion	High	Low
Box	Fall Over(a)	Fall Over(b)
Shampoo Container	Fall Over(c)	Slide(d)
Cylinder	Slide(e)	Slide(f)
Packing Tape	–	Roll(g)

1. The alphabets correspond to the items of PB data illustrated afterwards in Figure 6.

The second experiment consists of practical objects. A total of twenty objects shown in Figure 5, including balls of different sizes, were used for training and prediction. The robot pushed these objects at five different heights to generate rolling, falling over, sliding, and bouncing motions of the object. The

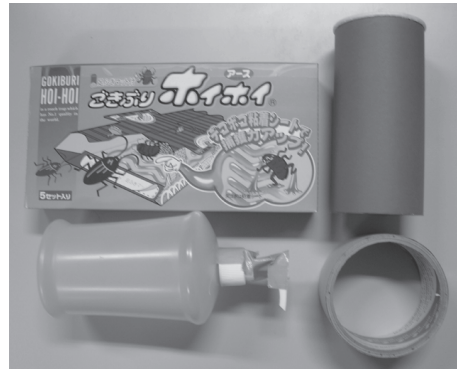
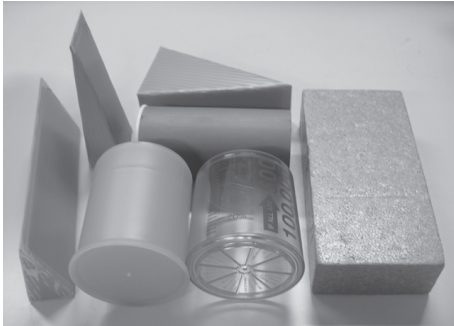


Figure 3: Objects used for Training (First Experiment)

Figure 4: Targets used for Dynamics Prediction (First Experiment)



Figure 5: Objects for Second Experiment

balls were placed at different heights to generate bouncing motions. A total of 115 motion sequences (each consisting of 15 steps acquired at 10 frames/sec) were acquired. 77 motion sequences were used for training the neural networks. The other 38 were used for prediction.

### 3.2 Configuration of the Neural Networks

The RNNPB model used in this experiment is set as a predictor for calculating the next sensory state  $S(t+1)$  as the output, from the current sensory state  $S(t)$ , the input. As inputs of RNNPB, the center position  $(x, y)$  and the inclination of the principal axis of inertia  $(\theta)$  of the object are used, which are extracted from sequentially acquired images. These data are normalized ( $[0,1]$  for center position,  $[0.25, 0.75]$  for inclination of the principal axis of inertia) before being input into RNNPB. Considering the case of the ball, which has no principal axis of inertia, in the second experiment,  $\theta$  was set to 0.05.

The input of the hierarchical neural network consists of the robot motion input and image input. The robot motion input consists of the shoulder pitch angle which determines the height of the arm to push the object. The image input of the hierarchical neural network, differs with the two experiments. For the first experiment, we considered an ideal condition, and used the grayscale image of the frontal view of the object (Resolution  $40 \times 30$  pixels). For the second experiment, we considered a more practical case, and used the grayscale image of the initial view of the object from the robot (Resolution  $50 \times 40$  pixels). The background was eliminated for both cases before inputting the image into the hierarchical neural network. All of the input data are normalized ( $[0,1]$ ) as with RNNPB.

The constructions of the neural networks for each experiment are shown in Table 2 and Table 3.

Table 2: Construction of RNNPB

	First Experiment	Second Experiment
Number of Input/Output Nodes	3	3
Number of Middle Nodes	10	40
Number of Context Nodes	10	40
Number of PB Nodes	2	3
Learning Constant $\varepsilon$	0.01	0.01

Table 3: Construction of Hierarchical Neural Network

	First Experiment	Second Experiment
Number of Input Nodes	$40 \times 30 + 1$	$50 \times 40 + 1$
Number of Middle Nodes	4	20
Number of Output Nodes	2	3



## 4 Analysis and Results for First Experiment

This section describes the result and evaluation of the first experiment.

