Proceedings of the 2007 IEEE/RSJ International
Conference on Intelligent Robots and Systems
San Diego, CA, USA, Oct 29 - Nov 2, 2007

Two-way Translation of Compound Sentences and Arm Motions by Recurrent Neural Networks

Tetsuya Ogata, Masamitsu Murase, Jun Tani, Kazunori Komatani, and Hiroshi G. Okuno

Abstract - **We present a connectionist model that combines motions and language based on the behavioral experiences of a real robot. Two models of recurrent neural network with parametric bias (RNNPB) were trained using motion sequences and linguistic sequences. These sequences were combined using their respective parameters so that the robot could handle many-to-many relationships between motion sequences and linguistic sequences. Motion sequences were articulated into some primitives corresponding to given linguistic sequences using the prediction error of the RNNPB model. The experimental task in which a humanoid robot moved its arm on a table demonstrated that the robot could generate a motion sequence corresponding to given linguistic sequence even if the motions or sequences were not included in the training data, and vice versa.**

I. INTRODUCTION

Language is a powerful tool in human communication because it can work as a static/definite symbol and/or dynamic/contextual symbol. However, the *symbol-grounding problem* proposed by Harnad is problematic because of these properties of language [1]. This is due to the ambiguity between actual actions/movement and symbols, that is, there is a many-to-many mapping between actual actions/movement and symbols. Ogata et al. proposed the use of the *quasi-symbols* to encode dynamic attractors as an approach to this problem [2]. Quasi-symbols can be automatically extracted by using a recurrent neural network (RNN), and be handled like real symbols in human-robot communication. However, little research has been under taken to investigate the relationship between quasi-symbols and natural language.

For example, the motions represented by “move the arm to center.” are different if the arm is on the right or left. The meaning of a sentence can vary significantly depending on the context in which they are used. This means that dynamical representation of language is key issue when translating sentences into motions.

We can describe a motion by using different sentences. We can also use multiple sentences to represent a motion by dividing the motion into stages. This means that articulating and allocating the actual actions/movement is the key for translating motions into sentences.

T. Ogata, M. Murase, K. Komatani, and H. G. Okuno is with the Department of Intelligence Science and Technology, Graduate School of Informatics, Kyoto University, Kyoto, Japan {ogata, komatani, okuno}@kuis.kyoto-u.ac.jp

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

Sugita et al. proposed a method to integrate robot motions and sentences using two RNNs that bind a part of the input layers [3]. They taught a robot nine motions with 18 corresponding sentences. The generalization ability of the RNNs enabled the robot to acquire many-to-many mappings between motions and sentences, even though some of them were not included in the training set. However, they did not deal with the problem of articulating motions and allocating sentences. The motions and sentences were prepared in advance. Moreover, they only dealt with translating sentences into motions and not translating motions into sentences.

In this paper, we present a method that solves articulating and allocating problems by using the prediction error obtained from RNNs trained with motions and sentences. Our method enables robots to translate motions into arbitrary numbers of sentences, and vice versa. Our method also enables new motions and new sentences to be created based on the generalization ability of the neural networks.

The rest of the paper is as follows: Section II discusses the validity of the use of RNN to deal with language by referring to Elman’s studies. Section III describes the detail of the RNN model used in our architecture. The model can articulate a robot motion by using the prediction errors obtained from the RNN and the number of given sentences. Section IV describes the learning methods used to acquire dynamic representation of motions and sentences. Section V proposes the methods used to translate motions into sentences, and vice versa. Section V explains the experimental task in which a humanoid robot moves its arm on a table and shows the results translating motions into sentences, and vice versa. Section VI concludes the paper with a summary of the key points and a look at our future work.

II. LANGUAGE LEARNING USING NEURAL NETWORKS

An essential problem when computation systems handle language and physical phenomena is a property difference between them. In general, language is represented as a stochastic system, and a physical phenomenon is represented as a dynamical system. To combine them effectively, these representations must be the same. Therefore, we used a RNN, one of the dynamical systems, to represent both systems.

