https://doi.org/10.20965/jrm.2000.p0459

Paper:

Motion Planning for Six-Legged Locomotion Robot Based on Hierarchical Knowledge Using Genetic Programming

Kentarou Kurashige*, Toshio Fukuda and Haruo Hoshino*****

*Dept. of Micro System Engineering Graduate School of Engineering, Nagoya University.
Furo-cho, Chikusa-ku, Nagoya, 464-8603, Japan
**Center for Cooperative Research in Advanced Science & Technology Nagoya University
***Takenaka Corporation
Otsuka 1-5-1, Insei-shi, Chiba, 270-13, Japan
[Received: December 24, 1999; accepted: April 11, 2000]

Much research has been done on the motion planning problem. In this field, main research is to generate the motion for specific robot and task without previously acquired motions. We research the motion planning reusing knowledge. It is our objective to realize hierarchical knowledge with reuse. In this paper, we adopt tree-based representation for expressing the knowledge of the motion and adopt genetic programming as a learning method. We construct the motion planning system using hierarchical knowledge. We apply the proposed method to the six-legged locomotion robot to show its availability.

Keywords: omni-directional walking robot, motion planning problem, genetic programming, hierarchical knowledge

1. Introduction

Robotics technology is constantly improving. Robots move more variously and more complexly, and are designed for use in various circumstances. According to these, making robot motions becomes difficult. It is demanded to generate robot motions automatically.

This problem has been approached from many directions. We focus on the planning problem in which a robot receives a task from a operator and perform the task in a work space including obstacles. The core in this research is to generate motion for performing the task without human assistance.

In this work, the trajectory planning problem is used [1] to generate the sequence of angle, velocity, acceleration or so on for performing a given task. Other work uses the task planning problem[2,3] in which the robot has symbolic commands for primitive motions, and the task given by human operators is represented as the sequence of primitive motions. Many researchers confront these problems[4-12] but common problems arise. One is to work for specific tasks individually. We must research these problems on the assumption that robots can do plural tasks because of their high ability.

To overcome these problems, we study the cycle of motion planning and reuse of generated motion (**Fig. 1**). We realize reuse mechanism of motion knowledge with this cycle. We propose the hierarchical motion knowledge as the suitable structure of knowledge for this cycle. To realize reuse, we proposed motion planning based on hierarchical knowledge as shown in **Fig. 2**[13]. This system consists of a motion generator and hierarchical knowledge. Hierarchical knowledge consists of primitive knowledge and acquired motion knowledge. Here we defined primitive knowledge as the movement of each joint of a robot and defined acquired motion knowledge as the motion acquired by learning tasks. The motion generator is planning for a given task with hierarchical knowledge.

In a previous paper, we gave this system tasks such as

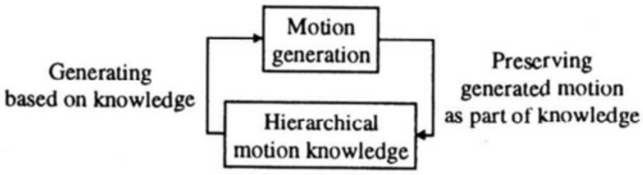

Fig. 1. Reuse mechanism of motion knowledge

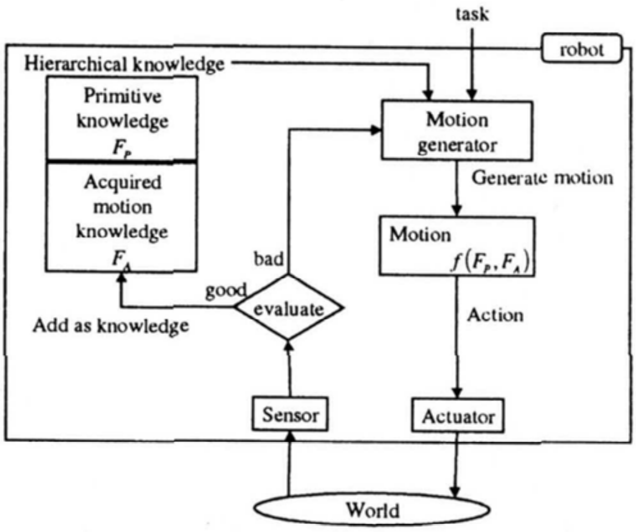

Fig. 2. Outline of motion planning system

© Fuji Technology Press Ltd. Creative Commons CC BY-ND: This is an Open Access article distributed under the terms of the Creative Commons Attribution-NoDerivatives 4.0 International License (http://creativecommons.org/licenses/by-nd/4.0/).

