robotics

MDPI

Article

Pose-Constrained Path Optimization for Manipulators with Contact Feedback from Sparse Waypoints

Yuta Kasahara [1,*], Yuichi Kobayashi [1,*], Tomohiro Hayakawa [1], Francisco Jesús Arjonilla García [1], Kazuki Yamamoto [1], Keiji Takeshita [2], Yuya Wada [2] and Yoichiro Nakamura [2]

[1] Department of Mechanical Engineering, Faculty of Engineering, Shizuoka University, 3-5-1 Johoku, Chuo-ku, Hamamatsu 432-8561, Japan
[2] Shibaura Machine Co., Ltd., 4-29-1 Hibarigaoka, Zama 252-0003, Japan
* Correspondence: kasahara.yuta.19@shizuoka.ac.jp (Y.K.); kobayashi.yuichi@shizuoka.ac.jp (Y.K.)

Abstract

Conventional robot motion teaching methods are time-consuming and place a significant burden on operators, especially when trajectory modification is required after contact with the environment. This paper proposes a six-dimensional automatic path correction method that considers both the position and orientation of the end-effector. The operator specifies only the start, goal, and via points of the motion path. Contact wrench information obtained during execution is incorporated into a cost function, and the control points of a spline-based trajectory are iteratively optimized. The method does not rely on an explicit environment model and autonomously updates the trajectory based on contact information. Simulation experiments were conducted on three industrial tasks: part box unloading, yarn cone removal, and part extraction. In all tasks, the proposed method successfully generated collision-free trajectories within a small number of correction iterations and with practical computation time. These results indicate that the proposed approach enables stable and efficient trajectory correction while reducing the teaching burden in industrial support tasks.

Keywords: manipulator; path generation; obstacle avoidance; optimization

1. Introduction

Manufacturing environments involve not only core operations like machining and assembly but also auxiliary tasks such as the transportation and placement of parts, materials, and finished products. Because these auxiliary tasks involve a wide variety of objects and environments, they are more difficult to automate with robots than core operations; thus, many production sites still rely heavily on human labor. Against the backdrop of recent labor shortages, the demand for automating such auxiliary tasks has been increasing. This study focuses on the automation of motion path generation, with the aim of reducing the task teaching time for robots, thus alleviating the burden on instructors when introducing production-support robots responsible for auxiliary tasks.

When introducing robots into production environments, motion teaching is indispensable. Existing approaches can be broadly classified into online teaching and offline teaching. A comprehensive review by Pan et al. [1] summarizes the characteristics and limitations of conventional teach–playback methods, CAD- or simulator-based offline approaches, and emerging sensor-based adaptive techniques. This body of work provides a systematic overview of how motion teaching has evolved alongside advances in sensing and modeling technologies.

Received: 16 March 2026
Revised: 14 May 2026
Accepted: 18 May 2026
Published: 29 May 2026

In online teaching, operators directly manipulate the robot using a teach pendant or similar device, and the recorded waypoints are replayed during execution. Because motions are determined while observing the real environment, this approach can achieve high positional accuracy. However, online teaching requires the interruption of production lines and depends heavily on skilled operators. Moreover, its flexibility is limited when task conditions frequently change—for example, in production environments where many different products are handled in small batches.

Offline teaching, in contrast, allows robot motions to be designed in advance using CAD models or virtual environments. Because programming can be performed without using the actual robot, the need to stop production lines during teaching can be reduced. Nevertheless, discrepancies between the virtual model and the real environment—such as calibration errors and positioning inaccuracies—must be compensated for during execution. As a result, constructing and maintaining highly accurate environment models often requires significant effort and expert knowledge.

Against this background, the importance of incorporating sensor information to enable adaptive motion generation has been increasingly recognized. Physical interaction with the environment provides direct and reliable information about environmental constraints, without relying on visual/laser sensing or precise geometric modeling. Contact force sensing offers a key advantage over visual and laser-based approaches in that it is not affected by occlusion or field-of-view limitations, enabling the robot to directly perceive obstacles that must be avoided. However, this approach is less suitable for environments that include fragile or easily movable objects. In addition, it may necessitate compliant control of the robot, which can introduce further challenges. In this paper, we propose an approach based on contact force sensing, with a particular focus on leveraging its aforementioned advantages.

