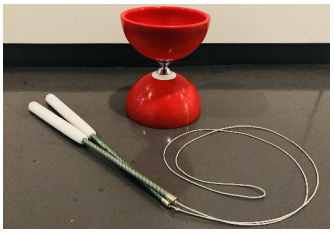
2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

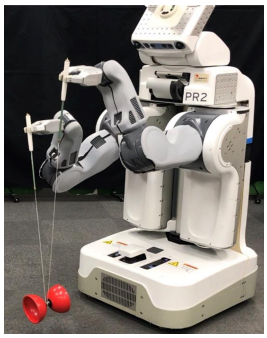
Diabolo Orientation Stabilization by Learning Predictive Model for Unstable Unknown-Dynamics Juggling Manipulation

Takayuki Murooka, Kei Okada and Masayuki Inaba

***Abstract*— Juggling manipulation is one of difficult manipulation to acquire since some of such manipulation is unstable and also its physical model is unknown due to the complex nonprehensile manipulation. To acquire these unstable unknown-dynamics juggling manipulation, we propose a method for designing the predictive model of manipulation with a deep neural network, and a real-time optimal control law with some robustness and adaptability using backpropagation of the network. In this study, we apply this method to diabolo orientation stabilization, which is one of unstable unknown-dynamics juggling manipulation. We verify the effectiveness of the proposed method by comparing with basic controllers such as P Controller or PID Controller, and also check the adaptability of the proposed controller by some experiments with a real life-sized humanoid robot.**



(a) Diabolo tools. A red diabolo top and two white sticks connected with a rope.



(b) Diabolo orientation stabilization by PR2.

Fig. 1. Diabolo orientation stabilization.

I. INTRODUCTION

Robots are expected to behave and manipulate like human, and in order to realize that, researchers have been tackling a wide range of motion planning or control with various robots. However there are a lot of manipulation left which robots haven't acquired yet because of its difficulty. In this study, we focus on juggling manipulation as one of such manipulation. Juggling manipulation by robots is considered to be important from the point of view of both its difficulty and its application. In regards to its difficulty, some of juggling manipulation are dynamic, and the dynamics is usually unknown because of the complex physical model of nonprehensile manipulation. Especially it is much more difficult to acquire unstable juggling manipulation. We refer to such juggling manipulation as unstable unknown-dynamics juggling manipulation. There are less studies about this manipulation, and in this study we aim to acquire diabolo orientation stabilization shown in Fig. 1 (b), which is one of unstable unknown-dynamics juggling manipulation.

In regards to its application of juggling manipulation, it can become one of entertainment for human. Not only working robots as an alternative for human but also performing robots as entertainment for human have been widely developed to enrich human life, such as dog-shaped robots or dancing humanoid robots. Also in the field of social psychology or physiology, it was reported that entertainment robots are useful as therapy for elderly people or children with autism.

T. Murooka, K. Okada and M. Inaba are with JSK Laboratory, Graduate School of Information Science and Technology, The University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656, Japan
t-murooka@jsk.imi.i.u-tokyo.ac.jp

A. Related Works

1) Robot as Entertainment: Entertainment robots have been widely researched and developed to enrich human life [1], [2], [3], and also entertainment robots have significant effects as therapy for elderly people or children with autism [4], [5], [6]. From the point of view of the research, dancing humanoid robots [7], [8], puppet humanoid robots [9] and juggling robots [10], [11], [12], [13] were developed. In [14], not only the detailed robot but also the platform to create robot motion for entertainment was developed. Involving some audience, there were also some discussions and researches about the robot theater [15], [16], [17].

From the point of view of the commercial business, the dog-shaped robot AIBO [18] and the seal-shaped robot PARO [19] were developed, and nowadays sold for entertainment. Recently a lot of other dog-shaped quadruped robots which can do not only just walking or running but also dynamic motion like backflip or jump have been developed [20], [21], [22]. It is expected that robots for entertainment will be developed more widely as technology advances, and this study becomes one of those applications.

2) Juggling Manipulation: Influenced by various researches about dexterous and dynamic manipulation, juggling manipulation have been researched steadily. In [10], the robotic hand realized feedback stabilization control of a rolling manipulation system called the disk-on-disk. In [11], the humanoid robot realized catching and throwing the ball, and finally realized ball juggling with people. In [12], the robotic arms realized devil-sticking manipulation with mechanical aid. In [13], the robotic hand realized a control of periodic up-and-down yo-yo motion.

In most of these studies, they used specialized robots or robotic hands for specific juggling manipulation, but it

978-1-7281-6211-9/20/$31.00 ©2020 IEEE

is better to use not the specialized robots but the general purpose robots which can do various tasks like human. Also these manipulation were almost realized by modelling its dynamics physically, which means these studies don't support unknown-dynamics juggling manipulation. Few studies, which tackled unknown-dynamics juggling manipulation, assumed linear approximation of unknown dynamics and lacked a wide range of expression of dynamics. Nowadays thanks to the development of deep learning, various learning-based methods and applications with robotic manipulation were come up all the way from reinforcement learning to supervised learning or unsupervised learning [23], [24], [25], [26]. In the filed of juggling manipulation, there are some studies using reinforcement learning [27]. However most of these studies using reinforcement learning, even out of juggling manipulation, were conducted in simulation since reinforcement learning takes much time to learn the control policy and that disadvantage doesn't match to the experiment of robots in the real environment. In this study, based on the supervised learning method [25], [28], we propose the method to realize unstable unknown-dynamics juggling manipulation by designing a network for such manipulation and a control law using the network. Also not only realization of such manipulation but also verification of the applicability of the method to unstable manipulation and the adaptability of the method to the unknown environments, which are missing in the previous studies, are conducted and discussed in this study.

B. Contributions of This Study

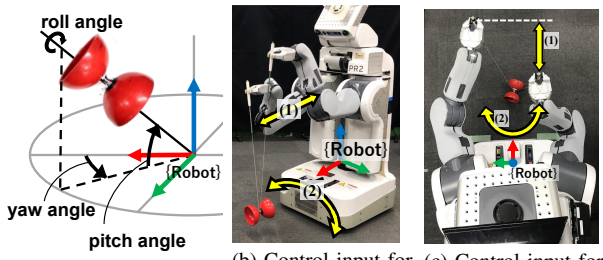
The main contributions of this study are shown as follows.

