Learning Control Method for Throwing an Object More Accurately with One Degree of Freedom Robot

Hideyuki Miyashita, Tasuku Yamawaki and Masahito Yashima

***Abstract*— The present paper proposes the throwing manipulation method which cannot only control the position and orientation of the rigid object at the final goal, but also make the object track the desired trajectory which minimizes the impact at landing to avoid damages to the object as much as possible. Due to a model uncertainty, it is generally difficult to obtain the exact model and find appropriate control inputs to realize the throwing manipulation by experiments. Instead of raising the accuracy of the throwing model itself, we overcome the problems of the approximate model with errors by proposing the position-based iterative learning control method. We show experimentally that the proposed control method can gain the high positioning performance for throwing a rigid object with one degree of freedom robot.**

I. INTRODUCTION

In the present logistics systems or automated assembly lines, many conveyors and vehicles are used for transporting packages or parts, which make the speed of the physical distribution and the production flow be slow and the plant and equipment costs be high. Transportation by throwing objects can produce the following advantages: (1) mechanical equipments are simplified and less required; (2) flexibility of the systems is enhanced; and (3) the transportation processes are speeded up [2]. We consider the applications of the throwing manipulation to transporting, orienting and packing the various types of objects.

Fig. 1 shows one of the application examples, in which a one joint robotic arm throws objects with various orientation carried by a belt-conveyer and stores them with appropriate orientation in a storage pallet with an array of nests, each nest holding a single object. The objects in the storage pallet will be then stored in a warehouse or will be grasped by a robot gripper in assembly lines.

Instead of a manipulation of an object with grasp, the throwing manipulation has drown attention recently in the field of robotics [3], [6], [7], [9], [10]. This is because the throwing manipulation cannot only manipulate the object to outside of the movable range of the robot, but also manipulate the position and orientation of the rigid object arbitrarily by a robotic arm with fewer control inputs.

Lynch et al [4] propose the motion planning for the dynamic manipulation such as throwing a rigid object and show its validity with the open-loop controller. However, the positioning and trajectory tracking performance, which are important for the throwing manipulation, are not discussed.

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

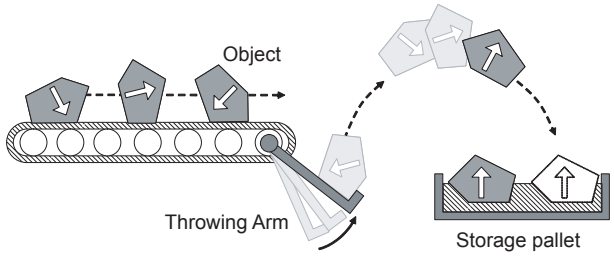

Fig. 1. Application of the throwing manipulation

They also discuss the controller for juggling by using nonlinear optimization techniques [5]. More precise system modeling is required to implement the controller. Therefore, the experimental results are less robust because of the modeling error and sensing error. Aboaf [1] applies a learning control method to the throwing of a point mass object to overcome these problems. However, constraint conditions such as actuator's performance limits are not considered to find the control inputs. The authors [6] propose the control method for throwing point mass object, which combines the learning control techniques and the nonlinear optimization techniques which take into account the actuator's performance limits. However, the control of object's orientation is not discussed though orienting objects is an important task as shown in Fig. 1. We also did not consider the collision impact at landing, which is crucial problem in the throwing manipulation.

The present paper proposes the iteration learning control method for the throwing manipulation which cannot only control the position and orientation of the rigid object at the final goal, but also make the object track the desired trajectory which minimizes the impact at landing to avoid damages to the objects as much as possible. We show experimentally that the proposed control method can gain the high positioning performance for throwing a rigid object with one degree of freedom robot.

II. THROWING MODEL

A. Definitions

We develop a model for throwing manipulation of a rigid object by the rotational one-degree-of-freedom robotic arm in the vertical plane, as shown in Fig. 2.