### 4.1 Self-Organized PB Space and Calculated PB Values of Target Objects

The authors carried out the experiment using a total of seventeen motion sequences as described in the previous section. RNNPB was trained by iterating the calculation 1,000,000 times which required approximately 20 minutes using Xeon, 2.66 GHz. Each motion sequence consists of seven steps, starting right before the robot arm has had contact with the object.

RNNPB was trained using the motion sequences of training objects. The PB values were self-organized according to the similarity of the dynamics. In other words, the PB values of the same object motions are allocated nearby. The PB space created by training RNNPB is shown in Figure 6. The seventeen teaching data are indicated as diamond-shaped marks. The remaining marks with alphabets are the calculated PB values of target objects.

For analysis, we divided the PB space into  $25 \times 25$  segments, and examined the object dynamics prediction of each area. Prediction is done by forming a closed loop RNNPB, feedbacking the output as the input of the next state. The motion sequence is calculated recursively from the initial position of the object. By evaluating the sequence, we labeled each segment into 4 motion categories (Unobserved, Fall Over, Slide, and Roll). Unobserved motions, which were not observed during acquisition of training data, consist of a combination of 2 trained motions or a trained motion in a different direction. The

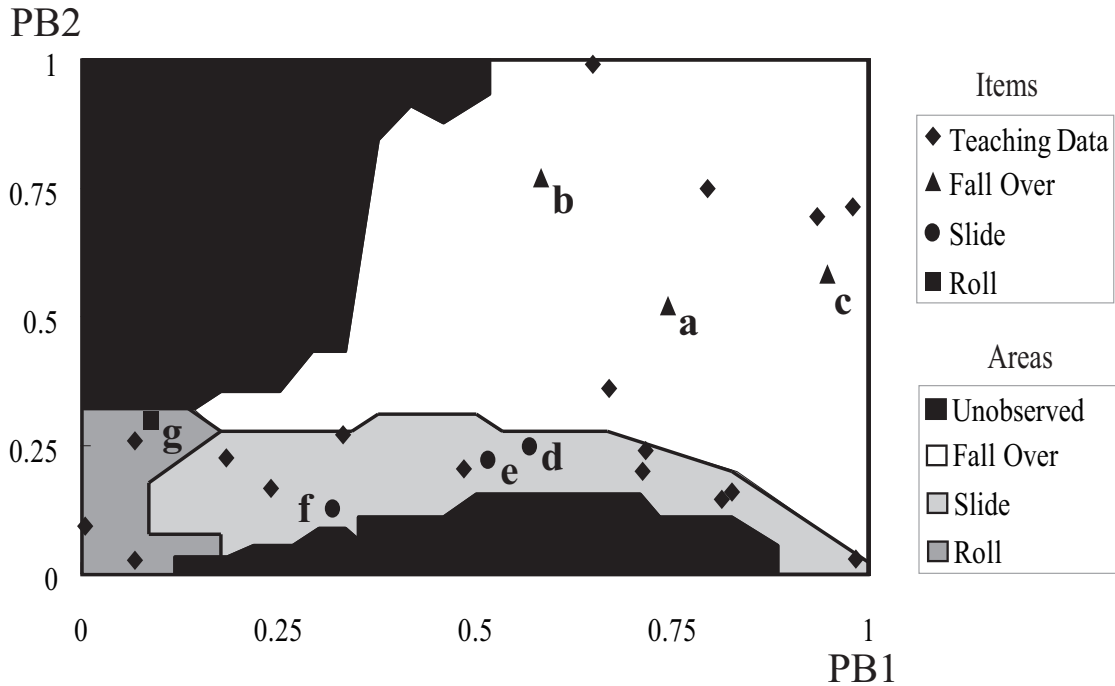


Figure 6: Constructed PB Space

four areas represent clusters of each category.

