The 2010 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 18-22, 2010, Taipei, Taiwan

Parts Assembly by Throwing Manipulation with a One-Joint Arm

Hideyuki Miyashita, Tasuku Yamawaki and Masahito Yashima

Abstract— **The present paper proposes the learning control method for the throwing manipulation which can control not only the position but also the orientation of the polygonal object more accurately and robustly by low-degree-of-freedom robotic arm. We show experimentally the validity of the proposed control method with the one-degree-freedom robotic arm. We also demonstrate the usefulness of the throwing manipulation by applying it to sorting task and assembly task on experiments.**

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 or less required; (2) flexibility of the systems is enhanced; and (3) the transportation processes are speeded up [3]. We consider the applications of the throwing manipulation to transporting, orienting, sorting and assembling 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 sorts 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 [1], [7], [8], [9], [10]. This is because the throwing manipulation cannot only manipulate the object outside the movable range of the robot, but also manipulate the position and orientation of the object arbitrarily by a robotic arm with fewer control inputs.

Lynch et al [5], [6] present the motion planning for the dynamic manipulation such as throwing an object and propose the open-loop controller based on nonlinear optimization techniques. However, they do not aim at throwing the object accurately and robustly. The experimental results show that the controller is subject to the modeling error and sensing error. Aboaf [2] applies a learning control method to the throwing of a point mass object to overcome these problems. However, constraints on dynamical robot system and actuator's performance limits, etc are not considered to find control inputs. The authors [7] propose the control method for throwing point mass object accurately, which combines the learning control techniques and the nonlinear optimization techniques. As far as I know, there has been no work which takes into consideration the orienting a polygonal object accurately by a throwing manipulation, which is very important in the case of sorting and assembling various types of objects, as shown in Fig. 1.

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

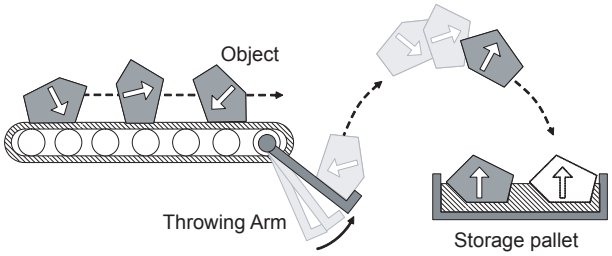

Fig. 1. Application of the throwing manipulation

The present paper proposes a learning control method for the throwing manipulation which can control not only the position but also the orientation of the polygonal object more accurately and robustly to uncertainties. We show experimentally the validity of the proposed control method and the feasibility of assembly process by the throwing manipulation.

The paper is organized as follows: In Section III we study the dynamical constraint on the throwing manipulation. Section IV presents the motion planning by optimization, and Section V proposes the learning control method. Finally, Section VI presents experiments with a one-degree-freedom robotic arm.

II. NOTATION AND ASSUMPTIONS

We consider a throwing manipulation of a polygonal 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 $-g < 0$ acts in the $-y$ direction. The position and orientation of the object in the x-y frame is described as $(x,\ y,\ \phi)$. The angle of the arm is θ, which is set to 0 deg when the arm is horizontal. The top surface of the arm intersects with the pivot point. The arm throws the object by counterclockwise rotation. It is assumed that the thrown object is captured by a storage pallet fixed at a goal without slipping, rolling or rebounding in order to simplify the analysis.

A frame u-v is located on the center of mass of the object, where the $+v$ direction is opposite to gravity when the arm angle is $\theta = 0$. The moment of inertia of the object and its