A reference frame x-y is fixed at the pivot point of the robotic arm. The gravitational acceleration in the x-y frame is $[0,\ -g]$ $(g > 0)$. The position and orientation of the object in the x-y frame is described as $\boldsymbol{z} = [x,\ y,\ \phi]^T \in \Re^3$. The angle of the arm is θ, which is set as 0 deg when the arm

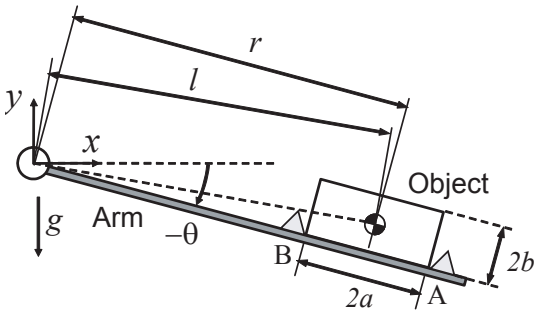

Fig. 2. Throwing model

is horizontal. The top surface of the arm intersects with the origin of the reference frame. The arm throws the object by counterclockwise rotation. We assume that the thrown object is captured by a storage pallet fixed at a final goal without slipping, rolling or rebounding at landing to simplify the analysis.

The object is assumed to be a rectangular and homogenous plate. The moment of inertia of the object and its mass are J and m and its width and height are $2a$ and $2b$, respectively. The distance between the center of mass of the object and the pivot point of the arm is l. When the arm is horizontal, the x location of the center of mass of the object is r.

The stoppers with a sharp edge are rigidly attached to the top surface of the arm to stop the object from sliding on the surface of the arm. We assume that there are no friction between the object and the stoppers and no collision with the stoppers when the object is released.

B. Constraints for Throwing Manipulation

The motion of the object relative to the arm is constrained in order not to slip, roll or break contacts before the object's release. The motion of the arm is also constrained by the joint actuator performance. This section shows the constraints of the object and the arm for the throwing manipulation.

1) Object Constraints: Since the object does not slide on the surface of the arm due to the stoppers, we consider only the vertical motion to the arm surface and the rolling motion about the vertex A and B.

The normal reaction force N acting on the object can be described as

$$N = mr\ddot{\theta} - md\dot{\theta}^2 + mgc\theta \tag{1}$$

where the upward direction is positive.

The reaction moment M_A and M_B from the arm about each vertex A and B can be described as, respectively,

$$\begin{aligned} M_A &= J\ddot{\theta} - mar\ddot{\theta} + mb^2\ddot{\theta} \\ &\quad + mab\dot{\theta}^2 + mbr\dot{\theta}^2 - magc\theta - mbgs\theta \end{aligned} \tag{2}$$

$$\begin{aligned} M_B &= J\ddot{\theta} + mar\ddot{\theta} + mb^2\ddot{\theta} \\ &\quad - mab\dot{\theta}^2 + mbr\dot{\theta}^2 + magc\theta - mbgs\theta \end{aligned} \tag{3}$$

where $s\theta = \sin\theta$ and $c\theta = \cos\theta$. The counterclockwise rotation about each vertex is positive.

To achieve the stable throwing, the condition for the dynamic grasp has to be satisfied from the start of the throwing motion until just before the object's release. Dynamic grasp occurs when a robotic arm accelerates such that there is no relative motion between the object and the arm [4]. In order to achieve the dynamic grasp, we should take into account the following constraints.

(i) Contact Constraints

$$N(t) > 0 \tag{4}$$

If the object satisfies this constraint, then the object can maintain the contact with the surface of the arm.

(ii) Un-rolling Constraints

$$M_A(t) < 0 \tag{5}$$

$$M_B(t) > 0 \tag{6}$$

If the object satisfies these constraints, then the object does not roll about the vertex A and B.

On the other hand, to release the object from the arm surface at time $t = t_t$, the following release constraint has to be satisfied.

(iii) Release Constraints

$$N(t_t) \leq 0 \tag{7}$$

If the arm decelerates so as to satisfy the constraint, then the object is released and starts the free flight.

2) Arm Constraints: The arm is subject to the constraints given by the actuator performance and the robot mechanism at all times during the throwing manipulation.

(i) Joint Angle Constraints

$$-\pi/2 < \theta(t) < 0 \tag{8}$$

The joint angle is subject to this constraint such that the object will be thrown to the right.

(ii) Joint Velocity Constraints

$$\left|\dot{\theta}(t)\right| < \dot{\theta}_{max} \tag{9}$$

where $\dot{\theta}_{max}$ is the maximum velocity of the actuator.

(iii) Joint Torque Constraints

$$|\tau(t)| < \tau_{max} \tag{10}$$

where τ_{max} is the maximum torque of the actuator.

