2009 IEEE International Conference on Robotics and Automation, Kobe, Japan, May, 2009.

Control for Throwing Manipulation by One Joint Robot

Hideyuki Miyashita, Tasuku Yamawaki and Masahito Yashima

***Abstract*— This paper proposes a throwing manipulation strategy for a robot with one revolute joint. The throwing manipulation enables the robot not only to manipulate the object to outside of the movable range of the robot, but also to control the position of the object arbitrarily in the vertical plane even though the robot has only one degree of freedom. In the throwing manipulation, the robot motion is dynamic and quick, and the contact state between the robot and the object changes. These make it difficult to obtain the exact model and solve its inverse problem. In addition, since the throwing manipulation requires more powerful actuators than the static manipulation, we should set the control input by taking consideration of the performance limits of the actuators. The present paper proposes the control strategy based on the iteration optimization learning to overcome the above problems and verifies its effectiveness experimentally.**

I. INTRODUCTION

In general, a robot manipulates an object with grasp. By grasping objects, the robot can gain high stability for manipulation. However, a grasp limits the flexibility of manipulation. For instance, the workspace of a grasp manipulation is limited by the movable range of a manipulator.

In this paper, we discuss the throwing manipulation by one joint robot which can control the object position to multiple goal positions as shown in Fig.1. If the robot can accomplish such throwing manipulation, the robot can not only manipulate the object to outside of the movable range of the robot, but also control the position of the object arbitrarily in the vertical plane, even though the robot has only one degree of freedom [1], [2].

Fig. 2 (a) shows one of the applications of the throwing manipulation, in which the one joint robot throws the various types of object carried by the belt-conveyer and stores them into each goal box container. As the box containers are placed at the apex point of the object's trajectory, the collision impact can be lessened. Fig. 2 (b) shows the juggling with the throwing manipulation. If the one joint robot can throw the object to different goal positions, the robot can perform a humanlike dexterous juggling.

In the throwing manipulation, the robot motion is dynamic and quick, and the contact state between the robot and the object changes [3]. These make it difficult to obtain the exact model and solve its inverse problem. In addition, since the throwing manipulation requires more powerful actuators than the static manipulation, we should set the control input by taking consideration of the performance limits of the actuators.

The authors are with Dept. of Mechanical Systems Engineering, National Defense Academy of Japan, 1-10-20, Hashirimizu, Yokosuka, Kanagawa, JAPAN {g47091, yamawaki, yashima}@nda.ac.jp

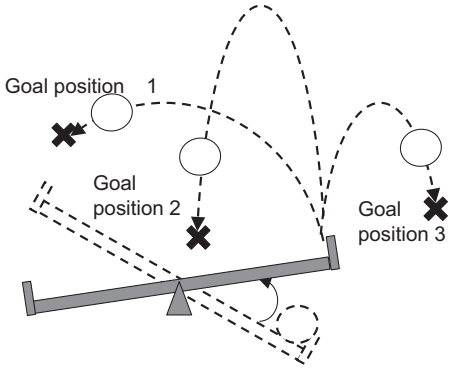

Fig. 1. Throwing manipulation by one joint robot

There are several researches discussing the manipulation with a throwing motion [1], [2], [4]-[9]. Lynch proposes the model-based controller which uses nonlinear optimization techniques [4]. The experimental results are less successful because of the modeling error. To overcome the modeling error problem, several researches apply the learning control to throwing manipulation [2], [5], [6]. However, these researches neglect the performance limits of the actuator and do not attempt the learning control for the multiple goals.

The present paper discusses the throwing manipulation method based on the iteration optimization learning, which can take into consideration the modeling error problem, the performance limits of the actuator and the stability of the learning method.

II. MODELING

In this section, we develop the throwing model, which maps the trajectory parameter that decides the arm trajectory to the apex point of the object where the object attains its highest elevation.