Elman et al. showed that a RNN has the ability to acquire grammar regulations [4-8]. The RNN was trained to predict the next word from in a sentence from the previous word. Result have generally shown that RNN can accept sentences that follow *regular grammar* conventions and sentences that follow *context–free* conventions, even though this network has sentence length limitations. Results have also shown that an Elman type RNN can deal with *sequences of tenses* and idiom sets that appear at distant positions in the same sentence. These reports imply that a RNN can be used to treat both language and robot motions.

III. RNNPB and Articulating Time Sequence Data

A. RNNPB Model

The recurrent neural network with parametric bias (RNNPB) model proposed by Tani and Ito is can predict which input is a current state-vector and which output is a next state-vector [9]. The model articulates complex motion sequences into motion units, which are encoded as limit cycling dynamics and/or fixed-point dynamics of the RNN. The model we used has the same architecture as the conventional Jordan-type RNN model [10], except that it has PB nodes in the input layer. Jordan-type RNN output vectors were generated from input vectors and context vectors with a recursive connection. Unlike other input nodes, the PB nodes have a constant value throughout each time sequence. They are used to implement mapping between fixed length values and time sequences. The network configuration of the RNNPB model is shown in Figure 1.

Like the Jordan-type RNN model, the RNNPB model learns data sequences in a supervised manner. The difference is that in the RNNPB model, the values that encode the sequences are self-organized in the PB nodes during the learning process. The common structural properties of the training data sequences are acquired as connection weights by using the back-propagation-through-time (BPTT) algorithm [11], and the specific properties of each individual time sequence are simultaneously encoded as PB values. As a result, the RNNPB model self-organizes the mapping between the PB values and the time sequences.

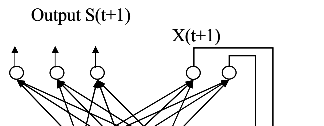
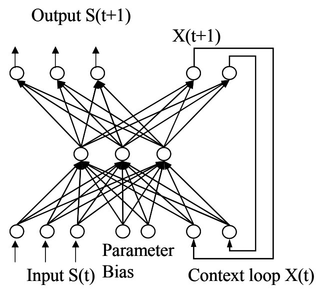

Figure 1 Network configuration of RNNPB model

B. Learning of PB Vectors

The learning algorithm for the PB values is a variant of the BPTT algorithm in which T denotes the step length of a sequence. For each sensory-motor output, the back-propagated errors with respect to the PB nodes are accumulated and used to update the PB values. The ith unit of the parametric bias is updated by using the following equations:

$$\delta\rho_i = \varepsilon \sum_{t=0}^{T} \delta_t^{bp^i} \qquad (1)$$

$$p_t = sigmoid(\rho_t / \zeta) \qquad (2)$$

In Eq. (1), δ_t^{bp} represents the delta error back propagated from the output nodes to the PB nodes and is integrated over period T steps. Internal value ρ_t is updated using the delta force, as shown in Eq. (2). The ε and ζ are learning coefficients. The current PB values are obtained from the sigmoidal outputs of the internal values.

In the generation mode, the PB value for a desired sequence is set to the PB node. The desired sequence is obtained by performing a forwarding-forward calculation called a *closed-mode* for RNNPB [12]. In the mode, the output in step $t-1$: $S(t-1)$ is the input data in the step t: $S(t)$. The input/output layer plays the same role as the context layer in the *closed-mode*.

The RNNPB model can also be used in recognition processes and in sequence generation processes. For a given sequence, the corresponding PB value can be obtained by using the update rules for the PB values without having to update the connection weight values. This inverse operation in generation is regarded as recognition.

C. Articulating Sequence Data

As mentioned in Section I, essential issues for combining language and robot motions are articulation and allocation.

One way to combine language and motions is based on using motion trajectories. Such methods usually cut off the parts of trajectories where the motion velocity is close to zero. Each part is approximated as a straight line, circle, and/or spline function. This technique is especially practical, when humanoid robots imitate motions, which has morphology quite similar to that of humans. However, this technique is unsuitable for our aim because it depends on static features. The motion trajectory must be articulated by using dynamical features so that its meaning can be changed based on the context used in the sentences.