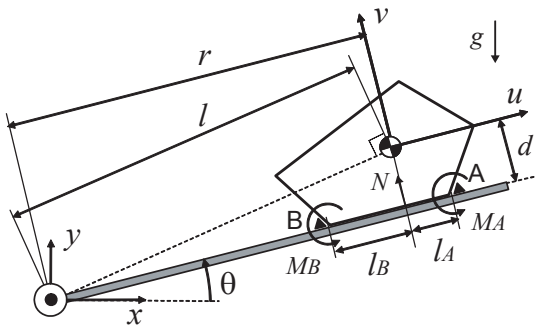

Fig. 2. Notation for throwing manipulation

mass are J and m, respectively. The polygonal object makes line contact with the arm surface, and the vertices of the contacting edge are A and B. The distance in the direction of u between the object center of mass and the vertices A and B are l_A and l_B, respectively. The height of the center of mass from the arm surface is d. The distance between the object center of mass and the pivot point is l. When the arm is horizontal, the x location of the object center of mass in the reference frame 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. We assume that there are no friction between the object and the stoppers.

III. Throwing Manipulation

The motion of a throwing manipulation is divided broadly into two phases before and after an object's release.

A. Motion before Object's Release

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.

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

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

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

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

$$\begin{aligned} M_A &= J\ddot{\theta} - ml_A r\ddot{\theta} + md^2\ddot{\theta} \\ &\quad + ml_A d\dot{\theta}^2 + mdr\dot{\theta}^2 - ml_A gc\theta - mdgs\theta \end{aligned} \tag{2}$$

$$\begin{aligned} M_B &= J\ddot{\theta} + ml_B r\ddot{\theta} + md^2\ddot{\theta} \\ &\quad - ml_B d\dot{\theta}^2 + mdr\dot{\theta}^2 + ml_B gc\theta - mdgs\theta \end{aligned} \tag{3}$$

where $s\theta = \sin\theta$ and $c\theta = \cos\theta$.

To achieve the stable throwing, the robotic arm has to accelerate such that there is no relative motion between the object and the arm before the object's release, which is called the *dynamic grasp*. We take into account the following constraints to achieve the dynamic grasp.

(i) Contact Constraints; which make the object maintain the contact with the surface of the arm.

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

(ii) Un-rolling Constraints; which prevent the object from rolling about the vertices A and B.

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

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

2) Arm Constraints: The arm is subject to the constraints on the actuator performance and the robot mechanism.

(i) Joint Angle Constraints

$$\theta_{min} < \theta(t) < \theta_{max} \tag{7}$$

To release the object in the right half space, we set $\theta_{max} = \pi/2$ and $\theta_{min} = -\pi/2$, respectively.

(ii) Actuator's Torque-Velocity Limits

$$\dot{\theta}_{min} < \dot{\theta}(t) < \dot{\theta}_{max} \tag{8}$$

$$\tau_{min} < \tau(t) < \tau_{max} \tag{9}$$

$$\tau_{max} = \xi(\dot{\theta}_{max}) \tag{10}$$

where (10) shows the relationship between the maximum torque and maximum velocity of an actuator, which is determined by the actuator's operation range.

B. Motion after Object's Release

After the object's release, the object's motion during a free-flight is determined by the arm's state $(\theta_t, \dot{\theta}_t)$, a throwing radius l_t at a release point and a flight time t_f.

Since we have four variables $(\theta_t,\ \dot{\theta}_t,\ l_t,\ t_f)$ as parameters in the throwing manipulation, the one-joint robotic arm can control four object's state variables in a six-dimensional planar state space $(x,\ y,\ \phi,\ \dot{x},\ \dot{y},\ \dot{\phi})$ through one throwing.

In this paper, in addition to the object's position and orientation $(x,\ y,\ \phi)$, we choose the vertical velocity $\dot{y}$ at a goal as the controlled object's variables. By specifying an appropriate velocity $\dot{y}$, we can find object paths with various loci which reach to the same object's position and orientation, which helps to plan the object's path so as to avoid obstacles in front of the goal. We can also adjust an impact velocity at landing. The condition for the velocity at the goal expands the feasibility of various throwing tasks.

We define the object's state variables in a flight time t_f, which can be controlled by the one-joint robotic arm, as

$$\boldsymbol{z} = [\ x(t_f),\ y(t_f),\ \phi(t_f),\ \dot{y}(t_f)\ ]^T = \boldsymbol{g}(\boldsymbol{u}) \in \Re^4, \tag{11}$$

which can be expressed by free-flight ballistic equations using parameters $\boldsymbol{u} = [\ \theta_t,\ \dot{\theta}_t,\ l_t,\ t_f\ ]^T \in \Re^4$.

