Journal of System Design and Dynamics

Vol.5, No.1, 2011

Parts Assembly and Sorting by Throwing Manipulation: Planning and Control*

Hideyuki MIYASHITA**, Tasuku YAMAWAKI** and Masahito YASHIMA**

** Dept. of Mechanical Systems Engineering, National Defense Academy of Japan

1-10-20 Hashirimizu, Yokosuka 239-8686, Japan

E-mail: yashima@nda.ac.jp

Abstract

This paper presents optimal trajectory planning and iterative learning control for a throwing manipulation which can control not only the position but also the orientation of a polygonal object robustly to uncertainties by low-degree-of-freedom robotic arm. We show experimentally the validity of the proposed method with the one-joint robotic arm. We also demonstrate the usefulness of the throwing manipulation by applying it to the parts assembly and sorting on experiments.

Key words : Throwing Manipulation, Parts Assembly, Sorting, Iterative Learning Control

1. Introduction

In the present automated assembly lines or logistics systems, conveyors and vehicles are used for transporting packages or parts, and industrial robots with grippers which are designed to grasp particular parts are working for parts assembly, which make the speed of production flow and physical distribution be slow and the plant and equipment costs be high. Arai et al.[2] discussed the use of non-grasping manipulations such as throwing, hitting, pushing, etc. in assembly to solve these problems caused by the conventional pick-and-place operations. Frank et al.[6] mentioned that the introduction of throwing in a manufacturing process can produce the following advantages : (1) mechanical equipments are simplified or less required; (2) flexibility of the systems is enhanced; and (3) the operation processes are speeded up. We consider the applications of the throwing manipulation to transporting, orienting, sorting and assembling a variety of objects.

Figure 1 shows one of the application examples, in which a one joint robotic arm which has no special grippers throws H-and T-shaped parts with various orientation carried by a belt-conveyer in order to assemble them with appropriate orientation and position in a storage pallet. The assembled parts in the storage pallet may then go through another operation process such as an inspection or may be transported to a warehouse.

Instead of a manipulation of an object with grasp, the throwing manipulation with a low-degree-of-freedom robot has drawn attention recently in the field of robotics: Throwing and Tossing[1],[7],[9]–[11],[13],[15]–[17], Juggling[4],[8],[14] and Casting[3]. 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 simple robotic arm with fewer control inputs. If the robotic arm can throw the object accurately and robustly to the goal, the throwing manipulation will be useful for assembling and sorting such small electro-mechanical parts as can tolerate some collision impact at landing, as shown in Fig. 1.

Lynch et al.[7] analyzed a dynamic manipulation such as throwing, snatching, rolling, etc., which dynamically moves a polygonal object's position and orientation using inertia force and gravity, and proposed an open-loop controller for throwing the polygonal object, taking into consideration various constraints on a dynamic manipulation, actuator's performances, etc. Tabata et al.[16] showed that a tossing manipulation can carry a spherical object to any

*Received 19 Aug., 2010 (No. 10-0365)
[DOI: 10.1299/jsdd.5.139]

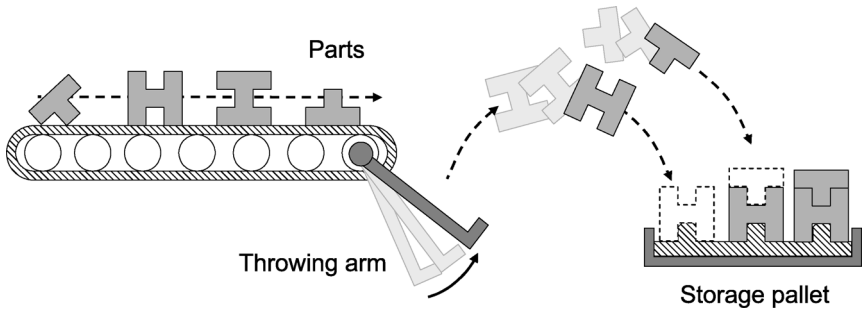


Fig. 1 Application of the throwing manipulation to parts assembly

goal position. Mori et al.[11] demonstrated that a one joint pitching robot can control three variables of a disk independently by simulations and experiments. Although it is actually difficult to develop an exact throwing model since there are dynamical interactions between the robot and the object, these works do not consider introducing any feedback. Aboaf[1] applied a learning control method with visual feedback to the planar robotic throwing to solve the problems of the modeling errors etc. and showed the effectiveness by experiments on throwing a rubber ball to a desired position. The control method, however, cannot find control inputs in consideration of constraints. In order to handle both the uncertainties and a variety of the constraints simultaneously, Yashima et al.[17] proposed a control method to throw the object to any position in the vertical plane by integrating the learning control method and optimization techniques. The above-mentioned works seem to partly succeed in the throwing manipulation, however, there are still some problems with regard to the following; controlled object's states and robustness, in order to realize assembly tasks by throwing with a real robot.

The main contributions of this paper are as follows:

(1) Based on our previous control method[17], we propose a control method for the throwing manipulation of a polygonal object, which can control not only the positions but also the orientation and the velocity at a goal, a total of four state variables in a six-dimensional planar object's state space. By adding the velocity at the goal in the controlled object's states, we can plan a variety of object paths such as avoiding obstacles.