C. Throwing Parameter

The free flight motion of the object thrown by the arm is given by a ballistic flight equation. If the object is released with the throwing angular velocity $\dot{\theta} = \dot{\theta}_t$ at the throwing angle $\theta = \theta_t$ and the throwing radius $l = l_t$, then the object's position and orientation $\boldsymbol{z}_f = [x_f,\ y_f,\ \phi_f]^T$ in the x-y frame after a lapse of flight time t_f can be written as

$$\boldsymbol{z}_f = \begin{bmatrix} l_t c\theta_{ta} - l_t\dot{\theta}_t s\theta_{ta} t_f \\ l_t s\theta_{ta} + l_t\dot{\theta}_t c\theta_{ta} t_f - gt_f^2/2 \\ \theta_t + \dot{\theta}_t t_f \end{bmatrix} \tag{11}$$

where $\theta_{ta} = \theta_t + \theta_a$ and $\theta_a = \cos^{-1}(r/l)$.

From (11), we can see that the object's position and orientation are expressed by using four variables

$$\boldsymbol{u} = [\ \theta_t,\ \dot{\theta}_t,\ l_t,\ t_f\ ]^T \in \Re^4 \tag{12}$$

where $\boldsymbol{u}$ is called the throwing parameter.

Eq. (11) can be rewritten as

$$\boldsymbol{z}_f = [x_f,\ \ y_f,\ \ \phi_f]^T = \boldsymbol{f}\,(\boldsymbol{u}) \tag{13}$$

The object's configuration $\boldsymbol{z}_f$ as well as the object's ballistic trajectory can be uniquely determined by specifying the throwing parameter $\boldsymbol{u}$ when the Jacobian $\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}$ is full rank. Since $\dim(\boldsymbol{u}) - \dim(\boldsymbol{z}_f) = 1$, there is one redundant control for the choice of trajectory reaching the specified final goal. For example, Fig.3 shows various object trajectories which reach the same final goal $\boldsymbol{z}_{fd} = [0.5\ \text{m},\ \ 0.2\ \text{m},\ \ 135\ \text{deg}]^T$ with different object's velocities. We can find an object's trajectory which satisfies specified throwing task requirements by using the redundancy.

III. Motion Planning

The motion planning of the throwing manipulation consists of the object trajectory planning and the arm trajectory planning. First, we find an optimal object trajectory to satisfy specified throwing requirements. Next, we find arm trajectories to realize the object motion under the constraints on the object and the arm. We show each motion planning algorithm below.

A. Object Trajectory Planning for Minimum Impact

An impact occurs when the thrown object is captured by a storage pallet fixed at a final goal. It is important to land the object with as less impact as possible in order to avoid damages. The magnitude of impact depends on the impact velocity at the instant of landing. Since there is the redundancy of the object's trajectory choice as shown in Fig.3, we can find the unique minimum impact trajectory and the corresponding trajectory parameter $\boldsymbol{u}$ by solving the following optimization problem.

$$\min: \quad \frac{1}{2}\dot{\boldsymbol{z}}_f(\boldsymbol{u})^T \boldsymbol{W} \dot{\boldsymbol{z}}_f(\boldsymbol{u}) \tag{14}$$

$$\text{subj. to:} \quad \boldsymbol{z}_f = \boldsymbol{f}\,(\boldsymbol{u}) \tag{15}$$

$$\boldsymbol{c} \le \boldsymbol{0} \tag{16}$$

$$\text{find:}\ \ \boldsymbol{u} = \left(\theta_t,\ \dot{\theta}_t,\ l_t,\ t_f\right)$$

where $\boldsymbol{W}$ is a weight matrix and $\dot{\boldsymbol{z}}_f$ is the impact velocity $(\dot{x}_f,\ \dot{y}_f,\ \dot{\phi}_f)$ at landing. The dynamic grasp constraints (4)∼(6) and the arm constraints (8)∼(10) at throwing are combined as shown in the inequality constraint (16).

When $\boldsymbol{W} = \text{diag}[\,m,\ m,\ J\,]$, the objective function is identical to the kinetic energy of the object at the impact. We use sequential quadratic programming (SQP) to solve the nonlinear programming. The trajectory of the white object in Fig.3 shows the minimum impact trajectory under the constraints on the object and the arm.