The object's four state variables $\boldsymbol{z}$ is uniquely determined by specifying the four-dimensional parameters $(\theta_t,\ \dot{\theta}_t,\ l_t,\ t_f)$, and vice versa. If we move the robotic arm so as to achieve the arm's state $(\theta_t,\ \dot{\theta}_t)$ at a release point,

then the object on the arm surface at a throwing radius l_t can be thrown to a desired state $\boldsymbol{z}$ in a time t_f after the object's release. In the next section, we present the motion planning of the robotic arm.

IV. Motion Planning

For a given initial state of the object and the robotic arm, the motion planning problem is to find a robotic arm's trajectory to throw the object to the goal state under the constraints on the dynamic grasping and robotic arm motion.

A. Joint Trajectory Parameterization with B-splines

We use uniform cubic B-splines to represent the joint trajectory since the continuity of the trajectory between adjacent B-spline segments is guaranteed [4].

The motion time interval $[0,\ t_t]$ from the start of the arm's motion until just before the object's release is divided into n knots with evenly spaced in time, Δt, such as $t_0(=0) < t_1 < \cdots < t_n(=t_t)$. The joint trajectory $\theta_i(t)\ (i=0,1,\cdots,n)$ on each interval $[t_i,\ t_{i+1}]$ is expressed by

$$\theta_i(s) = \sum_{j=i-1}^{i+2} \widehat{\theta}_j B_j(s) \tag{12}$$

where $t(s) = t_i + s\Delta t,\ (0 \le s \le 1)$, and $\widehat{\theta}_j$ and $B_j(s)$ are the control points and basis functions of the uniform cubic B-spline. The final joint angle is equivalent to the throwing angle such as $\theta_n = \theta_t$.

The release time $t_t\ (=t_n = n\Delta t)$ can be freely chosen. For a given n, Δt is free. Since the cubic B-splines lie within the convex hull formed by the control points $\widehat{\theta}_j$, the joint trajectory (12) passes near the control points. To obtain an optimal arm trajectory, we optimize the control points

$$\widehat{\boldsymbol{\theta}} = [\ \widehat{\theta}_{-1},\ \widehat{\theta}_0, \cdots, \widehat{\theta}_{n+1}\ ]^T \in \Re^{n+3} \tag{13}$$

and the time $\Delta t\ (< \Delta t_{max})$ including some of the joint's initial and final states by formulating the motion planning problem as a constrained optimization programming problem. We show the constraints and objective function of the optimization programming problem below.

B. Finite-dimensional Constraints

To prevent the robotic arm from violating the constraints on the dynamic grasping due to unknown disturbances which are not shown in the nominal model and from saturating control inputs, we find the joint trajectories which satisfy constraints (4)$\sim$(9) with sufficient tolerance. We introduce margin variables ΔN, ΔM_A, ΔM_B, $\Delta\theta$, $\Delta\dot{\theta}$, $\Delta\tau$ into each constraints, respectively, and maximize the magnitudes of the margin variables in the optimization problem.

We rewrite the dynamic grasping constraints (4)$\sim$(6) by using margin variables $\Delta N,\ \Delta M_B > 0,\ \Delta M_A < 0$. In addition, substituting (12) into them yields finite-dimensional constraints sampled at the knot points $t_0, \cdots, t_n$ as

$$N\big(\theta_i,\dot{\theta}_i,\ddot{\theta}_i\big) > \Delta N, \qquad i=0,\cdots,n \tag{14}$$
$$M_A\big(\theta_i,\dot{\theta}_i,\ddot{\theta}_i\big) < \Delta M_A, \qquad i=0,\cdots,n \tag{15}$$
$$M_B\big(\theta_i,\dot{\theta}_i,\ddot{\theta}_i\big) > \Delta M_B, \qquad i=0,\cdots,n \tag{16}$$