(2) Since the throwing manipulation requires dynamical and fast motion of the object, a slight unknown disturbance easily causes the failure of throwing by violating dynamical constraints. To enhance robustness and stability of the throwing manipulation, the throwing is planned so that the dynamical constraints can be satisfied with sufficient tolerance.

(3) We show experimentally the validity of the proposed method with a one-joint robotic arm and some applications. As far as the authors know, there has been no work which shows that the throwing manipulation is applicable to the parts assembly and sorting.

The paper is organized as follows: In Section 3 we study the dynamical constraint on the throwing manipulation. Section 4 presents the motion planning by optimization, and Section 5 proposes the iterative learning control method. Finally, Section 6 presents experiments with a one-degree-freedom robotic arm.

2. 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.

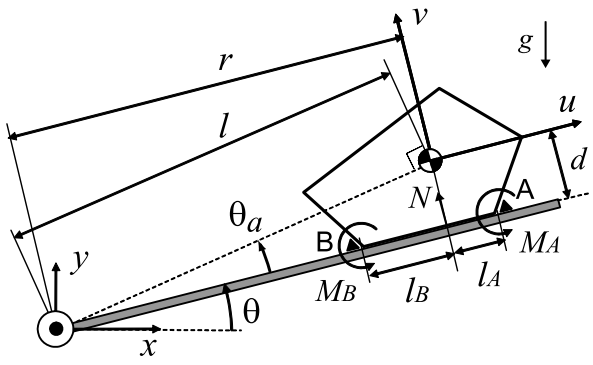


Fig. 2 Notation for throwing manipulation

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 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, which is called a throwing radius and is treated as a variable since the object's position in the direction of u can be adjusted. When the arm is horizontal, the x location of the object center of mass in the reference frame is r. The line connecting the pivot point and the center of mass has an offset angle θ_a with respect to the arm surface.

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.

3. Throwing Manipulation

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

3.1. 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.

3.1.1. 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,

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

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

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

To achieve the stable throwing, the robotic arm has to accelerate such that there is no

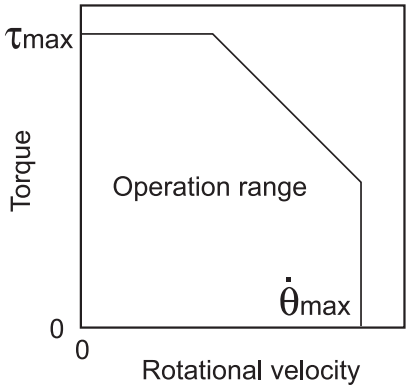


Fig. 3 Actuator's operation range for torque and velocity

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.

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

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

3.1.2. 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 $\dot{\theta}_{max}$, $\dot{\theta}_{min}$ and τ_{max}, τ_{min} are the maximum and minimum of velocity and torque of an actuator, respectively. Equation (10) shows the relationship between the maximum torque and maximum velocity of an actuator, which is determined by the actuator's operation range, as shown in Fig. 3.

3.2. 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 final object state variables in a flight time t_f as

$$\boldsymbol{z} = \begin{bmatrix} x(t_f) \\ y(t_f) \\ \phi(t_f) \\ \dot{y}(t_f) \end{bmatrix} = \begin{bmatrix} l_t c\theta_{ta} - l_t \dot{\theta}_t s\theta_{ta} t_f \\ l_t s\theta_{ta} + l_t \dot{\theta}_t c\theta_{ta} t_f - \frac{1}{2} g t_f^2 \\ \theta_t + \dot{\theta}_t t_f \\ l_t \dot{\theta}_t c\theta_{ta} - g t_f \end{bmatrix} = \boldsymbol{f}(\boldsymbol{u}) \in \Re^4 \tag{11}$$

which can be expressed by free-flight ballistic equations using parameters

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

where $\theta_{ta} = \theta_t + \theta_a$.

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.

4. 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.

4.1. Joint Trajectory Parameterization with B-splines

We use uniform cubic B-splines to represent the joint trajectory since the continuity of the trajectory of joint angle, velocity and acceleration between adjacent B-spline segments is guaranteed[5].

As shown in Fig. 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

$$\theta_i(s) = \sum_{j=i-1}^{i+2} \widehat{\theta}_j B_j(s) \tag{13}$$

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 Eq. (13) passes near the control points. To obtain an optimal arm trajectory, we optimize the control points

$$\widehat{\boldsymbol{\theta}} = [\ \widehat{\theta}_{-1},\ \widehat{\theta}_0, \cdots, \widehat{\theta}_{n+1}\ ]^T \in \Re^{n+3} \tag{14}$$

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.

4.2. Finite-dimensional Constraints

To prevent the robotic arm from violating the constraints on the dynamic grasping due to unknown disturbances and from saturating control inputs, we find the joint trajectories which satisfy constraints Eqs. (4)~(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.

