Robots' Motion Acquisition in Virtual Reality by Direct Teaching

Ryunosuke TOKUHISA* Tomohisa MANABE†
Shinya MATSUTOMO† Daisuke TANAKA‡

In this paper, we propose a direct teaching system using virtual reality (VR). Direct teaching is one of the effective methods for robot motion planning. However we consider the following concerns: 1) a real robot is required, 2) a robot breaks down during teaching, and 3) the size of the robot that can be taught is limited. We have developed a system that performs direct teaching with a virtual robot in VR space to overcome these problems. By using a virtual robot, a real robot will not be required in motion planning. A virtual robot also does not have to worry about breakdowns. In addition, the robot can be resized in VR space. Using the developed system, we have verified whether the motion taught by the virtual robot is executed correctly by the real robot (NAO). In this paper, we describe the details of the developed system and the results of an experiment in which the robot acquires a throwing motion using the system.

1. Introduction

Motion planning is necessary to make the robot which performs various motions, and it could be a difficult problem in robot development. There are many kinds of robots such as mobile robots and humanoid robots. Motion planning is especially hard for humanoid robots. One factor is the degrees of freedom (DoF) of the robot. A humanoid robot with a high DoF can perform a motion close to a human and sometimes required to move like a human [1, 2]. However, a high DoF complicates its control [3]. For example, walking and throwing with a humanoid robot can be executed with complex motions. In robot development, it is also important that easily performs motion planning.

Direct teaching is a method of intuitively deciding the trajectory of a robot and it can be a way to simplify motion planning. In this teaching method, the trajectory of the motion done by the robot is shown by a *teacher* by hand. (In this paper, a *teacher* denotes a person who plans the motion of a robot.) It is possible to plan the desired motion while holding the robot arm. In other words, it is easy to teach the desired motion even if DoF is high.

In Fig. 1, it shows the flow of motion acquisition performed in this paper. Various tasks are realized for humanoid robot and robot arm etc. from the flow of direct teaching to reinforcement learning [4, 5, 6, 7].

However, we regard the following three points as serious:

1) A real robot is required.
2) During robot teaching, the robot will break down or mechanical wear will occur.
3) There is a limitation on the size of the robot to which direct teaching can be applied.

Therefore, we propose a direct teaching method that does not require a real robot. We plan motions by using virtual robots instead of real robots. It can plan motion without worrying about mechanical wear. In addition, the size of a robot is variable in the Virtual Reality (VR) space. Robots larger than humans can be easily resized to be handled by humans.

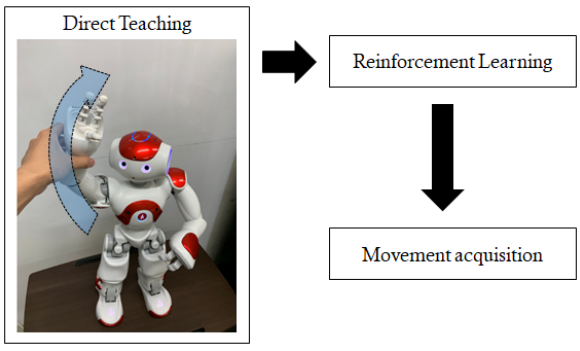


Figure 1 The flow of motion acquisition by direct teaching

令和 4 年 9 月 28 日受付 (Received Sept. 28, 2022)
* Advanced Electronic Engineering Program, National Institute of Technology (KOSEN), Niihama College, Niihama, 792–8580, Japan
† Department of Electronics and Control Engineering, National Institute of Technology (KOSEN), Niihama College, Niihama, 792–8580, Japan
‡ Department of Mechanical Engineering, National Institute of Technology (KOSEN), Niihama College, Niihama, 792–8580, Japan