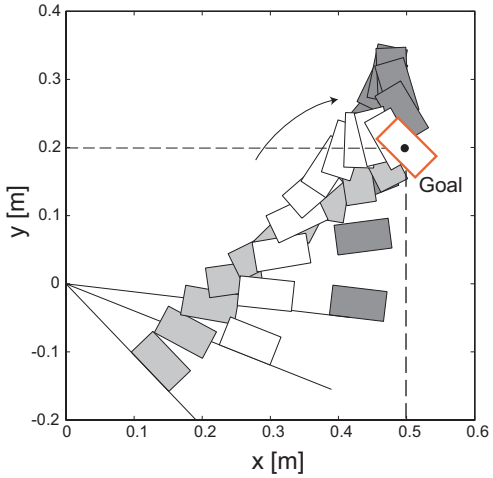

Fig. 3. Various object's trajectories which reach the same final goal. The trajectory of the white object shows the minimum impact trajectory.

B. Arm Trajectory Planning under Constraints

To throw the object at the throwing angle θ_t with velocity $\dot{\theta}_t$ obtained in Section III-A, the arm trajectory $\theta(t),\ t \in [0,\ t_t]$ has to satisfy the following boundary conditions.

$$\begin{aligned} \theta\,(t_t) &= \theta_t, \quad & \dot{\theta}\,(t_t) &= \dot{\theta}_t \\ \ddot{\theta}\,(t_t^-) &= \ddot{\theta}_t^-, \quad & \dot{\theta}(0) &= 0 \end{aligned} \tag{17}$$

where t_t is a release time which is a parameter to be chosen. $\ddot{\theta}_t^-$ is the joint acceleration at time $t = t_t^-$ immediately before object's release, which satisfies the dynamic grasp constraints (4)∼(6). The arm is rest at the initial time.

We apply a fifth-order polynomial about time to the arm trajectory to minimize the number of parameter.

$$\theta(t) = \sum_{i=0}^{5} a_i t^i, \quad t = [0,\ t_t] \tag{18}$$

where $a_0, \cdots, a_5 \in \Re$ are coefficients of the polynomial.

We find the arm trajectory which satisfies the boundary conditions and the constraints on the object and the arm by using the following optimization.

$$\begin{aligned} \max: \quad & w_1 L_{cnt} + w_2 L_{rollA} + w_3 L_{rollB} + w_4 L_{ang} && (19) \\ & + w_5 L_{vel} + w_6 L_{trq} + w_7 L_{acc} \\ \text{where:} \quad & L_{cnt} = \min N(t) \ge 0 && (20) \\ & L_{rollA} = \min\left(-M_A(t)\right) \ge 0 && (21) \\ & L_{rollB} = \min M_B(t) \ge 0 && (22) \\ & L_{ang} = \pi/2 - \max|\theta(t)| \ge 0 && (23) \\ & L_{vel} = \dot{\theta}_{max} - \max\left|\dot{\theta}(t)\right| \ge 0 && (24) \\ & L_{trq} = \tau_{max} - \max|\tau(t)| \ge 0 && (25) \\ & L_{acc} = -\left|\ddot{\theta}(0)\right| && (26) \end{aligned}$$

subj. to: eq. (17), (18)

where $w_1 \sim w_7$ are weight.

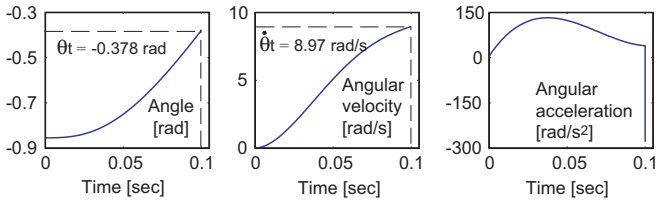

Fig. 4. Arm trajectories which satisfy the boundary conditions and the constraints on the object and the arm

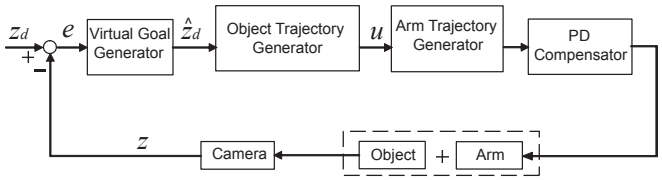

Fig. 5. Flowchart of iterative learning control

Eqs. (20)∼(22) contribute to expansion of minimal margins to satisfy the object constraints (4)∼(6). Eqs. (23)∼(25) contribute to expansion of minimal margins to satisfy the arm constraints (8)∼(10). Eq. (26) contributes to reduction in the magnitude of the initial acceleration $\ddot{\theta}(0)$.

