2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
Macau, China, November 4-8, 2019

Task-specific Self-body Controller Acquisition by Musculoskeletal Humanoids: Application to Pedal Control in Autonomous Driving

Kento Kawaharazuka[1], Kei Tsuzuki[1], Shogo Makino[1], Moritaka Onitsuka[1]
Koki Shinjo[1], Yuki Asano[1], Kei Okada[1], Koji Kawasaki[2], and Masayuki Inaba[1]

Abstract— **The musculoskeletal humanoid has many benefits that human beings have, but the modeling of its complex flexible body is difficult. Although we have developed an online acquisition method of the nonlinear relationship between joints and muscles, we could not completely match the actual robot and its self-body image. When realizing a certain task, the direct relationship between the control input and task state needs to be learned. So, we construct a neural network representing the time-series relationship between the control input and task state, and realize the intended task state by applying the network to a real-time control. In this research, we conduct accelerator pedal control experiments as one application, and verify the effectiveness of this study.**

I. INTRODUCTION

The musculoskeletal humanoid [1], [2], [3], [4] has many benefits such as the flexible under-actuated spine, ball joints without singular points, fingers with multiple degrees of freedom (multi-DOFs), variable stiffness control using redundant muscles, etc. At the same time, the modeling of its complex flexible body structure is difficult. To solve the problem, human-like reflex controls [5], [6] and body image acquisition methods [7], [8] have been developed. While the former is a temporary expedient to absorb the modeling error, the latter updates the self-body image online using the sensor information of the actual robot. However, even if we use the self-body image acquisition, we cannot completely match the actual robot and its self-body image. Thus, if we implement a control realizing a certain task in accordance with its modeling, the robot is hardly able to realize the task accurately.

In this study, we develop a method to realize a certain task more accurately, by acquiring the self-body controller specializing in the specific task. By using this method, we do not need to tune the controller manually, and the robot can acquire the optimal self-body controller from a few motion data.

The contributions of this study are as below.

- Discussion of the relationship between control input and state in musculoskeletal humanoids.
- The structure of DDC-Net (Dynamic Direct Control Network) representing the time-series relationship between control input and task state.
- The real-time control method realizing the intended task state using DDC-Net
- The evaluation of this method by accelerator pedal operation experiments, for autonomous driving by the musculoskeletal humanoid (Fig.1).

[1] The authors are with the Department of Mechano-Informatics, Graduate School of Information Science and Technology, The University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo, 113-8656, Japan. [kawaharazuka, tsuzuki, makino, onitsuka, shinjo, asano, k-okada, inaba]@jsk.t.u-tokyo.ac.jp
[2] The author is associated with TOYOTA MOTOR CORPORATION. koji_kawasaki@mail.toyota.co.jp

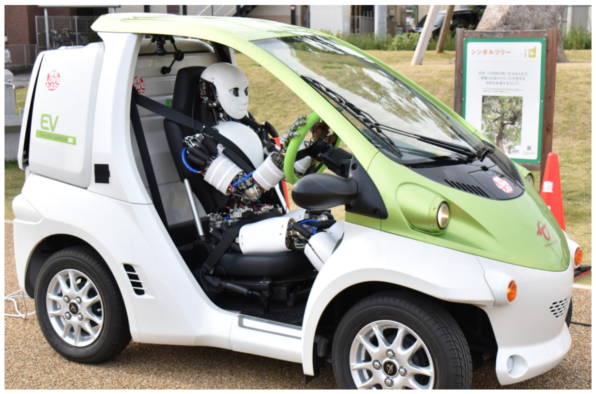
Fig. 1: Autonomous driving by the musculoskeletal humanoid, Musashi.

II. Musculoskeletal Humanoids, Self-body Image Acquisition, and Autonomous Driving by Humanoids

A. *Musculoskeletal Humanoids*

We show details of the musculoskeletal humanoid Musashi [9] used in this study (Fig.2). This is a successor of Kengoro [3], and is a modularized musculoskeletal humanoid platform composed of only joint modules, muscle modules, generic bone frames, and a few attachments. The muscles of its upper limb are composed of muscle wires using Dyneema and nonlinear elastic elements using O-ring, and there are no nonlinear elastic elements in its lower limb. Dyneema elongates like a spring, and the modeling of its friction, hysteresis, etc., is very difficult. In this study, we use only the ankle joint of Musashi.

B. *Self-body Image Acquisition*

We compare body controls between the ordinary axis-driven humanoid [10] and the musculoskeletal humanoid in Fig.3. In this comparison, we define control input and state for muscle space, joint space, and task space (exempting operational space), respectively.