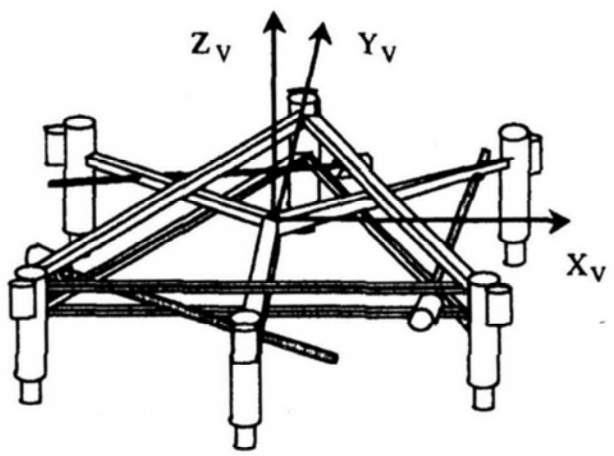

Fig. 3. Outline of six legged locomotion robot

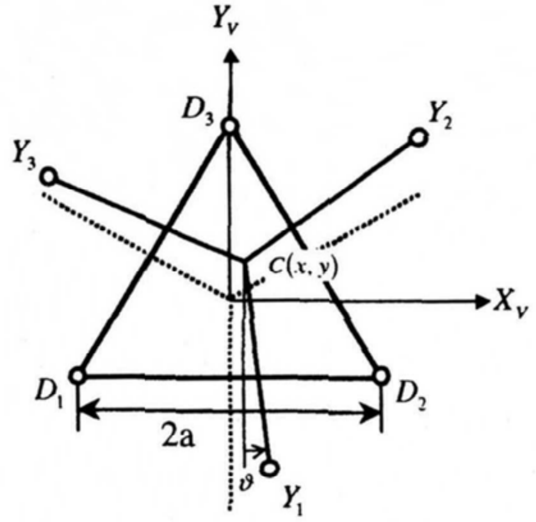

Fig. 4. Locomotion mechanism

actualized walking of a six-legged locomotion robot and these tasks are preserved as acquired knowledge. In this paper, we give additional tasks to this system and we show reuse of motion knowledge with simulation. We confirm use of hierarchical knowledge by simulation.

2. Six-legged locomotion robot

We show the six-legged locomotion robot[14-17] in **Fig.3**. This locomotion consists of a parallel link mechanism and is separated into two frames called by Δ-frame and Y-frame. By sliding two frames, this robot can walk in all directions. This robot always stands on at least three legs, so walking is stable.

Next we define the coordinates to treat states of robot as shown in **Fig. 3** and **Fig. 4**. Firstly, we define $\sum_V$ as the vehicle coordinates system fixed on the upper surface of Δ-frame. Based on this coordinates system, we define the position of center of Y-frame and the rotation angle

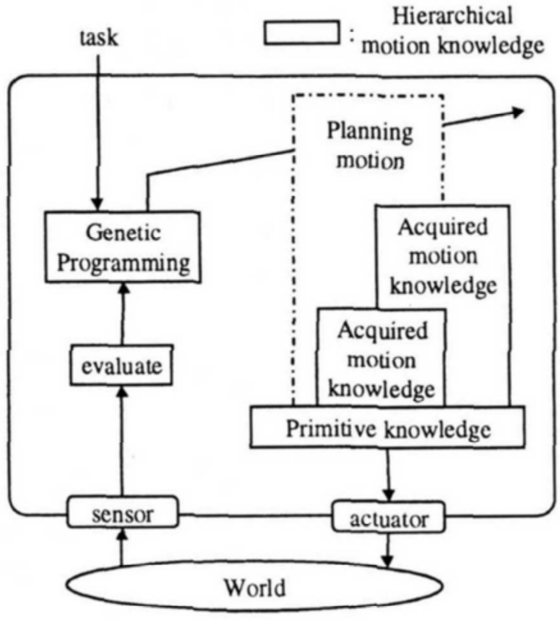

Fig. 5. Motion planning system

as follows.