- Network structure of Diabolo-Manipulation-Net which represents the predictive model of diabolo manipulation using a deep neural network
- Calculation process of optimal control using backpropagation of Diabolo-Manipulation-Net
- Estimation of diabolo orientation and the experiments for evaluation and comparison of the proposed Diabolo-Manipulation-Net Controller to other controllers from the point of view of the convergence and adaptability

In the following, Section II will explain diabolo manipulation including how we can acquire diabolo manipulation and how the robot can estimate the diabolo orientation. Section III will propose a Diabolo-Manipulation-Net Controller based on the representation of the predictive model of diabolo manipulation. Section IV will explain the results of experiments showing the effectiveness of proposed Diabolo-Manipulation-Net Controller compared to other controllers and also adaptability. Finally the discussion and conclusion will be explained in Section V and Section VI.

II. DIABOLO MANIPULATION

First of all, we explain diabolo manipulation. Diabolo is one of juggling manipulation, and the most basic trick is stabilizing the diabolo orientation. This diabolo manipulation is also difficult for us human since we have to learn how we can realize this manipulation, and also the dynamics of this



(a) Diabolo orientation in the coordinate system {Robot}.

(b) Control input for diabolo manipulation.

(c) Control input for diabolo manipulation seen from above.

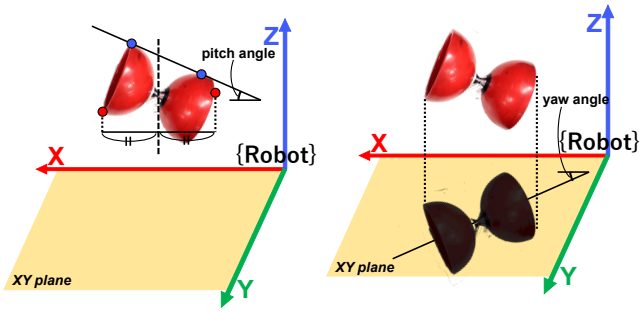
Fig. 2. Diabolo state and control input for diabolo orientation stabilization.

manipulation is unstable, which means that without proper control, the diabolo orientation tilts as time goes by and finally the rope will get entangled.

A. How To Manipulate Diabolo

For diabolo manipulation, we have to accelerate the diabolo rotation and stabilize the diabolo orientation. To accelerate the diabolo rotation, we move right and left sticks up and down alternately again and again. To stailize the diabolo orientation, we do some feedback manipulation. Generally the orientation is defined by roll, pitch and yaw angles. We define these three angles of the diabolo in the coordinate system {Robot} fixed to the robot base shown in Fig. 2 (a). The diabolo roll angle affects not the diabolo orientation but the diabolo rotation, so we consider only the diabolo pitch and yaw angles as the diabolo state to stabilize the diabolo orientation.

The relationship between the control input and the diabolo state is so complex about diabolo manipulation, but we can realize diabolo orientation stabilization with some approximation of the relationship. The diabolo pitch angle is strongly affected by the difference of longitudinal position of two sticks, and the diabolo yaw angle is strongly affected by the standing direction against the diabolo. Based on this simplified relationship, we can realize diabolo orientation stabilization as follows. When the diabolo pitch angle tilts forward, we move our right hand forward and our left hand backward, and when the diabolo pitch angle tilts backward,



(a) Estimation of the diabolo pitch angle.

(b) Estimation of the diabolo yaw angle.

Fig. 3. Visualization of estimation of the diabolo pitch and yaw angles in the coordinate system {Robot}.

we move our right hand backward and our left hand forward. At the same time we adjust the standing direction to stand just behind the diabolo according to the current diabolo yaw angle. As the control input of diabolo manipulation, we define the difference of longitudinal position of two sticks (Fig. 2(1)) and the angular velocity of the robot base (Fig. 2(2)) to change the standing direction. The difference of longitudinal position of two sticks is calculated by subtracting the position of the right stick on the X axis from the position of the left stick on the X axis in the coordinate system {Robot}.

However we cannot realize stabilization quite precisely in this way since complex dynamics of diabolo manipulation is not fully considered, which means the difference of longitudinal position of two sticks also affects the diabolo yaw angle and the standing direction also affects the diabolo pitch angle. We aim to consider these complex relationship and calculate optimal control of diabolo manipulation for more precise diabolo orientation stabilization in this study.

B. Estimation of Diabolo State

We estimate the diabolo pitch and yaw angles using a depth camera. At first, we get a point cloud from the depth camera, and we apply HSI color filter to that point cloud whose parameters are tuned to filter out all except the diabolo. Now we can get the point cloud of the diabolo. In the following paragraphs, we will explain the calculation process of estimating the diabolo pitch and yaw angles respectively in the coordinate system {Robot} shown in Fig. 3.

For estimating the diabolo pitch angle, first we calculate two points: one is the point whose value of the coordinate on the X axis is maximum among the point cloud, and the other is the point whose value of the coordinate on the X axis is minimum. These two points are drawn as red points in Fig. 3 (a). Second we define the virtual plane which is parallel to the YZ plane and passes through the point whose value of the coordinate on the X axis is the intermediate value between those two red points. By dividing the point cloud of the diabolo with this plane, we can get two point clouds. Third we calculate the point whose value of the coordinate of the Z axis is maximum about each two divided point cloud, which are drawn as blue points in Fig. 3 (a). Finally we connect those two blue points with one line and the angle formed by that line and the XY plane is estimated as the diabolo pitch angle.

For estimating the diabolo yaw angle, first we project the point cloud of the diabolo to the XY plane, which are drawn as a black region like a shadow in Fig. 3 (b). Second we apply linear regression to the projected points. The angle formed by the line of regression and the XZ plane is estimated as the diabolo yaw angle.

