Proceedings of The 20th Annual Conference of the Robotics Society of Japan 2002
October 12th-14th, 2002, Osaka, Japan

Recurrent Neural Network Language for Robot Learning

Riadh Zaier *, Fumio Nagashima **
*Fujitsu Automation LTD, ** Fujitsu Laboratories LTD.

Abstract — Recurrent neural network has been used in wide range of applications based on traditional programming language such as C. However, when it comes to complex system, such as a humanoid robot, it is hard for these languages to generate the motion pattern. In this paper, therefore, we present an RNN language, suitable for the programmer to reflect the biological process, easy to implement, and it can fit well the learning process.

Key Words: Recurrent Neural Network RNN, Central Pattern Generator CPG, Humanoid robot.

1. Introduction

Humanoid robot has recently attracted many robotics research groups who developed a variety of solutions dealing with its motion and behavior. In this direction, Fujitsu has made significant efforts tending toward better mechanical structure and real-time control software as well, by making its first humanoid robot named HOAP-1 [1, 2], where user demands were considered as a prime design concern.

In this research, we focused on the biological based models using the Recurrent Neural Network (RNN) theory. Consequently, we have developed an RNN language suitable for the programmers to reflect the biological process. With this notation, the design of RNN circuits becomes simple and easy to implement. The procedure adopted to construct a large RNN circuit is reduced to a simple connection of a set of basic RNN circuits, which are found by solving simple ordinary differential equations. In contrast to the mathematical notations [3, 4], this proposed language would be expected to express the learning process of HOAP-1 easily, just by changing connections, and their weights for a given RNN circuit. The basic idea behind this proposed notation is taken from neurobiology, where the spinal cord is assumed to solve problems using 4 types of operations namely, summation, multiplication by a constant, introduction of a time delay constant, and switching. Accordingly, the proposed RNN language is limited to these four operations. An example of a basic pattern generator of sine oscillation can be obtained by a special coupling of two neurons. Using this sine pattern generator, it is possible to generate the Central Pattern Generator (CPG) of HOAP-1.

The rest of this paper is organized as follows: Next section presents the proposed RNN language. Section 3 describes some basic RNN circuits. Section 4 explains how to construct the CPG of HOAP-1, how to compensate gear backlash, and how to generate a pitching pattern. Finally, section 5 presents our conclusion.

2. Proposed RNN Language

2.1 Mathematical model of a neuron

The proposed language is a kind of ML-Style language, which is inspired from a biological process. At first we define the mathematical model of a single neuron shown in Fig.1.

$$e_i \frac{dy_i}{dt} + y_i = \sum_j c_{ij} y_j \quad , (1)$$

where y_i is the output of neuron i, ε_i is a time delay constant, and y_j is the output of neuron j that represents an input of neuron i through a weighting factor c_{ij}. Notice that ε_i can be interpreted as the rise time constant of a step input [3].

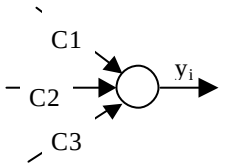

Fig. 1 Representation of a single neuron

2.2 Basic elements

To create the RNN language, we define 4 basic elements summarized in Fig.2, which are neuron, wire, cable-C, and cable-W. Their definitions are stated below

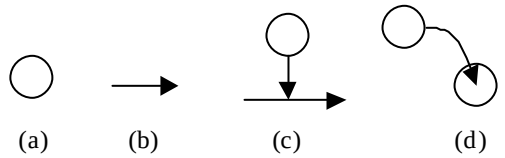

Fig. 2 Basic elements of the proposed RNN algorithm
(a): neuron, (b): wire, (c): cable-C, and (d): cable W

2.2.1 Neuron

The following is how to create a neuron with a time delay constant e and initial value V0,

```
var a(e) =  V0;
```

2.2 Wire

The function of a wire is to connect 2 neurons. For example, for a given neurons "a" and "b", neuron "a" is connected to "b" as follows

```
a := C * b;
```

where C is the wire's weighting factor.

2.2.2 Cable-C

Cable-C consists of changing other wire's weighting factor. It is created as follows,

```
var C(e) =  V0;
var a(e1);
var b(e1);
a := C * b;
```

The main purpose of this element is to activate the learning process of the system. In other words, this element belongs to the upper level layer.

2.2.4 Cable-W

The function of cable-W is to change other neuron's time delay constant.

```
var eps(e) =  V0;
var a(eps) = W0;
```

The main purpose of this element is also the same as cable-C that is to activate the learning process of the system. It belongs to the upper level layer too.