Position of center of Y-frame: C(x,y)

Rotation angle: θ

And we define about the length of each leg of Δ-frame and Y-frame as follows.

Length of legs of Δ-frame: (l_{d1}, l_{d2}, l_{d3})

Length of legs of Y-frame: (l_{y1}, l_{y2}, l_{y3})

3. Adoption for robot

In this section, we talk about adoption of the supposed system for six-legged locomotion robot. In this paper, we use tree-based representation as motion and knowledge representation [17]. Genetic programming (GP) is the method of learning and inference using this tree-based representation[18,19]. GP is expansion of GA to treat structural representation.

We propose the hierarchical motion knowledge to realize the cycle of motion generation and reuse of generated motion. We must define motion generation and how to reuse generated motion. We express the hierarchical motion knowledge finally.

We define motion generation as generation of a tree for given task and define reuse generated motion as *addition of function nodes expressing acquired motion.* Corresponding to this, hierarchical knowledge is divided into primitive knowledge and acquired motion knowledge. Primitive knowledge is defined as initial knowledge provided by the operator. For a six-legged locomotion robot, we define primitive knowledge as the movement of each of its joints. Acquired motion knowledge is defined as addition of function nodes we defined previously. As a result, a new motion consists of motions generated in the past. The whole structure of motion knowledge becomes hierarchical (**Fig. 5**).

Table 1. Function node

Node	Mean	Argument
Para1	The velocity of x	2
Para2	The velocity of y	2
Para3	The velocity of ϑ	2
Delta1	The velocity of $l_{\Delta 1}$	2
Delta2	The velocity of $l_{\Delta 2}$	2
Delta3	The velocity of $l_{\Delta 3}$	2
LegY1	The velocity of l_{Y1}	2
LegY2	The velocity of l_{Y2}	2
LegY3	The velocity of l_{Y3}	2
Para	Use child at same time	2
Prog	Use child one by one	2
Loop	Use children repeatedly	2

Table 2. Input node

Node	Mean
dXd	Target speed of the center of locomotion
t	Time to move.

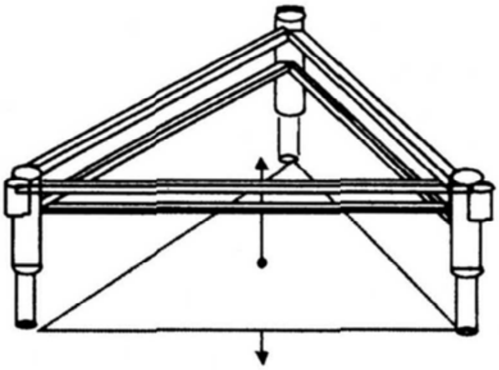

Fig. 6. Vertical movement

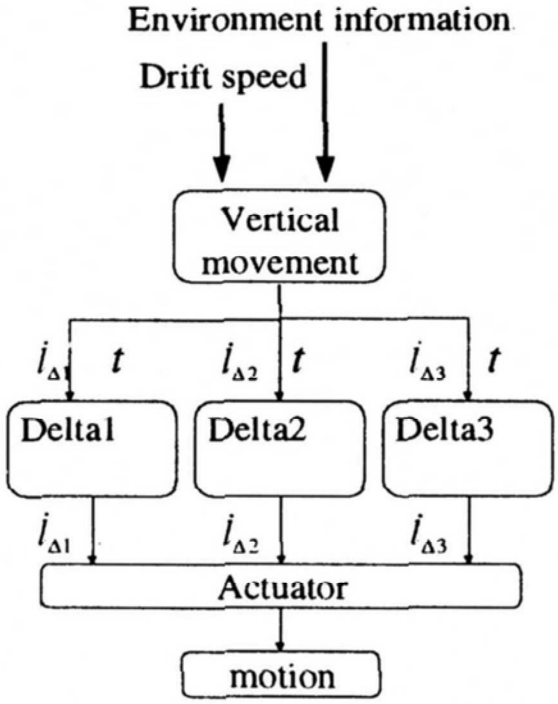

Fig. 7. Construction of vertical movement

We show the system in **Fig. 5**.