The diabolo pitch and yaw angles estimated with upper algorithms are so noisy because of observation variance of the depth camera. So we apply LPF (Low-Pass Filter) to the estimated pitch and yaw angles as follows.

$$\boldsymbol{x}_t^{lpf} = \frac{1}{N_{lpf}} \sum_{i=0}^{N_{lpf}-1} \boldsymbol{x}_{t-i} \tag{1}$$

$\boldsymbol{x}_t$ is the diabolo state (the diabolo pitch and yaw angles) at the timestep t, and $\boldsymbol{x}^{lpf}$ is the diabolo state applied LPF to. In this study we set $N_{lpf} = 5$.

III. DIABOLO-MANIPULATION-NET CONTROLLER

First we define the problem in this study, and then explain the calculation process of train, control and adaptation with Diabolo-Manipulation-Net shown in Fig. 4.

A. Problem Definition

We formulate the problem in this study. We represent the diabolo state with $\boldsymbol{x}$ whose dimension is $N_{\boldsymbol{x}}$ and the control input for diabolo manipulation with $\boldsymbol{u}$ whose dimension is $N_{\boldsymbol{u}}$. The predictive model of diabolo manipulation is formulated as follows, where $\boldsymbol{f}$ is the predictive model.

$$\boldsymbol{x}_{[t-T+1,t]} = (\boldsymbol{x}_{t-T+1}^T \ \boldsymbol{x}_{t-T+2}^T \ ... \ \boldsymbol{x}_t^T)^T \tag{2a}$$

$$\boldsymbol{x}_{t+1} = \boldsymbol{f}(\boldsymbol{x}_{[t-T+1,t]}, \boldsymbol{u}_t) \tag{2b}$$

t is the current timestep, and T is how long previous diabolo state to consider when predicting the diabolo state. For the dynamic system, the information about acceleration of the diabolo state is needed, so we set $T = 2$. In this diabolo manipulation experiments, we set $N_{\boldsymbol{x}} = 2$ (the diabolo pitch and yaw angles), $N_{\boldsymbol{u}} = 2$ (the difference of longitudinal position of two sticks and the angular velocity of the robot base).

We represent the predictive model with a neural network and that network is trained with real data of the diabolo state $\boldsymbol{x}$ and the control input $\boldsymbol{u}$. With the trained predictive model, we calculate the optimal control input as follows.

$$\boldsymbol{u}_t^{opt} = \underset{\boldsymbol{u}_t}{\operatorname{argmin}} \ J(\boldsymbol{x}_{t+1}^{ref}, \boldsymbol{x}_{t+1}^{pred}) \tag{3a}$$

$$s.t. \ \boldsymbol{u}_{min} \leq \boldsymbol{u}_t \leq \boldsymbol{u}_{max} \tag{3b}$$

$$\boldsymbol{x}_{t+1}^{pred} = \boldsymbol{f}(\boldsymbol{x}_{[t-T+1,t]}, \boldsymbol{u}_t) \tag{3c}$$

$\boldsymbol{u}_{max}, \boldsymbol{u}_{min}$ is the maximum and minimum value of $\boldsymbol{u}$. J is a cost function for optimizing the control input. $\boldsymbol{x}^{ref}, \boldsymbol{x}^{pred}$ is the reference and the prediction of $\boldsymbol{x}$. J must be calculated to make $\boldsymbol{x}_{t+1}^{pred}$ close to $\boldsymbol{x}_{t+1}^{ref}$, which means the diabolo state of the next step will get close to the reference diabolo state. Eq. 3 is calculated every cycle, and the optimized control input $\boldsymbol{u}_t^{opt}$ is commanded to the robot. We aim to acquire $\boldsymbol{u}^{opt}$ with some robustness and adaptability.

B. Network Structure of Predictive Model

The predictive model can be represented with any types of networks. For example, we can use sequence-based networks such as RNN or LSTM [29] like $\boldsymbol{x}_{t+1} = \boldsymbol{f}_{sequence}(\boldsymbol{x}_t, \boldsymbol{u}_t)$. With that sequence-based network, we can reduce the network size thanks to the sequence-based model. However the calculation time of optimizing the control input every cycle

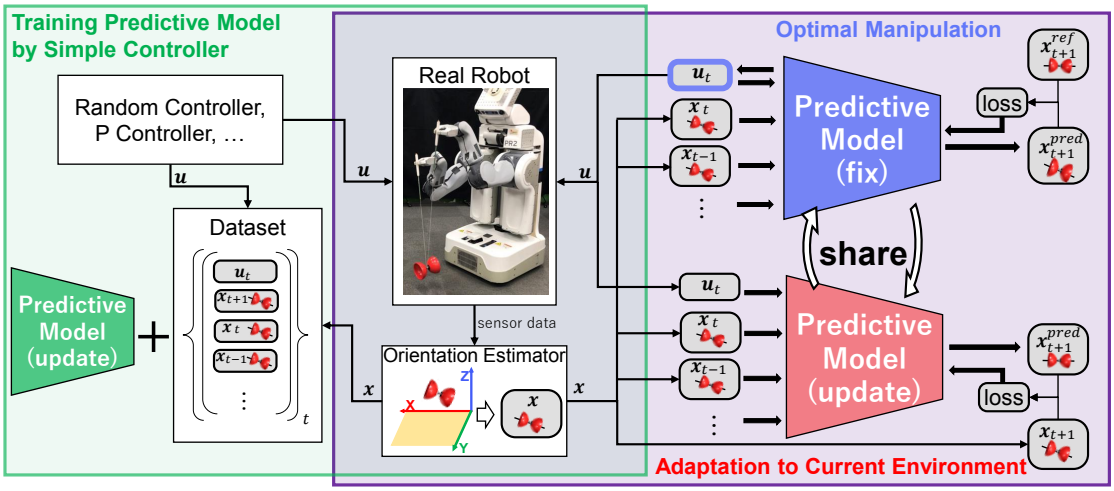


Fig. 4. The Overview of proposed system including Diabolo-Manipulation-Net. Fist we collect the dataset of diabolo manipulation with a simple controller like Random Controller or P Controller. In the train phase, we train the predictive model shown in the green trapezoid with the collected dataset. In the control phase, we calculate the optimal control input using backpropagation shown in the blue trapezoid and at the same time adaptation to the current environment shown in the red trapezoid. The predictive model is shared between optimal control and adaptation.

described in Eq. 3 would increase, which is fatal for real-time control of the dynamic system. So in this study, we use a five-fully-connected-layers network to represent the predictive model. This simple structure has an advantage to the computational cost compared to sequence-based networks. The dimension of the input layer is $N_{\boldsymbol{x}} \times T + N_{\boldsymbol{u}}$, and the dimension of the output layer is $N_{\boldsymbol{x}}$. The number of hidden layers are $\{50, 50, 50\}$, and we use Sigmoid for an activation function.