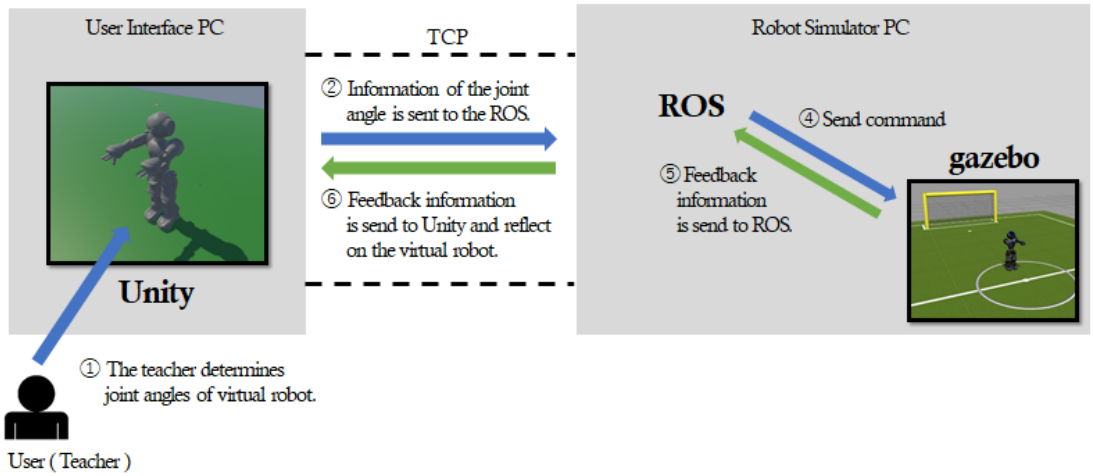


Figure 2　Overview of the system architecture. The serial number indicates the flow of angle information. First, the user moves the virtual robot in the Unity VR space to the desired position. The joint angle information is received by the ROS node, and the simulation is performed by Gazebo. Finally, the simulation result is feedback to Unity's virtual robot.

In this paper, we describe the features and configuration of the proposed system and show the results of direct teaching to a virtual robot using the system. Finally, we demonstrate the effectiveness of the proposed system showing that the trajectory taught in VR space and optimized by a reinforcement learning-like method (RL-like method) was successfully executed by a real robot.

2. Proposed System

2 – 1　VR space construction with exact robot dynamics

In this study, we used Unity to create a VR space. Unity is widely used in 3D game development. However, in VR space, there is a risk of teaching motions that cannot be executed by the real robot. The robot's 3D model used in this experiment did not have a joint angle limitation. With a virtual robot without a limit angle, a *teacher* may plan an undesired trajectory that cannot be executed by a real robot. In VR space, it is necessary to be able to plan motions without losing the characteristics of real robots as much as possible. Therefore, we use the system that feedbacks the information simulated using a robot simulator to the virtual robot in the VR space. The proposed system configuration integrating Unity and Gazebo is shown in Fig. 2. The Gazebo is an open-source robot simulator, and Robot Operating System (ROS) is a software platform for robots. In this system, ROS works on connecting Unity and Gazebo. Since the display of the VR space and the robot simulation are distributed between two PCs, a smooth VR display and real-time simulation were realized. The User Interface PC and the Robot Simulator PC are connected by Transmission Control Protocol (TCP), and mutually communicate angle information. We have created a TCP communication program using Unity C# and ROS C++. By giving the joint angle fed back to Unity, the joint angle limitation has attached to the virtual robot.

2 – 2　Equipment

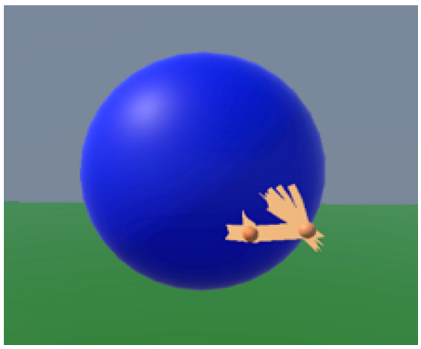


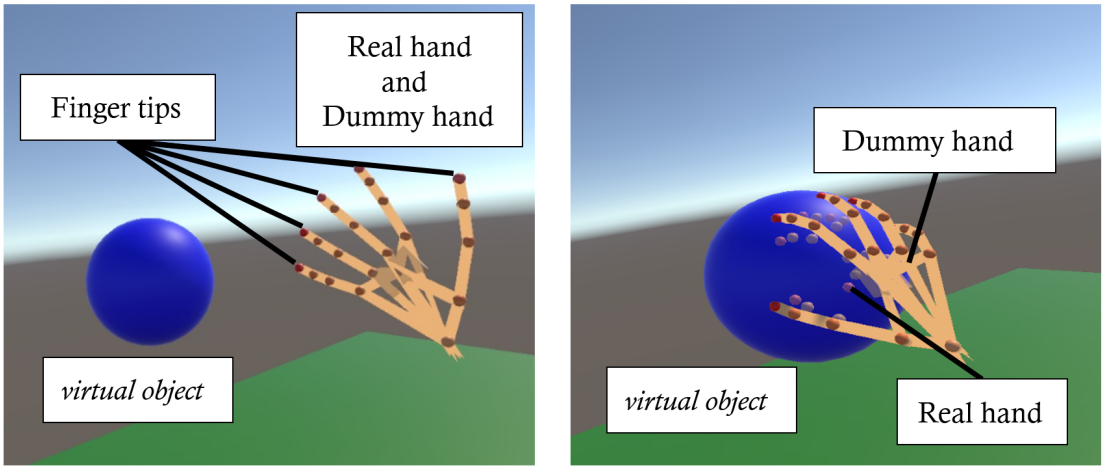
Figure 3　The virtual hand goes inside the ball.