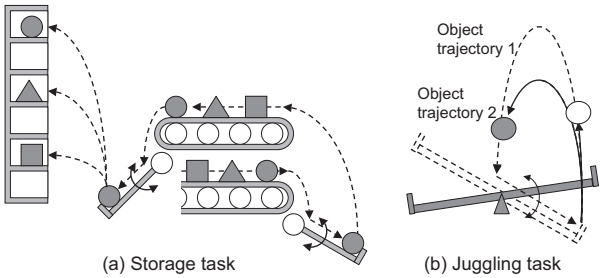

Fig. 2. Application of the throwing manipulation

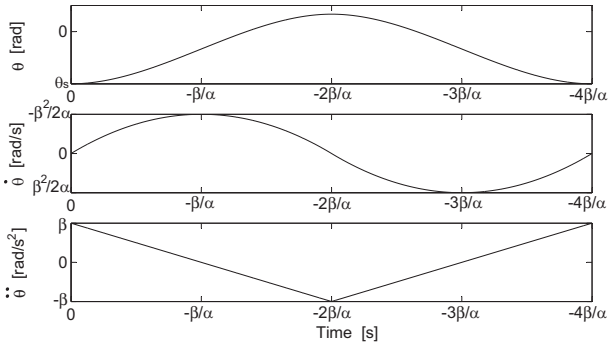

Fig. 3. Arm trajectory

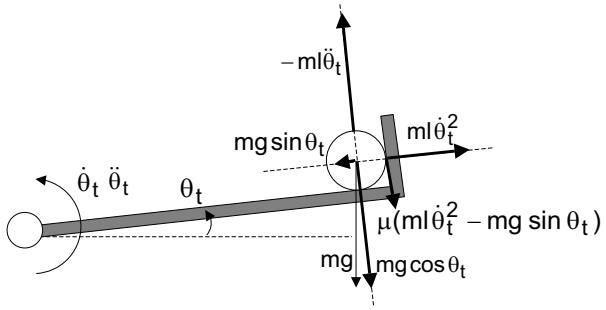

Fig. 4. Throwing condition

A. Arm trajectory

To simplify the modeling, the throwing motion remains in the vertical xy plane. We assume that the object is regarded as a point mass and air resistance can be neglected. The reference frame is located on the axis of the joint. When the arm is horizontal, the angle of the arm θ is defined as 0 deg. The counterclockwise rotation is set as positive.

It assumed that the arm trajectory is given by a third-order polynomial about time t.

$$\theta(t) = \frac{\alpha}{6}t^3 + \frac{\beta}{2}t^2 + \theta_s \tag{1}$$

$$\dot{\theta}(t) = \frac{\alpha}{2}t^2 + \beta t \tag{2}$$

$$\ddot{\theta}(t) = \alpha t + \beta \tag{3}$$

where θ_s is the initial angle, α and β are the jerk of the trajectory and the initial angular acceleration, respectively. From these equations, the arm trajectory can be given by three parameters, which are θ_s, α, and β. To enable the robot to throw the object in the counterclockwise direction, we define the range of α, β and θ_s as follows:

$$-\pi/2 < \theta_s < 0 \tag{4}$$

$$\alpha < 0 \tag{5}$$

$$\beta > 0 \tag{6}$$

The approaching arm trajectory to the maximum arm angle at the time $t = -2\beta/\alpha$ can be shown as Fig. 3. At the moment when the throwing condition, which is described in section II-D, is satisfied, the object is thrown in the air before $t = -2\beta/\alpha$. We set the return trajectory which is symmetrical to the approaching trajectory with respect to $t = -2\beta/\alpha$.

B. The apex point of the object trajectory

We formulate the motion of the thrown object. Assuming that the object is thrown with the angular velocity $\dot{\theta} = \dot{\theta}_t$ at the angle $\theta = \theta_t$ by the arm and that the motion of the object is given by the ballistic flight equation, the apex point of the free-flying trajectory where the object attains the highest point, (x_m, y_m), which is called the model's apex point, is written by

$$\begin{pmatrix} x_m \\ y_m \end{pmatrix} = \begin{pmatrix} l\cos\theta_t \\ l\sin\theta_t \end{pmatrix} + \begin{pmatrix} -l\dot{\theta}_t\sin\theta_t \\ l\dot{\theta}_t\cos\theta_t \end{pmatrix}\left(\frac{l\dot{\theta}_t\cos\theta_t}{g}\right) - \begin{pmatrix} 0 \\ \frac{g}{2} \end{pmatrix}\left(\frac{l\dot{\theta}_t\cos\theta_t}{g}\right)^2 \tag{7}$$

where l is the radius of the arm and g is the gravitational acceleration. As seen from (7), the position of the model's apex point (x_m, y_m) can be given by θ_t and $\dot{\theta}_t$.