2.3 Permitted Operations

By similarity to biological process, only 4 types of operations, as shown in Fig.3, are permitted in our proposed RNN language, namely summation, multiplication by a constant, introduction of a time delay, and switching. Notice that a threshold can be considered as a switcher.

```
var v(0.0);
var w(0.0);   } Dead neurons
var y1(e1);
var y2(e2);   } Create 4 neurons
y := C1 * y1;                #Multiply by a constant
y := C1 * y1 + C2 * y2;      #Summation
v := if(0.2 < y1) 1.0 * y;   #Switcher
w := 1.0 * (0.2 <) y;        #Threshold
```

Fig. 3 Permitted operation of the proposed language

Notice that "#" is used for comment out.

2.4 RNN Circuits

A Circuit is a part of a complex system such as HOAP-1. For example, to create a circuit "Joint" we use the syntax shown in Fig. 4.

```
input in[2];
output out[1];
circuit Joint {var p; …}
out[0] := 1.0 * ::Joint::p + 1.0 * in[0] +1.0 * in[1];
```

Fig. 4 Syntax to create a circuit

Remark

Both of cable-C and cable-W involve a nonlinear operation, This does not contradict our previous definition of the proposed language, since cables belong to a layer of upper level with a different time scale.

3. Examples of Basic RNN Circuits

3.1 Generator of sine pattern

Figure 5 shows 2 examples of circuits generating a sine pattern, where in case (a) the condition $C_1C_2 < 0$ must be satisfied, yet in the case of (b) all weighting factors must have the same absolute value of $\frac{12\sqrt[3]{-108+12\sqrt{93}}}{12-(-108+12\sqrt{93})^{2/3}}$.

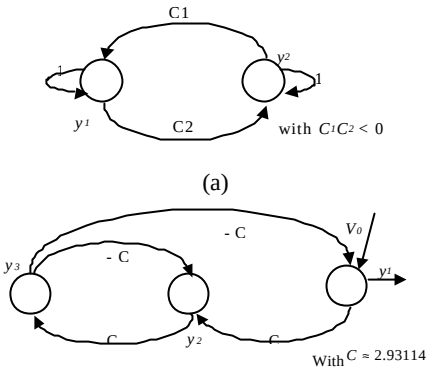

Fig. 5 Sine pattern generator

To demonstrate the result of Fig.5a, let's solve the following system of ordinary differential equations.

$$\begin{cases} e_1 \dfrac{dy_1}{dt} = C_2 y_2 & (2) \\ e_2 \dfrac{dy_2}{dt} = C_1 y_1 & (3) \end{cases}$$

By setting $\varepsilon_1 = \varepsilon_2 = \varepsilon$, the above equations will be reduced to the following

$$e^2 \frac{d^2 y_1}{dt^2} - C_1 C_2 y_1 = 0 \quad (4)$$

The general solution of eq.(4), under the condition $C_1C_2 < 0$, is given by,

$$y_1 = A_m \sin(wt + j), \quad (5)$$

where $w = \sqrt{C_1 C_2}/e$, A_m and j are parameters depending on the initial conditions.

The following is how to express the circuit of Fig.5a, using our developed RNN language

```
circuit sin1 {
   var y1(0.1) = 0.0;          # e1 = 0.1
   var y2(0.1) = 1.0;          # e2 = 0.1
   y1 := 1.0 * y1 + 1.0 * y2;  # C1 = 1.0
   y2 := -1.0 * y1 + 1.0 * y2 ; # C2 = 1.0
}
```

Fig. 6 A circuit that generate a sine pattern

3.2 Generator of polynomial pattern

Figure 7 shows how to generate a second order polynomial using 2 neurons.

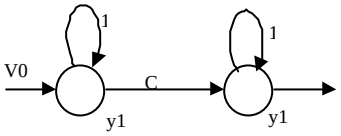

Fig. 7 Pattern generator of a second order polynomial

4. Design of HOAP-1 related circuits

4.1 Part of the CPG circuit

At first, we need to make some convention about the graphical representation of switcher, threshold, and dead neuron (a neuron with a null time delay constant), Fig.8.

