Workshop 1 in Japan: Design of Locomotion Patterns and Robot Contest on Flag Strike

Kubota Lab., Tokyo Metropolitan University, JAPAN

This workshop provides the participants with the practice on the design of locomotion patterns for multi-legged robots using ODE (Open Dynamics Engine, https://www.ode.org/). Participants don't need the high level of programming skill but they are asked to install ODE on Windows, Macintosh or UNIX PC beforehand.

0. Try It First

Try first: How to execute a sample program (time trial) [main.cpp].
refer to the Open Dynamic Engine Installation Guidelines; http://www.comp.sd.tmu.ac.jp/CcS2020/ode-installation.pdf

When you execute the generated file, a new window of the robot simulation by ODE is generated. You can control the robot by keyboard on the window.

-[ ] Space key, Change the viewpoint angle in simulation (Fig.0-1).
-[1-4] [5-8] Number key, the robot moves according to Locomotion text file (we will explain this in the following).

-[z] key, Exit the simulation with the output of data file, Pos-data.txt
-[x] key, Restart from start position.
-[v] key, Change the control method from the keyboard
(Posture-level control, Locomotion-level control and Locomotion-level Continuous Control).

You can use the posture-level control (Section 3.2) to move the robot. The robot can move forward by the slow keyboard inputs of [1] – [2] – [3] – [4], while the robot can move back by the slow keyboard inputs of [4] – [3] – [2] – [1]. The robot can turn right by the keyboard inputs of [5] – [6] – [7] – [8], while the robot can turn left by the slow keyboard inputs of [8] – [7] – [6] – [5]. The robot can take the regular position by [0].

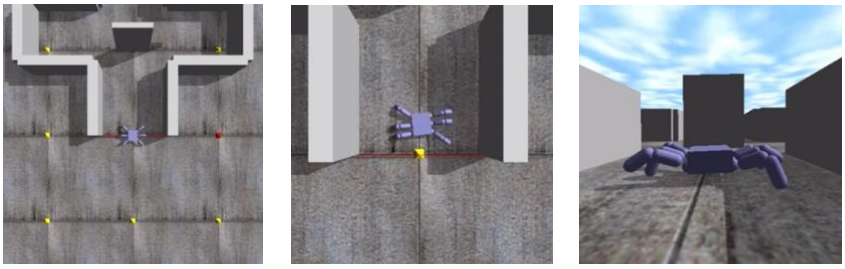


Fig.0-1. Viewpoints in the example (Time Trial)

- **Group Work** : A group involves 3-5 people (1 staff supports the development of locomotion patterns as a facilitator)

- **ODE Installation and Preliminary Execution** (Staff supports workshop participants to install ODE)

August 21 (Friday): 15:30 - 17:00 (After plenary talks)
Zoom Link: https://us02web.zoom.us/j/84777460779?pwd=emdZWkhydEtyZ2ltSFJHK0F2NkxHQT09
August 22 (Saturday): 14:00 - 14:30 (Coffee Break)
Zoom Link: https://us02web.zoom.us/j/83317809704?pwd=MUJDNi8xQjk1VE5RSFhZZnZNbjFCUT09

- **Schedule**

14:30 - 14:40 Introduction on Robotics Programming with ODE
14:40 - 15:00 One-leg Motion Planning
15:00 - 15:15 Group Discussion on the Design of Multi-legged Locomotion
15:00 - 16:00 Motion Planning (Forward, Right-turn, and Left-turn)
16:00 - 17:00 Online Robot Contests: (1) Robot Othello Game / (2) Flag Strike
Zoom Link (main): https://us02web.zoom.us/j/83317809704?pwd=MUJDNi8xQjk1VE5RSFhZZnZNbjFCUT09

Chair: WeiHong Chin (Tokyo Metropolitan University, Japan)
Commentator: Yusuke Nojima (Osaka Prefecture University, Japan)
Staffs (Tokyo Metropolitan University):
- Anom Besari (server management), - Noriko Takase - Lianchao Li
- Azhar Aulia Saputra - Fernando Ardilla