C. Forward model

It is assumed that the object is thrown at time $t = t_t > 0$ (throwing time). From (1), (2) and (3), the arm's throwing angle θ_t, angular velocity $\dot{\theta}_t$ and angular acceleration $\ddot{\theta}_t$ can be described by four parameters, α, β, θ_s and t_t. These parameters are called a trajectory parameter $\boldsymbol{u}$, which is expressed by

$$\boldsymbol{u} = (\alpha,\ \beta,\ \theta_s,\ t_t)^T \tag{8}$$

As seen from (7), the position of the model's apex point (x_m, y_m) can be expressed by using θ_t and $\dot{\theta}_t$. By substituting (1) and (2) into (7), the model's apex point (x_m, y_m) is described by the trajectory parameter $\boldsymbol{u}$. The forward model of the throwing, which relates the model's apex point (x_m, y_m) with the trajectory parameter $\boldsymbol{u}$, can be expressed by using a nonlinear function $\boldsymbol{f}$, which is

$$(x_m,\ y_m)^T = \boldsymbol{f}(\boldsymbol{u}) \tag{9}$$

D. Throwing condition

We derive the throwing condition from equilibrium of forces at the moment when the object is thrown in the air, as shown in Fig. 4. If the acceleration in the vertical direction with respect to the arm's surface satisfies the following equation

$$h_T(\boldsymbol{u}) = ml\ddot{\theta}_t + mg\cos\theta_t + \mu(ml\dot{\theta}_t^2 - mg\sin\theta_t) = 0 \tag{10}$$

then the object is thrown. The object's mass is m and the frictional coefficient is μ. Since (10) is expressed by the θ_t, $\dot{\theta}_t$ and $\ddot{\theta}_t$, the throwing condition $h_T(\boldsymbol{u})$ can be also described by the trajectory parameter $\boldsymbol{u}$.

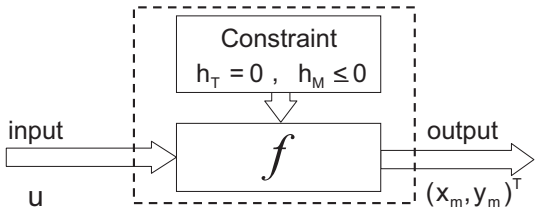

Fig. 5. Throwing model

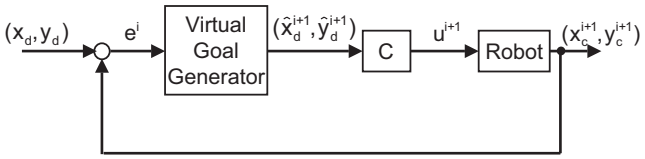

Fig. 6. Learning algorithm using virtual goal generator

E. Constraint on trajectory parameter

This section describes the constraint on the trajectory parameter. We should take into consideration the constraint on the actuator performance. As seen from Fig. 3, the maximum angular acceleration of the arm is β. This acceleration does not exceed the maximum torque $\tau_{\max}$ of the actuator, which is written by

$$I\beta \le \tau_{\max} \tag{11}$$

where I is moment of inertia of the arm and the rotor of the motor. From (6) and (11), we get the inequality constraint on the parameter β, which is

$$0 < \beta \le \tau_{\max}/I \tag{12}$$

Similarly, the maximum angular velocity of the arm is $-\beta^2/2\alpha$. This velocity does not exceed the maximum angular velocity $\dot{\theta}_{\max}$ of the actuator, which is written by

$$-\beta^2/2\alpha \le \dot{\theta}_{\max} \tag{13}$$

Linearizing (13) within the range of β shown in (12) yields

$$\alpha + \tau_{\max}\beta/2\dot{\theta}_{\max}I \le 0 \tag{14}$$

Therefore, the constraints on the actuator performance can be given by (12) and (14).