Similarly, the arm's motion constraints (7)$\sim$(9) can be converted into finite-dimensional constraints by using margin variables $\Delta\theta,\ \Delta\dot{\theta},\ \Delta\tau > 0$ as follows:

$$\theta_{min} + \Delta\theta < \theta_i < \theta_{max} - \Delta\theta, \qquad i=0,\cdots,n \tag{17}$$
$$\dot{\theta}_{min} + \Delta\dot{\theta} < \dot{\theta}_i < \dot{\theta}_{max} - \Delta\dot{\theta}, \qquad i=0,\cdots,n \tag{18}$$
$$\tau_{min} + \Delta\tau < \tau\big(\theta_i,\dot{\theta}_i,\ddot{\theta}_i\big) < \tau_{max} - \Delta\tau,\ i=0,\cdots,n \tag{19}$$

where the joint torque τ is expressed by the motion equation of the robotic arm, which is

$$\tau\big(\theta_i,\dot{\theta}_i,\ddot{\theta}_i\big) = J_R(\theta_i)\ddot{\theta}_i + h(\theta_i,\dot{\theta}_i) \tag{20}$$

where J_R is the inertia matrix and h represents the gravity, friction force and reaction force from the object.

C. Objective Function

Since the throwing manipulation is generally dynamic and fast and needs larger joint driving torques, it is easy for the object to roll on the surface of the arm or to break contacts with the arm during the manipulation by violating the dynamic grasp constraints. We are greatly concerned with making the throwing manipulation more robust and stable to disturbances and trajectory errors, etc.

In order that the constraints on the dynamic grasping and the arm's motion can be satisfied with sufficient tolerance, we maximize the margin variables

$$\boldsymbol{d} = \begin{bmatrix}\Delta N & \Delta M_A & \Delta M_B & \Delta\theta & \Delta\dot{\theta} & \Delta\tau\end{bmatrix}^T \in \Re^6 \tag{21}$$

in the optimization problem shown below.

D. Optimization Programming Problem

For a given object's goal state $\boldsymbol{z}_d$, we find an optimal arm trajectory by solving the following optimization problem.

$$\max: \quad \frac{1}{2}\boldsymbol{d}^T\boldsymbol{W}\boldsymbol{d} \tag{22}$$
$$\text{subj. to:} \quad \boldsymbol{z}_d = \boldsymbol{g}(\boldsymbol{u}) \tag{23}$$
$$\boldsymbol{c} \le \boldsymbol{0} \tag{24}$$
$$\dot{\theta}_0 = 0 \tag{25}$$
$$\text{find:} \quad \widehat{\boldsymbol{\theta}} \ \text{ and } \ \Delta t$$

Eq. (22) is the objective function to maximize the margin variables, where $\boldsymbol{W}$ is a weight matrix. The free-flight model is used as the equality constraint (23). The inequality constraints shown in Section. IV-B are combined as shown in (24). As boundary conditions, the initial angular velocity is specified by (25). In contrast, the initial angle and the final angle and velocity are derived from this optimization. We use sequential quadratic programming (SQP) to solve the nonlinear programming. Once we find the arm trajectory parameters $(\widehat{\boldsymbol{\theta}},\ \Delta t)$, we can calculate the joint trajectory using (12).

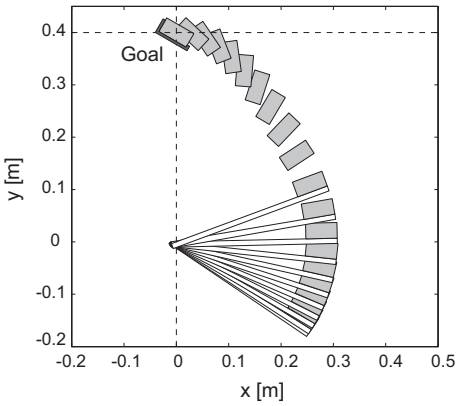

Fig. 3. Throwing manipulation found by the optimization.