Regarding the axis-driven humanoid, encoders are equipped with joints, and we execute a feedback control to match the target joint state and encoder value. So, in this

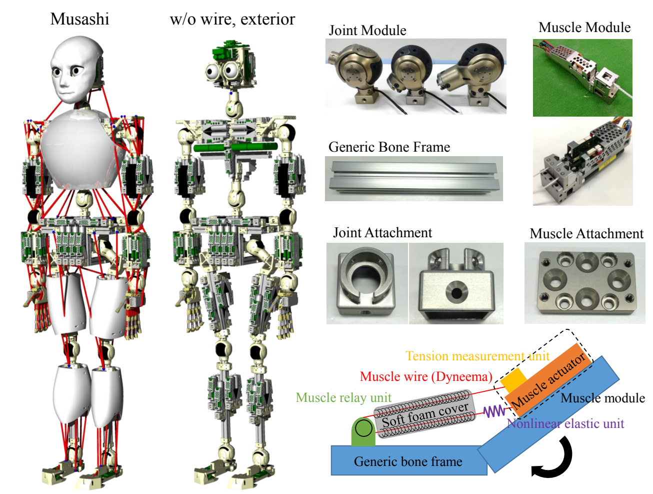

Fig. 2: Details of the musculoskeletal humanoid, Musashi.

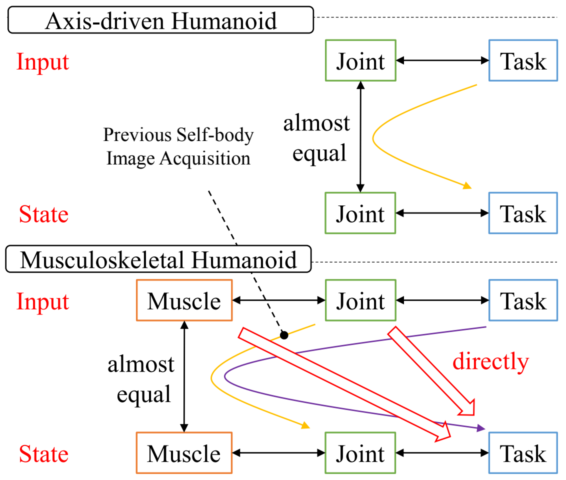

Fig. 3: Comparison of robot controls between musculoskeletal humanoids and ordinary axis-driven humanoids.

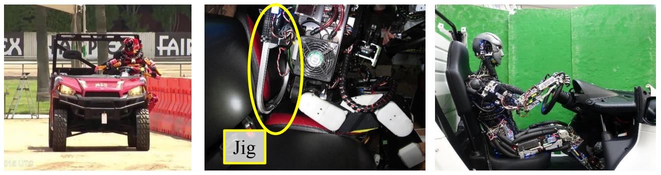

Fig. 4: Comparison of autonomous driving between using musculoskeletal humanoids and ordinary axis-driven humanoids. In DARPA robotics challenge [11], jaxon [16] needed a jig to sit on the seat.

system, we can almost completely match the control input and state. When using task space as control input, the control input in joint space is decided from the task input, the joint state follows it, and the task state changes. In this situation, the control input and state in joint space are almost equal, but the control input and state in task space will have some errors, because the task space cannot be handled directly. The task failures in DARPA Robotics Challenge [11] are similar to this phenomenon.

On the contrary, the musculoskeletal humanoid has a more complex control structure with an additional layer of muscle space. In this system, the joint space of the axis-driven humanoid and the muscle space of the musculoskeletal humanoid work almost equally, and the control input and state in those spaces match almost completely. Thus, to accurately realize the control input in joint space in the musculoskeletal humanoid is as difficult as to realize the control input in task space in the axis-driven humanoid. There are many previous studies that attempts to solve this problem. In [12], the authors construct joint-muscle mapping offline using a table of joint angles and muscle lengths of the actual robot. In [7], the authors express joint-muscle mapping by a neural network, and update it online using vision. In [8], the authors update the relationship between joint angles, muscle tensions, and muscle lengths online, considering the body tissue softness unique to the musculoskeletal humanoid. By using these methods, the musculoskeletal humanoid can realize the intended joint angles to a certain degree, and realize object manipulation. However, these methods end in the learning of the relationship between joint space and muscle space, and to handle the control input in task space is more difficult. For example, these method cannot control a trajectory of a ball when thrown, and control the car velocity when driving.

Thus, in this study, we develop a novel control method by directly representing the relationship between control input and task state. By using this method, the error intermediating multiple layers are absorbed, and the robot becomes to able to realize the task as intended. Also, while the previous studies [12], [7], [8] only handle the static state, we control the robot dynamically considering the time-series information. There are previous studies such as EMD Net [13] and SE3-Pose-Nets [14], but these studies do not handle the time-series information. Also in [15], there is no time-series output of task state and it does not consider regulations of control input.

C. *Autonomous Driving by Musculoskeletal Humanoids*

In the driving task of DRC [11], the ordinary axis-driven humanoid JAXON [16] needed a jig to sit on the car seat, as shown in Fig.4. On the contrary, the musculoskeletal humanoid has human-like proportions, so it can move in the car and adapt its flexible body to the environment. Therefore, studying autonomous driving by the musculoskeletal humanoid is important, and we handle the accelerator pedal operation in this study. In our experiments, we operate the accelerator pedal by the ankle pitch joint to control the car velocity. Also, the feedback frequency of human pedal operation is slow, the frequency being above about 3.5 Hz [17], and we control the robot by 5 Hz in this study.