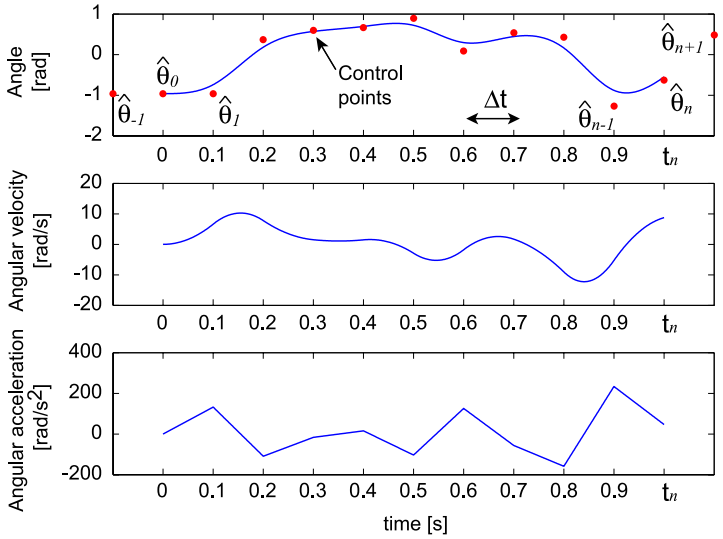


Fig. 4 Generation of joint trajectories using B-spline functions.

We rewrite the dynamic grasping constraints Eqs. (4)~(6) by using margin variables ΔN, $\Delta M_B > 0$, $\Delta M_A < 0$. In addition, substituting Eq. (13) into them yields finite-dimensional constraints sampled at the knot points $t_0, \cdots, t_n$ as

$$N(\theta_i, \dot{\theta}_i, \ddot{\theta}_i) > \Delta N, \qquad i = 0, \cdots, n \tag{15}$$

$$M_A(\theta_i, \dot{\theta}_i, \ddot{\theta}_i) < \Delta M_A, \qquad i = 0, \cdots, n \tag{16}$$

$$M_B(\theta_i, \dot{\theta}_i, \ddot{\theta}_i) > \Delta M_B, \qquad i = 0, \cdots, n \tag{17}$$

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

$$\theta_{min} + \Delta\theta < \theta_i < \theta_{max} - \Delta\theta, \qquad i = 0, \cdots, n \tag{18}$$

$$\dot{\theta}_{min} + \Delta\dot{\theta} < \dot{\theta}_i < \dot{\theta}_{max} - \Delta\dot{\theta}, \qquad i = 0, \cdots, n \tag{19}$$

$$\tau_{min} + \Delta\tau < \tau(\theta_i, \dot{\theta}_i, \ddot{\theta}_i) < \tau_{max} - \Delta\tau, \quad i = 0, \cdots, n \tag{20}$$

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

$$\tau(\theta_i, \dot{\theta}_i, \ddot{\theta}_i) = J_R(\theta_i)\ddot{\theta}_i + h(\theta_i, \dot{\theta}_i) \tag{21}$$

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

The constraints on parameters are given as follows:

$$0 < \Delta t < \Delta t_{max} \tag{22}$$

$$t_t \ (= t_n) = n\Delta t \tag{23}$$

$$t_f > 0 \tag{24}$$

$$0 < l_t < l_{max} \tag{25}$$

$$l_t^2 = r^2 + d^2, \ r > 0 \tag{26}$$

where Δt_{max} is the maximum of Δt, l_{max} is the maximum throwing length, and Eq. (26) shows geometrical constraints on the object putting on the arm surface.

4.3. Objective Function

Since the throwing manipulation generally requires rapid acceleration/deceleration as well as larger joint driving torques, a slight disturbance easily cause the object to roll on

the arm surface 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

$$\boldsymbol{\delta} = \begin{bmatrix} \Delta N & \Delta M_A & \Delta M_B & \Delta\theta & \Delta\dot{\theta} & \Delta\tau \end{bmatrix}^T \in \Re^6 \tag{27}$$

in the optimization problem shown below.

4.4. 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.

$$\max : \ \frac{1}{2}\boldsymbol{\delta}^T \boldsymbol{W} \boldsymbol{\delta} \tag{28}$$

$$\text{subj. to:} \quad \boldsymbol{z}_d = \boldsymbol{f}(\boldsymbol{u}) \tag{29}$$

$$\boldsymbol{c}_I \leq \boldsymbol{0} \tag{30}$$

$$\boldsymbol{c}_E = \boldsymbol{0} \tag{31}$$

$$\dot{\theta}_0 = 0 \tag{32}$$

$$\text{find:} \quad \widehat{\boldsymbol{\theta}},\ \Delta t,\ \boldsymbol{u},\ \boldsymbol{\delta}$$

$$\text{given:} \quad \boldsymbol{z}_d,\ n,\ \theta_{min},\ \theta_{max},\ \dot{\theta}_{min},\ \dot{\theta}_{max},\ \tau_{min},\ \tau_{max},\ \Delta t_{max}$$

Equation (28) 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 Eq. (29). The inequality and equality constraints shown in § 4.2 are combined as shown in Eqs. (30) and (31). As boundary conditions, the initial angular velocity is specified by Eq. (32). In contrast, the initial angle and the final angle and velocity are derived from this optimization. We use sequential quadratic programming (SQP)[12] to solve the nonlinear programming. Once we find the arm trajectory parameters $(\widehat{\boldsymbol{\theta}},\ \Delta t)$, we can calculate the joint trajectory using Eq. (13).