Another way is based on the dynamics used to generate the trajectories. Systems using this approach usually consist of dynamic recognizers that predict the target sequences. The motion is articulated based on the predictability of the recognizer. The method we used to articulate a robot's motion using the prediction error of RNNPB model and the number of given sentences is described as follows: We consider the problem of articulating a motion sequence, $X(t)$, of which length is T into N sections, which are represented as $S_0, S_1, ..., S_{N-1}$. The boundary time between S_{i-1} and S_i is represented by $t = s_i$, that is, S_i is defined as $[s_i, s_{i+1}]$.

Step 1: Initializing

The given motion sequence is divided into N sections. Each section has the same length.

$$s_i \leftarrow T \cdot i / N \qquad (i = 0, ..., N) \qquad (3)$$

Step 2: RNNPB Training

The connection weights and PB values of the RNNPB model are updated with the given sequence, while the PB values keep constant in each section, S_i.

Step 3: Calculating of prediction errors

In each S_i, the prediction errors of the RNNPB model $P(t)$ are calculated, and the maximal error E_i is obtained as follows:

$$E_i \leftarrow \max_{t \in S_i} \| X(t) - P(t) \| \qquad (i = 0, ..., N) \qquad (4)$$

Step 4: Updating the length of each section

The boundary time, s_i, is updated by using the following rules:

$$s_{i+1} \leftarrow \begin{cases} s_{i+1} - ds & if\ E_i \geq E_{i+1} \\ s_{i+1} + ds & if\ E_i < E_{i+1} \end{cases} \qquad (5)$$

Where, ds is the parameter used to update the section length.

Step 5: Repeating Steps 1 to 4 until the whole error is less than the threshold.

If a motion is generated by using simple dynamics, the prediction error of the RNNPB is small, even when the PB values are fixed. However, if a motion is generated by using multiple dynamics, the prediction error at the boundary between dynamics increases as shown in Fig. 2. The developed algorithm can decrease the error by modifying the position of each boundary. The annealing technique used to decrease ds based on the increase of learning steps is implemented for stable learning.

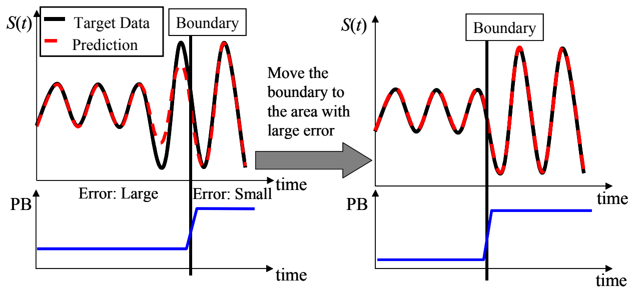

Figure 2 Articulating into multiple sequences

IV. Learning to Combine Motions and Sentences

A. Two RNNPBs for Motions and Sentences

Similar to Sugita, we use two RNNPBs to combine robot motions and sentences. One RNNPB, called behavioral RNNPB, is used to train the robot motion, and the other RNNPB, called linguistic RNNPB, is used to train the multiple sentences that explain the motion.

The training data of the behavioral RNNPB consists of a series of joint degrees and camera images downloaded into the robot. The sequence data is articulated by using our algorithm, described in Section II-C, and the number of given sentences.

Like Elman's RNN, the linguistic RNNPB predicts the words in the given sentences. All words in the given sentences correspond to nodes in the input layer. If the sentence consists of three words (move, left, right), "move" is an input vector (1, 0, 0) and "left" is (0, 1, 0). Since the given sentences are articulated by "period" in advance, the algorithm described in Section II-C was not used in the linguistic RNNPB.

These RNNPBs are bind at the PB nodes, as shown in Fig. 4. The PB values of both RNNPBs are updated by using the following equations. Here, $PB_{M,i}$ represents the ith PB node in the behavioral RNNPB, and $PB_{L,i}$ represents the ith PB node in the linguistic RNNPB.

$$\delta PB_{M,i} = \alpha \cdot error_{M,i} + \beta_M (PB_{L,i} - PB_{M,i}) \qquad (6)$$
$$\delta PB_{L,i} = \alpha \cdot error_{L,i} + \beta_L (PB_{M,i} - PB_{L,i}) \qquad (7)$$

Where, α, β_M, and β_L are the learning coefficients, and $error_{M,i}$ and $error_{L,i}$ are the delta errors at the ith PB node back-propagated in the motion and linguistic RNNPBs. The updating of the PB values enables the robot to acquire dynamic representation shared in both the robot motions and sentences.