3.1. Primitive knowledge

We show the definition of primitive knowledge as function nodes for the six-legged locomotion robot as in Table 1. Function nodes we define here correspond to the robot structure and show about the motion of each joint. The argument column shows number of argument. Nodes of "Para1", "Para2", "Para3" show the motion about relative movement of Y-frame against Δ-frame. Inputs of each node are target speed and time of movement, $(\dot{x}, t), (\dot{y}, t), (\dot{\theta}, t)$. Similarly we define the nodes from "Delta1" to "LegY3" as the motion about each leg of Δ-frame and Y-frame and define argument as target speed and time of movement. Next we define the function nodes as node operator.

Function node named "Para" has two arguments. In case these arguments are the motions equal function nodes, this node executes the motions simultaneously. In case either argument shows terminal node or variable or constant number, this argument is converted to the motion expressing motion of stop. The value this argument has is utilized as the stopping time. In case both arguments arg1 and arg2 show terminal nodes, etc., this node carries out operation as follows and becomes the node showing this value.

$$value = \max(\text{arg } 1, \text{arg } 2)$$

Function node named "Prog" has two arguments. In case neither argument shows terminal nodes, etc., this node execute the motions of argument sequentially. In this case, terminal nodes, etc., is converted to the motion means stop in the same way of node "Para". In case both arguments arg1 and arg2 show terminal nodes, etc., this node carries out operation as follows and become the node shows this value.

$$value = (\text{arg1} + \text{arg2})$$

Function node named "Loop" has two arguments, arg1 as function node and arg2 as terminal nodes, etc.

This node means the motion that arg1 is executed during time of arg2. In case arg2 shows motion, function node, arg2 is converted to value as time arg2 takes. And in case arg1 shows terminal nodes, etc., arg1 is converted to the motion means stop in the same way of node "Para".

3.2. Vertical movement

We explain the motion knowledge of vertical movement acquired by a learning task. The process of acquisition of this and next knowledge of one-step walking is introduced in a previous paper[13].

We consider the vertical movement with Δ-frame or Y-frame. We talk as for max-frame after here. We can

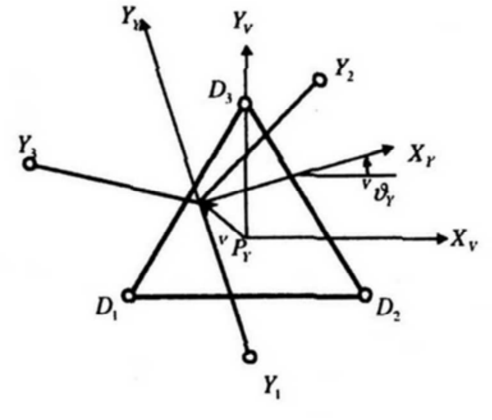

Fig. 8. One step walk

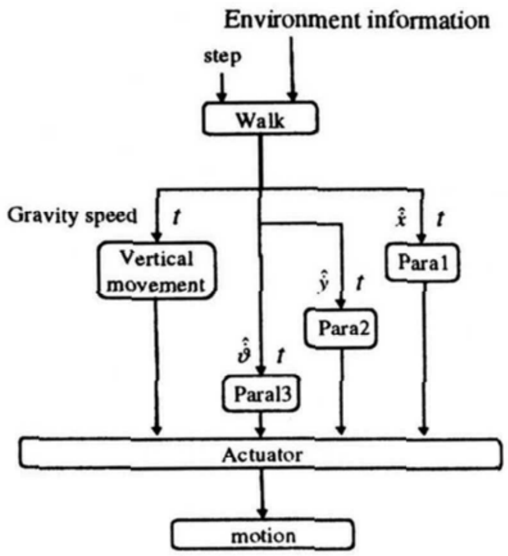

Fig. 9. Construction of one step walk

talk as for Y-frame in the same way of Δ-frame.