4.5. 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 6. 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.

Figure 5 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 $(x,\ y,\ \phi) = (0\,\text{m},\ 0.4\,\text{m},\ 150\ \text{deg})$ 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. If we set different velocities $\dot{y}$ at the goal, object paths to the same goal position and orientation, which are different from Fig. 5, are obtained.

Figure 6 shows the corresponding arm trajectories from the initial state through the state just before the object's release. The arm is maximally decelerated at the release time t_t 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. 6.

To keep the un-rolling constraint about the vertex A of the object is crucial in this simulation. It is required to find arm trajectories so that the margin ΔM_A become sufficiently large. Figure 7 (a) and (b) show the simulation results with and without taking into consideration the enlargement of the margin ΔM_A, respectively. The latter case is obtained by solving the nonlinear equations Eqs. (29) ~ (32) using the Gauss-Newton method. Colored regions in these figures indicate the tolerable region of the margin ΔM_A. Compared with Fig. 7 (b) which

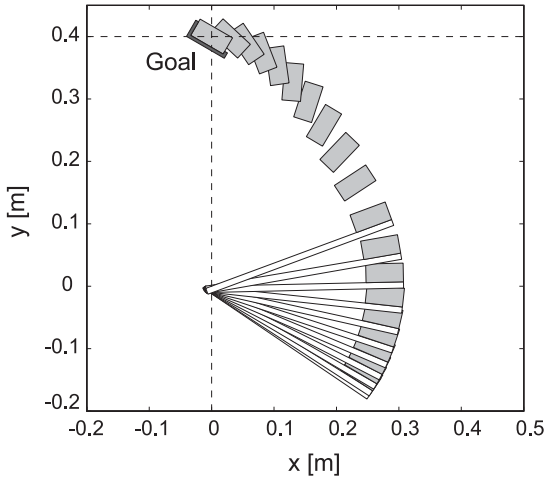


Fig. 5 Throwing manipulation found by the optimization.

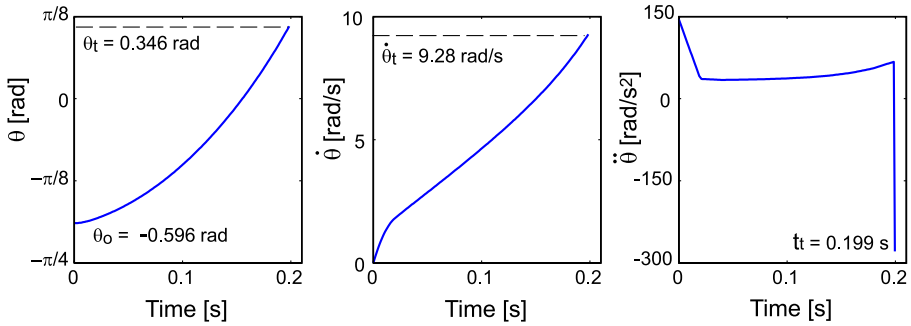


Fig. 6 Arm trajectories found by the optimization.

shows the margin ΔM_A often becomes near zero, Fig. 7 (a) shows that wide tolerable regions are obtained by the proposed optimization method, which serves a robust and stable throwing.

Figure 8 shows the accessible space $(x,\ y,\ \phi)$ with the velocity $\dot{y} = 0$, which is found by solving the optimization Eqs. (28) ~ (32) at discrete points with 2 cm and 5 deg intervals in a finite space. The accessible space is considerably narrower than expected due to the dynamic constraints and the actuator's performance limits. It would be important to set an object's goal state appropriately in the throwing manipulation.

5. Iterative 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.

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.

5.1. Introduction of Virtual Goal

To find an optimal arm trajectory, instead of setting an object's goal state z_d for the optimization programming problem, we introduce a virtual goal state $\widehat{z}_d$, which is updated at each throwing trial in order that the robotic arm can throw the object to the goal state z_d as

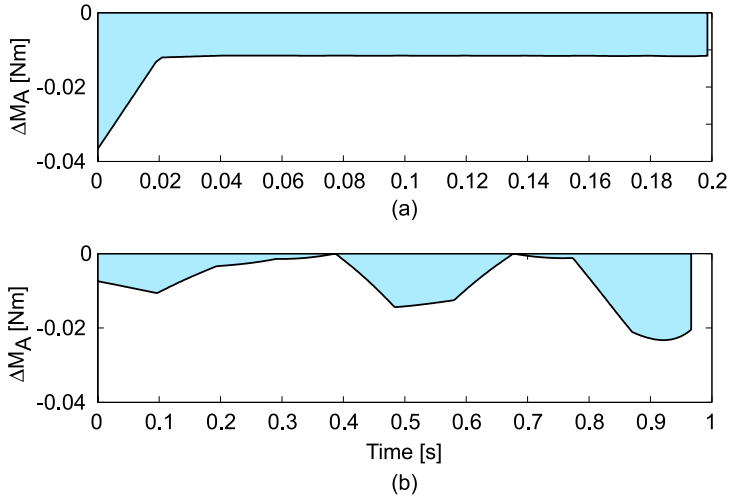


Fig. 7 Tolerable region of margin ΔM_A. (a) With consideration of expanding the margin. (b) Without consideration of those.