In this study, we propose a method that combines minimal human instruction with autonomous motion generation. Specifically, only the start pose, end pose, and a minimal number of intermediate waypoints are provided by the operator. The robot then generates its trajectory autonomously and incrementally corrects it based on environmental information acquired during execution.

2. Related Works

In the field of robot motion generation, approaches based on Vision–Language–Action (VLA) models have attracted considerable attention, aiming to learn general-purpose policies from large-scale datasets [2]. RT-1 [3] and RT-2 [4] demonstrated end-to-end control by leveraging vision–language pretraining, while SayCan [5] and PaLM-E [6] integrated large language models into action generation frameworks. Furthermore, the application of VLA-based methods to mobile manipulation has also been reported [7].

Although these methods exhibit strong generalization capabilities, they rely on large-scale data and pretrained models, and they do not explicitly consider frameworks for incrementally modifying trajectory shapes in response to unexpected contacts that occur during execution. In contrast to such large-scale pretraining-based approaches, this study focuses on reliable execution of specific tasks with minimal teaching effort and without relying on large pretrained models, rather than pursuing broad generalization from the outset. Therefore, this study first focuses on establishing a robust path generation method that reliably executes a given task. After confirming its effectiveness, the ability to generalize the method to different tasks and environments is considered in subsequent studies.

Reinforcement learning (RL) has also been widely adopted as a framework for acquiring control policies through trial and error based on reward functions [8]. For example, positive rewards may be assigned to goal achievement and negative rewards to collisions

or failures, enabling robots to learn appropriate behaviors [9]. Deep RL approaches that directly learn policies from high-dimensional sensory inputs such as visual observations have also been reported [10]. Furthermore, efforts to improve learning efficiency through sim-to-real transfer and integration with imitation learning have been investigated [11,12]. However, achieving satisfactory performance generally requires extensive interaction data, and such large-scale learning may not be efficient for task-specific path generation or online correction.

In the context of path planning, sampling-based approaches such as RRT [13] and its optimal variant RRT* [14], along with artificial potential field methods [15], have been widely adopted. However, these methods typically assume that an accurate model of the environment and obstacles is available in advance. Therefore, they are not well suited to situations where the path must be corrected during execution based on contact information obtained from the robot.

Optimization-based approaches for refining an initial trajectory—such as CHOMP [16], STOMP [17], TrajOpt [18], and GPMP2 [19]—have demonstrated effectiveness in improving trajectory smoothness and collision avoidance. Avoidance of contact via task scheduling has been also discussed [20]. However, in these methods, the distance to obstacles is essential for generating effective paths, typically requiring a complete environment model. In contrast, our method leverages local contact feedback via a repulsive cost field instead of distance information to determine appropriate directions for trajectory modification.

Another line of research focuses on generating motion trajectories based on demonstrated paths obtained through teaching, rather than optimizing trajectories purely from scratch. For example, Laha et al. [21] proposed a user-guided constrained path planning method that utilizes trajectories obtained through online teaching. Based on a successfully demonstrated trajectory and specified initial and goal positions, their framework automatically generates motion trajectories for tasks exhibiting similar motion characteristics. In this approach, obstacle information is incorporated using environmental data acquired from visual sensors. However, perception based on visual sensing inherently depends on sensor placement and field of view, and environmental information may become incomplete due to occlusions.

We propose a six-dimensional automatic path correction method that explicitly accounts for end-effector orientation and does not require an explicit environment model or large-scale prior training. By incorporating contact information obtained during execution into the cost function, the trajectory is iteratively corrected and progressively refined. Unlike studies on path generation using contact force (e.g., [22,23]), our method targets robot task execution in production scenarios and therefore emphasizes simple geometric trajectory adaptation rather than detailed dynamic modeling of interactions.