A grid search method is used to solve the optimization problem. Since six coefficients $a_0 \sim a_5$ are available in (18), it is possible to impose three variables $\theta(0),\ \ddot{\theta}(0),\ t_t$ besides the four boundary conditions (17). We find the coefficients which maximize the objective function (19) by exploring numerous grid points specified in the $(\theta(0),\ \ddot{\theta}(0),\ t_t)$- search space.

To release the object from the arm surface at release time $t = t_t$, we make the arm decelerate instantaneously at $t = t_t$. The joint acceleration $\ddot{\theta}_t$ at $t = t_t$ is determined so as to satisfy the release constraint (7). After the time t_t, the arm is moved with the joint acceleration until the arm stops.

Fig. 4 shows the simulation results. The obtained arm trajectories satisfy the boundary conditions $\theta_t = -0.378$ rad and $\dot{\theta}_t = 8.97$ rad/s at $t_t = 0.1$ s, which is derived from (14) for the final goal $\boldsymbol{z}_{fd}$ used in Fig.3. The arm trajectories also satisfy the constraints on the object and the arm at all time during the manipulation. The acceleration $\ddot{\theta}_t$ changes instantaneously to release the object at $t_t = 0.1$ s.

IV. Iterative Learning Control

A. Control Problem

The purpose of the control for the throwing manipulation is not only to throw the object to a desired final goal, but also to reduce the impact at landing.

If we have an exact throwing model, then we can calculate the throwing parameter $\boldsymbol{u}$ of (12) which accomplishes the control purpose by using the method in Section III-A. To obtain the exact model, we need not only to implement parameter identification whenever the change of an object, but also to consider kinematics and dynamics including the interaction of the robot and the object, as well as the correction of the vision sensor etc [1]. In general, it is impossible to obtain such exact model.

Instead of raising the accuracy of the throwing model itself, we overcome the problems of the approximate model with errors by using the position-based iterative learning control method [6]. If the object can track a desired minimum impact trajectory, then the desired object's states at landing can be achieved. To move the object along the desired trajectory by using the position-based iterative learning control method [6], we assign a sequence of desired path points along the desired minimum impact trajectory.

B. Introduction of Virtual Goal

In place of setting a desired final goal $\boldsymbol{z}_{fd}$ and k desired path points $\boldsymbol{z}_{jd}\ (j = 1, \cdots, k)$ on the minimum impact trajectory to find the throwing parameter $\boldsymbol{u}$ of (12), we newly set virtual goals $\widehat{\boldsymbol{z}}_{fd}$ and $\widehat{\boldsymbol{z}}_{jd}\ (j = 1, \cdots, k)$, which correspond to $\boldsymbol{z}_{fd}$ and $\boldsymbol{z}_{jd}\ (j = 1, \cdots, k)$, respectively, for the object trajectory generator, as shown in Fig. 5.

The virtual goals are updated at each throwing trial in order to enable the robotic arm to throw the object so as to pass through the desired final goal and the desired path points as closely as possible. The basic idea of the virtual goal is shown in [1].

The virtual goals are derived as follows. The errors between the desired position and orientation $\boldsymbol{z}_{fd}$, $\boldsymbol{z}_{jd}\ (j = 1, \cdots, k)$ and the actual position and orientation $\boldsymbol{z}_f$, $\boldsymbol{z}_j\ (j = 1, \cdots, k)$ are respectively given by

$$\boldsymbol{e}_f^i = \boldsymbol{z}_{fd} - \boldsymbol{z}_f^i \tag{27}$$

$$\boldsymbol{e}_j^i = \boldsymbol{z}_{jd} - \boldsymbol{z}_j^i, \quad (j = 1, \cdots, k) \tag{28}$$

where i denotes the trial number.

The virtual goals $\widehat{\boldsymbol{z}}_{fd}^{i+1}$ and $\widehat{\boldsymbol{z}}_{jd}^{i+1}\ (j = 1, \cdots, k)$ of the $i+1$th throwing are updated by the following equations.

$$\widehat{\boldsymbol{z}}_{fd}^{i+1} = \widehat{\boldsymbol{z}}_{fd}^{i} + \boldsymbol{K}_f \boldsymbol{e}_f^i \tag{29}$$

$$\widehat{\boldsymbol{z}}_{jd}^{i+1} = \widehat{\boldsymbol{z}}_{jd}^{i} + \boldsymbol{K}_j \boldsymbol{e}_j^i \quad (j = 1, \cdots, k) \tag{30}$$

where the diagonal matrices $\boldsymbol{K}_f$ and $\boldsymbol{K}_j$ are the gains which contribute to the convergence of the learning.