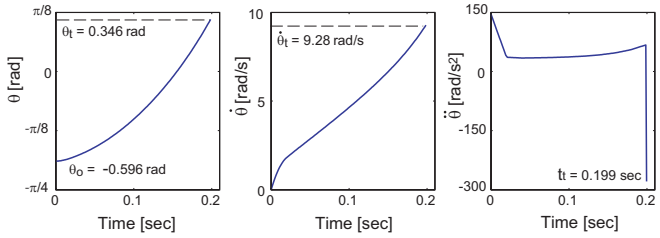

Fig. 4. Arm trajectories found by the optimization.

E. Simulation Results

Consider a throwing manipulation of a rectangular object to a goal. The parameter values of the object and robotic arm are described in Section VI. The optimization programming problem is formulated with $n = 10$. Since a local optimum found by SQP depends on the initial guess, SQP is solved with many different initial guesses given at random.

Fig. 3 shows the simulation result found by the optimization which represents the motions of the object and robotic arm. The object is thrown to the goal $(x,\ y,\ \phi) = (0\,\mathrm{m},\ 0.4\,\mathrm{m}, 150\,\mathrm{deg})$ with the velocity $\dot{y} = 0$. We assume that the object lands at the storage pallet located at the goal without slipping and rebounding to simplify the analysis.

Fig. 4 shows the corresponding arm trajectories, which have the initial state $(\theta_0, \dot{\theta}_0) = (-0.596\,\mathrm{rad}, 0\,\mathrm{rad/s})$ and final state $(\theta_t, \dot{\theta}_t) = (0.346\,\mathrm{rad}, 9.28\,\mathrm{rad/s})$ at a release time $t_t = 0.199\,\mathrm{s}$, respectively. The arm is maximally decelerated at the release time to set the object free instantaneously. After the release time, the robot arm is moved with the joint acceleration at the release time until the arm stops, which is not described in Fig. 4.

Fig. 5 shows the arm trajectories obtained for different initial guesses. The arm trajectory in Fig. 4, which has the maximum value of the objective function, is chosen among them.

V. Learning Control

If we have an exact throwing model, then we can obtain an optimal arm trajectory to throw an object to the goal by using the motion planning method discussed in the previous section. To model the throwing manipulation system accurately, we are required not only to implement parameter identification whenever the change of the object, but also to consider the dynamics of the interaction between the robot and the object, as well as the correction of the vision sensor etc. In general, it is impossible to obtain exact models.

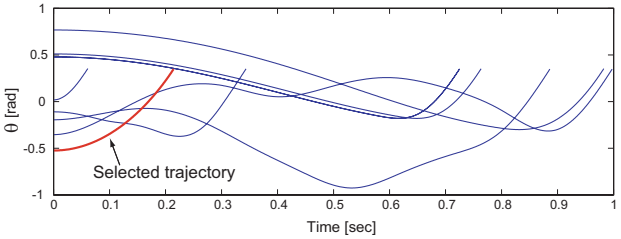

Fig. 5. Arm trajectories found for different initial guesses, where $\Delta t_{max} = 100$ msec.

In the present paper, instead of raising accuracy of an approximate model itself, we overcome the problems of the model with errors by applying an iterative learning control method to the throwing manipulation.

A. Introduction of Virtual Goal

To find an optimal arm trajectory, instead of setting an object's goal state $\boldsymbol{z}_d$ for the optimization programming problem, we introduce a virtual goal state $\widehat{\boldsymbol{z}}_d$, which is updated at each throwing trial in order that the robotic arm can throw the object to the goal state $\boldsymbol{z}_d$ as closely as possible. The basic idea of the learning control using the virtual goal is shown in [2].

The virtual goal is obtained as follows. The error between the goal state $\boldsymbol{z}_d$ and the actual state $\boldsymbol{z}$ is given by

$$\boldsymbol{e}^j = \boldsymbol{z}_d - \boldsymbol{z}^j \tag{26}$$

where j denotes the trial number.