III. Task-specific Self-body Controller Acquisition

We show the whole system of the task-specific self-body controller acquisition method in Fig.5. We will explain the network structure of DDC-Net, its training phase, and the control phase using it.

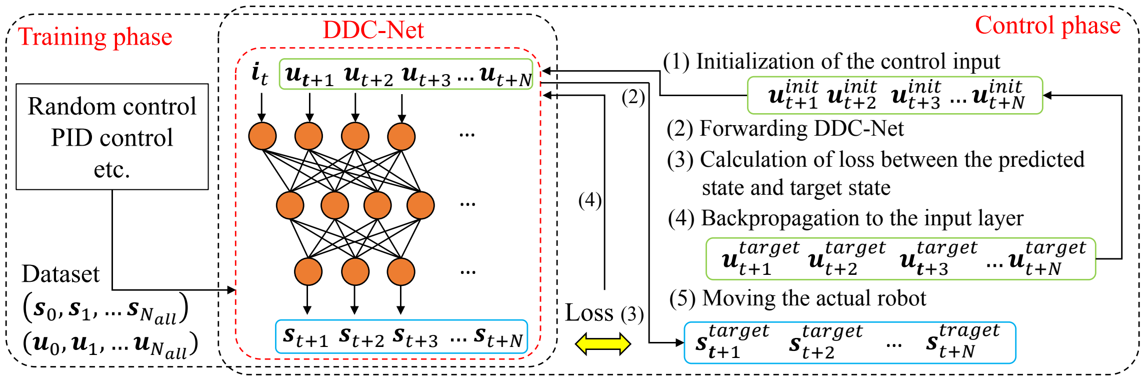
Training phase
Random control
PID control
etc.
Dataset
$(s_0, s_1, \dots s_{N_{all}})$
$(u_0, u_1, \dots u_{N_{all}})$
DDC-Net
i_t
$u_{t+1}\ u_{t+2}\ u_{t+3} \dots u_{t+N}$
$s_{t+1}\ s_{t+2}\ s_{t+3} \dots s_{t+N}$
Control phase
(1) Initialization of the control input
$u^{init}_{t+1}\ u^{init}_{t+2}\ u^{init}_{t+3} \dots u^{init}_{t+N}$
(2) Forwarding DDC-Net
(3) Calculation of loss between the predicted state and target state
(4) Backpropagation to the input layer
$u^{target}_{t+1}\ u^{target}_{t+2}\ u^{target}_{t+3} \dots u^{target}_{t+N}$
(5) Moving the actual robot
$s^{target}_{t+1}\ s^{target}_{t+2} \dots s^{traget}_{t+N}$
(2)
(4)
Loss (3)

Fig. 5: Overview of the task-specific self-body controller acquisition method.

A. Network Structure

DDC-Net in this study is equal to the function $\boldsymbol{f}$ as below:

$$\begin{aligned}\left[\boldsymbol{s}^T_{t+1} \quad \boldsymbol{s}^T_{t+2} \quad \cdots \quad \boldsymbol{s}^T_{t+N}\right]^T &= \\ \boldsymbol{f}(\left[\boldsymbol{i}^T_t \quad \boldsymbol{u}^T_{t+1} \quad \boldsymbol{u}^T_{t+2} \quad \cdots \quad \boldsymbol{u}^T_{t+N}\right]^T) \end{aligned} \tag{1}$$

where $\boldsymbol{s}$ is the task state with N_s dimensions, $\boldsymbol{i}$ is the initial state with N_i dimensions, N is the length of time-series information to consider, and $\boldsymbol{u}$ is the control input with N_u dimensions. By feeding the time-series control input to the initial state, we can predict all the time-series task states until N steps ahead. We generally set $\boldsymbol{i}_t = \boldsymbol{s}_t$, but we can increase the initial state information.

Although any network structures that express Eq. 1 are fine, we use the network with 5 fully-connected layers in this study. For all the layers except the last layer, we apply batch normalization [18] after each layer, and use sigmoid as activation functions. The dimension of the input layer is $N_i + N \times N_u$, and the dimension of the output layer is $N \times N_s$.

In our pedal operation experiment, we set $N = 30$, $N_s = 1$ (car velocity), $N_i = 4$ (car velocity, car acceleration, joint angle of ankle pitch, and joint velocity of ankle pitch), $N_u = 1$ (joint angle of ankle pitch), and the number of units of the hidden layers are $\{80, 50, 20\}$.

B. Training Phase

In the training phase, we first accumulate the motion data when controlling the robot by random control input. We can obtain the motion data of $\boldsymbol{s}$ and $\boldsymbol{u}$, convert them to the forms of $(\boldsymbol{i}_t, \boldsymbol{u}_{t+1}, \boldsymbol{u}_{t+2}, \cdots, \boldsymbol{u}_{t+N})$ and $(\boldsymbol{s}_{t+1}, \boldsymbol{s}_{t+2}, \cdots, \boldsymbol{s}_{t+N})$, and train DDC-Net (we make $\boldsymbol{i}$ from $\boldsymbol{s}$, $\boldsymbol{u}$, their differential values, etc.). In this study, we use Mean Squared Error (MSE) as the loss function, and Adam [19] as the update method. We use the 1/5 of the data as test data, and use the model with the lowest loss in 100 epoch for the control phase.