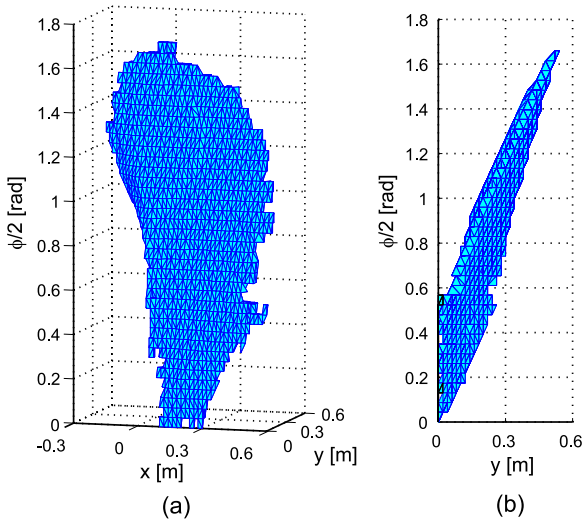


Fig. 8 Object's accessible space (x, y, ϕ) with velocity $\dot{y} = 0$

closely as possible. The basic idea of the learning control using the virtual goal is shown in Ref. (1).

The virtual goal is obtained as follows. The error between the goal state z_d and the actual state z is given by

$$\boldsymbol{e}^j = z_d - z^j \tag{33}$$

where j denotes the trial number.

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

$$\widehat{z}_d^{j+1} = \widehat{z}_d^{j} + \boldsymbol{K}\boldsymbol{e}^i \tag{34}$$

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

Replacing the goal state z_d in the left-hand side of Eq. (29) with the virtual goal state $\widehat{z}_d$, we find an optimal arm trajectory by iteratively solving the optimization programming problem Eqs. (28)~(32) at each throwing trial.

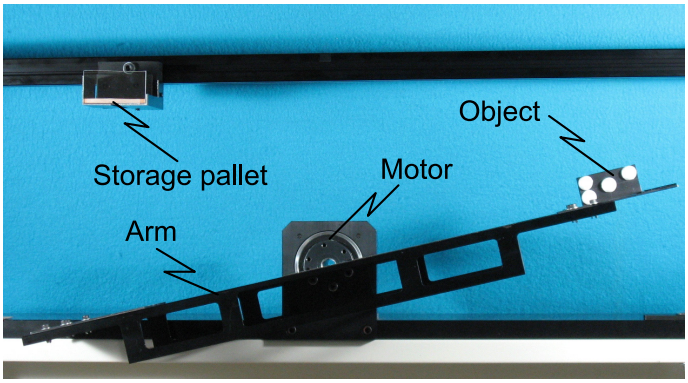


Fig. 9 Throwing robot system

5.2. 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 estimate 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 Eq. (33). 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 $\widehat{\boldsymbol{z}}_d^{j+1}$ is updated by using Eq. (34).

(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.

6. Experiment

6.1. Throwing Robot System

We have developed the throwing robot system which can perform the throwing in the vertical plane as shown in Fig.9. 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 (Harmonic Drive Systems Inc., FHA-14C-50) with a harmonic drive gear, which has the maximum torque $\tau_{\max} = 18\,\mathrm{Nm}$ and the maximum speed $\dot{\theta}_{\max} = 4\pi\,\mathrm{rad/s}$. The angle of the arm is measured by the encoder integrated in the motor. 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 × 3 cm × 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 four circular white markers attached to the object during the flight with a camera (Hamamatsu Photonics K.K., C8201) at a rate of 1 kHz to obtain the position and orientation of the object. The object's ballistic trajectory is estimated through the least square approximations by using the measured data. The final object state is derived from the estimated object trajectory (see Appendix). The concave storage pallet of wood is fixed at a goal in the vertical plane.