C. Training Predictive Model

In the train phase, in order to calculate the predictive model, we collect the dataset of $\boldsymbol{x}$ and $\boldsymbol{u}$ whose form of each data is $\{\boldsymbol{x}_{[t-T+1,t]}, \boldsymbol{u}_t, \boldsymbol{x}_{t+1}\}$. In this study, for collecting the dataset, we constructed two types of control laws: Random Controller and P Controller. We formulate Random Controller as follows.

$$\boldsymbol{u}_t = random(-\boldsymbol{u}_{min}, \boldsymbol{u}_{max}) \tag{4}$$

$random(\boldsymbol{a}, \boldsymbol{b})$ is a function which returns a value randomly from $\boldsymbol{a}$ to $\boldsymbol{b}$. Also we formulate P Controller as follows according to the simplified relationship between $\boldsymbol{x}$ and $\boldsymbol{u}$ discussed in Subsec. II-A.

$$\boldsymbol{u}_t = \boldsymbol{K}_p \boldsymbol{e}_t \tag{5a}$$
$$\boldsymbol{u}_t = \max(\boldsymbol{u}_{min}, \min(\boldsymbol{u}_{max}, \boldsymbol{u}_t)) \tag{5b}$$
$$\boldsymbol{K}_p = diag(k_{p1}, k_{p2}) \tag{5c}$$

$\boldsymbol{K}_p$ is a parameter matrix of the controller and k_{p*} is detailed parameters. $\boldsymbol{e}_t$ is $\boldsymbol{x}_t^{ref} - \boldsymbol{x}_t$. The control frequency of these Random Controller and P Controller is 30Hz which is the same as the frequency of depth image of Kinect. With these controllers, We collect the data for 5 minutes. With those dataset, we train the predictive model by setting the batch size as C_{batch}^{train} and the number of epochs as C_{epoch}^{train} When training, we use Mean Squared Error for the loss function, and Adam [30] for the optimization algorithm. In this study, we set $C_{batch}^{train} = 10, C_{epoch}^{train} = 100$

D. Real-time Optimal Control

We will explain how to calculate the optimal control input by solving Eq. 3. The optimization problem in Eq. 3 is nonlinear optimization, and we cannot solve it analytically. By calculating not $\boldsymbol{u}_t$ but $\Delta\boldsymbol{u}_t$ and update $\boldsymbol{u}_t$ again and again, we can optimize $\boldsymbol{u}_t$ recursively as follows.

$$\Delta\boldsymbol{u}_t = \underset{\Delta\boldsymbol{u}_t \in \boldsymbol{U}}{\operatorname{argmin}} \; J(\boldsymbol{x}_{t+1}^{ref}, \boldsymbol{x}_{t+1}^{pred}) \tag{6a}$$
$$s.t.\ \boldsymbol{u}_{min} \leq \boldsymbol{u}_t + \Delta\boldsymbol{u}_t \leq \boldsymbol{u}_{max} \tag{6b}$$
$$\boldsymbol{x}_{t+1}^{pred} = \boldsymbol{f}(\boldsymbol{x}_{[t-T+1,t]}, \boldsymbol{u}_t + \Delta\boldsymbol{u}_t) \tag{6c}$$
$$\boldsymbol{u}_t \leftarrow \boldsymbol{u}_t + \Delta\boldsymbol{u}_t \tag{6d}$$

We conduct this calculation of the optimal control input using the trained predictive model. Eq. 6a means that minimizing the cost function J by searching $\Delta\boldsymbol{u}_t$ within some constraint $\boldsymbol{U}$. We describe the detailed process of Eq. 6.

1) Set initial $\boldsymbol{u}_t$ of optimization as $\boldsymbol{u}_{t-1}$
2) Predict the state of next step $\boldsymbol{x}_{t+1}^{pred}$ by forwardpropagation
3) Calculate the cost function J
4) Optimize only $\boldsymbol{u}_t$ using backpropagation
5) Repeat 2)-4), and command $\boldsymbol{u}_t$ to the real robot.

In 1), we expect $\boldsymbol{u}_t^{opt}$ won't be too far from $\boldsymbol{u}_{t-1}$ since the robot cannot realize any $\boldsymbol{u}$ because of the mechanical time constant. So 1) initialization of optimization is important. After that initialization, the process of 2)-4) is calculated repeatedly. In 3), a cost function should be calculated as follows.

$$J(\boldsymbol{x}_{t+1}^{ref}, \boldsymbol{x}_{t+1}^{pred}) = \frac{1}{2}\|\boldsymbol{x}_{t+1}^{ref} - \boldsymbol{x}_{t+1}^{pred}\|^2 \tag{7}$$

In this study, we aim to stabilize the diabolo orientation, which means the diabolo pitch and yaw angles should be

zero degree, so $\boldsymbol{x}^{ref} = (0\ 0)^T$ In addition, in order to transit the control input from $\boldsymbol{u}_{t-1}$ to $\boldsymbol{u}_t$ smoothly, we expand the cost function as follows.

$$J(\boldsymbol{x}_{t+1}^{ref}, \boldsymbol{x}_{t+1}^{pred}, \boldsymbol{u}_t, \boldsymbol{u}_{t-1}) = \frac{1}{2}\|\boldsymbol{x}_{t+1}^{ref} - \boldsymbol{x}_{t+1}^{pred}\|^2 + \frac{1}{2}W_{smooth}\|\boldsymbol{u}_t - \boldsymbol{u}_{t-1}\|^2 \quad (8)$$