After the PB values have been self-organized, the hierarchical neural network was trained using the  $(40 \times 30)$  static images, shown in Figure 7, and shoulder pitch angle. The calculation was iterated 30,000 times, which was decided heuristically to prevent over-training. The static images of target objects, shown in Figure 8, and shoulder pitch angle for predicting the object motion were input into the neural network to calculate the PB values of the target object.

The calculated PB values of target objects are indicated as triangles, circles, and a square in Figure 6. The alphabets next to the marks correspond to the labels of target objects in Table 1. The triangles, circles, and square each represent the distinction of dynamics, where they fell over, slid, and rolled. As can be seen, the PB values of target objects reside in the area corresponding to their actual motions.

## 4.2 Motion Prediction from the PB Values

The estimated sequences are shown in Figure 9. The sequences are calculated by first inputting the static image of the object into the hierarchical neural network to calculate the PB. The PB is then used to “associate” the sequence. In other words, the RNNPB recursively calculates the motion sequence (transition of the center position and inclination of the principal axis of inertia) by inputting the output of the current step recursively into the RNNPB to calculate the sequence. Each sequence is drawn by forming a rectangle using the predicted center position and inclination of the principal axis of inertia for each step. The rectangle represents the principal axis of inertia of the object. Figure 9(a), 9(b), 9(c) are motion sequences of objects that fell over. Figure 9(d), 9(e), 9(f) are motion sequences of objects

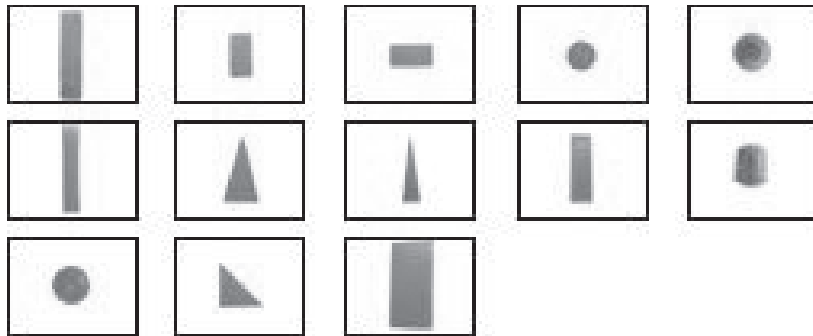


Figure 7: Static Images of Training Objects

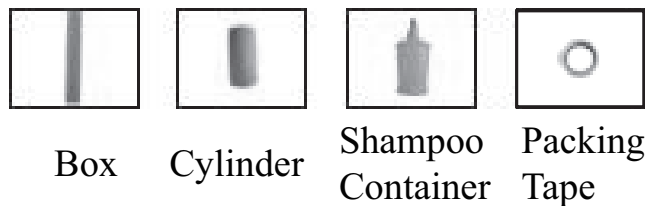
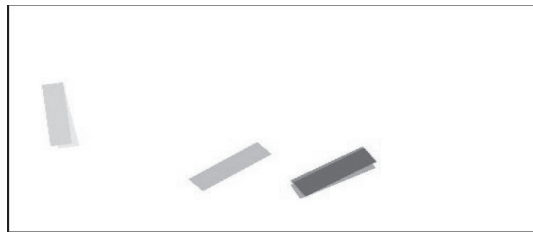
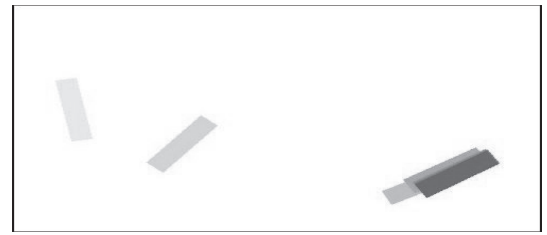