This motion is executed by expansion and contraction of each leg of Δ-frame. We use gravity speed of Δ-frame as the objective value about this motion. We constrain the six-legged locomotion robot to keep the upper surface horizontal for stable motion.

Input nodes for this movement are defined in **Table 2**. Node named "dXd" shows the target gravity speed and node named "t" shows the time to move.

3.3. One-step walking

Walking is basic motion of multi-legged robots. In case of a six-legged locomotion robot, walking pattern is restricted to the pattern executed by sliding Δ-frame and Y-frame and vertical movement. Here we consider one-step walking for the second step to the walking. We made the motion planning system generating one-step walking using the motion knowledge of vertical movement.

After here, we talk about the one-step walking motion of Δ-frame under the condition that Y-frame is fixed. The motion exchanging Δ-frame for Y-frame can be described in the same way.

We use the step ${}^V P_Y$ as the objective value of this motion. Here, we consider only straightforward walking, not rotational walking. The six-legged locomotion robot can move in all directions in the same way, so the step ${}^V P_Y$ is vector is used ${}^V P_Y$ based $\sum_V$ coordinates.

The walking pattern is periodic motion basically, so continuity of walking pattern is needed. Here, we represent this continuity with equality in gravity point of frame between before and after one-step walking and keeping the upper surface horizontal.

Input nodes for this movement are defined in **Table 3**. These values (*Step, Direction*) are points based on polar coordinates.

Table 3. Input node

Node	Mean
Step	Target step
Direction	*Direction of movement*

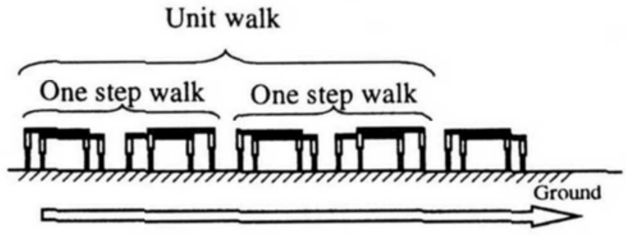

Fig. 10. Difference between one step walk and unit walk

4. Motion generation

In this paper, we give the supposed system the task of motion "unit walk" for third step to the motion "walking" using the motion acquired in past.

4.1. Unit walk

We define the motion of unit walking as the unit cycle of walking. As explaining in a previous chapter, the walking pattern of six-legged locomotion robots is restricted in the pattern executed by sliding Δ-frame and Y-frame and vertical movement. The difference between one-step walking and unit walk is whether it uses one frame or both frame as shown in **Fig.10**.

We use the gravity position $pos_d\ (x_d, y_d)$ as the transfer goal position about this task. We show the fitness function for this task in Eq.(1). The first member of this equation expresses achieving this task and second member expresses stable walking. The third member push on achieving task in less time. In this equation, max step means the largest step size of the six-legged locomotion robot.

$$fitness = k_1 \,|\, \mathrm{pos_d} - \mathrm{pos} \,| + k_2 \,|\, gh_{after} - gh_{before} \,| + k_3 time \quad (1)$$

Table 4. Function node

Node	Mean	Argument
Δ one step walk	With Δ frame	2
Y one step walk	With Y frame	

Table 5. Terminal node

Node	Mean
x_d	Goal position of x-axis
y_d	Goal position of y-axis
(variable number)	Variable number.