C. Control Phase

There are five steps in the control phase as below.

(1) Get the current task state and decide the initial value of the target control input.
(2) Feed them into DDC-Net.
(3) Calculate the loss between the predicted task state from DDC-Net and target task state.
(4) Backpropagate the loss to the initial value of the target control input.
(5) Repeat (2)–(4), and finally send the calculated target control input to the robot.

In (1), we decide the current state $\boldsymbol{i}_t$ and initial value of the target control input $\boldsymbol{u}^{init}_{seq} = \{\boldsymbol{u}^{init}_{t+1}, \boldsymbol{u}^{init}_{t+2}, \cdots, \boldsymbol{u}^{init}_{t+N}\}$. We set $\boldsymbol{u}^{init}_{seq}$ as $\{\boldsymbol{u}^{pre}_{t+1}, \boldsymbol{u}^{pre}_{t+2}, \cdots, \boldsymbol{u}^{pre}_{t+N-1}, \boldsymbol{u}^{pre}_{t+N-1}\}$ which shifts the optimized target control input of the previous time step $\{\boldsymbol{u}^{pre}_{t}, \boldsymbol{u}^{pre}_{t+1}, \cdots, \boldsymbol{u}^{pre}_{t+N-1}\}$ and replicates the last term $\boldsymbol{u}^{pre}_{t+N-1}$.

In (2)–(3), we first feed the obtained values from (1) into DDC-Net. Next, we calculate the loss (L) between the predicted task state $\boldsymbol{s}^{predicted}_{seq}$ and target task state $\boldsymbol{s}^{target}_{seq}$. In this study, we calculate the loss as below:

$$L = \text{MSE}(\boldsymbol{s}^{predicted}_{seq}, \boldsymbol{s}^{target}_{seq}) + \alpha \text{AdjacentError}(\boldsymbol{u}^{init}_{seq}) \tag{2}$$

where MSE is the Mean Squared Error, α is the weight of losses, AdjacentError is the MSE of each difference of target control inputs at adjacent time steps. We added the last term to decrease the changes in control input and obtain smooth control input.

In (4), we update $\boldsymbol{u}^{init}_{seq}$ by backpropagating the loss, as below:

$$\boldsymbol{g} = dL/d\boldsymbol{u}^{init}_{seq} \tag{3}$$

$$\boldsymbol{u}^{init}_{seq} = \boldsymbol{u}^{init}_{seq} - \beta \frac{\boldsymbol{g}}{|\boldsymbol{g}|} \tag{4}$$

where $\boldsymbol{g}$ is the gradient of loss(L) for $\boldsymbol{u}^{init}_{seq}$, and β is the learning rate constant. By repeating (2)–(4), we update $\boldsymbol{u}^{init}_{seq}$.

In (5), we send $\boldsymbol{u}^{target}_{t+1}$ of the finally calculated $\boldsymbol{u}^{target}_{seq}$ to the actual robot.

We mentioned the overview as above, and we add some changes to calculate the target control input by dividing (1)–(4) to two stages. In the first stage, we add random noises from $-\delta \boldsymbol{u}_{batch}$ to $\delta \boldsymbol{u}_{batch}$ in $\boldsymbol{u}^{init}_{seq}$, construct N_{batch} number of samples, and set the $N_{batch}+1$ number of samples, including the sample without noise, as the initial control input. We repeat (2)–(4) N_1 times, by setting $\beta = \beta_1$. In the second stage, we use the updated control input with the lowest loss in the first stage, as the initial control input. We set $\beta = \beta_2$ ($\beta_2 < \beta_1$), repeat (2)–(4) N_2 times, and calculate the final $\boldsymbol{u}^{target}_{seq}$.

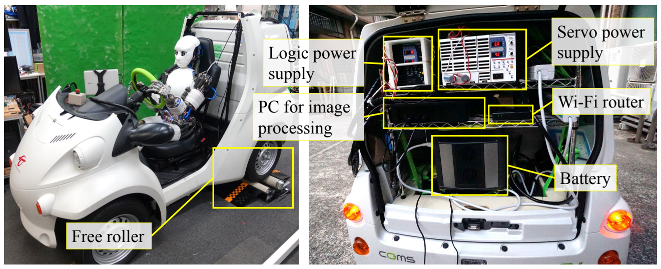

Fig. 6: Experimental setup of COMS.

Since $\boldsymbol{u}$ has the minimum and maximum values, we filter $\boldsymbol{u}$ to fit from $\boldsymbol{u}_{min}$ to $\boldsymbol{u}_{max}$. Also, before the first calculation, there are no values of $\boldsymbol{u}^{pre}_{seq}$, so we use N_{batch} number of samples as $\boldsymbol{u}^{init}_{seq}$ by filling the same random value from $\boldsymbol{u}_{min}$ to $\boldsymbol{u}_{max}$ at every time.