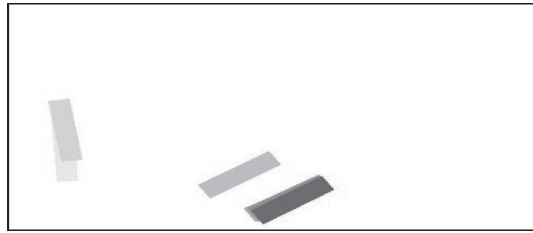
Figure 8: Static Images of Target Objects



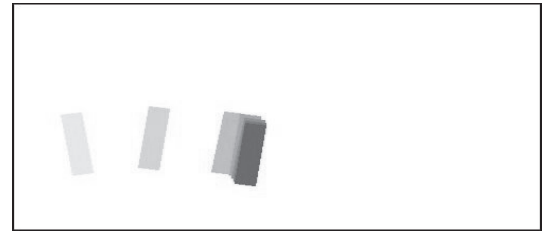
(a) Motion Sequence of Object a(Box, High)



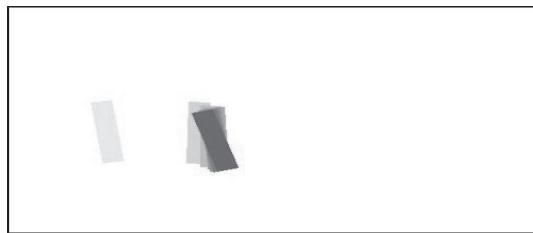
(b) Motion Sequence of Object b(Box, Low)



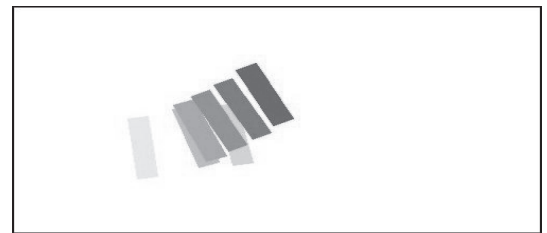
(c) Motion Sequence of Object c(Shampoo, High)



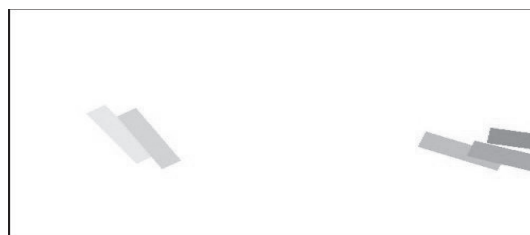
(d) Motion Sequence of Object d(Shampoo, Low)



(e) Motion Sequence of Object e(Upright Cylinder, High)



(f) Motion Sequence of Object f(Upright Cylinder, Low)



(g) Motion Sequence of Object g(Tape)

Figure 9: Predicted Sequences of Target Objects

that slid. Figure 9(g) is the motion sequence of the object that rolled. The seven predicted motion sequences represent well the actual motions indicated in Table 1.

### 4.3 Discussions

#### 4.3.1 Analyzing the Hierarchical Neural Network

The hierarchical neural network functions as a filter to extract static features from images that affect the dynamics of objects. In this subsection, we investigate what types of features were extracted during the training process by analyzing the weights of the hierarchical neural network.

The weights of the hierarchical neural network are presented in Figure 10. We analyze the weights from the image input and robot motion input, each stated “Static Feature” and “Motion”, separately to evaluate the effects of each input. “Static Feature” is an image created by normalizing the weights to image format([0, 255]) for each of the image pixels.

“Static Feature” represents filters for extracting the static features from object images. It is notable that these filters are created by combining portions of the input images of training objects in Figure 7. Features that affect positively are displayed white, while those that affect negatively are displayed black. The images in “Static Feature” are applied to calculate the roundness and stability (fineness ratio, ratio of upper and lower surface of object) of the object, which affect the dynamics (fall over, slide, roll) considered in the experiment. Features that induce the fall over motion of the objects are relatively colored black possessing a negative effect on the “Static Feature”. The edge of the objects are specifically intensified which denotes that the neural network has learned that the edge is a prominent static feature for predicting object dynamics.