In this study, we use HTC VIVE for the Head Mounted Display (HMD). It comes with two controllers, which can hold objects in the VR space. However, we used the method described in Section 2.3 to hold objects in VR space without using these controllers.

In direct teaching, the *teacher* teaches the trajectory by using hands. Trajectory teaching can be more intuitive by using hand instead of the controller. For this reason, we used Leap Motion to display the hands of a *teacher* in the VR space. It is possible to teach the trajectory by hand as in actual direct teaching.

2 – 3　Object Holding in VR Space

The objects held by the *teacher* in the VR space have no feeling of touch. In this paper, the objects that have no feeling of touch



(a) Before holding a *virtual object* (b) After holding a *virtual object*

Figure 4 Virtual hands by Leap Motion. It shows two states that before holding a *virtual object* and after holding a *virtual object*. Real Hands and Dummy Hands has overlapped in Fig. 4(a). Real Hands and Dummy Hands are displayed at different position in Fig. 4(b).

in VR space is defined as *virtual object*. Therefore, when holding *virtual object*, sometimes the hand model may go through surface of *virtual object* (Fig. 3). We propose a method that uses two types of hand models. One is the Real Hand model and the other is the Dummy Hand model. The Real Hand model always tracks the hands of a *teacher*. The Dummy Hand model behaves the same motion with the Real Hand model. When Finger tips touch the surface some *virtual objects*, the Dummy Hand model has fixed on the surface. If all Finger tips have touched the surface, the *virtual object* moves the Real Hand model together.

In Fig. 4, it shows virtual hands that displayed by Leap Motion and holding a *virtual object*. In Fig. 4 (a), the Real Hand model and the Dummy Hand model are combined. In Fig. 4 (b), the Real Hand model and the Dummy Hand model are separated.

In this research, we also use this method for robot teaching. We made an object for collision detection on the arm of the robot used in the experiment. When all the fingers of the *teacher's* touch the object, the *teacher's* can create an arbitrary posture of the robot using the method in this section. Note that this virtual hands is not linked to the robot's hand. This is a representation of the teacher's hand in the VR space, and is not used to create a ball-grabbing motion with the robot's hand.

3. Experiment and Result

We verified whether the developed system could teach the robot motion. The throwing motion was taught to the virtual robot in the VR space, and the real robot executed the acquisition motion. The robot used in the experiment is Aldebaran-Robotics NAO with 25 DoFs.

3 – 1 Trajectory Teaching to Virtual Robot

The throwing motion was taught to the virtual robot (NAO), and optimization was performed in a RL-like method.

(1) Teaching Trajectory

The *teacher* performs motion planning that makes the trajectory of the motion. In this experiment, the throwing motion is selected as the task. The throwing motion constructed with two motions: swinging the arm to the robot head and throwing it down.

(2) Orbit Time Function

This section describes the time function representation of time-series data. It is assumed that the trajectory of the right arm as a time function.

The function considers an n-th order polynomial that is the least squares optimal approximation. The trajectory executed by the robot is represented by the following function $p(t)$.

$$p_{\text{joint}}(t) = a_n t^n + a_{n-1} t^{n-1} + \cdots + a_2 t^2 + a_1 t + a_0 \qquad (1)$$
$$\text{joint} \in \{\text{RSP}, \text{RSR}, \text{RER}, \text{REY}, \text{RWY}\}$$

The time function of the orbit is composed of the coefficient a_n and the order n. Where a_k is a coefficient and jointis a joint name (RSP, RSR, RER, REY), and RWY stand for Right Shoulder Pitch, Right Shoulder Roll, RER is Right Elbow Roll, REY is Right Elbow Yaw, and Right Wrist Yaw respectively.