In our pedal operation experiment, we set $u_{min} = 0$ [deg], $u_{max} = 50$ [deg], $N_{batch} = 10$, $N_1 = 10$, $N_2 = 20$, $\delta u_{batch} = 5$ [deg], $\alpha = 30.0$, $\beta_1 = 3.0$ [deg], and $\beta_2 = 0.5$ [deg].

In this study, we use a joint angle as the control input. Therefore, after we convert the joint angle to the muscle length by [8], we send it to the actual robot. There is no need of joint angle sensors.

IV. Pedal Control Experiments

First, we will explain the experimental setup. Next, we will conduct accelerator pedal control experiments using an ordinary PID control and extended PID control with the acceleration estimation. Although various model-based control methods have been developed, we cannot apply those due to the difficult modeling of this experiment, and used PID controllers. Finally, we will conduct the developed task-specific self-body controller acquisition method, and verify its effectiveness.

A. Experimental Setup

The car used in this experiment is B.COM Delivery of extremely small EV COMS (Chotto Odekake Machimade Suisui) series. For the safety, the motor torque is limited to 5 Nm on its software, and the emergency stop button is equipped.

The outside appearance is shown in the left figure of Fig.6. Although we should obtain the current car velocity by visual odometry, etc. from image information, we conduct experiments in the indoor environment for the safety, and we obtain the car velocity from CAN-USB from ROS. The rear drive wheels are on the free roller, and the front wheels are fixed by stoppers.

Also, the inside appearance is shown in left figure of Fig.6. In the car, power supplies for the logic and motors of the robot, batteries, Wi-Fi rooter are equipped. We use special batteries for the power supply, but in the future, we examine to directly obtain the power from the electric vehicle COMS.

B. Basic Control

First, we show the result of the car velocity transition when using a basic PID control (we call it **PID1**) in Fig.7. *PID*1

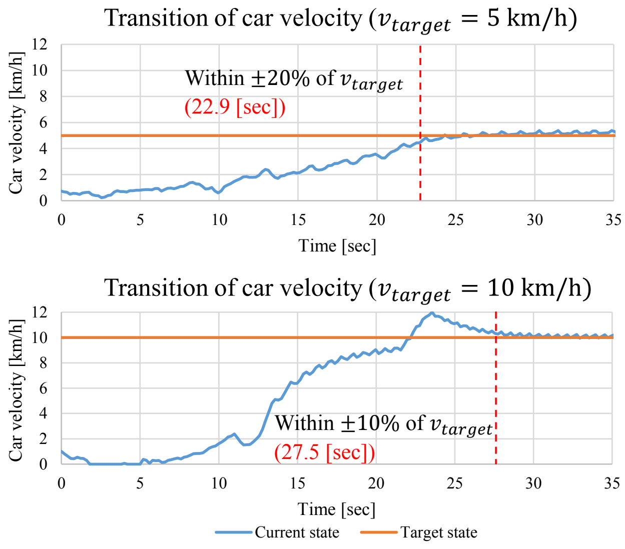

Fig. 7: The result of the basic PID control (**PID1**).

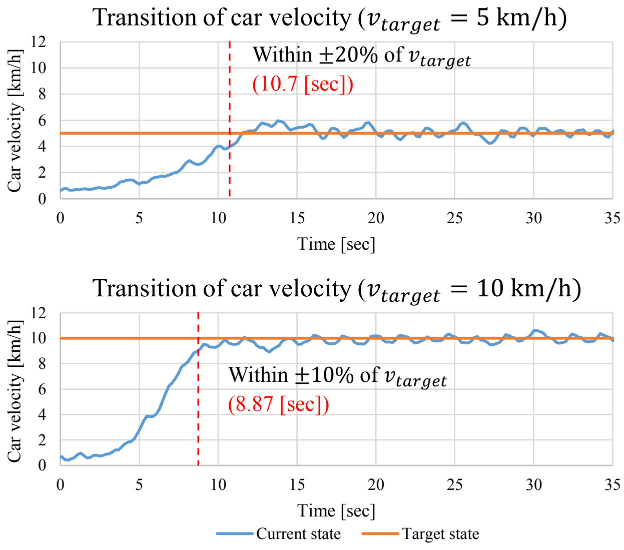

Fig. 8: The result of the PID control with acceleration estimation (**PID2**).

is the control as shown below.

$$e_v(t) = v_{target}(t) - v(t) \tag{5}$$

$$u = \theta_{ankle} = k^1_p e_v(t) + k^1_i \int e_v(t) + k^1_d \dot{e}_v(t) \tag{6}$$

Where k^1_p, k^1_i, k^1_d are P, I, D gain respectively, v is the current car velocity, and $u = \theta_{ankle}$ is the joint angle of the ankle pitch as the control input. We conduct all experiments to track the constant target car velocity of 5 km/h or 10 km/h. We can see that **PID1** is slow to follow the target car velocity from Fig.7. If we define $T^{conv}_{A\%}$ as the first time that the error between the current and target car velocity falls within A% from then, $T^{conv}_{20\%} = 22.9$ [sec] when $v_{target} = 5$ [km/h], and $T^{conv}_{10\%} = 27.5$ [sec] when $v_{target} = 10$ [km/h]. Because the tracking control is difficult when the target car velocity is slow, we basically set $A = 20$ when $v_{target} = 5$, and $A = 10$ when $v_{target} = 10$ [km/h], for the experimental evaluation.