Finally, combining (4), (5), (12) and (14) yields the constraint on the trajectory parameter, which is written by

$$\boldsymbol{h}_M(\boldsymbol{u}) \le \boldsymbol{0} \tag{15}$$

F. Throwing model

Summing (9), (10) and (15) yields the throwing model (16), which relates the trajectory parameter $\boldsymbol{u}$ with the object's apex point $(x_m,\ y_m)$ under the constraint conditions, as shown in Fig. 5.

$$\begin{cases} (x_m,\ y_m)^T = \boldsymbol{f}(\boldsymbol{u}) \\ h_T(\boldsymbol{u}) = 0 \\ \boldsymbol{h}_M(\boldsymbol{u}) \le \boldsymbol{0} \end{cases} \tag{16}$$

III. ITERATIVE LEARNING CONTROL USING OPTIMIZATION

A. Learning control based on virtual goal apex point

If we can obtain the exact model of the robot, we can calculate the motion pattern for the arm to throw the object to the goal. To obtain the exact model, we need to consider kinematics and dynamics including the interaction of the robot and the object, and the correction of the vision sensor etc. In general, it is impossible to obtain such exact model. The throwing model (16) is obtained by simplifying the actual motion. Therefore, we cannot control the apex point of the object trajectory to the goal apex point (x_d, y_d) by applying the control input (trajectory parameter) $\boldsymbol{u}$ obtained from the above-mentioned throwing model to the robot. Instead of setting (x_d, y_d) as the goal apex point for the controller, we set a virtual goal apex point $(\widehat{x}_d, \widehat{y}_d)$ for the controller C as shown in Fig. 6, which is updated at each throwing trial in order to enable the robot to throw the object to the goal apex point (x_d, y_d). The virtual goal apex point $(\widehat{x}_d, \widehat{y}_d)$ is obtained by the task-level learning approach [6].

The difference between the goal apex point (x_d, y_d) and the measured apex point (x_c^i, y_c^i) with a camera can be written as

$$\boldsymbol{e}^i = (x_d,\ y_d)^T - (x_c^i,\ y_c^i)^T \tag{17}$$

where the subscript i indicates the ith throwing.

The virtual goal apex point $(\widehat{x}_d^{i+1},\ \widehat{y}_d^{i+1})$ of the i+1th throwing is updated by using the ith error $\boldsymbol{e}^i$ of (17) as

$$(\widehat{x}_d^{i+1},\ \widehat{y}_d^{i+1})^T = (\widehat{x}_d^{i},\ \widehat{y}_d^{i})^T + k\boldsymbol{e}^i \tag{18}$$

where $0 < k \le 1$ is a constant parameter which affects the convergence of the learning.

B. Optimization of trajectory parameter

The controller C in Fig. 6 finds the trajectory parameter $\boldsymbol{u}$ which achieves throwing to the virtual goal apex $(\widehat{x}_d, \widehat{y}_d)$ by using the throwing model (16), which is called the inverse problem of the throwing manipulation. It is necessary to consider the various control purposes as well as the throwing condition and the actuator's constraints to improve the performance of the throwing task. Therefore, in order to obtain the trajectory parameter $\boldsymbol{u}^{i+1}$ of the i+1th throwing, the inverse problem is solved by using the nonlinear programming problem described by the following equations.

$$\begin{aligned} \min : J =& w_x(\widehat{x}_d^{i+1} - x_m(\boldsymbol{u}^{i+1}))^2 + w_y(\widehat{y}_d^{i+1} - y_m(\boldsymbol{u}^{i+1}))^2 \\ &+ w_T(-\beta^{i+1}/\alpha^{i+1})^2 + \Delta\boldsymbol{u}^T\boldsymbol{W}_u\Delta\boldsymbol{u} \\ \text{subj. to:}\quad & \text{eq. (16)} \end{aligned} \tag{19}$$

where w_x, w_y, w_T are weights and $\boldsymbol{W}_u$ is a diagonal weighting matrix.

Here, (19) is the objective function. The first and second terms of the right-hand side, which indicate the difference between the virtual goal apex point $(\widehat{x}_d^{i+1}, \widehat{y}_d^{i+1})$ and the