In the teaching of the throwing motion in this experiment, the motion has been realized using five joints (RSP,RSR,RER,REY,RWY). Table 1 shows the order of the time function of the trajectory of each joint.

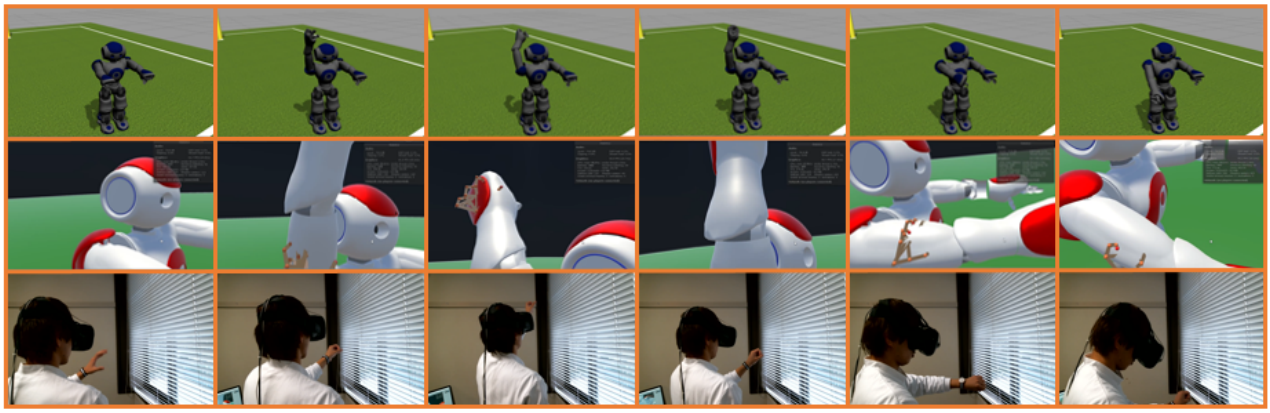


Figure 5　The state of teaching the throwing motion by using developed system. The upper part is the robot simulator (Gazebo), the middle part is the VR (created by Unity) image seen by the *teacher*, and the lower part is a state in which the *teacher* is teaching the trajectory.

Table 1　Degree of interpolation formula for each joint

Joint name	order
Right Shoulder Pitch (RSP)	5
Right Shoulder Roll (RSR)	7
Right Elbow Roll (RER)	9
Right Elbow Yaw (REY)	9
Right Wrist Yaw (RWY)	7

The coefficient a_k of the time function $p(t)$ can be obtained by creating a Vandermonde matrix and solving a linear equation ($\mathrm{a} = \mathrm{V}^{-1}\mathrm{y}$).

$$\begin{pmatrix} t_1^n & t_1^{n-1} & \cdots & 1 \\ t_2^n & t_2^{n-1} & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ t_m^n & t_m^{n-1} & \cdots & 1 \end{pmatrix} \begin{pmatrix} a_n \\ a_{n-1} \\ \vdots \\ a_0 \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix} \tag{2}$$

Where t is time, m is the number of data, n is the order of the polynomial function, and y is the numerical value of the joint angle. The time function is obtained using 213 data ($m = 213$). The time function of each joint is obtained from the time-series data and the order by executing `polyfit` in MATLAB.

(3) Trajectory Optimization in a Way Like Reinforcement Learning

This section describes trajectory optimization.

We have tried to optimize the time function and obtain the optimal trajectory. In this study, the optimality is defined as to be a trajectory with a short time from the first movement to the end. The optimal trajectory can be obtained if the time the robot executes by learning becomes shorter. A RL-like method was applied to achieve that. It is better to apply reinforcement learning such as PI^2 [8, 9, 10]. However, in this study, we use a simple learning method because to simplify learning problems. The time required to execute the pitching motion is measured and defined as the execution time. In this learning method, the execution time is defined as the time from the start to the end of the operation, and learning was performed to shorten it.

The time function is characterized by its coefficients. Changing each coefficient, the trajectory will also be changed. The changed trajectory would retrain the characteristics with infinitesimal change of each coefficient. Also, the execution time of the motion may change when the time function of the trajectory changes.