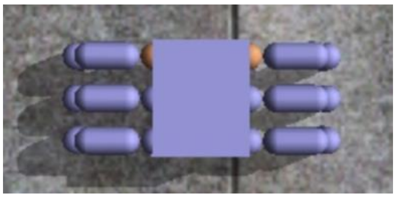
1. Introduction

Various types of robots have been applied to educational fields. Basically, there are three different aims in robot edutainment (education with entertainment). One is to develop knowledge and skill of students through the project-based learning by the development of robots (Learning on Robots). Students can learn basic knowledge on robotics itself by the development of a robot. The next one is to learn the interdisciplinary knowledge on mechanics, electronics, dynamics, biology, and informatics by using robots (Learning through Robots). The last is to apply human-friendly robots instead of personal computers for computer assisted instruction (Learning with Robots). A student learns (together) with a robot. In addition to this, such a robot can be used for supporting teachers by the teaching to students and the monitoring of the learning states of students. An educational partner robot can teach something through interaction with students in daily situation. Furthermore, the robot can observe the state of friendship among students. This is very useful information for teachers, because it is very difficult for a teacher to extract such information from the daily communication with students. The human-robot co-learning is a kind of Learning with Robots and we have developed various types of robot partners. To enhance the natural communication and interaction with students, we have to design the robot motion. Therefore, we focus on Learning through Robots in this workshop.

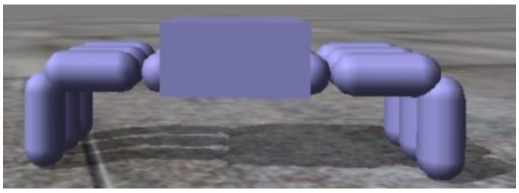
This workshop provides the participants with the practice on the design of locomotion patterns for multi-legged robots using ODE (Open Dynamics Engine, https://www.ode.org/) from the viewpoint of Learning through Robots. Basically, participants don't need the programming skill, but we assumed that participants install ODE on Windows, Macintosh or UNIX PC beforehand. First, participants learn the basic mathematical formulation of robot geometry and kinematics by trigonometric functions. Next, participants understand how to conduct multi-legged locomotion by computer simulations with ODE, and design locomotion patterns by text files as a group work. Finally, participants join a flag strike robot contest.

2. Open Dynamics Engine (ODE)

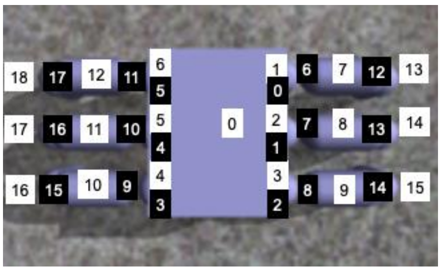
We conduct programming by using ODE (Open Dynamics Engine, https://www.ode.org/), which is available for free. ODE is a suitable platform for learning physics because it can perform various 3D dynamics simulations at high speed on Windows, Linux and Mac. Since both 3D graphics using Open GL and collision detection mechanism are included as main functions, ODE can facilitate hardware design of complicated shapes such as locomotion robots, and difficult design and motion planning of control systems with multiple degrees of freedom. In general, we try to conduct the design to complicated locomotion patterns after understanding the local physical phenomena by simulating the physical system from the design of a simple robot with few degrees of freedom. However, thanks to using ODE, it is possible for us to proceed from the intuitive design of locomotion patterns in the opposite way to the understanding of the physical system simulation and the theoretical design of robot shapes and the control system design.



(a) Top view



(b) Front view



(c) ID

(White: Parts ID, Robot_Body, Black: Joint ID, R_joint)

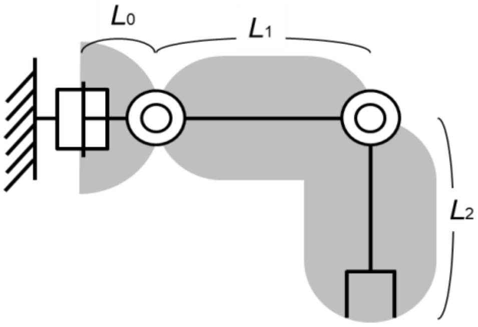
Fig.3-1 Six-legged locomotion robot

3. Multi-legged Locomotion

3-1. A Six-legged Robot Simulation

We use a six-legged robot shown in Fig.3-1. The robot has 3 joints on each leg. Each leg is assigned its corresponding identification number (white number) by clockwise from the right front leg in Fig.3-1(c). Basically, we need to set 18 joint angles to make a posture in the locomotion. The color of 1st and 6th parts of the robot is different from others for you to know the moving direction.