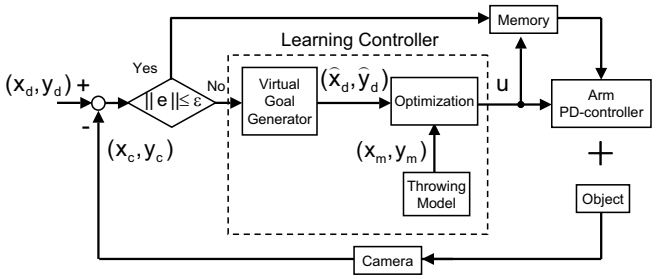

Fig. 7. Iterative learning control algorithm

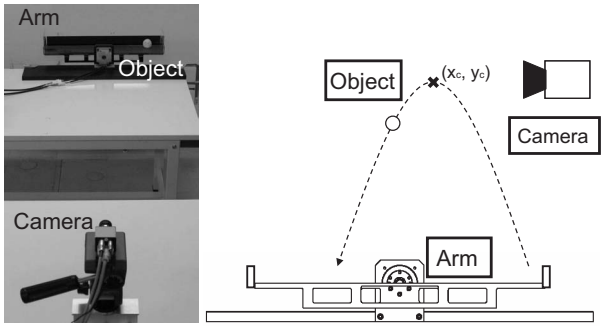

Fig. 8. Throwing robot system

model's apex point (x_m, y_m), contributes to the derivation of the trajectory parameter achieving the virtual goal apex point. The third term $(-\beta^{i+1}/\alpha^{i+1})$, which indicates the time when the arm reaches the maximum angular velocity, contributes to the improvement of the motion performance of the arm. The variable $\Delta \boldsymbol{u}$ in the fourth term indicates the change of the trajectory parameter, which is described as

$$\Delta \boldsymbol{u} = \boldsymbol{u}^{i+1} - \boldsymbol{u}^{i} \tag{20}$$

Therefore, the fourth term helps the trajectory parameter to avoid a drastic change and helps the stability of the control system to be improved.

We solve the nonlinear optimization problem expressed by (19) and (16) at each throwing, and update the i+1th trajectory parameter $\boldsymbol{u}^{i+1}$. We use the sequential quadratic programming (SQP) method of which the advantage is the high convergence [10].

C. Learning algorithm

Fig. 7 shows the flow of the iterative learning control algorithm, whose details of the procedure are shown below.

1) We obtain the trajectory parameter $\boldsymbol{u}^1$ for the first throwing. We set the goal apex point (x_d, y_d) as the first virtual goal apex point $(\widehat{x}_d^1, \widehat{y}_d^1)$, and obtain the trajectory parameter $\boldsymbol{u}^1$ achieving the first virtual goal apex point by solving (19) under (16). We input the first trajectory parameter $\boldsymbol{u}^1$ to the robot. The robot throws the object. We measure the apex point (x_c^1, y_c^1) by using the camera. In the SQP method, the solutions depend on the initial value. Thus we give a variety of the initial value and obtain all trajectory parameters given by them. We chose the trajectory parameter that has the minimum value of the objective function as the first trajectory parameter. In this procedure, we set the 4th term of the right-hand side of (19) as $\boldsymbol{W}_u = \boldsymbol{0}$.
2) We calculate the error $\boldsymbol{e}^i$ between the apex point (x_c^i, y_c^i) measured with camera and the goal apex point (x_d, y_d) by using (17). If the norm of the error is less than the value of the threshold ε, we assume that the robot accomplishes the desired throwing task and stop the learning, otherwise we progress to the step 3).
3) We derive the virtual goal apex point $(\widehat{x}_d^{i+1}, \widehat{y}_d^{i+1})$ of the i+1th trial from the ith measured apex point by using the virtual goal generator (18).
4) We solve the nonlinear optimization problem (19) by taking into account the throwing model (16) and obtain the trajectory parameter $\boldsymbol{u}^{i+1}$.
5) Substituting the trajectory parameter α, β and θ_s of $\boldsymbol{u}^{i+1}$ into (1) and (2) yields the desired arm trajectories, $\theta_d(t)$ and $\dot{\theta}_d(t)$. The robot is controlled with a PD-compensator along with the desired trajectories and the object is thrown.
6) We measure several positions of the flying object with the camera and estimate the object's ballistic trajectory through the least square approximations. Then we calculate the apex point of the object trajectory $(\widehat{x}_c^{i+1}, \widehat{y}_c^{i+1})$ from the estimated trajectory and return to the step 2).

IV. LEARNING CONTROL EXPERIMENT OF THROWING

A. Experimental condition

We verify that the proposed learning control is still effective even though the goal apex point is switched in the way of learning. Let the first goal apex point be $(x_{d1}, y_{d1}) = (0.1, 0.3)$ m, the second goal apex point be $(x_{d2}, y_{d2}) = (0.15, 0.35)$ m, the weighting factors be $w_x = w_y = 1$, $w_T = 0.01$, $\boldsymbol{W}_u = \mathrm{diag}(0.1, 0.1, 0.1, 0.1)$, the parameter of (18) be $k = 0.7$, and the threshold value of the error norm be $\varepsilon = 5 \times 10^{-3}$ m. To inspect the stability of the learning control after the error norm satisfies the termination condition $||\boldsymbol{e}|| \leq \varepsilon$, we continue the learning experiment 10 times additionally.