“Motion” value appends a negative effect on “Static Feature”. The larger the magnitude, the larger the effect. This result implies that the neural network has learned that objects would tend to fall over as the robot pushes the object at a higher spot.

To analyze the relation between the input and middle layers, we inputted rectangles of different sizes and a circle, and evaluated the values of the middle layer. The sizes of the rectangles (horizontal  $\times$  vertical) were,  $4 \times 28$ ,  $6 \times 24$ ,  $8 \times 20$ ,  $10 \times 16$ ,  $12 \times 12$ , and  $14 \times 8$ .

First, we evaluated each input with a fixed motor value. The values of the middle layer and PB for each input are stated in Table 4. The Unobserved slide motion for the rectangle of the size  $8 \times 20$  was a motion not only sliding to the right, but also sliding away from the robot. By observing the results

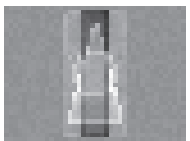
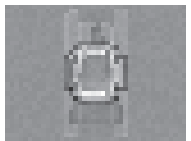
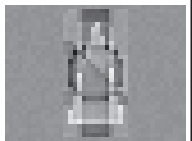
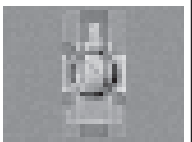
Node	1	2	3	4
Static Feature				
Motion	-188.9	-153.7	-72.6	-143.6

Figure 10: Weights of Hierarchical Neural Network

Table 4: Middle Layer and PB for Fixed Motor Value

Size of Rect. or Circle	Node 1	Node 2	Node 3	Node 4	PB 1	PB 2	Predicted Dynamics
$4 \times 28$	0.175	0.570	0.227	0.981	0.712	0.695	Fall Over
$6 \times 24$	0.083	0.647	0.113	0.991	0.656	0.782	Fall Over
$8 \times 20$	0.959	0.964	0.924	0.999	0.524	0.075	Slide(Unobserved)
$10 \times 16$	0.999	0.951	0.492	0.990	0.308	0.148	Slide
$12 \times 12$	0.999	0.976	0.020	0.826	0.101	0.245	Roll
$14 \times 8$	0.993	0.318	0.217	0.997	0.583	0.272	Slide
Circle	0.996	0.996	0.635	0.092	0.064	0.023	Roll

in Table 4, we can discover the roles of each middle node. Node 1 possesses a small value for the Fall Over motion and a large value for the others. Therefore, this node calculates the stability of the object. The value of Node 2 increases as the shape of the image becomes close to a circle, becoming maximum with the circle. This node calculates the roundness (often calculated as *squared perimeter/area*) of the object. Node 4 possesses a small value for the circle and large value for the others. This node calculates the sharpness of the corners. Node 3 possesses a large value for rectangle  $8 \times 20$  and a smaller value for the others. From Figure 10, it is notable that this node is affected greatly by the motor value.

To investigate the role of Node 3, we changed the motor values for the rectangle  $8 \times 20$ . The results are given in Table 5. We can see from Table 5 that the decrease of Node 3 is smaller compared to Node 1 and Node 2 when the motor value increases. The Predicted Dynamics also alters from the Sliding motion to Fall Over motion. Therefore we can denote the role of Node 3 as a filter to estimate the alteration of dynamics according to the motor value. This node did not alter the dynamics of the rectangles larger than  $8 \times 20$ , nor did it alter that of  $10 \times 16$  for different motor values.