Kinematic model of the robot's leg is shown in Fig. 3-2. The initial posture (every joint angle is 0°) is shown in Fig.3-1 (a). The structure of each leg is the same with others. In Fig.3-2, the link length of the leg L_0 is 15 [mm], 1st link L_1 and 2nd link L_2 are 70 [mm], respectively. Each joint is assigned its corresponding identification number (black number) by clockwise from the right front leg in Fig.3-1(c). Fig. 1-3 shows an example where θ_0 is set0°, θ_6 is set 30°, and θ_{12} is set 30°. The axis of each revolute joint is as shown in Fig.3-3. The movable range is between -150° and +150 °.



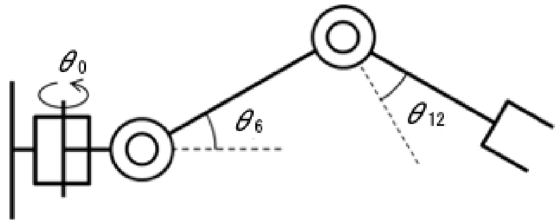


Fig.3-2 Kinematic Robot Leg Model Fig.3-3 Around the Axis (Right Front Leg)

3-2. Control of Six-legged Locomotion Robot

You can control the locomotion by keyboard inputs using the locomotion data (Loco-data.txt). We aim to design the locomotion patterns in the following. In this workshop, you can use two different types of motion control by keyboard; (A) posture-level control and (B) locomotion-level control. One number key is used to move to a single posture in (A) posture-level control, while one number key is used to take a series of postures step by step in (B) locomotion-level control.

In the beginning, you can use (A) posture-level control to confirm the design of a single posture one by one. The robot can move forward by the slow keyboard inputs of [1] – [2] – [3] – [4], while the robot can move back by the slow keyboard inputs of [4] – [3] – [2] – [1]. The robot can turn right by the keyboard inputs of [5] – [6] – [7] – [8], while the robot can turn left by the slow keyboard inputs of [8] – [7] – [6] – [5]. The robot can take the regular position by [0]. Thus, the robot can be controlled by a time series of postures.

In case of (B) locomotion-level control, one key is corresponding to the automatic change of postures used in the posture-level control. For example, [2] in (B) locomotion-level control is the sequential postures of [1] – [2] – [3] – [4] in (A) posture-level control. Furthermore, we can use (C) locomotion-level continuous control as a cyclic and continous locomotion pattern until the system receives the next keyboard input.

4. Design of Locomotion Patterns

4-1. Exercise of Posture Design

We use "Loco-data.xlsx" to support the design of "Loco-data.txt". "Loco-data.xlsx" is composed of 2 sheets: "EX (input)" and "Loco-data". If you change the values in the "EX" sheet (Fig.1-4), its change is reflected in "Loco-data" automatically.

After of changing the value, you should move to "Loco-data" sheet and select "File"→ "Save as". Then, you should choose "Loco-data.txt" as "File Type"→" tub separate text data on Excel's ribbon menu. To avoid overwriting the locomotion data file that was created in the past, please change the name of previous files as a backup beforehand. Please set text file that was made own in same directory (or folder) with the executable file.

Num of Posture	9																	
0 Key	45	0	-45	-45	0	45	20	20	20	20	20	20	0	0	0	0	0	0
1 Key	0	0	-45	0	0	45	20	20	20	20	20	20	40	0	40	0	40	0
2 Key	45	0	0	-45	0	0	20	20	20	20	20	20	40	0	40	0	40	0
3 Key	45	0	0	-45	0	0	20	20	20	20	20	20	0	40	0	40	0	40
4 Key	0	0	-45	0	0	45	20	20	20	20	20	20	0	40	0	40	0	40
5 Key	45	0	0	0	0	45	20	20	20	20	20	20	40	0	40	0	40	0
6 Key	45	45	0	-45	0	0	20	20	20	20	20	20	40	0	40	0	40	0
7 Key	45	45	0	-45	0	0	20	20	20	20	20	20	0	40	0	40	0	40
8 Key	45	0	0	0	0	45	20	20	20	20	20	20	0	40	0	40	0	40
ID of Joint	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Position	R,F	R,C	R,B	L,B	L,C	L,F	R,F	R,C	R,B	L,B	L,C	L,F	R,F	R,C	R,B	L,B	L,C	L,F
	Root						1st						2nd					

Fig.4-1 Excel（「Laco-data.xlsx」）

4-2. Exercise of Locomotion Design

The design process of locomotion patterns is divided into the following phases.