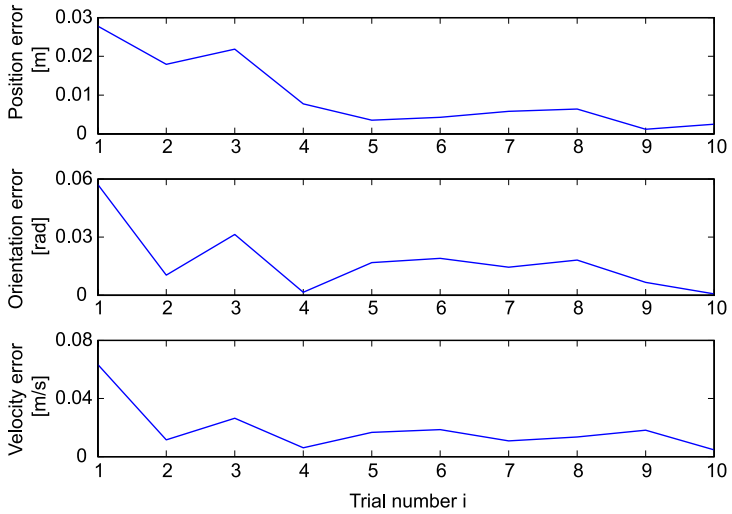


Fig. 10 Transitions of error norm of object's state at the goal

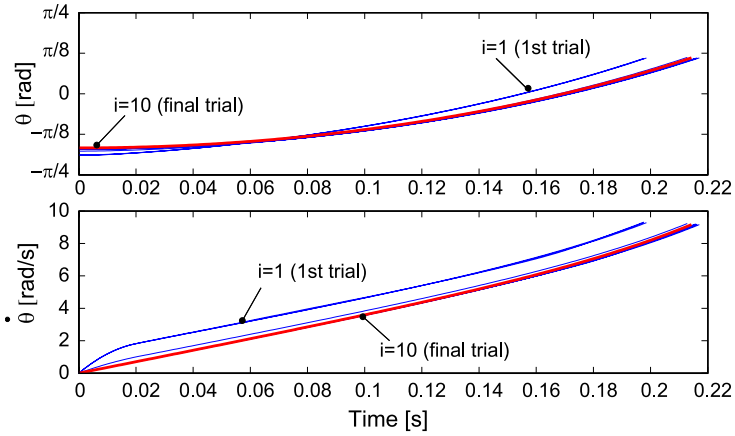


Fig. 11 Arm trajectories found by the optimization at each trial

6.2. Experimental Results

The goal state z_d is set to (0 m, 0.4 m, 150 deg, 0 m/s), which is the same as used in the simulations of § 4.5. Figure 10 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. Figure 11 shows the transitions of arm trajectories found by the optimization at each iteration. The trajectories are modified by repeating the learning. Figure 12 shows snapshots of the successful throwing motion into the storage pallet. The thrown object landed without rebounding from the storage pallet.

The trials of the throwing are almost successful with the arm trajectories which are finally found in the learning control. The occasional failures are emerged when the out-of-vertical-plane rotation of the object occurs during the free-flight, probably due to the initial placement error of the object on the arm surface.

As mentioned in § 4.5, maintaining the un-rolling constraints is crucial to throw the object to the specified goal. Since the arm trajectories are found so that the margins for constraint conditions could be sufficiently large at each iteration, the object did not start rolling on the arm surface in this experiment.

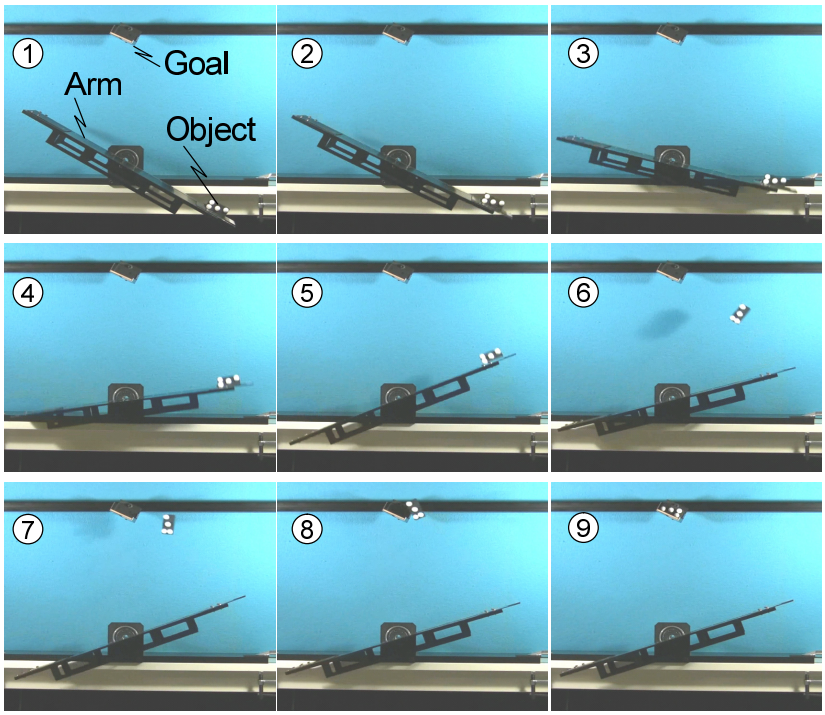


Fig. 12 Snapshots of the throwing of the rectangular object into the pallet

6.3. Application to Sorting/Assembly Tasks

Figure 13 shows the sorting task in which the robotic arm throws three objects into storage pallets with an appropriate orientation. In the first throwing (Fig. 13(a)), the zero vertical velocity is set at the first goal. On the other hand, in the second throwing (Fig. 13(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 throwing (Fig. 13(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 appropriately. If the negative vertical velocity with a larger (smaller) magnitude should be set, the object will collide with the upper (lower) pallet. Specifying appropriate desired vertical velocity at the goal helps to find the object path which avoids the obstacles. Figure 14 shows corresponding arm trajectories for the third throwing found by the optimization. Figure 15 shows the snapshots of the successful sorting task by three successive throwing manipulation.

Figure 16 shows the parts assembly task with H- and T-shaped parts. First, the robotic arm throws the H-shaped part and slots it onto the T-shaped part fixed upside down (Fig. 16(a)). Next, the robotic arm throws the other T-shaped part and inserts it into the former H-shaped part (Fig. 16(b)). In order not only to prevent the flying part from colliding with the previously-inserted parts from the side but also to make it possible to insert the part into the other part smoothly from above, the desired vertical velocity at the goal is set so that the translational velocity at the goal could point toward the $-y$ direction as much as possible. Figure 17 shows corresponding arm trajectories for the second throwing found by the optimization. Figure 18 shows the snapshots of the successful assembly task by two successive throwing manipulation.