Table 6. GP parameter

Population size	2000
Selection method	TOURNAMENT
Tournament size	13
Crossover rate	0.2
Mutation rate	0.7

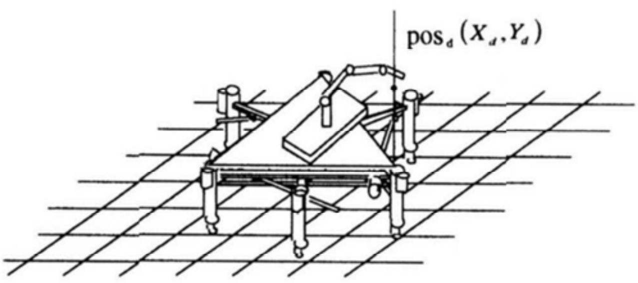

Fig. 11. Unit walk

-max step $\leq \boldsymbol{pos}_d \leq$ max step

$\boldsymbol{pos}_d$: Target gravity position

$\boldsymbol{pos}_d$: Gravity position

gh_{after} : Gravity position after movement

gh_{before} : Gravity position before movement

time : Time for moving

We show the additional function node for this task in Table 4 and show the terminal node as the inputs for this task in Table 5 . The function node "Δ one-step walking" is one-step walking by Δ frame touching Y frame on the ground and the function node "Y one-step walking" is by Y frame touching Δ frame on the ground. These function have two arguments as the input order in Table 3 . For terminal node as input for this task, we use the target transfer goal position (x_d, y_d) and we show the constraint for this robot as follow:

• The telescopic range of legs of each frame.

$0 \leq l_{\Delta i} \leq 150$ mm

$0 \leq l_{Y i} \leq 150$ mm

$l_{\Delta i}$: Leg i of Δ frame

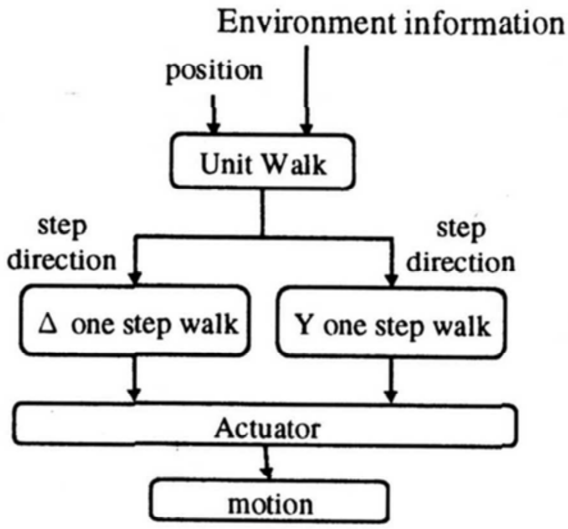

Fig. 12. Construction of unit walk

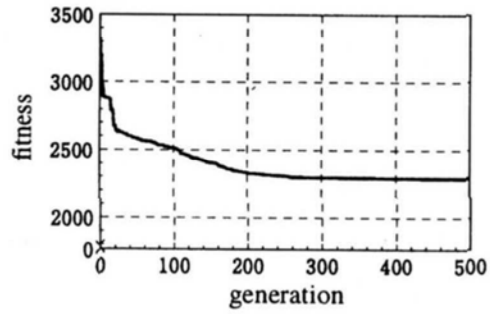

Fig. 13. Transition of fitness.

$l_{Y i}$: Leg i of Y frame $\quad i = 1,2.3$

• The distance between center of each frame.

$$0 \leq \sqrt{x^2 + y^2} \leq 144\text{mm}$$

(x , y):Center position of Y frame based on Δ frame

• The landform

$h(X,Y)=0$

$h(X,Y)$:The height of the ground in (X,Y)

4.2. Simulation results

We show the simulation results for the unit walk. Parameters of GP are shown in Table 6 . Inputs of this motion for learning are made randomly.

We show the transition of fitness in Fig.13 and show the motion of joints for unit walk given inputs as (x_d, y_d)=(60,90) in Fig.14. Here the transition of fitness does not converge to zero because the fitness function has the third member expressing the time robot moves.