According to the value of the error, the virtual goal $\widehat{\boldsymbol{z}}_d^{j+1}$ of the $j+1$th throwing is updated as

$$\widehat{\boldsymbol{z}}_d^{j+1} = \widehat{\boldsymbol{z}}_d^{j} + \boldsymbol{K}\boldsymbol{e}^i \tag{27}$$

where the diagonal matrix $\boldsymbol{K}$ is gain parameters.

Replacing the goal state $\boldsymbol{z}_d$ in the left-hand side of (23) with the virtual goal state $\widehat{\boldsymbol{z}}_d$, we find an optimal arm trajectory by solving iteratively the optimization programming problem (22)-(25) at each throwing trial.

B. Learning Control Algorithm

The algorithm of the learning control is shown below.

1) The optimization programming problem is solved to obtain the arm trajectory for the first throwing ($j = 1$). In the same manner described in step 5 and 6, we make the robotic arm throw the object and measure the actual object state $\boldsymbol{z}^1$ with a camera.
2) We calculate the error $\boldsymbol{e}^j$ of the j th throwing by using (26). If the error norm is greater than the value of the threshold, we move to the next step. Otherwise we

assume that the arm succeeds in throwing the object to the goal, and the learning control is terminated.

3) The virtual goal $\hat{\boldsymbol{z}}_d^{j+1}$ is updated by using (27).
4) The optimization programming problem is solved to obtain the arm trajectory for the j+1 th throwing.
5) To throw the object, the robotic arm is controlled with a PD-compensator so that the arm can track the trajectory obtained in step 4.
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 state $\boldsymbol{z}^{j+1}$ from the estimated trajectory and return to step 2.

VI. Experiment

A. Throwing Robot System

We have developed the throwing robot system which can perform the throwing in the vertical plane as shown in Fig.6. The aluminum arm is centrally-mounted and 62 cm long and 15 cm wide plate. The arm is controlled with PD compensators and is driven by the AC motor with a harmonic drive gear, which has the maximum torque $\tau_{\max} = 18$ Nm and the maximum speed $\dot{\theta}_{\max} = 4\pi$ rad/s. The throwing radius can be changed by adjusting the position of a thin plate on the arm surface. The object on the plate is constrained with 2 mm high stoppers so as not to slide on the arm surface.

We use a rectangular parallelepiped wooden block as the object whose mass and size are 29 g and 6 cm $\times$ 3 cm $\times$ 3 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 trajectory. The concave storage pallet of wood is fixed at a goal in the vertical plane.

B. Experimental Results

The goal state $\boldsymbol{z}_d$ is set to $(0\,\text{m},\ 0.4\,\text{m},\ 150\,\text{deg},\ 0\,\text{m/s})$. Fig. 7 shows the transitions of the error norms of the position, orientation and vertical velocity at the goal. The value of the error norm is reduced gradually by repeating the learning. The error norm of the 10th throwing is less than 2.5 mm, 0.1 deg and 0.01 m/s, respectively. These experimental results show that the proposed method enables the robotic arm to throw the object to the goal accurately. Fig.8 shows the transitions of arm trajectories found by the optimization at each iteration. The trajectories are modified by repeating the learning. Fig. 9 shows snapshots of the successful throwing motion.