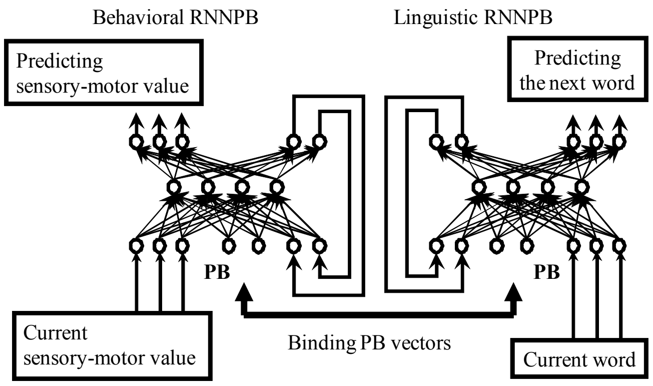

Figure 3 Configuration of RNNPBs binding the parametric bias

B. Learning Algorithm

The RNNPBs are trained with several pairs of motions M_i and sentences L_i as follows: Here, L_i consists of multiple sentences, l_1, l_2, ..., l_{ni}.

Step 1: Articulating the motion

Motion M_i is divided into the n_i sections, m_1, m_2, ..., m_{ni} by using the algorithm shown in Section II-C.

Step 2: Learning of the RNNPB models

The behavioral RNNPB is trained with m_i, and the linguistic RNNPB is trained with l_i, while both PB values are bind using equation (6) and (7).

Step 3: Repeating Steps 1 and 2 during $i = 1, 2, ..., N$, until the whole error is less than the threshold. Once this occurs, the motions are allocated to the given sentences.

V. Translating between Motions and Sentences

A. Translating Sentences into Motion

When N sentences, $l_1, l_2, ..., l_N$ are given, the motion is generated as follows:

Step 1: Obtaining the PB vector
Sentence l_i is recognized by the linguistic RNNPB, and the PB vector is obtained.

Step 2: Generating motion
The current joint angles, camera image, and obtained PB vector are set to the input and PB layer of the motion RNNPB. After initializing, performing forwarding-forward calculation of the RNNPB generates the motion corresponding to sentence l_i.

Step 3: By repeating Steps 1 and 2 for $i = 1, 2, ..., N$, the motions corresponding to L are generated.

B. Translating Motion into Sentences

When motion M and the number of sentences, N, are given, sentences L are generated as follows:

Step 1: Articulating the motion
Motion M is divided into N sections, $m_1, m_2, ..., m_N$.

Step 2: Obtaining the PB vector
The motion section, m_i, is recognized by the behavioral RNNPB, and the PB vector is obtained.

Step 3: Generating the sentence
Performing forwarding-forward calculation of the RNNPB with the obtained PB vector generates the sentence corresponding to motion section m_i.

Step 4: By repeating Steps 1, 2, and 3 for $i = 1, 2, ..., N$, the motions corresponding to M is generated.

In Step 3, there are some specific cases in which plural nodes in the output layer are simultaneously activated. For example, some motions can be described using different words of "direction" and "name of place". This ambiguity is one of the fundamental issues of symbol-grounding problems, as mentioned in Section I. In our algorithm, if n words, $w_1, w_2, \ldots, w_n$, are outputted in Step 3, an appropriate word is selected as follows:

Step 3-1: The l_i is generated by assuming that the RNNPB model only outputted w_i.

Step 3-2: The pseudo motion, $\widetilde{m}_i$, is reproduced from the l_i by using the algorithm described in Section V-A.

Step 3-3: Repeating Steps 3-1 and 3-2 for $i = 1, 2, \ldots, n$. A word, w_i, with minimal errors between $\widetilde{m}_i$ and original m_i is selected as the appropriate word.

V. Experiment for Combining Motions and Sentences

A. Experimental Setup and Training Data

We used a modified version of the Robovie-IIs humanoid robot as the platform for our experiments [13]. Robovie-IIs is a refined version of Robovie-II, which was developed at ATR [14]. The original Robovie-II has three degrees of freedom (DOF) to control its neck and four to control each arm. It also has two CCD cameras on its head.