(1) Posture Design of One Leg (Trial Phase):
Focus on joints of the right-hind-leg (Black 0, 6 and 12) of the [1] key. Change their values randomly, and confirm the actual posture of their change. You can intuitively understand the relationship between joint angles and its corresponding posture by trial and error.

(2) Posture Design of Three Legs (Thinking Phase):
Focus on joints of three legs swing (e.g., left-middle-leg, right-fore-leg and right-hind-leg) of the [1] key to design . Basically, the robot can stand when three legs contact on the ground.

(3) Half cycle of Locomotion Design by two postures (Trial Phase)
Focus on the design of half cycle of locomotion by two postures of [1] and [2] keys. The position of robot should move forward after taking two postures.

(4) One cycle of Locomotion Design by four postures (Thinking Phase)
The robot should repeatedly and smoothly change half cycle of locomotion from right side to left side. Furthermore, you have to consider the continuity from the end of one locomotion step to the start of the next.

(5) Concept Design of Locomotion (Imagination Phase)
Natural creatures can take various types of locomotion patterns. This means that it is difficult for you to consider how to design. Therefore, you should consider the concept for the locomotion design, e.g., [pretty], [elegant], [modern], [fearful], [suspicious], [strange], … Decide the design concept in a group discussion.

(6) Creative Design of Locomotion (Creativity Phase)
Decide the role of locomotion design in a group; Forward [1-4] and Right-turn [5-8].

4-3. Towards Online Robot Contests

We will use locomotion-level continuous control using the file, “LocoTra-data.txt” in the online robot contest. The file is composed of a set of postures corresponding to one keyboard input from [0] to [9] at maximal in the following.

1	4																	
2	1																	
3	45	0	-45	-45	0	45	20	20	20	20	20	20	0	0	0	0	0	0
4	4																	
5	45	0	0	0	0	45	20	20	20	20	20	20	0	40	0	40	0	40
6	45	45	0	-45	0	0	20	20	20	20	20	20	0	40	0	40	0	40
7	45	45	0	-45	0	0	20	20	20	20	20	20	40	0	40	0	40	0
8	45	0	0	0	0	45	20	20	20	20	20	20	40	0	40	0	40	0
9	4																	
10	0	0	-45	0	0	45	20	20	20	20	20	20	40	0	40	0	40	0
11	45	0	0	-45	0	0	20	20	20	20	20	20	40	0	40	0	40	0
12	45	0	0	-45	0	0	20	20	20	20	20	20	0	40	0	40	0	40
13	0	0	-45	0	0	45	20	20	20	20	20	20	0	40	0	40	0	40
14	4																	
15	45	0	0	0	0	45	20	20	20	20	20	20	40	0	40	0	40	0
16	45	45	0	-45	0	0	20	20	20	20	20	20	40	0	40	0	40	0
17	45	45	0	-45	0	0	20	20	20	20	20	20	0	40	0	40	0	40
18	45	0	0	0	0	45	20	20	20	20	20	20	0	40	0	40	0	40[EOF]

Fig.4-2 Data structure for the locomotion-level continuous control

L.1	[4]	: The number of keys used for the control, e.g. 4 keys from [0] to [3] in this example.
L.2	[1]	: The number of postures used for the locomotion corresponding to the key [0].
L.3		: Posture data set (Joint angle group of 1 posture corresponding to L.2.)
L.4	[4]	: The number of postures used for the locomotion corresponding to the key [1].
L.5-8		: Posture data set (Joint angle group of 4 postures corresponding to L.4.)
L.9	[4]	: The number of postures used for the locomotion corresponding to the key [2].
L.10-13		: Posture data set (Joint angle group of 4 postures corresponding to L.9.)
L.14	[4]	: The number of postures used for the locomotion corresponding to the key [3].
L.15-18		: Posture data set (Joint angle group of 4 postures corresponding to L.14.)

In this example, the robot can move forward by the keyboard inputs [2]. The robot can turn right by the keyboard inputs of [3], while the robot can turn left by the keyboard inputs of [1]. The robot can take the regular position by [0].