We think that the reason of the slow tracking control by

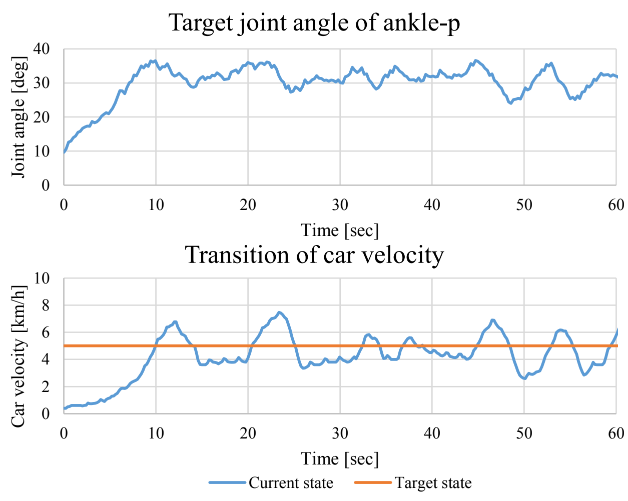

Fig. 9: The result of the random controller (**Random**).

PID1 is because the amount of pedal step actually effects not the car velocity directly but the car acceleration. So, we consider to estimate the target car acceleration and to execute PD control to the value (we call it **PID2**). In detail, we conduct a control as shown below.

$$a_{target}(t) = (v_{target}(t) - v(t))/t_{delay} \tag{7}$$

$$a(t) = (v(t) - v(t-1))/t_{interval} \tag{8}$$

$$e_a(t) = a_{target}(t) - a(t) \tag{9}$$

$$u = \int (k^2_p e_a(t) + k^2_d \dot{e_a}(t)) \tag{10}$$

Where, a_{target} is the target acceleration, t_{delay} is the time delay until realizing the target acceleration, $t_{interval}$ is the control period, a is the current car acceleration, and k^2_p, k^2_d is the P and D gain, respectively. We show the result of **PID2** in Fig.8. We can see that $T^{conv}_{20\%} = 10.7$ [sec] when $v_{target} = 5$ [km/h], and $T^{conv}_{10\%} = 8.87$ [sec] when $v_{target} = 10$ [km/h]. Compared to **PID1**, the time delay to follow the target velocity is improved in **PID2**.

C. Task-specific Self-body Controller Acquisition

We show the result of the task-specific self-body controller acquisition method (we call it **Proposed**). At first, we show the control result by the random controller as shown below (we call it **Random**), in Fig.9.

$$\begin{aligned} &if \quad v(t) \leq v_{target}(t) \\ &\quad u = u + \text{Random}(u^{random}_{min}, u^{random}_{max}) \quad (11) \\ &else \\ &\quad u = u - \text{Random}(u^{random}_{min}, u^{random}_{max}) \quad (12) \end{aligned}$$

Where $\text{Random}(u^{random}_{min}, u^{random}_{max})$ is the value ranging from u^{random}_{min} to u^{random}_{max}. In this study, we set $u^{random}_{min} = -1$ [deg] and $u^{random}_{max} = 2$ [deg]. As shown in Fig.9, the ankle joint and car velocity undulate, and we train DDC-Net by using these data. We execute this **Random** for 60 sec, so we can obtain 300 step data, because the control frequency is 5 Hz stated in Subsection II-C and $t_{interval} = 0.2$ [sec].

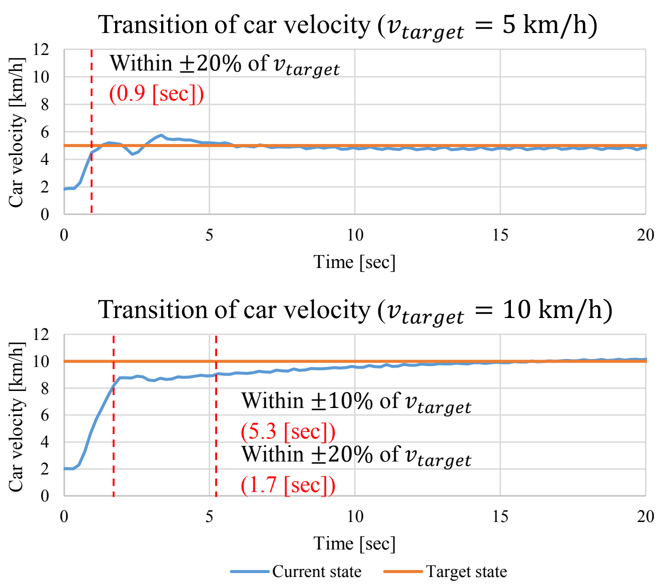

Fig. 10: The result of the task-specific self-body controller acquisition method (**Proposed**).