Table 5: Middle Layer and PB for Rectangle  $8 \times 20$ 

Motor Value	Node 1	Node 2	Node 3	Node 4	PB 1	PB 2	Predicted Dynamics
0.1	0.959	0.964	0.924	0.999	0.524	0.075	Slide(Unobserved)
0.2	0.900	0.925	0.894	0.999	0.557	0.091	Slide(Unobserved)
0.3	0.778	0.850	0.854	0.997	0.629	0.129	Slide
0.4	0.577	0.725	0.803	0.994	0.742	0.215	Slide
0.5	0.346	0.550	0.739	0.988	0.850	0.359	Fall Over

### 4.3.2 Dynamics Prediction using RNNPB

The proposed method was capable of predicting the dynamics of four unknown objects. Experiments with the packing tape, which possesses a cavity in the center, and shampoo container, which resemble a combination of a cuboid and a triangle pole for the input image, express the effectivity of the generalization capability of the neural networks. The analysis has proved that the appropriate features were extracted for predicting the object dynamics. Although the second node of the neural network predicted cubes as rolling objects, it is notable that it acts as a filter for calculating the roundness of the front view of the object. It would be necessary to include a cube as a training object to distinguish the dynamics between the cube and cylinder.

A difficult issue in training RNNPB is to decide the number of iterations for training. Large numbers of iteration would result in over-training which diminish the capability of generalization, while small numbers result in under-training. The training we have conducted created restricted space for the rolling motion. Although the neural network was capable of calculating an appropriate PB value for the packing tape, a wider area of the rolling motion in the PB space would result in better prediction for the motion sequence. We would like to refine the training process to create a more sparse PB space for improving the capability of the method.

## 5 Experimental Results of the Second Experiment

The second experiment is conducted to evaluate the prediction capability with a variety of practical objects. It consists of a total of 115 motion sequences. Training was conducted with 77 sequences: 35 sliding motions, 14 rolling motions, 19 falling motions, 9 bouncing motions. The target sequences for prediction consist of 10 sliding motions, 4 rolling motions, 8 falling motions, and 16 bouncing motions.

### 5.1 Dynamics Prediction Results

The authors compared the predicted object motions with the actual object motions. Table 6 shows the number of successful prediction for target sequences. Considering sequences used for training, every predicted sequence accurately corresponded to the actual sequence. Prediction of sequences used for

Table 6: Prediction Results for Target Sequences

Object Motion	Successful Prediction	Total
Slide	9	10
Roll	2	4
Fall Over	8	8
Bounce	16	16
Total	35	38

training were accurate for every sequence. There were few failures in prediction for sliding and rolling motions.

Figure 11 and Figure 12 illustrate the predicted sequences of a bouncing ball used for training. Figure 13 illustrates the predicted sequence for a ball which was not included in training. The motion sequences of the inclination of the principal axis of inertia of balls were described as 0.05. Therefore, sequences with an inclination of the principal axis under 0.1 are illustrated as circles formed around the center point of each step. Motion sequences with an inclination of the principal axis above 0.1 are illustrated as described in the previous section. As can be seen from Figure 13, the trajectory of the ball represent well the bouncing sequence of a ball. The successful prediction results for sliding, rolling, and falling over are omitted since they are similar to those shown in Figure 9.

The predicted sequences of object dynamics that had failed are shown in Figure 14, Figure 15, and Figure 16. Figure 14 shows a bouncing motion after a falling motion. Figure 17 shows the initial state of the object. The robot pushed the very bottom of the object from the left. It is notable that even a human would mistake the sliding motion of the object with a falling motion. Figure 15 and Figure

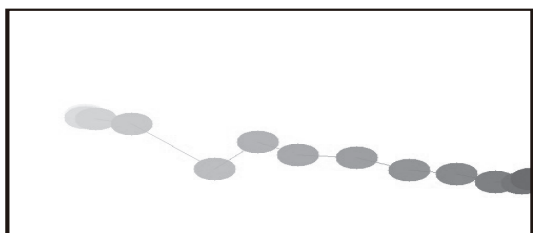


Figure 11: Bouncing Prediction 1 (Trained)