B. Throwing robot system

Fig. 8 shows the throwing robot systems applied to the experiment. The arm is driven by the AC motor containing a harmonic drive gear. The angle of the arm is measured by the encoder. The arm is performed with simple PD compensators. The motor has the maximum torque $\tau_{\max} = 18$ Nm, and the maximum rotational speed $\dot{\theta}_{\max} = 4\pi$ rad/s. The radius length of the arm is 0.3 m. To enable the robot to keep the object on the arm's edge, we install the arm wall of which the height is 0.03 m on the arm's edge. The total moment of inertia including that of the arm and that of the rotor of the actuator is $I = 0.074\ \mathrm{kgm}^2$. Several positions of the flying object in the vertical plane are measured with the

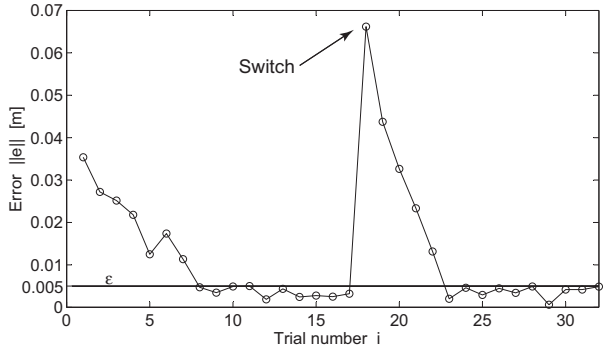

Fig. 9. Performance error at each trial

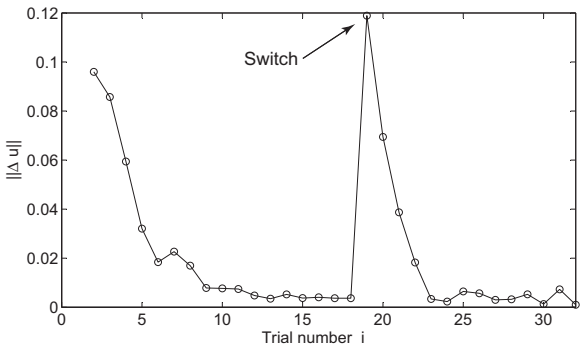

Fig. 11. Transition of $||\Delta \boldsymbol{u}||$ at each trial

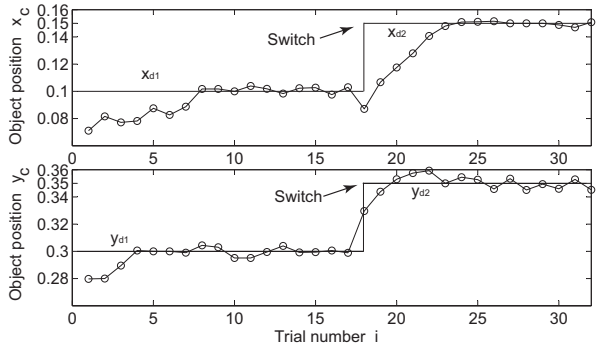

Fig. 10. Observed apex point of the object at each trial

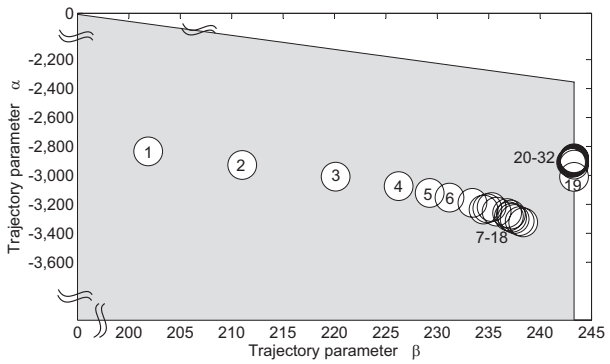

Fig. 12. Transition of value of α and β at each trial

high-speed camera at a rate of 1 kHz, which is developed by Hamamatsu Photonics K. K. These object's positional data are used to estimate the object's apex point (x_c, y_c). The sampling rate of the controller is 1 ms. We use a beanbag as the throwing object whose mass and diameter are 50 g and 0.05 m, respectively.