We show the result of **Proposed** in Fig.10. $T^{conv}_{20\%} = 0.9$ [sec] when $v_{target} = 5$ [km/h], and $T^{conv}_{10\%} = 5.3$ [sec] when $v_{target} = 10$ [km/h]. Also, $T^{conv}_{20\%} = 1.7$ [sec] when $v_{target} = 10$ [km/h]. So, while there are a little error, the car velocity converges to the target task state much better than **PID1** and **PID2**.

V. Discussion

First, we consider the experimental results of Section IV. From these experiments, we can see that the convergence to the target velocity is fast in **Proposed**, compared to in **PID1** and **PID2**. Although there remains a possibility of improving the convergence time of **PID1** and **PID2** by tuning gains, the car velocity vibrates largely when increasing the gain in our experiments. **Proposed** can be constructed only by conducting **Random** for 60 seconds. Although there are some parameters in **Proposed**, we only had to tune α after setting all parameters. When α, which is the parameter that limits the changes of control input, is small, the convergence time shortens and the vibration increases, and vice versa. Although we set the loss function as Eq. 2 in this study, there are many variations of its setting. For example, it is possible to limit joint angles within a certain range or inhibit the acceleration changes of the control input.

Second, we consider the applicable scope of this study. In this study, although we focused on the accelerator pedal control by the musculoskeletal humanoid, there are many other problems to which it can be applied. For example, in autonomous driving, this method can be applied to not only the accelerator pedal, but the brake pedal, handle operation, and compensation of body tilt during driving. Also, we can change the kind of control input. We can use not only the joint angle, but the joint velocity, joint torque, and muscle tension directly. Also, although we conducted experiments to track the constant car velocity, we can change the transition of the target car velocity $v_{target}(t)$. For example, in the case of applying a brake to smoothly stop at the intended place, we can decide the target transition of the car velocity by minimizing jerk, and conduct a control that tracks it.

Finally, we consider further developments of **Proposed**. In this study, in accordance with the increase in the dimension of control input, the control input space that needs to be learned increases exponentially, and to handle multi-DOFs is difficult. So, we must reduce the control input space by using concepts of muscle synergy [20], AutoEncoder [21], etc. Also, although we constructed DDC-Net by a simple structure from the viewpoint of stability and cost of learning, we can express this network by recurrent networks such as LSTM [22]. Although we start to develop this method in the context of the accelerator pedal control by the musculoskeletal humanoid, there are many other appropriate behaviors and robots for it to be applied.

VI. CONCLUSION

In this study, we proposed the task-specific self-body controller acquisition method, and verified the effectiveness through the accelerator pedal experiments by the musculoskeletal humanoid Musashi. It is difficult for the error between the actual robot and its self-body image to completely vanish by merely learning the relationship between joints and muscles, and the task realization is even more difficult. Thus, we construct the novel network (DDC-Net) whose input is the current task state and time-series target control input, and whose output is the time-series predicted task state. By learning the DDC-Net using the random motion data and backpropagating the error between the network output and intended task state to the target control input, the optimal control input is acquired. Using this method, we do not need to tune the PID gain regarding the pedal control, and can realize the intended car velocity stably and dynamically. In future works, we would like to analyze the stability of this neural network based controller for the actual autonomous driving. Also, we would like to realize long series consecutive tasks by the musculoskeletal humanoid.

REFERENCES

[1] S. Wittmeier, C. Alessandro, N. Bascarevic, K. Dalamagkidis, D. Devereux, A. Diamond, M. Jäntsch, K. Jovanovic, R. Knight, H. G. Marques, P. Milosavljevic, B. Mitra, B. Svetozarevic, V. Potkonjak, R. Pfeifer, A. Knoll, and O. Holland, "Toward Anthropomimetic Robotics: Development, Simulation, and Control of a Musculoskeletal Torso," *Artificial Life*, vol. 19, no. 1, pp. 171–193, 2013.

[2] Y. Nakanishi, S. Ohta, T. Shirai, Y. Asano, T. Kozuki, Y. Kakehashi, H. Mizoguchi, T. Kurotobi, Y. Motegi, K. Sasabuchi, J. Urata, K. Okada, I. Mizuuchi, and M. Inaba, "Design Approach of Biologically-Inspired Musculoskeletal Humanoids," *International Journal of Advanced Robotic Systems*, vol. 10, no. 4, pp. 216–228, 2013.

[3] Y. Asano, T. Kozuki, S. Ookubo, M. Kawamura, S. Nakashima, T. Katayama, Y. Iori, H. Toshinori, K. Kawaharazuka, S. Makino, Y. Kakiuchi, K. Okada, and M. Inaba, "Human Mimetic Musculoskeletal Humanoid Kengoro toward Real World Physically Interactive Actions," in *Proceedings of the 2016 IEEE-RAS International Conference on Humanoid Robots*, 2016, pp. 876–883.