Algorithm 1 reinforcement learning-like method

Input: $p_{\mathrm{joint}}(t)$
Output: $p_{\mathrm{joint}}(t)$ (Optimized)
T_0 execution time of $p(t)$
for $j = 1$ to 300 **do**
  for $i = 1$ to 20 **do**
    $p'_{i_{\mathrm{joint}}}(t) = a'_n t^n + a'_{n-1} t^{n-1} + \cdots + a'_1 t + a'_0$
    $(a'_k = a_k + \mathrm{noise})$
    T'_i = the execution time of $p'_{i_{\mathrm{joint}}}(t)$
  end for
  $T_j = \min_i \ T'_i$
  if $T_j < T_{j-1}$ **then**
    $p_{\mathrm{joint}}(t) \leftarrow p'_{i_{\mathrm{joint}}}(t) \qquad (i = \arg\min_i T'_i)$
  else
    $T_j = T_{j-1}$
  end if
end for

Algorithm 1 shows the algorithm of a RL-like method. The inner loop performs the following processing. First, a random number ϵ_k is added to the parameter a_k of the time function $p_{\mathrm{joint}}(t)$. The time function in which a random number ϵ_k is

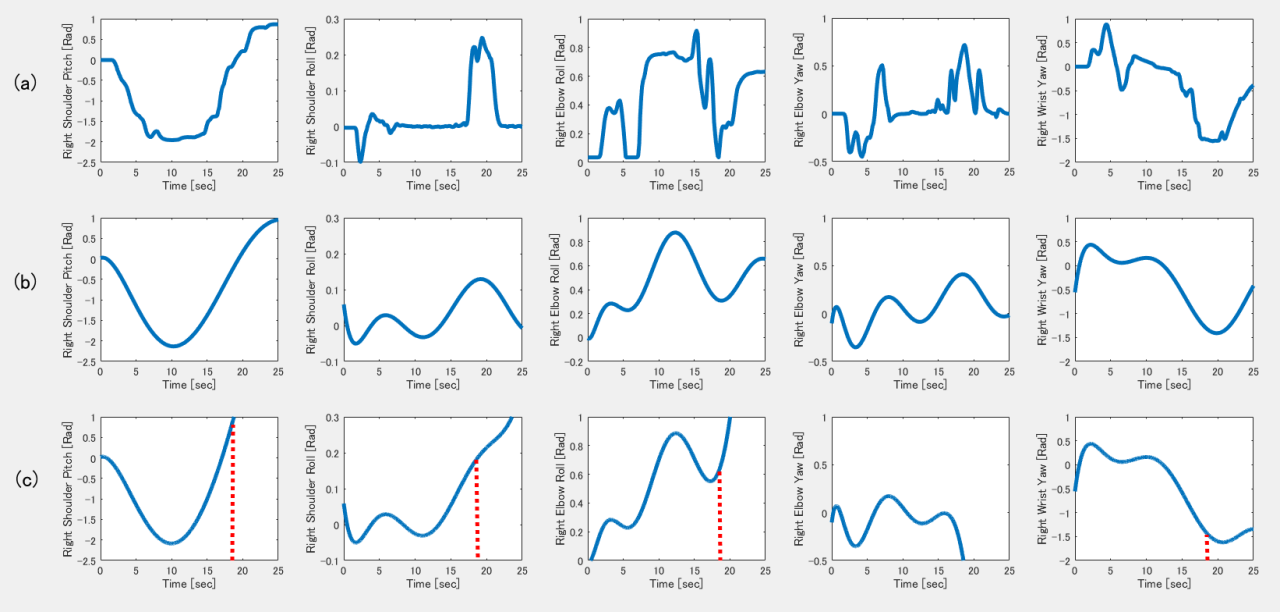


Figure 6 3 types of trajectory. (a) is the teaching trajectory, (b) is the time function trajectory, and (c) is the optimized trajectory. The red dotted line in (c) is the angle at the time when the operation is completed. Execution time is reduced by optimization.

added to each coefficient is defined as $p'_{i_{\text{joint}}}(t)$.

$$p'_{i_{\text{joint}}}(t) = (a_n + \epsilon_n)t^n + (a_{n-1} + \epsilon_{n-1})t^{n-1} + \cdots + (a_1 + \epsilon_1)t + (a_0 + \epsilon_0) \quad (3)$$
$$\text{joint} \in \{\text{RSP}, \text{RSR}, \text{RER}, \text{REY}, \text{RWY}\}$$

where i $(i = 1, 2, \cdots, 20)$ is the number of loop. Next, the time function $p'_{i_{\text{joint}}}(t)$ is executed by Gazebo (Robot simulator), and the execution time is measured. $Time'_i$ is defined as the time required to execute $p'_{i_{\text{joint}}}(t)$. In the inner loop, 20 data sets of the time function with random numbers and the time required to execute the trajectory are formed.