C. Object Trajectory Generator for Learning Control

Unlike the case mentioned in Section III-A, the object trajectory generator in the proposed learning control is required to find the throwing parameter $\boldsymbol{u}$ of (12) which achieves throwing to the virtual goals $\widehat{\boldsymbol{z}}_{fd}$ and $\widehat{\boldsymbol{z}}_{jd}\ (j = 1, \cdots, k)$. Therefore, (14) and (15) are rewritten as (31) and (32), respectively.

$$\min: \ \frac{1}{2} \sum_{j=1}^{k} \left(\widehat{\boldsymbol{z}}_{jd}^{i+1} - \boldsymbol{z}_j(\boldsymbol{u}^{i+1}) \right)^T \boldsymbol{W}_j \left(\widehat{\boldsymbol{z}}_{jd}^{i+1} - \boldsymbol{z}_j(\boldsymbol{u}^{i+1}) \right) \tag{31}$$

$$\text{subj. to:} \quad \widehat{\boldsymbol{z}}_{fd}^{i+1} = \boldsymbol{f}(\boldsymbol{u}^{i+1}) \tag{32}$$

$$\boldsymbol{c} \leq \boldsymbol{0}$$

$$\text{find:} \quad \boldsymbol{u} = \left(\theta_t,\ \dot{\theta}_t,\ l_t,\ t_f \right)$$

where $\boldsymbol{W}_j$ is a weight matrix.

Since the arm has only one redundant control input as shown in Sec. II-C, it is difficult to find a trajectory parameter such that the object can pass through all desired path points. We try to find a trajectory parameter by using the objective function (31) such that the object will pass through the path points as closely as possible. In contrast, the object can be thrown to the final goal accurately since a trajectory parameter is found by using the equality constraint (32).

D. Control Algorithm

Fig. 5 shows the flowchart of the iterative learning control algorithm, whose details are shown below.

1) The robot performs the first throwing. The trajectory parameter $\boldsymbol{u}^1$ for the first throwing trial is derived from (14). The arm trajectory is obtained by solving (19) for the trajectory parameter $\boldsymbol{u}^1$. In the same manner described in the step 5 and 6, we make the robot throw the object, and measure the actual object's position and orientation $\boldsymbol{z}_f^1$ and $\boldsymbol{z}_j^1$ $(j = 1, \cdots, k)$ at the final goal and k path points, respectively, with a camera.
2) We calculate the error norm between the goals $\boldsymbol{z}_{fd}$, $\boldsymbol{z}_{jd}$ and the actual object's position and orientation $\boldsymbol{z}_f^i$, $\boldsymbol{z}_j^i$ of the ith throwing by using (27) and (28). If the error norm is greater than the value of the threshold, we progress to the step 3. Otherwise we assume that the robot accomplishes the desired throwing, and the learning control is terminated.
3) We derive the virtual goals $\hat{\boldsymbol{z}}_{fd}^{i+1}$, $\hat{\boldsymbol{z}}_{jd}^{i+1}$ of the i+1th throwing trial by using (29) and (30).
4) We solve the nonlinear optimization problem (31) and obtain the throwing parameter $\boldsymbol{u}^{i+1}$.
5) The desired arm trajectory $\theta(t)$ and $\dot{\theta}(t)$ are obtained by solving (19) for the parameter $\boldsymbol{u}^{i+1}$. The robotic arm is controlled with a PD-compensator along with the desired arm trajectories, and the object is thrown.
6) We measure several positions and orientations of the flying object with the camera, and estimate the object's ballistic trajectory through the least square approximations. We obtain the actual object's position and orientation $\boldsymbol{z}_f^{i+1}$ and $\boldsymbol{z}_j^{i+1}$ $(j = 1, \cdots, k)$ from the estimated trajectory, and return to the step 2.

V. Experiment

A. Throwing Robot System

To verify the effectiveness of the proposed learning control, we develop the throwing robot system which can perform the throwing in the vertical plane as shown in Fig.6. The robotic 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 controlled 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 throwing radius can be changed by adjusting the position of the stopper, and its maximum radius is 0.31 m. The moment of inertia of the arm is 0.0762kgm^2.