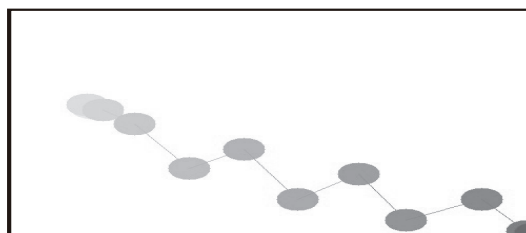


Figure 12: Bouncing Prediction 2 (Trained)

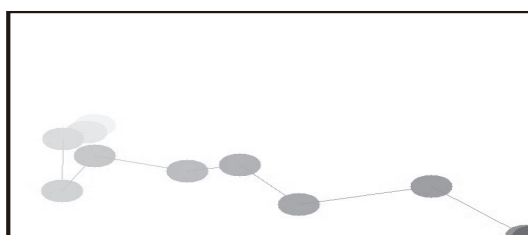


Figure 13: Bouncing Prediction 3 (Untrained)

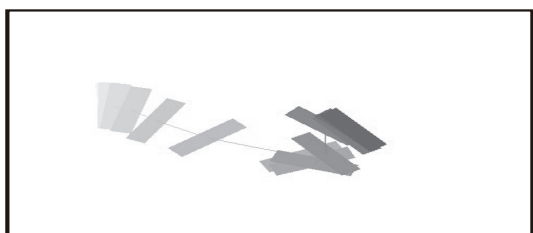


Figure 14: Failure of Sliding Prediction

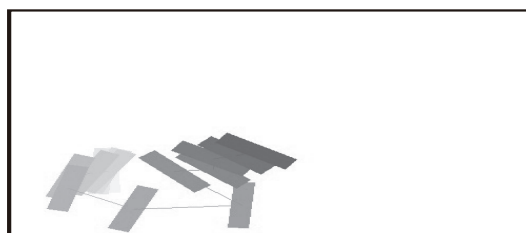


Figure 15: Failure of Rolling Prediction 1

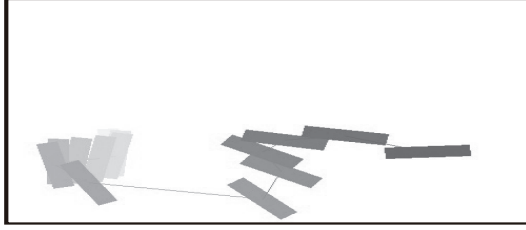


Figure 16: Failure of Rolling Prediction 2



Figure 17: Initial Object Posture for Sliding Prediction Failure

16 show a sliding motion after a rolling motion. This failure is due to the restricted area of the rolling motion in the PB space.

## 5.2 Discussions

From the experimental results, the method was capable of predicting about 90% of the unknown object sequences. The results have shown that the prediction capability of the method is stronger with dynamical object motions, such as fall over or bounce. Therefore, prediction failures have occurred to rolling and sliding motions. Here, we discuss the factors that have led to failure in prediction.

The most prominent factor that led to prediction failure is in the predefined dynamic object features. In this experiment, the authors have used center point and inclination of the principal axis of inertia for describing object dynamics. However, the sliding motion and rolling motion possess fairly similar dynamical properties when these object features are used: the difference exists only in the magnitude of the movement of the center position. This is due to insufficiency of object features to describe object dynamics. Therefore, these motions cannot be prominently classified into the PB space which has led to prediction errors.

One means of solving this issue is to adaptively extract object features for inputs of RNNPB from image sequences, based on the robot's active sensing experiences. This would provide the object features that appropriately describes the object dynamics used for training. We plan to implement the method



in our future work.