In 4), we acquire $\Delta\boldsymbol{u}_t$ approximately using backpropagation. We calculate backpropagation of the predictive model with the loss as a cost function J. Usually we calculate the derivative of weights $\boldsymbol{W}$ of the network as $\frac{\partial J}{\partial \boldsymbol{W}}$, but in this phase we calculate the derivative of $\boldsymbol{u}_t$ as $\frac{\partial J}{\partial \boldsymbol{u}_t}$ and then calculate $\Delta\boldsymbol{u}_t$ as follows.

$$\boldsymbol{g} = \frac{\partial J}{\partial \boldsymbol{u}_t} \quad (9a)$$

$$\Delta\boldsymbol{u}_t = -\epsilon\frac{\boldsymbol{g}}{\|\boldsymbol{g}\|} \quad (9b)$$

$\boldsymbol{g}$ is the gradient of J for $\boldsymbol{u}_t$, and ϵ is the constant learning rate for updating $\boldsymbol{u}_t$. By repeating the process of 2)-4) C_{epoch}^{opt} times with C_{batch}^{opt} batch size, finally we can acquire the optimal control input $\boldsymbol{u}_t^{opt}$. We conduct this whole calculation process of the optimal control input by 30Hz which is the same as the frequency of depth image of Kinect. In this study, we set $C_{batch}^{opt} = 10$, $C_{epoch}^{opt} = 10$, $W_{smooth} = 0.05$.

E. Adaptation to New Environments

In the control phase, at the same time as the optimal control calculation, we also conduct adaptation of the predictive model to the current environment every cycle. This calculation are executed in a different thread from the optimal control calculation thread. In order to update the predictive model in real time, we collect the dataset of $\boldsymbol{x}$ and $\boldsymbol{u}$ whose form of each data is $\{\boldsymbol{x}_{[t-T+1,t]}, \boldsymbol{u}_t, \boldsymbol{x}_{t+1}\}$. We hold a memory buffer whose size is C^{memory} to put some time-series data in it. When training the predictive model online for adaptation, we use dataset whose batch size is C_{batch}^{adapt} selected randomly from a memory buffer and train C_{epoch}^{adapt} times. In this study, we set $C^{memory} = 500$, $C_{batch}^{adapt} = 10$, $C_{epoch}^{adapt} = 10$.

IV. DIABOLO MANIPULATION EXPERIMENTS

First, we will explain the experimental setting. Next we will formulate P Controller and conduct an experiment of diabolo orientation stabilization. Then we will deal with PID Controller in the same way. After that we will conduct an experiment of diabolo orientation stabilization with our proposed Diabolo-Manipulation-Net Controller, and verify its effectiveness including adaptability.

A. Experimental Setting

In this study, a life-sized humanoid robot PR2 will conduct the experiments. We use a diabolo top and sticks which are connected with a rope shown in Fig. 1 (a). The weight of the diabolo is 287 [g]. The length of the rope is 1270 [mm]. These tools for performing a diabolo are designed for human, and in this study there is no mechanical modification of the diabolo tools for robots. It is a great advantage for humanoid robots to use tools designed for human.

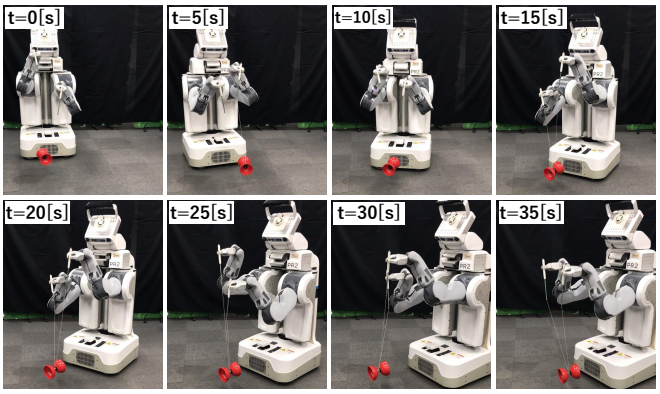


Fig. 5. Snapshots of PR2 conducting diabolo orientation stabilization with Diabolo-Manipulation-Net Controller.

In the following, we will calculcate control indices to measure the convergence of each controller, MSE^{pitch} and MSE^{yaw}. MSE is a mean square error through time between the diabolo state and the reference diabolo state, and we calculate this error about the diabolo pitch and yaw angles respectively. We evaluate each controller in each environment five times, and summarize the average of these control indices in Table I.

B. P Controller

First, we show the diabolo orientation transition in Fig. 6 with P Controller. The detailed formulation of P Controller is shown in Eq. 5. As for P Controller, MSE is so large and especially there is stationary deviation about the yaw angle since P Controller is a very simple controller. So we will design PID Controller in the following.

C. PID Controller

We formulate PID Controller as follows.

$$\boldsymbol{u}_t = \boldsymbol{K}_p\boldsymbol{e}_t + \boldsymbol{K}_i\int \boldsymbol{e}_t dt + \boldsymbol{K}_d\dot{\boldsymbol{e}}_t \quad (10a)$$

$$\boldsymbol{K}_p = diag(k_{p1}, k_{p2}) \quad (10b)$$

$$\boldsymbol{K}_i = diag(k_{i1}, k_{i2}) \quad (10c)$$

$$\boldsymbol{K}_d = diag(k_{d1}, k_{d2}) \quad (10d)$$

$\boldsymbol{e}_t$ is $\boldsymbol{x}_t^{ref} - \boldsymbol{x}_t$. $\boldsymbol{K}_p$, $\boldsymbol{K}_i$ and $\boldsymbol{K}_d$ are parameter matrices of the controller, and k_{p*}, k_{i*} and k_{d*} are detailed parameters. We show the diabolo orientation transition with PID Controller in Fig. 7. We can see that the convergence with PID Controller is better than that with P Controller, and there is less stationary deviation about the yaw angle compared to P Controller. According to Table I, any control indices were improved compared to P Controller.