C. Experimental results and discussion

1) Control stability: Fig. 9 shows the transition of the error norm $||\boldsymbol{e}||$. The value of the error norm is reduced gradually by repeating the learning, and the error norm of the 8th throwing satisfies the termination condition, $||\boldsymbol{e}|| \leq \varepsilon$. After that, the error norms of the 9-17th throwings can keep the termination condition.

The goal apex point is switched at the 18th throwing. Due to the change of the goal, the error jumps. However, the error is attenuated gradually by repeating the learning, and the error norm of the 23rd throwing is less than ε and satisfies the termination condition. After that, the error norms of the 24-32nd can be kept less than ε.

Fig. 10 shows the apex point (x_c^i, y_c^i) measured with the high-speed camera. The measured apex point can get close to the goal apex point gradually by repeating the throwing, even though the goal apex point is switched at the 18th throwing.

Fig. 11 shows the norm of the change of the trajectory parameter $\| \Delta \boldsymbol{u} \|$, of which value decreases gradually by repeating the throwing. This result yields that the trajectory parameter can converge at the local optimal point despite the goal apex point is switched. The convergence of the trajectory parameter provides not only the convergence of the error norm $||\boldsymbol{e}^i||$ as shown in Fig. 9 but also the stability of the learning control.

2) Constraints on actuator's performance: Fig. 12 shows the transition of the trajectory parameters, α and β, which relate to the performance of the actuator as shown in II-E. The plotting numbers indicate the iteration number, and the allowable range of α and β, which is given by (12) and (14), is painted in gray color. This figure shows that α and β can keep staying within the allowable range.

At the 19th throwing after the goal apex is switched to be higher, the trajectory parameter significantly changes onto the borderline of the allowable range. This indicates that the robot learns the trajectory parameter making best use of the actuator's performance.

For comparison, Fig. 13 shows the transition of α and β for the learning control without consideration of the actuator's performance limits. Unlike with Fig. 12, the trajectory parameter exceeds the allowable range of the actuator's performance after the change of the goal apex. Therefore, the arm fails to track the desired trajectory corresponding to the trajectory parameter. Fig. 14 shows the transition of the error norm. The learning failed as seen from the results that the error norm oscillates drastically and is not reduced after the change of the goal apex.

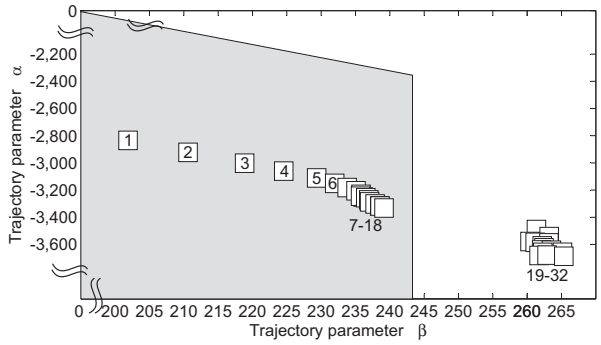

Fig. 13. Transition of value of α and β by iterative learning control without consideration of the actuator's performance limits

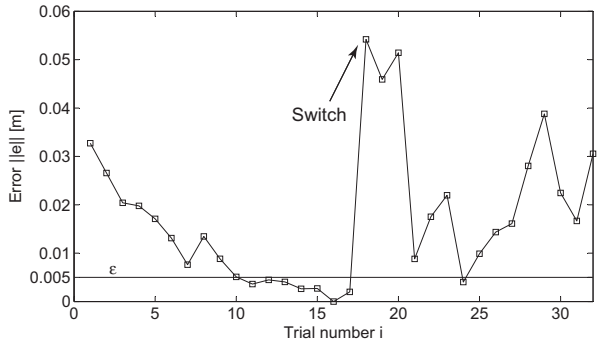

Fig. 14. Performance error by iterative learning control without consideration of the actuator's performance limits

D. Attached video

The attached video shows three types of the throwing manipulation utilizing the proposed learning control.

The first task is that the robot throws the object into the goal box container which is set at the goal apex point. We switch the position of the target box container in the way of learning. The robot succeeds in throwing the object into the goal box container by repeating the throwing, even though the position of the goal box container is switched. Fig. 15 shows snapshots of its successful throwing motion.