The comparison with other approaches discussed above is summarized as follows:

- Large-model- and RL-based approaches can be highly general and powerful; however, they are not always suitable for rapid adaptation to case-specific practical applications. In contrast, our approach is lightweight and can be adapted quickly.
- Conventional motion planning approaches require explicit object and environment models. Our approach eliminates the need for such model preparation.
- Vision-based approaches are susceptible to occlusions and limitations in the sensor's field of view. In contrast, our approach relies solely on contact information and is therefore unaffected by visual occlusions.
- Existing contact-based motion generation methods have not been specifically designed for auxiliary motion generation tasks in manufacturing environments.

The main contributions of this study are summarized as follows:

- A six-dimensional trajectory correction framework considering both position and orientation;

- A model-free formulation without explicit geometric environment models;
- An incremental correction strategy based on contact information during execution.

The effectiveness of the proposed method is verified through simulation.

3. Problem Definition

In this study, we address a motion path generation problem for the end-effector of a six-degree-of-freedom manipulator in three-dimensional space, considering both position and orientation (six degrees of freedom), while holding an object with a gripper, as shown in Figure 1.

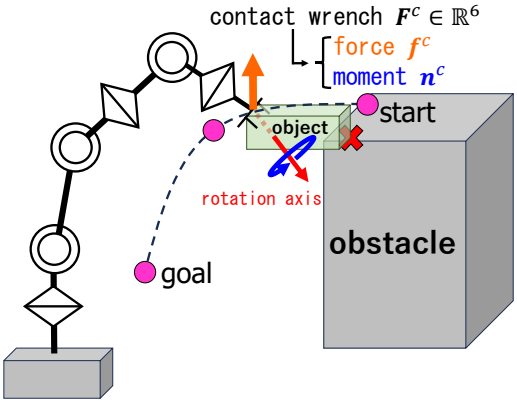


Figure 1. Contact information observed at collision between object and obstacle.

Regarding the robot and its environment, we make the following assumptions:

- The robot does not have an environmental model (shapes of objects and obstacles).
- The robot can detect collision between the object and an obstacle by its force/torque sensor.

Contact is detected as wrench $\boldsymbol{F}^C \in \mathbb{R}^6$ at the end-effector, which consists of force and moment indicated by orange and blue arrows, respectively, and is supposed to be estimated from either a force–torque sensor on its wrist or torque sensors at the six joints. The force component of the contact wrench is used to judge the detection of force, where a certain threshold is applied to the norm of the force for the judgment.

The objective is to automatically modify an initial path that is determined by a human operator and generate a collision-free path such that the object held by the robot does not come into contact with obstacles. To determine initial path, the operator once sets a start pose, a goal pose, and intermediate waypoints (not mandatory). A continuous path curve is then obtained by spline interpolation of these points. During path correction, no prior information—such as an environmental model—is used. Instead, the method relies solely on the end-effector pose at the moment of contact and the contact wrenches obtained from the sensors.

4. Contact-Based Path Generation

4.1. Representation of Path Curves

The path curve is represented as a six-dimensional curve consisting of the end-effector's position and orientation. The representation of the position and orientation is shown in Figure 2. The start pose, goal pose, and via poses specified by the instructor are referred to as control points. Each control point is defined by the end-effector position $\boldsymbol{p} = (x, y, z)$ and the end-effector orientation $\boldsymbol{r} = (\phi v_x, \phi v_y, \phi v_z)$, and expressed as the following vector, where, ϕ denotes the rotation angle from the initial orientation, and $\boldsymbol{v} = (v_x, v_y, v_z)$ represents the rotation axis:

$$\boldsymbol{x} = \begin{bmatrix} x & y & z & \phi v_x & \phi v_y & \phi v_z \end{bmatrix}^T \tag{1}$$

The shape of the path curve is defined by a set of N control points $\boldsymbol{x} \in \mathbb{R}^6$, denoted as $\boldsymbol{X} = \{\boldsymbol{x}_i \mid i = 1, \cdots, N\}$ (control point sequence). The path curve is generated by applying spline interpolation to the control point sequence $\boldsymbol{X}$.