## 6 Conclusions and Future Works

This paper proposed a dynamics prediction method through active sensing experiences combining RNNPB and a hierarchical neural network. The method is composed of two phases, one for self-organizing the object dynamics by RNNPB and the other for linking static images and robot action to the self-organized dynamics using the hierarchical neural network. Training of RNNPB was conducted with motion sequences of object features, acquired during the pushing motion of the robot. Using the self-organized PB values, the hierarchical neural network was trained by inputting the reduced static images and shoulder pitch value. The hierarchical neural network, which links static images and motor values to PB values, proved efficient for extracting prominent static features that affect the dynamics of the object. The motion sequences of unknown objects were predicted by inputting the PB values to the closed loop RNNPB. The analysis proved that the appropriate static features were extracted during the training process. The method was capable of prediction 90practical objects.

Our next goal is to apply the prediction results to manipulation strategies based on reliable predictability. We plan to increase the number of motions the robot could take, and investigate the capability of the method with other types of motions, such as grasping. Further on, the work will be extended to automatic extraction of dynamical object features based on the robot's experience. We believe that these works would develop our system to a more sophisticated technique with large capabilities of application to studies in motion planning.

## 7 ACKNOWLEDGMENTS

This research was partially supported by the Ministry of Education, Science, Sports and Culture, Grant-in-Aid for Young Scientists (A)(No. 17680017, 2005-2007), and Kayamori Foundation of Informational Science Advancement.

## REFERENCES

- [1] P. E. Agre, The Symbolic Worldview: Reply to Vera and Simon, *Cognitive Science*, Vol. 17, No. 1, pp. 61-69, 1993.
- [2] G. Rizzolatti and L. Craighero, The Mirror-Neuron System, *Annual Review of Neuroscience*, Vol. 27, pp. 169-192, 2004.
- [3] J. Hawkins and S. Blakeslee, On Intelligence, *Times Books*, 2004. ISBN: 0805078533.
- [4] J. J. Gibson, "The Ecological Approach to Visual Perception," 1979. ISBN: 0898599598.

- [5] R. Bajcsy, "Active Perception," *IEEE Proceedings, Special issue on Computer Vision*, Vol. 76, No. 8, pp. 996-1005, 1988.
- [6] K. Noda, M. Suzuki, N. Tsuchiya, Y. Suga, T. Ogata, and S. Sugano, "Robust Modeling of Dynamic Environment based on Robot Embodiment," *IEEE ICRA 2003*, Taipei, Taiwan, pp. 3565-3570, 2003.
- [7] T. Kohonen, "Self-Organizing Maps," *Springer Series in Information Science*, Vol. 30, Springer, Berlin, Heidelberg, New York, 1995.
- [8] T. Ogata, H. Ohba, J. Tani, K. Komatani, and H. G. Okuno, "Extracting Multi-Modal Dynamics of Objects using RNNPB," *IEEE/RSJ IROS 2005*, Edmonton, Alberta, Canada, pp. 160-165, 2005.
- [9] S. Takamuku, Y. Takahashi, and M. Asada, "Lexicon Acquisition based on Behavior Learning," *IEEE ICDL 2005*, Osaka, Japan, TALK 14, 2005.
- [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 Conference of the Cognitive Science Society*, Hillsdale, New Jersey, USA, 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. McClelland, editors Parallel Distributed Processing (Cambridge, MA: MIT Press)*, 1986.
- [13] T. Ogata, S. Matsumoto, J. Tani, K. Komatani, and H. G. Okuno, "Human-Robot Cooperation using Quasi-symbols Generated by RNNPB Model," *Proc. IEEE ICRA 2007*, Rome, Italy, pp. 2156-2161, 2007.
- [14] H. Ishiguro, T. Ono, M. Imai, T. Maeda, T. Kanda, and R. Nakatsu, "Robovie: an interactive humanoid robot," *Int. Journal of Industrial Robotics*, Vol. 28, No. 6, pp. 498-503, 2001.
- [15] T. Miyashita, T. Tajika, K. Shinozawa, H. Ishiguro, K. Kogure, and N. Hagita, "Human Position and Posture Detection based on Tactile Information of the Whole Body," *IEEE/RSJ IROS 2004 Workshop*, Sendai, Japan, 2004.