D. Diabolo-Manipulation-Net Controller

We implemented Diabolo-Manipulation-Net Controller and its calculation process of train, control and adaptation with Chainer [31]. We trained the predictive model from

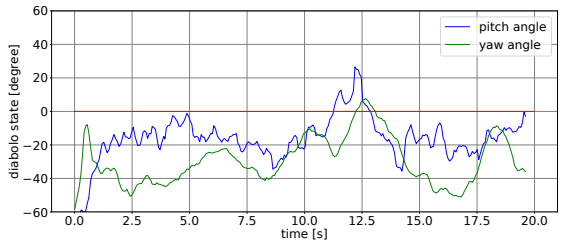


Fig. 6. The diabolo orientation transition with **P Controller**. The red line is the reference state.

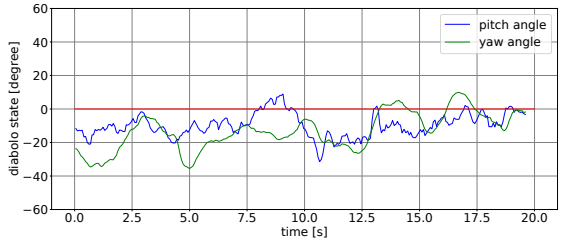


Fig. 7. The diabolo orientation transition with **PID Controller**. The red line is the reference state.

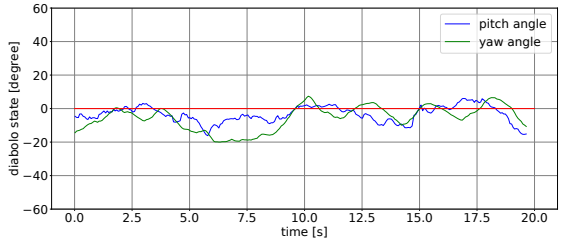


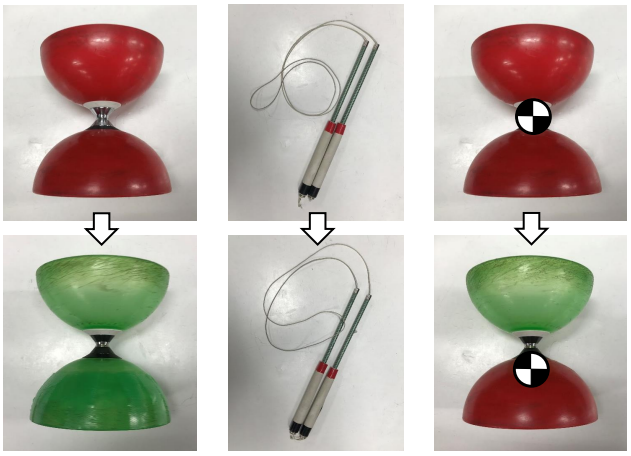
Fig. 8. The diabolo orientation transition with **Diabolo-Manipulation-Net Controller**. The red line is the reference state.

Random Controller and P Controller respectively shown in Subsec. III-C. The number of trials to collect enough dataset for training the predictive model to construct Diabolo-Manipulation-Net Controller was ten for Random Controller and one for P Controller. We can see that P Controller is a very simple controller but very useful for training. We refer to Diabolo-Manipulation-Net Controller trained by the dataset collected with Random Controller as DMN1 Controller, Diabolo-Manipulation-Net Controller trained by the dataset collected with P Controller as DMN2 Controller in Table I. We show the diabolo orientation transition of DMN2 Controller in Fig. 8 and also snapshots of diabolo orientation stabilization in Fig. 5. According to Table I, any control indices with DMN1 Controller or DMN2 Controller were improved compared to those with P Controller and PID Controller, and there are almost no difference about control indices between DMN1 Controller and DMN2 Controller.

TABLE I

CONTROL INDICES OF DIABOLO ORIENTATION STABILIZATION.

	P	PID	DMN1	DMN2
$MSE^{pitch}[degree]$	24.29	13.87	**7.02**	9.11
$MSE^{yaw}[degree]$	33.19	17.24	9.41	**9.01**



(a) Case1. The weight of the diabolo decreases. (b) Case2. The length of the rope becomes shorter. (c) Case3. The CoG moves backward.

Fig. 9. Various experimental settings.

E. Adaptability of Controllers

We also check the adaptability of controllers by conducting some experiments in different environments (Case1 - Case3) by changing experimental settings from the original environment of Subsec. IV-D. For Case1 we change the weight of the diabolo from 287 [g] to 243 [g], by just changing a diabolo top shown in Fig. 9 (a). For Case2 we change the length of the rope from 1270 [mm] to 1000 [mm] shown in Fig. 9 (b). For Case3 we change CoG (center of gravity) 6.25 [mm] backward from original CoG, by changing a diabolo top at only oneside shown in Fig. 9 (c). We conduct experiments of upper three settings with PID Controller and Diabolo-Manipulation-Net Controller before adaptation, and Diabolo-Manipulation-Net Controller after adaptation, which is the controller which collected the dataset in the new environment and updated the predictive model online enough to realize diabolo orientation stabilization in the new environment. These two Diabolo-Manipulation-Net Controller are trained beforehand with the dataset collected in the original environment. We refer to each controller as PID Controller, DMN Controller and DMN* Controller, and summarize the control indices with each controller in Table II. T_{keep} means how long diabolo orientation stabilization was kept without failure up to one minute. It is judged as failure when the rope gets entangled and the diabolo orientation cannot be stabilized any more. - means no data because of failure of diabolo orientation stabilization.