5. Conclusion and projected work

In this paper, we talk about the cycle of motion planning and reuse of generated motion. With this cycle, we try to make true hierarchical motion knowledge and mo-

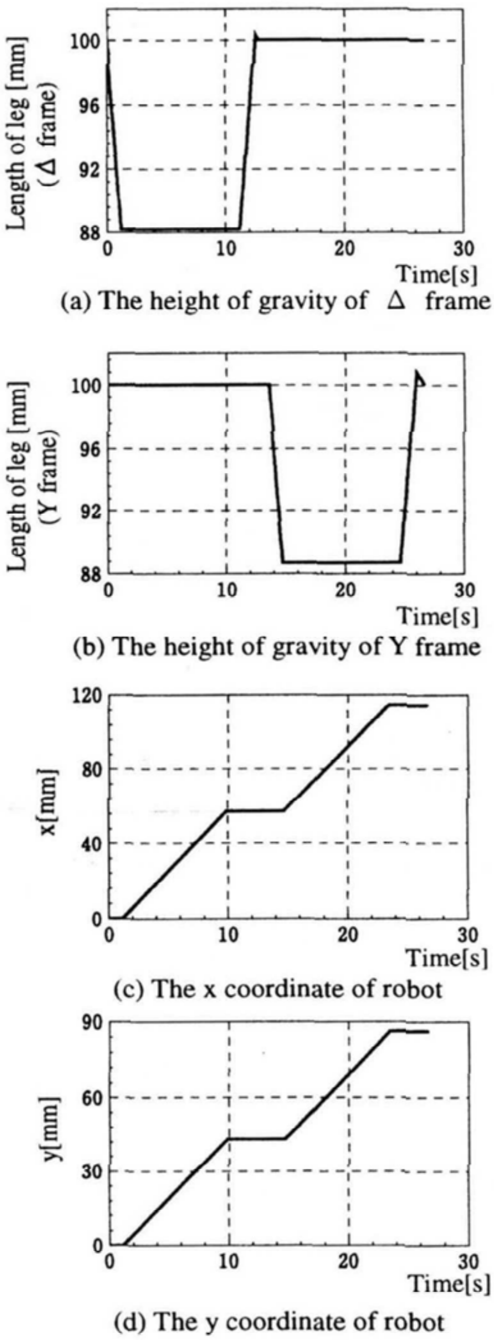

Fig. 14. Generated motion

tion generation for complex task easily. Realizing this, we use tree-based representation as motion knowledge and construct a system using tree-based representation.

With this system, we simulate motion generation hierarchically for six-legged locomotion robots using Genetic Programming. As acquired motion knowledge, the proposed system has "vertical movement" and "one-step walking" introduced previously. We give the task "unit walk" to the system with primitive and acquited motion knowledge.

In this paper, we use the tree-based representation as motion knowledge. We will consider other representation for motion knowledge in future work and try to generate other motion and study hierarchical motion knowledge thoroughly.