During the experiment, the robot moved its head so as to capture data of its left-hand by using forward control with the joint angles of the arm. The sensory data collected were the area ratio of each color (red, blue, green, and white) captured by using a CCD camera with a resolution of 320 x 240 pixels (four dimensions) and the joint angles of the right arm and head (four dimensions). The data were then normalized ([0-1]) and synchronized (10 frame/s) for use by the RNNPB model.

We performed an experiment in which the robot moved its right hand to one of four areas in turn on a table; the areas were marked red (R), blue (B), yellow (Y), and white (W). Figure 3 shows an actual image of the experiment in progress. The experimental setting enables the correspondence between motions and sentences to be easily detected.

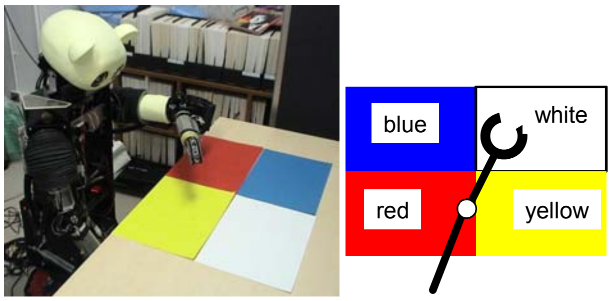

Figure 4 Image of experiment in progress and layout of color area

The behavioral RNNPB has eight neurons in the input/output layer, which are four joint angles (shoulder roll, shoulder yaw, head pitch, and head yaw) and four color areas, 50 neurons in the middle layer, and eight neurons in the context layer.

The linguistic RNNPB has 17 neurons in the input/output layer, which corresponds to certain words: {move, to, go, red, blue, yellow, white, slowly, fast, left, right, up, bottom, upper-right, upper-left, lower-left, and lower-right}. These words are translated to 17-bit vectors and set in the input layer. The RNNPB has 20 neurons in the middle layer and eight neurons in the context layer. The number of PB nodes in both RNNPB is four.

Table 1 lists the motion patterns of the robot arm and their corresponding sentences. Each motion was allocated to multiple sentences with use of "Move" or "Go" and use of direction or color. The "fast" patterns took one second to complete the motion, whereas the "slow" patterns took two seconds. Furthermore, some patterns with the same sentences had different motions, such as the first sentences in (5) and (6) in Table 1. There were many-to-many mappings between motions and sentences. Although there were 12 movement patterns between four color areas, only 10 patterns were used for training the two RNNPB models.

Table 1 Training data pairs of motion and linguistic sequences

	Motion pattern	Linguistic sequences
(1)	Y→R→B→W (slow)	Move/Go to red/left slowly. Move/Go to blue/up slowly. Move/Go to white/right slowly.
(2)	Y→R→B→R (slow)	Move/Go to red/left slowly. Move/Go to blue/up slowly. Move/Go to red/down slowly.
(3)	Y→R→Y→R (fast)	Move/Go to red/left fast. Move/Go to yellow/right fast. Move/Go to red/left fast.
(4)	Y→R→W→R (fast)	Move/Go to red/left fast. Move/Go to white/upper-right fast. Move/Go to red/lower-left fast.
(5)	Y→B→R→Y (slow)	Move/Go to blue/upper-left slowly. Move/Go to red/down slowly. Move/Go to yellow/left slowly.
(6)	Y→B→W→Y (slow)	Move/Go to blue/upper-left slowly. Move/Go to white/right slowly. Move/Go to yellow/down slowly.
(7)	Y→B→Y→R (fast)	Move/Go to blue/upper-left fast. Move/Go to yellow/lower-right fast. Move/Go to red/left fast.
(8)	Y→B→W→R (fast)	Move/Go to blue/upper-left fast. Move/Go to white/right fast. Move/Go to red/lower-left fast.