According to Table II, PID Controller and DMN Controller can keep diabolo orientation stabilization only about Case1 and Case2. This is because the dynamics of Case1 and Case2 is not so different from the the dynamics of the original

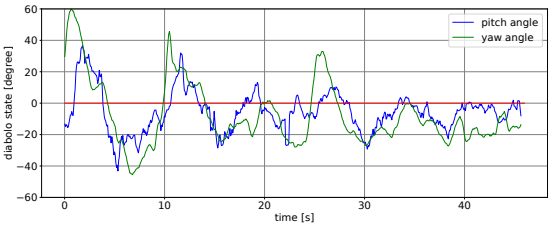


Fig. 10. The diabolo orientation transition with Diabolo-Manipulation-Net Controller in the sixth trial of adaptation to Case3. The red line is the reference state.

environment, and robustness of each controller absorbed the dynamics error, so the robot could keep diabolo orientation stabilization in the new environment without failure. As for adaptation of DMN* Controller, Case1 and Case2 need only one trial to adapt to the new environment. Adapting to the new environment means that updating the predictive model online again and again until realization of diabolo orientation stabilization. However Case3 needs six trials to adapt to the new environment. This is because of a large difference between the original environment and Case3. At first, DMN* Controller couldn't keep diabolo orientation stabilization, but the predictive model adapted to the new environment through time by updating weights of the network. We show the diabolo orientation transition with DMN* Controller in the sixth trials of adaptation to Case3, which is the final trials, in Fig. 10. We can see that the error between the diabolo state and the reference diabolo state gets less through time.

TABLE II

CONTROL INDICES OF DIABOLO ORIENTATION STABILIZATION WITH VARIOUS EXPERIMENTAL SETTINGS.

		PID	DMN	DMN*
Case1	$T_{keep}[s]$	60	60	60
	$MSE^{pitch}[degree]$	15.2	**8.8**	11.2
	$MSE^{yaw}[degree]$	17.1	11.1	**9.7**
Case2	$T_{keep}[s]$	60	60	60
	$MSE^{pitch}[degree]$	21.3	14.8	**6.9**
	$MSE^{yaw}[degree]$	28.8	17.4	**13.5**
Case3	$T_{keep}[s]$	3.7	4.2	**60**
	$MSE^{pitch}[degree]$	-	-	**14.5**
	$MSE^{yaw}[degree]$	-	-	**17.0**

V. DISCUSSION

Comparing to basic controllers, Diabolo-Manipulation-Net Controller has the good convergence according to Table I. The error with Diabolo-Manipulation-Net is about half of that with PID Controller. This is because the formulation of Diabolo-Manipulation-Net Controller is not restricted, which means Diabolo-Manipulation-Net Controller can represent highly wide types of controllers, though the formulation of PID Controller is restricted like Eq. 10. This ability to represent highly wide types of controllers is one of advantages of Diabolo-Manipulation-Net Controller. Also it is easy for Diabolo-Manipulation-Net Controller to adapt to the new environment since all we have to do is collecting data in the new environment and train the predictive model online. It is also one of advantages of Diabolo-Manipulation-Net Controller not to construct a control law directly but to construct the predictive model.

In this study, we trained the predictive model from P Controller and Random Controller, and checked the convergence and adaptability. According to the result of Table I, there are almost no difference of the convergence and adaptability about which controller to use for training the predictive model. This is because the predictive model, which we want to approximate with a neural network, is determined uniquely based on the laws of physics, unlike a controller or a control policy which we can represent in various forms. As for the number of trials to train the predictive model, though it is imagined easily, Random Controller needs much more trials than P Controller. Diabolo manipulation is unstable, so it is very efficient to use some feedback controller even if it is very simple like P Controller.

Next, we discuss the difficulty of setting parameters of controllers. As for PID Controller, we have to tune parameters, and such manual tuning of PID Controler is hard because of unstable and unknown-dynamics juggling manipulation. As for Diabolo-Manipulation-Net Controller, the allowable range of parameters is not so severe as PID Controller, since the parameters we have to set such as the network structure are not related to the controller directly and the parameters related to the controller directly such as the weight of the network are tuned automatically and empirically. As for tuning parameters in different environments, Diabolo-Manipulation-Net Controller can adapt automatically and empirically by collecting data and updating the network online although P Controller and PID Controller has to be tuned manually again.

Finally, we discuss the applicability to other control tasks. The basic idea of the calculation process of the optimal control and the adaptation to other environments is a general method, so this calculation process can be applied to other control problems. In addition, by extending the network to predict the much later state, we can get the learning-based controller which is similar to the model predictive controller.

VI. CONCLUSIONS

In this study, we proposed Diabolo-Manipulation-Net Controller and the calculation process to train that network, acquire the optimal control input for such manipulation and adapt to the new environments. Diabolo-Manipulation-Net is the predictive model of that diabolo manipulation, and we can calculate the control input by backpropagating the difference between the predicted diabolo state and the reference diabolo state. With some experiments of diabolo orientation stabilization, we verified the effectiveness of Diabolo-Manipulation-Net Controller, which is good at both convergence and adaptability to the different environments without hard tuning compared to basic controllers like PID Controller. We also verified its robustness to a certain extent though there is no structure in the network for robustness, which is one of future works.

REFERENCES

[1] Henrik Hautop Lund. Adaptive robotics in the entertainment industry. In *Proceedings 2003 IEEE International Symposium on Computational Intelligence in Robotics and Automation. Computational Intelligence in Robotics and Automation for the New Millennium (Cat. No. 03EX694)*, Vol. 2, pp. 595–602. IEEE, 2003.

[2] Christoph Bartneck and Michio Okada. Robotic user interfaces. In *Proceedings of the Human and Computer Conference*, pp. 130–140. Citeseer, 2001.

[3] Masahiro Fujita, Hiroaki Kitano, and Toshi Doi. Robot entertainment. *Robots for kids: Exploring new technologies for learning*, pp. 37–72, 2000.

[4] Toshiyo Tamura, Satomi Yonemitsu, Akiko Itoh, Daisuke Oikawa, Akiko Kawakami, Yuji Higashi, Toshiro Fujimooto, and Kazuki Nakajima. Is an entertainment robot useful in the care of elderly people with severe dementia? *The Journals of Gerontology Series A: Biological Sciences and Medical Sciences*, Vol. 59, No. 1, pp. M83–M85, 2004.

[5] Kazuyoshi Wada and Takanori Shibata. Robot therapy in a care house-its sociopsychological and physiological effects on the residents. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pp. 3966–3971. IEEE, 2006.

[6] Lykke Brogaard Bertel and Dorte Malig Rasmussen. Peers at play: A case study on persuasive educational and entertainment robotics in autism education. In *Proceedings of the International Workshop on EuroPLOT Persuasive Technology for Learning, Education, and Teaching IWEPLET*, pp. 161–168, 2013.