The following processing is performed in the outer loop. MIN_{Time} is defined as the shortest execution time among the 20 data sets. $Time$ is compared with MIN_{Time}, and if MIN_{Time} is shorter, the time function $p'_{i_{\text{joint}}}(t)$ of MIN_{Time} is substituted for $p(t)$. Otherwise, there is no change in $p(t)$. After the comparison, return to the inner loop and re-evaluate the trajectory. The time function $p(t)$ at the end of the 300 outer loops is the acquisition trajectory. This RL-like method was implemented using MATLAB.

(4) Experimental Result

In Fig. 6, it shows a trajectory that teaches the pitching motion (a), a trajectory obtained by converting the teaching data into a time function (b), and a trajectory optimized by learning (c). It can be seen that each of the five joints is represented as a time function. The learning results is shown in Fig. 7. The motion is realized by using Pitch and Yaw on the right shoulder, Roll and Yaw on the right elbow, and Yaw on the right wrist. In Fig. 5, it shows the state of teaching the throwing motion by using developed system. From Fig. 5, it can be seen that the *teacher* can teach the throwing motion without using the real robot.

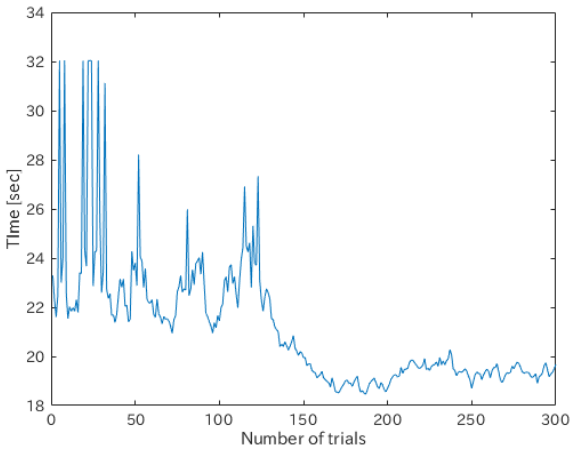


Figure 7 The state of RL-like method. The execution time before optimization is 24 seconds. It can be seen that the execution time after optimization is 19 seconds and the execution time is short.

Information of each joint angle at each time was also recorded at the same time, and time-series data of the joint angles was created. When the trajectory was executed by a robot on a robot simulator (Gazebo), the execution time was about 24 seconds. If the execution time is shorter than 24 seconds, we can regard the trajectory as better. The running time of the trajectory finally acquired by learning was about 19 seconds. Thus, we succeeded in reducing the trajectory execution time by 5 seconds using a RL-like method. However, the execution time did not become shorter than 19 seconds in 300 times of learning. The reason may be that the random number seed of the random noise was

not appropriate and that the learning efficiency was poor because it was not perfect reinforcement learning.

3 – 2 Execute the Acquisition trajectory with Real Robot

We verify that the trajectory obtained by the optimization can be executed by a real robot (NAO).

(1) Execution Environment

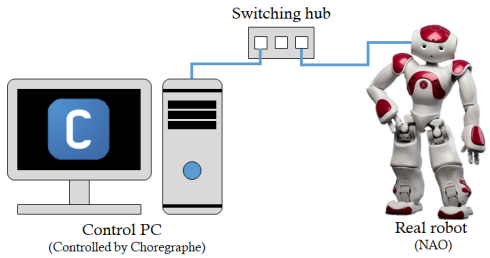


Figure 8　The environment for controlling the real robot.

The real robot executes the acquired trajectory using the environment shown in Fig. 8. The NAO control application Choregraph was installed on the control PC (Ubuntu14.04), and a program to execute the trajectory by the actual robot using Python was described. The acquired trajectory is a time function that outputs the joint angle at the time t. The output trajectory is executed by using the control PC.

(2) Experimental result

In Fig. 9, it shows the real robot performing the acquired trajectory. From left to right, the photography have been taken every 3 seconds. The throwing motion of swinging the right arm from the initial position to the back of the head and throwing it down was executed on the real robot. This result shows that our proposed system is able to perform direct teaching system that does not require a real robot.