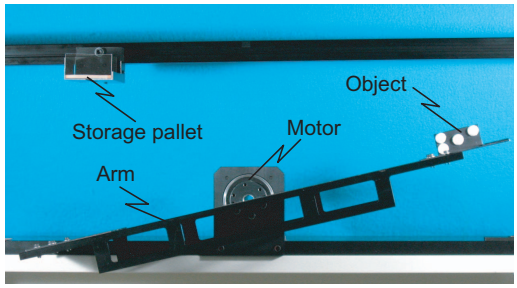

Fig. 6. Throwing robot system

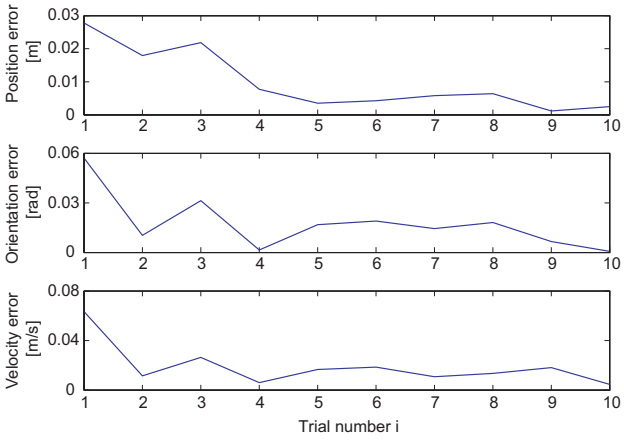

Fig. 7. Transitions of error norm of object's state at the goal

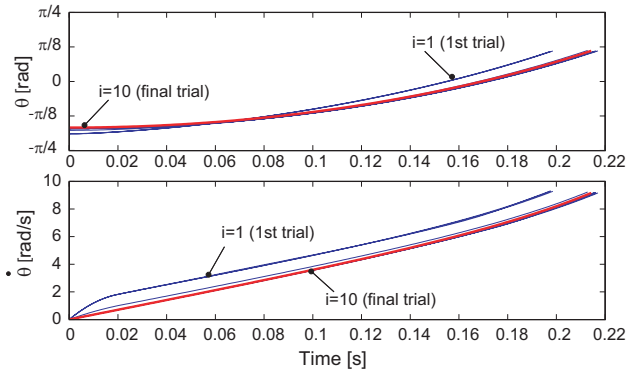

Fig. 8. Arm trajectories found by the optimization at each trial

In this experiment, it is crucial for the stable and robust throwing to keep the un-rolling constraint about the vertex A of the object. In order that the un-rolling constraint (5) can hold and the object will not start rolling on the arm surface, we find the robotic arm trajectory so that the margin ΔM_A become sufficiently large in the optimization problem.

C. Application to Sorting/Assembly Tasks (Attached Video)

The first video shows the trial throwing motion in the iterative learning mentioned above. By repeating the throwing, the robotic arm succeeds in throwing the object into the storage pallet with the zero vertical velocity, which is located on the apex point of the object's ballistic trajectory.

The second video shows the sorting task in which the robotic arm throws three objects into storage pallets with an appropriate orientation, as shown in Fig. 10. In the first throwing (Fig. 10(a)), the zero vertical velocity is set at the first goal. On the other hand, in the second throwing (Fig. 10(b)) we set the negative vertical velocity at the second goal. The object reaches the pallet with descending after passing through the apex point of the ballistic trajectory. In the third