References

1) T. Oomichi, M. Higuchi, K. Ohnishi, Design Method of Multi-fingered Master Manipulator with Force and Tactile Feed-back Free from Operation Restriction by Mechanism, J. of the Robotics Society of Japan, Vol.16, No.7, pp.942-950, 1998
2) J. Han, W. K. Chung, Y. Youm and S. H. Kim, Task Based Design of Modular Robot Manipulator using Efficient Genetic Algorithm, Proc.1997 IEEE International Conference on Robotics and Automation, pp.507-512, 1997
3) J. P. Desai, V.Kumar, Nonholonomic Motion Planning for Multiple Mobile Manipulators, Proc.1997 IEEE International Conference on Robotics and Automation, pp.3409-3414, 1997
4) F. Saito, T. Fukuda, Reinforcement Learning for Motion Control of Real Robots, JRSJ Vol.13 No.1,pp.82?88,1995
5) T. Shibata, T. Fukuda, K. Kosuge, F. Arai, Selfish and Coordinative Planning for Multiple Mobile Robots by Genetic Algorithm, Proc. of the 31th IEEE Conf. On Decision and Control, Tucson ,Vol3, pp. 497-503,1992
6) Hsuan Chang, A New Technique To Handle Local Minimum For Imperfect Potential Field Based Motion Planning, Proc. of 1996 IEEE International Conference on Robotics and Automation,pp.108-112,1996
7) Jung. D, Gupta, KK. Octree-Based Hierarchical Distance Maps for Collision Detection, Proc. Of 1996 IEEE International Conference on Robotic and Automation, pp.454-459, 1996
8) C. W. Warren, Fast Path Planning Using Modified A* Method, IEEE Int. Conf. on Robotics and Automation, Vol.2, pp.662-667, 1993
9) J. C. Latombe, Robot motion planning, Kluwer Academic Publications, 1991
10) T. Tsubouchi, M. Rude, Motion planning for mobile robots in a time-varying environment : a survey, J. of Robotics and Mechatoronics, Vol.8, No.1, pp.15-24, 1996
11) T. Flash, N. Hogan, The coordination of arm movements, J. of Neuroscience, Vol.5, pp.1688-1703, 1985
12) R. E. Fikes, P. E. Hart, N. J. Nilsson, Learning and Executing Generalized Robot Plans, Artificial Intelligence, Vol.3, No.4, pp.251-288, 1972
13) K. Kurashige, T. Fukuda, H. Hoshino, "Motion planning based on hierarchical knowledge using Genetic Programming", Proc. of 1999 IEEE International Conference on Robotic and Automation, pp.2464-2469
14) Y. Fujisawa, H. Hoshino, T. Fukuda, K. Kosuge, E. Muro, K. Kikuchi, Omnidirectional Walking Mechanism(1st Report, Control of Moving with Coordination of Actuators), Trans. of the Japan Society of Mechanical Engineers, Vol.60, No.571, C, pp.964-969, 1994
15) T. Fukuda, Y. Adachi, H. Hoshino, K. Kosuge, E. Muro, Isao MATSUNAGA and Fumihito ARAI, Omnidirectional Walking Mechanism (2nd Report, Inclination Control while Walking on Rough Terrain), Tran. JSME, Vol.61, No.589,C(1995), pp.3620-3626
16) T. Fukuda, Y. Adachi, H. Hoshino, E. Muro, Omnidirectional Walking Mechanism(3rd Report, Redundancy and Trajectory Control), Trans. of the Japan Society of Mechanical Engineers, Vol.63, No.607, C, pp.952-959, 1997
17) Koza, J., Genetic programming, On the Programming of Computers by means of Natural Selection, MIT Press, 1992
18) David J. Montana, "Strongly Typed Genetic Programming", BBN Technical Report #7866, 1994
19) H. P. Schwefel, On the Evolution of Evolutionary Computation, IEEE Press, New York, 1994.

Name:
Kentarou Kurashige

Affiliation:
Dept. of Micro System Engineering Graduate School of Engineering, Nagoya University

Address:
1 Furo-cho, Chikusa-ku, Nagoya 464-8603, Japan

Brief Biographical History:
1996- Graduated from Nagoya University
1998- Received M.S from Nagoya University
1888- Studied at the Graduate School of Engineering, Nagoya University

Name:
Haruo Hoshino

Affiliation:
Research & Development Institute, Takenaka Corp.

Address:
Otsuka 1-5-1, Insei-shi, Chiba, 270-13, Japan

Brief Biographical History:
1974- Graduated from Science University of Tokyo
1974- Joined Takenaka Corp.

Membership in Learned Societies:
- The Robotics Society of Japan
- Architectural Institute of Japan

Name:
Toshio Fukuda

Affiliation:
Professor, Center for Cooperative Research in Advanced, Science and Technology, Nagoya University

Address:
1 Furo-cho, Chikusa-ku, Nagoya 464-8603, Japan

Brief Biographical History:
1972- Received M.S. from the University of Tokyo
1977- Received Dr. Eng. from the University of Tokyo
1979- Joined the National Mechanical Engineering Laboratory
1981- Joined the Science University of Tokyo
1989- Joined Nagoya University

Main works:
- He is mainly engaging in the research filed of intelligent robotic system, self-organizing system, mechatronics and micro robotics.

Membership in Learned Societies:
- The IEEE Industrial Electronics Society
- IFSA Vice President
- IEEE Robotics and Automation Society President