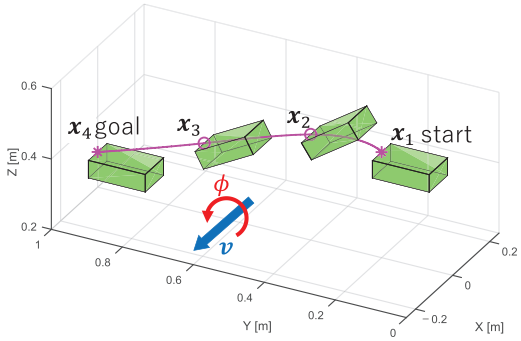


Figure 2. Representation of position and orientation.

4.2. Contact Information

An overview of the contact information is illustrated in Figure 1. The contact pose of the manipulator end-effector at the i-th contact is denoted by $\boldsymbol{x}_i^c \in \mathbb{R}^6$ and is defined by the contact position $\boldsymbol{p}_i^c \in \mathbb{R}^3$ and the contact orientation $\boldsymbol{r}_i^c \in \mathbb{R}^3$ as follows:

$$\boldsymbol{x}_i^c = \begin{bmatrix} \boldsymbol{p}_i^c & \boldsymbol{r}_i^c \end{bmatrix}^T. \tag{2}$$

The contact wrench at the i-th contact is denoted by $\boldsymbol{F}_i^c \in \mathbb{R}^6$ and is defined by the force $\boldsymbol{f}_i^c \in \mathbb{R}^3$ and the moment $\boldsymbol{n}_i^c \in \mathbb{R}^3$ acting on the end-effector, as follows:

$$\boldsymbol{F}_i^c = \begin{bmatrix} \boldsymbol{f}_i^c & \boldsymbol{n}_i^c \end{bmatrix}^T. \tag{3}$$

The contact information used for path correction is defined as

$$\boldsymbol{c}_i = (\boldsymbol{x}_i^c, \boldsymbol{F}_i^c). \tag{4}$$

4.3. Procedure for Automatic Path Correction

An overview of the overall procedure for path generation is illustrated in Figure 3. In the proposed method, a collision-free motion path is generated by iteratively repeating path following, acquisition of contact information, and path correction. First, the operator specifies an initial sequence of control points $\boldsymbol{X}^{\text{init}}$ for initial path generation. This sequence is set as the control point sequence for path following, denoted by $\boldsymbol{X}^{\text{follow}}$, and the manipulator follows the motion path generated from these control points. During robot motion, contact with obstacles is monitored. When a contact is detected, the contact information at that moment, denoted by $\boldsymbol{c}_i$, is acquired and recorded in the contact information history $\boldsymbol{C}$. The contact information history is defined as

$$\boldsymbol{C} = \{\boldsymbol{c}_i \mid i = 1, \cdots, N_c\}, \tag{5}$$

where N_c denotes the number of contact events. Next, an objective function L that reflects the contact information history $\boldsymbol{C}$ is defined, and the control point sequence $\boldsymbol{X}$ is updated by solving an optimization problem that minimizes this objective function, thereby correcting the path:

$$\boldsymbol{X}^* = \arg\min_{\boldsymbol{X}} L(\boldsymbol{X}, \boldsymbol{C}). \tag{6}$$

The optimized control point sequence $\boldsymbol{X}^*$ is then set as a new control point sequence for path following, $\boldsymbol{X}^{\text{follow}}$. After returning the robot to the start point of the path, the same procedure of path following is performed again. By repeating this process, the motion path is gradually corrected based on the accumulated contact information until a collision-free path is obtained. On the other hand, if no contact is detected during path following, a goal-reaching condition is evaluated, and the path generation process is terminated when the robot reaches the target position.

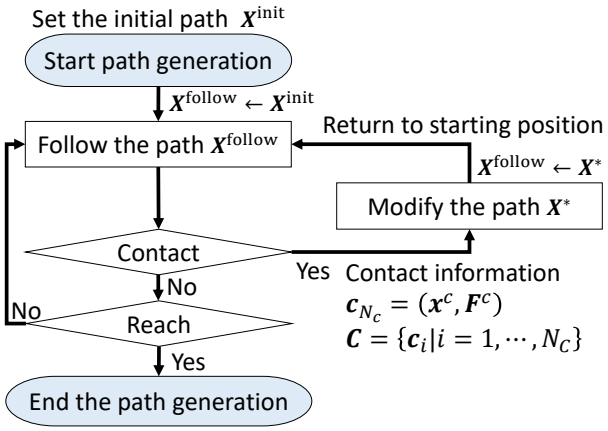


Figure 3. Path generation process.