[4] M. Jäntsch, S. Wittmeier, K. Dalamagkidis, A. Panos, F. Volkart, and A. Knoll, "Anthrob - A Printed Anthropomimetic Robot," in *Proceedings of the 2013 IEEE-RAS International Conference on Humanoid Robots*, 2013, pp. 342–347.

[5] Y. Asano, T. Shirai, T. Kozuki, Y. Motegi, Y. Nakanishi, K. Okada, and M. Inaba, "Motion Generation of Redundant Musculoskeletal Humanoid Based on Robot-Model Error Compensation by Muscle Load Sharing and Interactive Control Device," in *Proceedings of the 2013 IEEE-RAS International Conference on Humanoid Robots*, 2013, pp. 336–341.

[6] K. Kawaharazuka, M. Kawamura, S. Makino, Y. Asano, K. Okada, and M. Inaba, "Antagonist Inhibition Control in Redundant Tendon-driven Structures Based on Human Reciprocal Innervation for Wide Range Limb Motion of Musculoskeletal Humanoids," *IEEE Robotics and Automation Letters*, vol. 2, no. 4, pp. 2119–2126, 2017.

[7] K. Kawaharazuka, S. Makino, M. Kawamura, Y. Asano, K. Okada, and M. Inaba, "Online Learning of Joint-Muscle Mapping using Vision in Tendon-driven Musculoskeletal Humanoids," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 772–779, 2018.

[8] K. Kawaharazuka, S. Makino, M. Kawamura, A. Fujii, Y. Asano, K. Okada, and M. Inaba, "Online Self-body Image Acquisition Considering Changes in Muscle Routes Caused by Softness of Body Tissue for Tendon-driven Musculoskeletal Humanoids," in *Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2018, pp. 1711–1717.

[9] K. Kawaharazuka, S. Makino, K. Tsuzuki, M. Onitsuka, Y. Nagamatsu, K. Shinjo, T. Makabe, Y. Asano, K. Okada, K. Kawasaki, and M. Inaba, "Component Modularized Design of Musculoskeletal Humanoid Platform Musashi to Investigate Learning Control Systems (in press)," in *Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2019.

[10] K. Hirai, M. Hirose, Y. Haikawa, and T. Takenaka, "The Development of Honda Humanoid Robot," in *Proceedings of the 1998 IEEE International Conference on Robotics and Automation*, 1998, pp. 1321–1326.

[11] "DARPA Robotics Challenge," http://archive.darpa.mil/roboticschallenge/.

[12] S. Ookubo, Y. Asano, T. Kozuki, T. Shirai, K. Okada, and M. Inaba, "Learning Nonlinear Muscle-Joint State Mapping Toward Geometric Model-Free Tendon Driven Musculoskeletal Robots," in *Proceedings of the 2015 IEEE-RAS International Conference on Humanoid Robots*, 2015, pp. 765–770.

[13] D. Tanaka, S. Arnold, and K. Yamazaki, "EMD Net: An Encode-Manipulate-Decode Network for Cloth Manipulation," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1771–1778, 2018.

[14] A. Byravan, F. Leeb, F. Meier, and D. Fox, "SE3-Pose-Nets: Structured Deep Dynamics Models for Visuomotor Planning and Control," in *Proceedings of the 2018 IEEE International Conference on Robotics and Automation*, 2018, pp. 3339–3346.

[15] K. Kawaharazuka, T. Ogawa, J. Tamura, and C. Nabeshima, "Dynamic Manipulation of Flexible Objects with Torque Sequence Using a Deep Neural Network," in *Proceedings of the 2019 IEEE International Conference on Robotics and Automation*, 2019, pp. 2139–2145.

[16] K. Kojima, T. Karasawa, T. Kozuki, E. Kuroiwa, S. Yukizaki, S. Iwaishi, T. Ishikawa, R. Koyama, S. Noda, F. Sugai, S. Nozawa, Y. Kakiuchi, K. Okada, and M. Inaba, "Development of life-sized high-power humanoid robot JAXON for real-world use," in *Proceedings of the 2015 IEEE-RAS International Conference on Humanoid Robots*, 2015, pp. 838–843.

[17] T. van der Sande, I. Besselink, and H. Nijmeijer, "Steer-by-wire: a study into the bandwidth and force requirements," in *Proceedings of the 11th International Symposium on Advanced Vechicle Control*, 2012, pp. 1–6.

[18] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," in *Proceedings of the 32nd International Conference on Machine Learning*, 2015, pp. 448–456.

[19] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *Proceedings of the 3rd International Conference on Learning Representations*, 2015.

[20] C. Alessandro, I. Delis, F. Nori, S. Panzeri, and B. Berret, "Muscle synergies in neuroscience and robotics: from input-space to task-space perspectives," *Frontiers in Computational Neuroscience*, vol. 7, no. 43, pp. 1–16, 2013.

[21] G. E. Hinton and R. R. Salakhutdinov, "Reducing the Dimensionality of Data with Neural Networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[22] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.