4. Conclusions

This paper presented a direct teaching system that not requires a real robot. Our system overcomes the following concerns: 1) a real robot is required, 2) a robot breaks down during teaching, 3) the size of the robot that can be taught is limited.

We used virtual robots instead of real robots. A real robot is not required during direct teaching. And it could overcome the concern that the robot would break down. In addition, virtual robots in VR space can be resized.

We verified whether the developed system could teach the robot motion. First, we taught throwing motion with a virtual robot in VR space. Second, the change in the joint angle of the taught motion was converted into time-series data, and the trajectory of the motion was expressed as a time function. Third, we optimized the time function. Execution time was reduced from 24 seconds to 19 seconds using a RL-like method. Finally, we were able to execute the trajectory obtained by learning with the real robot (NAO). Therefore, we succeeded in developing a direct teaching system that does not require a real robot.

Future tasks include the introduction of well-known reinforcement learning method and a review of trajectory evaluation items. However, our RL-like method reduced the trajectory execution time by only 5 seconds. This learning method may be less efficient than complete reinforcement learning. Next, it is necessary to consider a method for evaluating the trajectory evaluation item that is not the execution time. NAO on the simulator (Gazebo) used in this study could not hold the ball. For this reason, we optimized the trajectory execution time. Throwing the ball farther or throwing the target position is a more practical trajectory. It is necessary that the robot holds the ball and uses these as evaluation items for acquiring the trajectory.

Bibliography

[1] Kazuo Hirai, Masao Hirose, Yuji Haikawa, and Toru Takenaka. : “The development of Honda humanoid robot”, *Proc. 1998 IEEE Int. Conf. Robotics and Automation*, pp. 1321–1326, Leuven, Belgium (1998)

[2] Kenji Kaneko, Kensuke Harada, Fumio Kanehiro, Go Miyamori, and Kazuhiko Akachi. : “Humanoid robot HRP-3”, *Proc. 2008 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pp. 2471–2478, Nice, France (2008)

[3] Shuji Kurebayashi and Monami Iguchi. : “A Study of the Effects of Teaching Computer-Aided Measurement and Control Using a 16-Degree-of-Freedom Humanoid Robot”, *Shizuoka University Faculty of Education Research Report*, No.45, pp. 117–130, (2014) (in Japanese)

[4] Jens Kober, and Jan Peters. : “Learning Motor Primitives for Robotics”, *Proc. 2009 IEEE Int. Conf. on Robotics and Automation*, pp. 2112–2118, Kobe, Japan (2009)

[5] Petar Kormushev, Sylvain Calinon, and Darwin G. Caldwell. : “Robot Motor Skill Coordination with EM-based Reinforcement Learning”, *Proc. 2010 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pp. 3232–3237, Taipei, Taiwan (2010)

[6] Jan Peters, Jens Kober, Katharina Mülling, Oliver Krämer, and Gerhard Neumann. : “Towards Robot Skill Learning: From Simple Skills to Table Tennis”, *in Machine Learning and Knowledge Discovery in Databases*, pp. 627–631, Springer, (2013)

[7] Elena Gribovskaya, and Aude Billard. : “Combining Dynamical Systems Control and Programming by Demonstration for Teaching Discrete Bimanual Coordination Tasks to a Humanoid Robot”, *Proc. 2008 3rd ACM/IEEE Int. Conf. on Human-Robot Interaction*, pp. 33–40, Amster-

Figure 9 The state of RL-like method. The right arm is swung up to the back of the head, and the action of throwing down is executed by the real robot.

dam, Netherlands (2008)

[8] Evangelos Theodorou, Jonas Buchli, and Stefan Schaal. : "Learning Policy Improvements with Path Integrals", *Proc. Int. Conf. on Artificial Intelligence and Statistics* , pp. 828–835, Sardinia, Italy (2010)

[9] Freek Stulp, Jonas Buchli, Evangelos Theodorou, and Stefan Schaal. : "Reinforcement Learning of Full-body Humanoid Motor Skills", *Proc. 2010 10-th IEEE-RAS Int. Conf. on Humanoid Robots*, pp. 405–410, Nashville, TN, USA (2010)

[10] John Schulman, Sergey Levine, Philipp Moritz, Michael Jordan, and Pieter Abbeel. : "Trust Region Policy Optimization", *Proc. the 32nd Int. Conf. on Machine Learning*, pp. 1889–1897, Lille, France (2015)