Y: yellow area, R: red area, B: blue area, W: white area

B. Result 1: Comparison with PB spaces

Figure 5 shows the distribution of the PB values for the 12 motion sequences listed in Table 1. Figure 6 shows the distribution of the PB values obtained from four sentences: "Move to red slowly.", "Move to yellow slowly.", "Move to white slowly.", and "Move to blue slowly." Two nodes were selected from a PB vector with four dimensions to coordinate on X-Y plane. The results show that the PB vectors in two spaces correspond each other appropriately.

C. Result 2: Translating sentence into motion

Figure 7 shows the three trajectories generated by the robot using the sentence, "Move to white slowly." The initial positions of the robot hand in movements 1 and 2 were in the red and blue areas. These motions were included in the training data, as listed in Table 1. The robot generated movement 3 from the yellow area, even though the trajectory was not included in the training data. The generalization ability of the RNN enabled the robot to generate unknown motions from a given sentence.

D. Result 3: Translating motions into sentences

Table 2 lists the sentences corresponding to the 24 motions generated by our method. We obtained appropriate sentences corresponding to 20 trained motions and four motions that were unknown for the robot. When the movement from blue to yellow area slowly was input to the linguistic RNNPB, for example, it generated three sentences, "Move to red slowly.", "Move to yellow slowly.", and "Go to yellow slowly." Our system eventually selected "Move to yellow slowly." by using the method described in Section V-B.

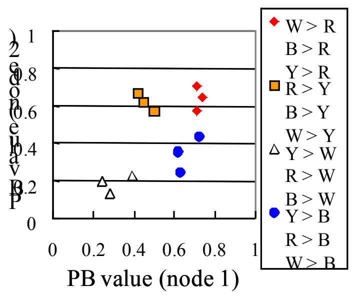

Figure 5 PB space acquired from 12 motions

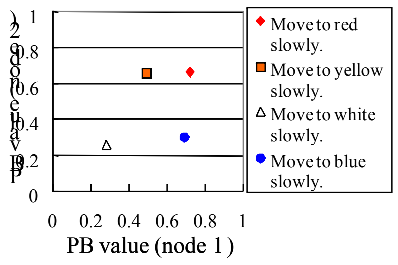

Figure 6 PB space acquired from four sentences

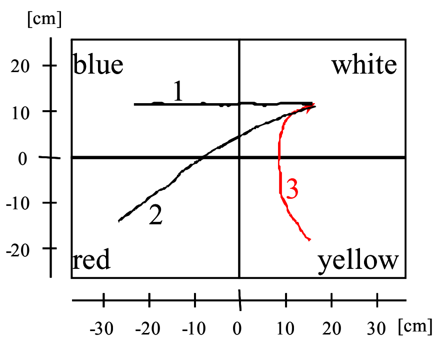

Figure 7 Motion trajectories generated from a sentence:
1: Motion from blue to white (included in training data)
2: Motion from red to white (included in training data)
3: Motion from yellow to white (not included in training data)

Table 2 Sentences generated from motion sequences
('*' denotes "not included in training data")

Motion pattern	Generated sentence
Y→R (slow)	Move to red slowly.
Y→R (fast)	Go to red fast.
Y→B (slow)	Go upper-left slowly.
Y→B (fast)	Go upper-left fast.
Y→W (slow)*	Move to white slowly.
Y→W (fast)*	Go to white fast.
R→Y (slow)	Move to yellow slowly.
R→Y (fast)	Go to yellow fast.
R→B (slow)	Move to blue slowly.
R→B (fast)	Move to blue fast.
R→W (slow)	Move upper-right slowly.
R→W (fast)	Move upper-right fast.
B→Y (slow)	Move lower-right slowly.
B→Y (fast)	Move lower-right fast.
B→R (slow)	Go to red slowly.
B→R (fast)	Move to red slowly.
B→W (slow)	Move to white slowly.
B→W (fast)	Move to white fast.
W→Y (slow)	Go to yellow slowly.
W→Y (fast)	Go to yellow fast.

W→R (slow)	Move lower-left slowly.
W→R (fast)	Go lower-left fast.
W→B (slow)*	Go to blue slowly.
W→B (fast)*	Move to blue fast.

We found trends in the words selected in the generated sentences. The words associated with area color tended to be selected in sentences corresponding to horizontal and vertical movements. Whereas, the words associated with motion direction tended to be selected in sentences corresponding to diagonal movements.