4.4. Path Correction via Optimization

In this study, the path is corrected by updating the control point sequence $\boldsymbol{X}$, which represents the shape of the path, through minimization of an objective function L defined on $\boldsymbol{X}$. By incorporating the contact information history $\boldsymbol{C}$ into the objective function L, a control point sequence is generated so as to avoid the position and orientation at which contact has previously occurred. The Nelder–Mead method is employed as the optimization algorithm to solve the resulting optimization problem [24]. The reason for this is that the cost function includes orientation-dependent terms that make gradient computation difficult, and the method is known to be robust in low-dimensional settings. The objective function used for path correction is composed of four cost functions G_i $(i = 1, 2, 3, 4)$, defined with respect to the following criteria:

1. Collision avoidance based on contact information.
2. Deviation between the via points of the corrected path and those of the initial path to keep the corrected path close to the taught path.
3. Total length of the generated path.
4. Uniformity of the interval between adjacent control points in the pose space.

The cost function G_1 reflects the contact information history $\boldsymbol{C}$ and penalizes re-contact at positions and orientations where contact has previously occurred.The cost functions G_i $(i = 2, 3, 4)$ are designed to correct and regularize the overall path shape. Each cost function is weighted by a coefficient α_i $(i = 1, 2, 3, 4)$, and the objective function L is defined as the weighted sum of these cost functions:

$$L(\boldsymbol{X}, \boldsymbol{C}) = \alpha_1 G_1(\boldsymbol{X}, \boldsymbol{C}) + \sum_{i=2}^{4} \alpha_i G_i(\boldsymbol{X}) \tag{7}$$

In the following subsections, the detailed definitions of each cost function are described.

4.4.1. Cost Function Incorporating Contact Information

The cost function G_1 is designed to generate a path that avoids contact at the position and orientation where contact has occurred previously. Specifically, it is formulated such that the cost becomes high in regions that are likely to contain obstacles that caused the contact. To evaluate this cost along the path, a set of poses is sampled along the path at regular intervals, and these sampled poses are referred to as evaluation poses. The cost G_1 is then defined as the sum of the costs evaluated at these evaluation poses. As a result, the value of G_1 increases for paths that enter high-cost regions. Consequently, a path that does not enter such regions is generated, enabling avoidance of re-contact at previously contacted poses. Based on this concept, we define the cost function G_1 as follows:

Let the sequence of evaluation poses along the path be $\boldsymbol{X}^q = \left\{ \boldsymbol{x}_k^q \mid k = 1, \cdots, N_q \right\}$, where $\boldsymbol{x}_k^q \in \mathbb{R}^6$ denotes the k-th evaluation pose and N_q is the number of evaluation poses. Each evaluation pose represents the position and orientation of the end-effector and is written as $\boldsymbol{x}_k^q = (\boldsymbol{p}_k^q, \boldsymbol{r}_k^q)$, where $\boldsymbol{p}_k^q \in \mathbb{R}^3$ and $\boldsymbol{r}_k^q \in \mathbb{R}^3$ denote the position and orientation components, respectively. The path is obtained by spline interpolation of the control point sequence $\boldsymbol{X}$. Therefore, each evaluation pose $\boldsymbol{x}_k^q$ can be computed from $\boldsymbol{X}$. Since G_1 is defined as the sum of the costs G_1^q at the evaluation poses, it can be written as

$$G_1(\boldsymbol{X}, \boldsymbol{C}) = \sum_{k=1}^{N_q} G_1^q(\boldsymbol{x}_k^q(\boldsymbol{X}), \boldsymbol{C}). \tag{8}$$