The second task is storing the object into three target box containers by the throwing manipulation as shown in Fig. 2 (a). The robot learns the trajectory parameters for each of three target box containers in advance. By applying the obtained three trajectory parameters to the robot, the robot can throw the object to each of the goal box containers.

The third task is one ball juggling, in which the robot throws the object to different apex points as shown in Fig. 2 (b). The robot learns the trajectory parameters for each of the two goal apex in advance. By applying alternately these two trajectory parameters to the robot, the robot achieves the one ball juggling.

V. CONCLUSIONS AND FUTURE WORKS

The present paper proposed a throwing manipulation method for a robot with one revolute joint. Task-level learning was used to learn the robot's control (the trajectory parameter) for throwing an object so that the object can reach a goal position. The learning method used a throwing model which relates a robot's control with the apex point of the object. The actuator's constraints are taken into consideration in the throwing model. The validity of the proposed method was verified experimentally.

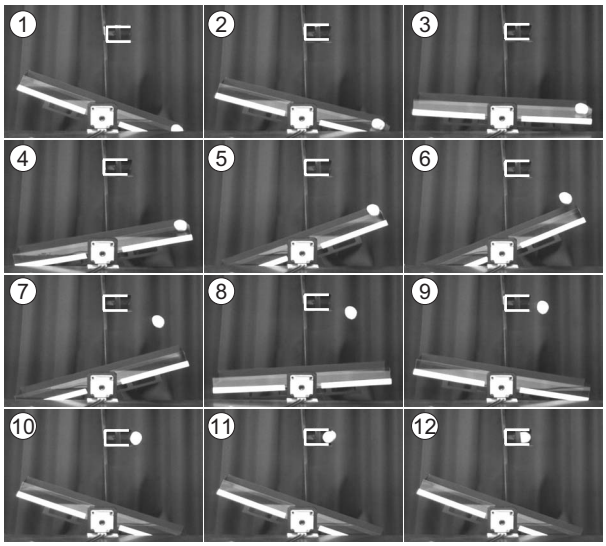

Fig. 15. Snapshots of the throwing of the object into the goal box container

In the present paper, we discussed the position control of the throwing object. However, for the storage task, the control of object's orientation is also important. Our future work is to consider the orientation control for a storage task.

REFERENCES

[1] K. M. Lynch and M. T. Mason, "Dynamic Nonprehensile Manipulation: Controllability, Planning, and Experiments, " Int. J. of Robotics Research, Vol.18, No.1, pp.64-92, January 1999.

[2] T. Tabata and Y. Aiyama, "Passing Manipulation by 1 Degree-of-Freedom Manipulator, " Proc. of IEEE Int. Conf. on Intelligent Robotics and Systems, Vol.3, pp.2920-2925, October 2003.

[3] M. T. Mason, "Mechanics of Robotic Manipulation, " The MIT Press 2001.

[4] K. M. Lynch and C. K. Black, "Control of Underactuated Manipulation by Real-Time Nonlinear Optimization, " Int. Symp. on Robotics Research, Snowbird UT, October 1999.

[5] T. Sakaguchi, M. Fujita, H. Watanabe and F. Miyazaki, "Motion planning and control for a robot performer, " Proc. of IEEE Int. Conf. on Robotics and Automation, Vol.3, pp.925-931, May 1993.

[6] E. W. Aboaf, "Task-Level Robot Learning, " Technical Report 1079, MIT Artificial Intelligence Laboratory, 1988.

[7] S. Schaal and C. G. Atkeson, "Open Loop Stable Control Strategies for Robot Juggling, " Proc. of IEEE Int. Conf. on Robotics and Automation, Vol.3, pp.913-918, May 1993.

[8] K. M. Lynch and C. K. Black, "Recurrence, Controllability, and Stabilization of Juggling, " IEEE Trans. on Robotics and Automation, Vol.17, No.2, pp.113-124, April 2001.

[9] M. Buehler, D. E. Koditschek and P. J. Kindlmann, "Planning and Control of Robotic Juggling and Catching Tasks, " Int. J. of Robotics Research, Vol.13, No.2, pp.101-118, April 1994.

[10] M. J. D. Powell, "The convergence of variable metric methods for nonlinearly constrained optimization calculations, " Nonlinear Programming 3, Academic Press, pp.27-63, 1978.