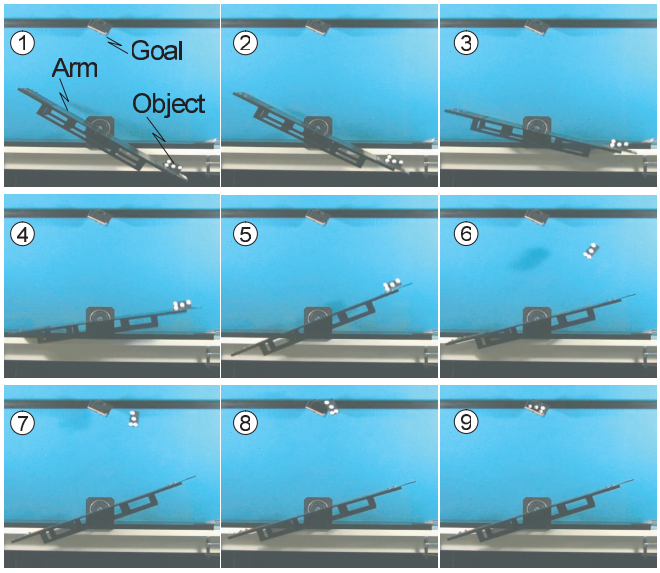

Fig. 9. Snapshots of the throwing of the rectangular object into the pallet

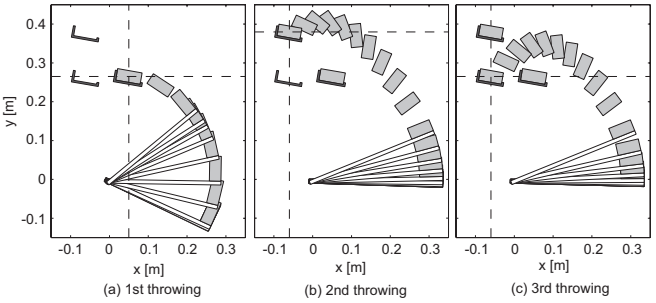

Fig. 10. Sorting task application on video

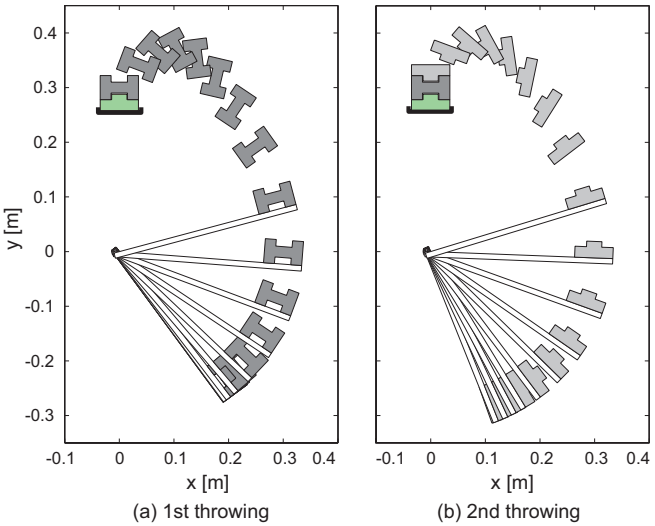

Fig. 11. Parts assembly task application on video

throwing (Fig. 10(c)), the robotic arm can throw the object to the third goal avoiding the first and second pallets by adjusting the negative vertical velocity at the goal. If the negative vertical velocity with a larger (smaller) magnitude should be set, the object will collide with the upper (lower) pallet.

The third video shows the parts assembly task with H- and T-shaped parts as shown in Fig. 11. First, the robotic arm throws the H-shaped part and slots it onto the T-shaped part fixed upside down (Fig. 11(a)). Next, the robotic arm throws the other T-shaped part and inserts it into the former H-shaped part (Fig. 11(b)). In order to prevent the flying part from colliding with the previously-inserted parts from the side, the vertical velocity at the goal is set so that the linear velocity at the goal may put toward the vertical direction as much as possible.

These videos show the throwing manipulation controlled by the proposed learning algorithm can achieve the high positioning performance.

VII. Conclusion

This paper proposed the learning control method for the throwing manipulation which can control not only the position but also the orientation of the polygonal object more accurately and robustly to uncertainties. We showed experimentally the validity of the proposed control method with the one-degree-freedom robotic arm. We demonstrated the usefulness of the throwing manipulation by applying it to sorting task and assembly task on experiments.

In future, we will consider the design of storage pallets or catching devices which can surely hold the object without rebounding and slipping at landing. Integrating throwing devices and catching devices will help to construct robust flexible manufacturing systems (FMS).

References

[1] H. Arisumi, K. Yokoi and K. Komoriya, Casting Manipulation, Midair Control of a Gripper by Impulsive Force, *IEEE Trans. on Robotics*, Vol.24, No.2, 2008, pp.402-415.

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

[3] 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.

[4] Y-C Chen, Solving Robot Trajectory Planning Problems with Uniform Cubic B-Splines, *Optimal Control Applications and Methods*, Vol.12, 1999, pp.247-262.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] T. Senoo, A. Namiki and M. Ishikawa, High-speed Throwing Motion Based on Kinetic Chain Approach, *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robotics and Systems*, 2008, pp.3206-3211.

[10] 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, 2003, pp.2920-2925.