These experimental results show not only that the throwing manipulation controlled by the proposed learning algorithm can achieve the high positioning performance but also that the throwing manipulation can be applied to the assembly and sorting of parts.

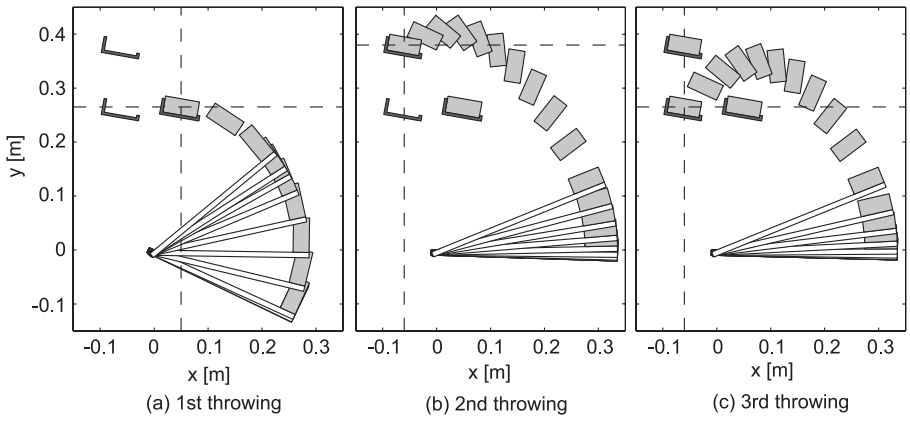


Fig. 13 Sorting task: Throwing manipulation found by the optimization

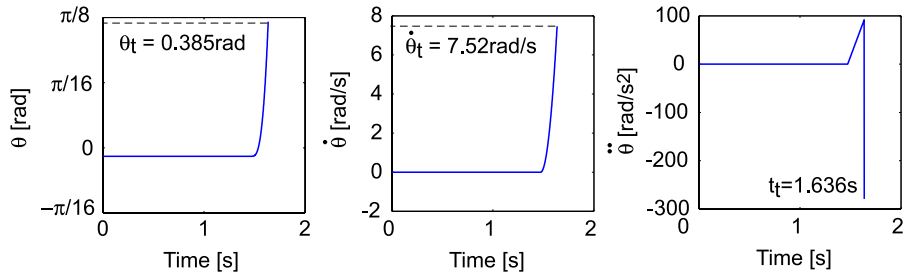


Fig. 14 Sorting task: Arm trajectories for the third throwing found by the optimization

7. Conclusion

This paper presented the optimal motion planning to solve for the arm trajectory and the learning control method for the throwing manipulation which can control not only the position but also the orientation of the polygonal object robustly to uncertainties. We showed experimentally the validity of the proposed control method with the one-degree-of-freedom robotic arm. We also 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).