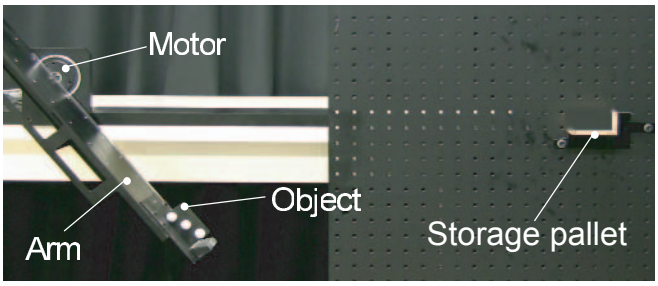

Fig. 6. Throwing robot system

We use a rectangular parallelepiped wooden block as the object whose mass and size are 63.5 g and $8\text{cm} \times 4\text{cm} \times 4\text{cm}$, respectively. The moment of inertia of the wooden block is estimated on the assumption that the block is homogenous and symmetrical. We measure the positions of markers put on the object during the flight with a camera at a rate of 1 kHz to estimate the object's position and orientation trajectories easily in this experiment. The L-shaped storage pallet is fixed to the wall in the vertical plane.

B. Experimental Condition

We lay the object on the arm surface sideways at the initial time. The storage pallet is located at the final goal $\boldsymbol{z}_{fd} = [0.5 \text{ m}, \ 0.2 \text{ m}, 135\text{deg }]$ with respect to the reference frame. We set five desired path points $\boldsymbol{z}_{jd}$ $(j = 1, \cdots, 5)$ on the minimal impact trajectory. Let the weight for the optimization be $\boldsymbol{W}_j = \text{diag}\left[(\pi/2)^{-1}, \dot{\theta}_{max}^{-1}, l_{max}^{-1}\right]$ $(j = 1, \cdots, 5)$, and the parameters for the iterative learning be $\boldsymbol{K}_f = \text{diag}\,[0.1, 0.1, 0, 1]$ and $\boldsymbol{K}_j = \text{diag}\,[0.3, 0.3, 0, 3]$ $(j = 1, \cdots, 5)$. Since throwing to the final goal is achieved much easier than that to the path points in the proposed control method, it is not necessary to weight $\boldsymbol{K}_f$ more than $\boldsymbol{K}_j$.

C. Experimental Results

Fig. 7 shows the transition of the error norms of the position and orientation at the final goal. The value of the error norm is reduced gradually by repeating the learning. The error norm of the 21st throwing is less than 3.0 mm and 1.0 deg, respectively, and the final object configuration is sufficiently close to the desired final goal.

Fig.8 shows the object's trajectories at each trial. The desired minimal impact trajectory is described by a heavy line. Although the final trial trajectory deviates somewhat from the desired trajectory, the object's trajectories gradually approach the desired trajectory by repeating the learning. Since the arm has only one redundant control input for tracking the desired trajectory, we cannot completely make the trial trajectory coincide with the desired trajectory.

Fig. 9 shows snapshots of the successful throwing motion at the 21st trial. The object was thrown to the storage pallet along near the minimal impact trajectory and landed without rebounding from the storage pallet.