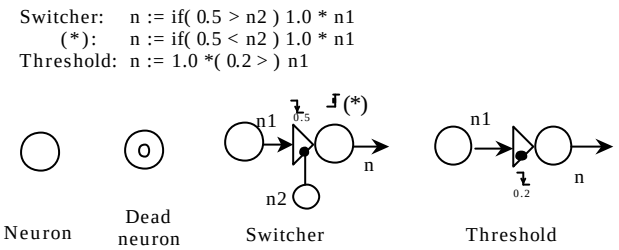

Fig. 8 Different types of neurons.

Figure 9 shows a part of HOAP-1's CPG circuit, where the gear backlashes of both legs at the hip level are compensated.

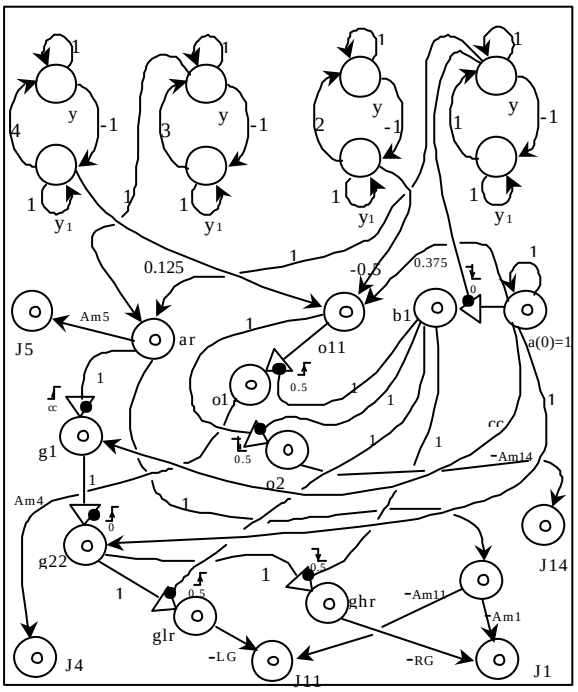

Fig. 9 Part of CPG circuit

Using the proposed RNN language, the circuit of Fig.9 can be written as shown in Fig 10. Figure 11 shows the neurons' outputs.

```
include "Cst.cpg" # file that contains values of constants
circuit sin1 {
   const C1 = 1.0;
   var y1(0.1) = 1.0;
   var y2(0.1) = 0.0;
   y1 :=   1.0 * y1 + C1 * y2;
   y2 := - C1 * y1 + 1.0 * y2;
}
circuit sin2 {the same as sin1 with C1=2.0}
circuit sin3 {the same as sin1 with C1=3.0}
circuit sin4 {the same as sin1 with C1=4.0}
# define a switcher b
var a(0.0) = 1.0;
a := 1.0 * a;
var b1;
b1 := if( 0 > ::sin1::y ) 1.0 * a;
var o11(0.0);
o11 :=0.125*::sin4::y2-0.5*::sin2::y2+0.375 * ::const1::y;
var o1(0.0);
o1 := if(0.5 < b1) 1.0 * o11 ;
var o2(0.0);
o2 := if(0.5 > b1) 1.0 * o11 ;
# rolling Ankel
var ar(0.0);
ar :=+ 1.0 * ::sin1::y        + 0.1 * ::sin3::y;
var g1(0.0);
g1 := 1.0 * (CC<) ar + CC * ::const1::y;
var g22(0.0);
g22 := 1.0 * (0.0<) g1 - CC * ::const1::y;
# gear hip right
var ghl(0.0);
ghl := if(0.5 < b1) 1.0 * g22 ;
# gear hip Left
var ghr(0.0);
ghr := if(0.5 > b1) 1.0 * g22;
var J5(0.0);
J5 := Am5 * ar;
var J4(0.0);
J4 := Am4 * o1;
# Joint J1 with gear compensation
var J1(0.0);
J1 := - Am1 * ar - RG * ghr;
var J14(0.0);
J14 := Am14 * o2;
# JOINT J11 with gear compensation
var J11(0.0);
J11 :=   - Am11 * ar - LG * ghl;
```

Fig. 10 RNN program of the circuit of

Remark

To smooth the velocity of the knee joint, we choose $(\sin\omega)^4$ as a profile of its angle position. Where ω is the frequency of the hip rolling motion in [rad/sec].