We then designed a motion that consists of two movements: from the yellow to red and from red to blue (Fig. 8). The speed of both motions was 'fast'. Here, we changed the number N of output sentences for this motion.

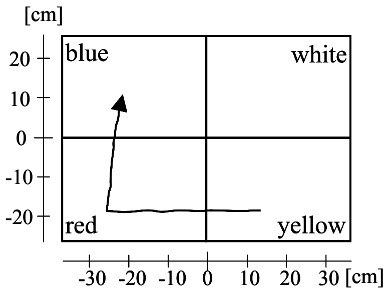

Figure 8 Motion trajectory for generating sentences

When the N was set to '2', the output sentences were as follows:

Move to red ***fast***. Move to blue ***fast***.

These sentences are suitable for matching our intuition well. An interesting output was obtained when the N was set to '1' as follows:

Move to blue ***slowly***.

When the motion was regarded as one movement, the goal color defined the sentence. Furthermore, since the motion took twice as long time as the 'fast (1 second)' motion, the word 'slowly (2 seconds)' was suitable to express the motion.

VI. Summary and Future work

We developed a method that enables a two-way translation between motions and sentences by using the behavioral and linguistic RNNPB models that contain binding PB layers. In translating sentences into motions, our method enabled the robot to generate known and unknown motions. Our method includes two essential methods: One method articulates/allocates motions using prediction errors from RNN and the number of sentences. The other method selects word to solve ambiguity in generated sentences.

Sentences were roughly categorized into 'simple sentences', 'compound sentences' and 'complex sentences.' A simple sentence has a minimal sentence unit. A compound sentence consists of multiple simple sentences and has a coordination relationship. Although complex sentence also consists of multiple sentences, they have a dependency relationship. The sentences generated by our method do not have dependency relationships and follow a time sequence. Therefore, our current method can only handle compound sentences. One area of our future works will be to enhance our method to be able to handle complex sentence. We will use our method in more complex tasks and sophisticated humanoids.

Acknowledgment

This research was 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] S. Harnad, The symbol grounding problem. *Physica D*, *42*, 335–346, 1990.
[2] T. Ogata, M. Matsunaga, S. Sugano, and J. Tani, "Human Robot Collaboration Using Behavioral Primitives," *IEEE/RSJ IROS 2004*, pp. 1592-1597, 2004.
[3] Y. Sugita and J. Tani, "Learning semantic combinatoriality from the interaction between linguistic and behavioral processes", *Adaptive Behavior*, Vol.13, No.1, pp.33-52, 2005.
[4] J. Elman et al., "A PDP approach to processing center-embedded sentences", *Proc. of the Fourteenth Annual Conference of the Cognitive Science Society, Hillsdale, NJ,* 1993.
[5] J. Elman, "Finding structure in time", *Cognitive Science*, *14*, 179–211, 1990.
[6] J. Elman, "Distributed representations, simple recurrent networks, and grammatical structure", *Machine Learning*, pp.195-225, 1991.
[7] J. Elman, "Learning and development in neural networks: The importance of starting small", *Cognition*, pp.71-99, 1993.
[8] P. Rodrigues, J. Wiles, and J. Elman, "A recurrent network that learns to count", *Connection Science*, Vol. 11, No. 1, pp. 5-40, 1999.
[9] J. Tani and M. Ito, "Self-Organization of Behavioural Primitives as Multiple Attractor Dynamics: A Robot Experiment," *IEEE Transactions on SMC Part A*, Vol. 33, No. 4, pp. 481-488, 2003.
[10]M. Jordan, "Attractor dynamics and parallelism in a connectionist sequential machine," *Eighth Annual Conference of the Cognitive Science Society* (Erlbaum, Hillsdale, NJ), pp. 513-546, 1986.
[11]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.
[12]T. Ogata, S. Sugano, and J. Tani, "Open-end Human-Robot Interaction from the Dynamical Systems Perspective -Mutual Adaptation and Incremental Learning, Advanced Robotics," VSP and Robotics Society of Japan, Vol.19, No. 6, pp. 651-670, July, 2005.
[13]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.
[14]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 Work Shop*, 2004