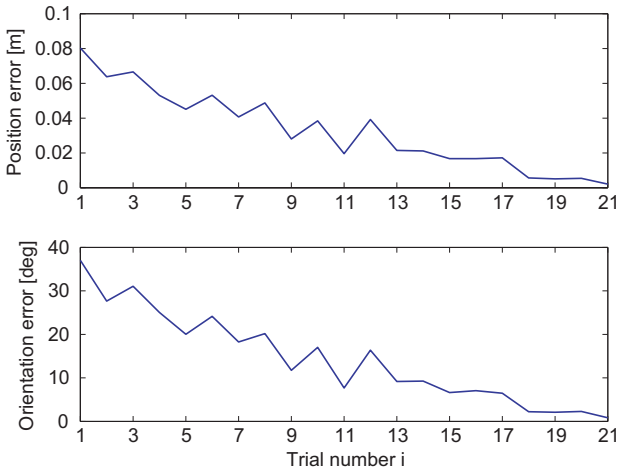

Fig. 7. Performance error of the final goal at each trial

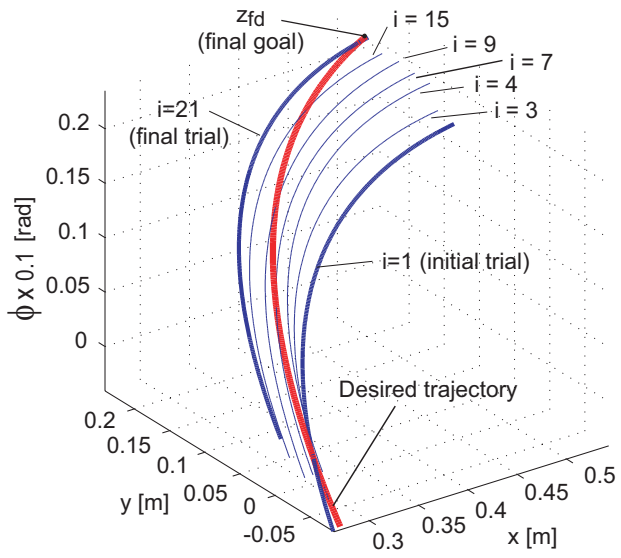

Fig. 8. Object's trajectories at each trial

VI. Conclusion

This paper proposed the iterative learning control for the throwing manipulation which can not only control the position and orientation of the object at the final goal, but also move the object along the minimal impact trajectory as closely as possible to reduce the impact at landing. We showed experimentally the validity of the proposed control method.

It is not strictly appropriate to define the minimal impact trajectory calculated using a model as a desired trajectory. When the model is inaccurate, the obtained trajectory will not be a minimal impact trajectory in reality. We will develop a method for generating a desired trajectory without suffering from model uncertainty.

In future, we will consider the design of catching devices which can hold the object without rebounding and slipping at landing. Integrating throwing devices and catching devices

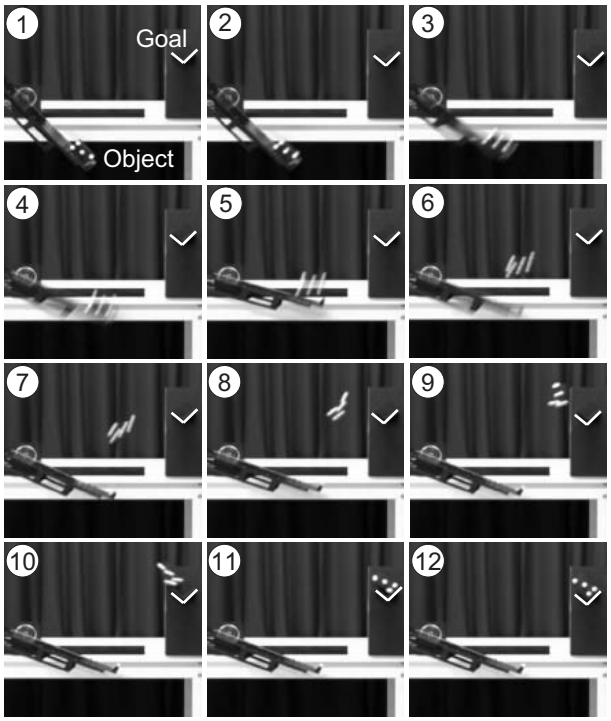

Fig. 9. Snapshots of the throwing of the rectangular object into the storage pallet at final goal

will help to construct robust flexible manufacturing systems (FMS).

References

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

[2] H. Frank, N. W-Wojtasik, B. Hagebeuker, G. Novak and S. Mahlknecht, Throwing Objects - A bio-inspired Approach for the Transportation of Parts, *Proc. of IEEE Int. Conf. on Robotics and Biomimetics*, 2006, pp.91-96.

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

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

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

[6] H. Miyashita, T. Yamawaki and M. Yashima, Control for Throwing Manipulation by One Joint Robot, *Proc. of IEEE Int. Conf. on Robotics and Automation*, 2009, pp.1273-1278.

[7] W. Mori, J. Ueda and T. Ogasawara, 1-DOF Dynamic Pitching Robot that Independently Controls Velocity, Angular Velocity, and Direction of a Ball: Contact Models and Motion Planning, *Proc. of IEEE Int. Conf. on Robotics and Automation*, 2009, pp.1655-1661.

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

[9] 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, 1993, pp.925-931.

[10] 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, 1993, pp.913-918.