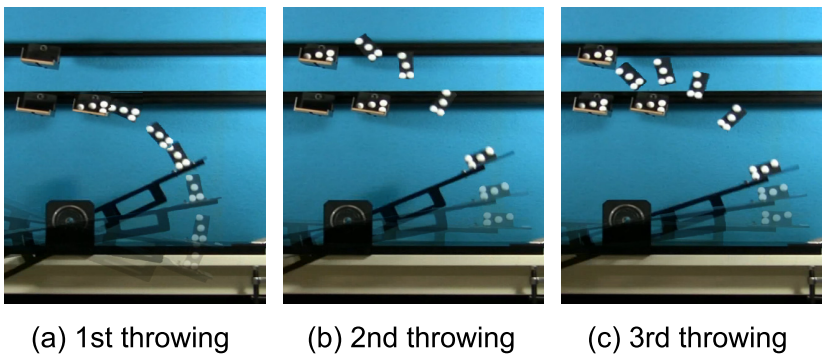


Fig. 15 Sorting task: Implementation

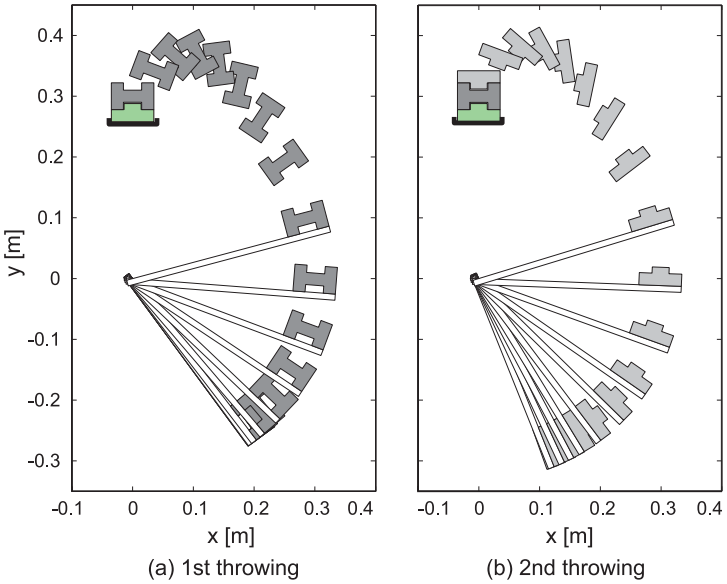


Fig. 16 Parts assembly task: Throwing manipulation found by the optimization

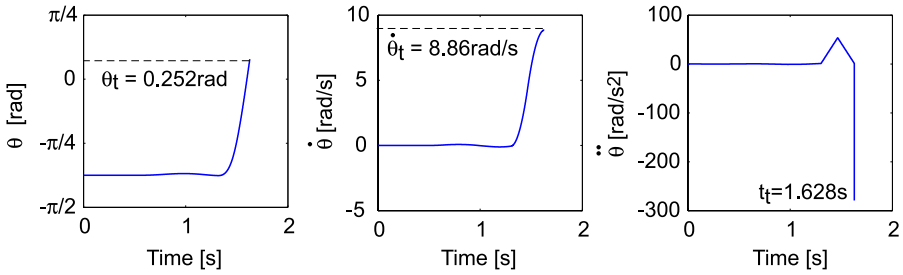


Fig. 17 Parts assembly task: Arm trajectories for the second throwing found by the optimization

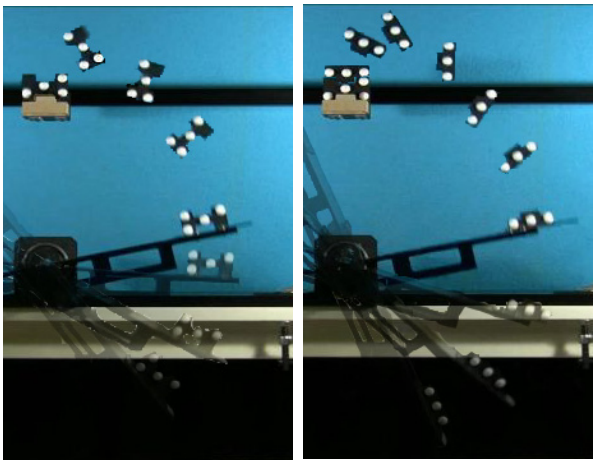
Appendix: Derivation of Final Object State

The actual final object state z is derived from the object ballistic trajectory equation $z_c(t)$, which is estimated by using several measured data on the object's positions and orientations. The main purpose of the throwing manipulation is to transfer the object from the initial state to the goal state as close as possible. In the present paper, the final object state z is defined as a state on the estimated object trajectory $z_c(t)$, which is the nearest to the goal state z_d. Since the final object state z is equivalent to a state $z_c(t_f)$ at the final time t_f, which is described as Eq. (11), the vector $z_d - z_c(t_f)$ must be perpendicular to the tangent vector $\dot{z}_c(t_f)$. This gives the following equation.

$$(\bar{z}_d - \bar{z}_c(t_f))^T \dot{\bar{z}}_c(t_f) = 0 \tag{35}$$

where $\bar{z}_d = \boldsymbol{T} z_d$ and $\bar{z}_c(t_f) = \boldsymbol{T} z_c(t_f)$. $\boldsymbol{T}$ is a diagonal matrix which scales the object state including multiple physical units.

Solving the third degree polynomial Eq. (35) for the time t_f and finding a real solution t_f which minimizes $\left\|\bar{z}_d - \bar{z}_c(t_f)\right\|$ among multiple solutions yields the final object state as $z = z_c(t_f)$.



(a) 1st throwing (b) 2nd throwing

Fig. 18 Parts assembly task: Implementation

References

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

(2) T. Arai, J. Ota and Y. Aiyama, Assembly by Nongrasping Manipulation, *CIRP Annals - Manufacturing Technology*, Vol.46, No.1 (1997), pp.15-18.

(3) 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.

(4) 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.

(5) Y.-C. Chen, Solving Robot Trajectory Planning Problems with Uniform Cubic B-Splines, *Optimal Control Applications and Methods*, Vol.12 (1991), pp.247-262.

(6) 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.

(7) 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.

(8) 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.

(9) 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.

(10) H. Miyashita, T. Yamawaki and M. Yashima, Learning Control Method for Throwing an Object More Accurately with One Degree of Freedom Robot, *Proc. of IEEE/ASME Int. Conf. on Advanced Intelligent Mechatronics*, (2010), pp.397-402.

(11) W. Mori, J. Ueda and T. Ogasawara, A 1-d.o.f Dynamic Pitching Robot that Independently Controls Velocity, Angular Velocity, and Direction of a Ball, *Advanced Robotics*, Vol.24, No.5-6 (2010), pp.921-942.

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

(13) 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.

(14) 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.

(15) 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.

(16) T. Tabata, and Y. Aiyama, Tossing Manipulation by 1 Degree of Freedom Manipulator, *J. of the Robotics Society of Japan*, Vol.20, No.8 (2002), pp.876-882.

(17) M. Yashima, H. Miyashita and T. Yamawaki, Throwing Manipulation by One Joint Robot, *Trans. of the Japan Society of Mechanical Engineers, Series C*, Vol.75, No.759 (2009-11), pp.3005-3010.