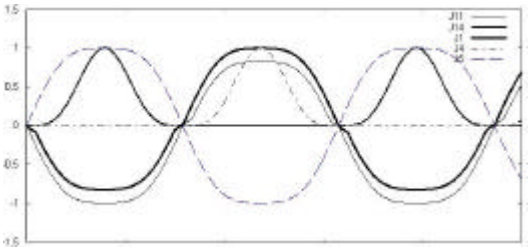

Fig. 11 Neurons Outputs using the circuit of Fig.9

4.2 Pitching pattern generator

To enable the robot to walk forward (or backward), we generate the pitching pattern of Fig.12. The stance leg has a motion profile of a third order polynomial, while the swing leg motion profile is a polynomial of first order. Figure 13 shows an example of a program that constructs the pitching pattern of Fig. 12. Notice that the rolling pattern of the hip for HOAP-1, is considered as a base motion.

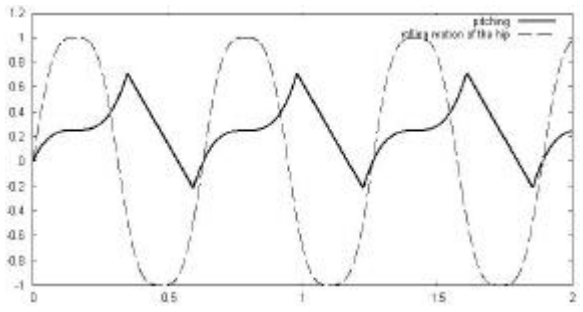

Fig. 12 Pitching pattern of HOAP-1

```
include "Cst.cpg" # file that contains values of constants
circuit sin1P {
   const C = 1.0;
   var y(0.1) = 0.9396926207859;
   var y1(0.1) = 0.342020143325;
   y   :=  1.0 * y + C * y1;
   y1 := - C * y + 1.0 * y1;
}
circuit sin2P {
   var y(0.1) = -0.9396926207859;
   var y1(0.1) = 0.342020143325;
   y   :=  1.0 * y + C * y1;
   y1 := - C * y + 1.0 * y1;
}
var a = 1.0;
 a := 1.0 * a;
#_____Switcher_______
var t1(0.0);
var t2(0.0);
t1 :=if(0 > ::sin1P::y & ::sin2P::y > 0) -1.0 * a + 0.5 * a;
t2 :=if(0 > ::sin1P::y & ::sin2P::y > 0) -1.0 * a + 0.38888 * a;
#_____________________________________
var p1(0.1) = 0.0;
p1 := 1.0 * p1+ 1.0 * t2;
#_____________________________________
var p2(0.0);
p2 :=  if(0.2 < t2) 1.0 * p1 - if(0.2 < t2) 0.611865238 * a;
#_____________________________________
var p3(0.1) = 0.0;
var p33(0.0);
var y(0.1) = 0.0;
p3 :=  1.0 * p3              + 1.0 * p2;
p33 := if(0.2 < t1) 1.0 * p3+ 0.481 * a- if(0.2 > t1) 0.8595 * a;
y := + 1.0 * y+ 1.0 * p33;
```

Fig. 13 Pitching pattern of HOAP-1

5. Conclusion

In this paper, we proposed an RNN language suitable for the programmers to reflect the biological process. It is based on 4 elements namely: neuron, wire, cable-C, and cable-W. The function of cables is to change the time delay constant of a neuron and the weighting factor of a connection between two neurons. They have different time scale compared with the elements they change.

On the other hand, similarly to biological process, this notation allows only 4 types of operations, which are summation, multiplication by a constant, introduction of a time delay constant, and switching. Using this RNN language we generated the CPG circuit of humanoid robot, and we succeeded in dealing with the gear backlash problem. Moreover, it is expected that this language can contribute well in the development of the learning process and the reflex as well, while being away from handling high-order ordinary differential equations.

The simplicity and reliability of this language in reflecting the biological model over the traditional language will be the key for the development of intelligent robot in the near future.

References

1) S. Jiang and F. Nagashima, “Biologically Inspired Spinal locomotion Controller for Humanoid Robot”, 19th Annual Conference of the Robotics Society of Japan, p. 517-518 (2001).

2) Y. Murase, Y. Yasukawa, K. Sakai, and M. Ueki, “Design of Compact Humanoid Robot as a Platform”, 19th Annual Conference of the Robotics Society of Japan, p. 789-790 (2001).

3) K. Matsuoka, “Mechanisms of Frequency and Pattern Control in the Neural Rhythm Generators”, Biol. Cybern. 56, p. 345-353 (1987).

4) G. Taga, “A model of the neuro-musculo-skeletal system for human locomotion, I. Emergence of basic gait”, Boil. Cybern. 73, p. 97-111 (1995).