[7] Kuniya Shinozaki, Yousuke Oda, Satoshi Tsuda, Ryohei Nakatsu, and Akitsugu Iwatani. Study of dance entertainment using robots. In *International Conference on Technologies for E-Learning and Digital Entertainment*, pp. 473–483. Springer, 2006.

[8] Kunio Kojima, Shunichi Nozawa, Kei Okada, and Masayuki Inaba. Dance-like humanoid motion generation through foot touch states classification. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1788–1793. IEEE, 2014.

[9] Elizabeth Jochum, Jarvis Schultz, Elliot Johnson, and Todd D Murphey. Robotic puppets and the engineering of autonomous theater. In *Controls and Art*, pp. 107–128. Springer, 2014.

[10] Ji-Chul Ryu, Fabio Ruggiero, and Kevin M Lynch. Control of nonprehensile rolling manipulation: Balancing a disk on a disk. *IEEE Transactions on Robotics*, Vol. 29, No. 5, pp. 1152–1161, 2013.

[11] Jens Kober, Matthew Glisson, and Michael Mistry. Playing catch and juggling with a humanoid robot. In *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012)*, pp. 875–881. IEEE, 2012.

[12] Stefan Schaal, Christopher G Atkeson, and Sethu Vijayakumar. Scalable techniques from nonparametric statistics for real time robot learning. *Applied Intelligence*, Vol. 17, No. 1, pp. 49–60, 2002.

[13] Takuma Nemoto, Yuta Noguchi, Hokuto Miyakawa, and Masami Iwase. Control of robotic yoyo with energy compensation based on an integrated model of a robot and a yoyo. *Journal of Advanced Simulation in Science and Engineering*, Vol. 2, No. 2, pp. 329–348, 2015.

[14] Shin'ichiro Nakaoka, Kanako Miura, Mitsuharu Morisawa, Fumio Kanehiro, Kenji Kaneko, shuuji Kajita, and Kazuhito Yokoi. Toward the use of humanoid robots as assemblies of content technologies -realization of a biped humanoid robot allowing content creators to produce various expressions-:-realization of a biped humanoid robot allowing content creators to produce various expressions- (in japanese). *Synthesiology (in Japanese)*, Vol. 4, No. 2, pp. 80–91, 2011.

[15] Hiroshi Ishiguro and Oriza Hirata. Robot theater. *Journal of the Robotics Society of Japan (in Japanese)*, Vol. 29, No. 1, pp. 35–38, 2011.

[16] Chyi-Yeu Lin, Chang-Kuo Tseng, Wei-Chung Teng, Wei-Chen Lee, Chung-Hsien Kuo, Hung-Yan Gu, Kuo-Liang Chung, and Chin-Shyurng Fahn. The realization of robot theater: Humanoid robots and theatric performance. In *2009 International Conference on Advanced Robotics*, pp. 1–6. IEEE, 2009.

[17] Garth Zeglin, Aaron Walsman, Laura Herlant, Zhaodong Zheng, Yuyang Guo, Michael C Koval, Kevin Lenzo, Hui Jun Tay, Prasanna Velagapudi, Katie Correll, et al. Herb's sure thing: A rapid drama system for rehearsing and performing live robot theater. In *2014 IEEE International Workshop on Advanced Robotics and its Social Impacts*, pp. 129–136. IEEE, 2014.

[18] Masahiro Fujita. Aibo: Toward the era of digital creatures. *The International Journal of Robotics Research*, Vol. 20, No. 10, pp. 781–794, 2001.

[19] Takanori Shibata. Ubiquitous surface tactile sensor. In *IEEE Conference on Robotics and Automation, 2004. TExCRA Technical Exhibition Based.*, pp. 5–6. IEEE, 2004.

[20] Simone C Niquille. Regarding the pain of spotmini: Or what a robot's struggle to learn reveals about the built environment. *Architectural Design*, Vol. 89, No. 1, pp. 84–91, 2019.

[21] Péter Fankhauser and Marco Hutter. Anymal: a unique quadruped robot conquering harsh environments. *Research Features*, No. 126, pp. 54–57, 2018.

[22] Benjamin Katz, Jared Di Carlo, and Sangbae Kim. Mini cheetah: A platform for pushing the limits of dynamic quadruped control. In *2019 International Conference on Robotics and Automation (ICRA)*, pp. 6295–6301. IEEE, 2019.

[23] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[24] Kentaro Wada, Shingo Kitagawa, Kei Okada, and Masayuki Inaba. Instance segmentation of visible and occluded regions for finding and picking target from a pile of objects. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2048–2055. IEEE, 2018.

[25] Kento Kawaharazuka, Toru Ogawa, Juntaro Tamura, and Cota Nabeshima. Dynamic manipulation of flexible objects with torque sequence using a deep neural network. In *2019 International Conference on Robotics and Automation (ICRA)*, pp. 2139–2145. IEEE, 2019.

[26] Frederik Ebert, Chelsea Finn, Sudeep Dasari, Annie Xie, Alex Lee, and Sergey Levine. Visual foresight: Model-based deep reinforcement learning for vision-based robotic control. *arXiv preprint arXiv:1812.00568*, 2018.

[27] Jason Chemin and Jehee Lee. A physics-based juggling simulation using reinforcement learning. In *Proceedings of the 11th Annual International Conference on Motion, Interaction, and Games*, pp. 1–7, 2018.

[28] Daisuke Tanaka, Solvi Arnold, and Kimitoshi Yamazaki. Emd net: An encode–manipulate–decode network for cloth manipulation. *IEEE Robotics and Automation Letters*, Vol. 3, No. 3, pp. 1771–1778, 2018.

[29] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, Vol. 9, No. 8, pp. 1735–1780, 1997.

[30] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[31] Seiya Tokui, Kenta Oono, Shohei Hido, and Justin Clayton. Chainer: a next-generation open source framework for deep learning. In *Proceedings of workshop on machine learning systems (LearningSys) in the twenty-ninth annual conference on neural information processing systems (NIPS)*, Vol. 5, pp. 1–6, 2015.