The point-wise cost is defined using the contact information $\boldsymbol{c}$. Because multiple contact events may be available after the second and subsequent corrections, a cost g_1 is defined for each contact information instance $\boldsymbol{c}$. For an evaluation pose $\boldsymbol{x}^q$, we compute $g_1(\boldsymbol{x}^q, \boldsymbol{c})$ for all contact information instances, and the maximum value is used as the cost at $\boldsymbol{x}^q$, denoted by $G_1^q(\boldsymbol{x}^q, \boldsymbol{C})$:

$$G_1^q(\boldsymbol{x}^q, \boldsymbol{C}) = \max_{i=1,\cdots,N_c} g_1(\boldsymbol{x}^q, \boldsymbol{c}_i). \tag{9}$$

The cost function g_1 is designed to avoid contact with obstacles by encouraging corrections of either the position or the orientation of the end-effector. Based on the contact information, g_1 is defined as the sum of a cost g_1^p that promotes position correction and a cost g_1^r that promotes orientation correction, as follows, where, w_1^p and w_1^o are weighting parameters for the position- and orientation-related contact information, respectively:

$$g_1(\boldsymbol{x}^q, \boldsymbol{c}) = w_1^p g_1^p(\boldsymbol{p}^q, \boldsymbol{c}) + w_1^o g_1^r(\boldsymbol{x}^q, \boldsymbol{c}) \tag{10}$$

In the following text, we describe the definitions of the cost terms for position correction and orientation correction, respectively.

Cost for Position Correction

First, we describe the position-related cost g_1^p of the cost function g_1. An example of the position-related cost g_1^p is illustrated in Figure 4. The cost g_1^p is designed to avoid regions where obstacles are predicted to exist by correcting the end-effector position. Since an obstacle is expected to exist in the direction opposite to the force $\boldsymbol{f}^c$ acting on the end-effector at the time of contact, the cost is constructed such that it decreases along the direction of the force by employing a bump function centered at the contact position $\boldsymbol{p}^c$. The shape of the bump function is illustrated in Figure 5. Meanwhile, in the directions perpendicular to the force, the cost g_1^p is defined using a Gaussian function, so that it smoothly decreases as the distance from the contact position increases.

$$g_1^p = \prod_{i=1}^{3} g_{1,i}^p, \qquad 0 \le g_1^p \le 1 \tag{11}$$

$$g_{1,1}^p(\boldsymbol{p}^q, \boldsymbol{c}) = \exp\left(1 - \frac{1}{1 - \left(\frac{(z_1(\boldsymbol{p}^q, \boldsymbol{c}) - \mu_p) - \gamma_p}{\beta_p}\right)^2}\right) \tag{12}$$

$$g_{1,i}^p(\boldsymbol{p}^q, \boldsymbol{c}) = \exp\left(-\frac{z_i^2(\boldsymbol{p}^q \mid \boldsymbol{c})}{2\sigma_p^2}\right), \qquad i = 2,3 \tag{13}$$

The parameters β_p, γ_p, and σ_p are predefined constants, where β_p controls the width of the bump function, γ_p determines the location of its peak, and σ_p governs the smoothness of the transition.

Here, $z_i(\boldsymbol{p}^q, \boldsymbol{c})$ $(i = 1,2,3)$ are defined as follows: Let $\boldsymbol{m}_1 = \boldsymbol{f}^c / \|\boldsymbol{f}^c\|$ denote the unit vector in the direction of the force. Let $\boldsymbol{m}_2$ and $\boldsymbol{m}_3$ be orthonormal vectors spanning the plane perpendicular to $\boldsymbol{m}_1$. Then,

$$z_1(\boldsymbol{p}^q, \boldsymbol{c}) = \boldsymbol{m}_1 \cdot (\boldsymbol{p}^q - \boldsymbol{p}^c), \qquad (\boldsymbol{p}^c, \boldsymbol{f}^c) \in \boldsymbol{c} \tag{14}$$

$$z_i(\boldsymbol{p}^q, \boldsymbol{c}) = \boldsymbol{m}_i \cdot (\boldsymbol{p}^q - \boldsymbol{p}^c), \qquad (\boldsymbol{p}^c, \boldsymbol{f}^c) \in \boldsymbol{c},\ i = 2,3 \tag{15}$$

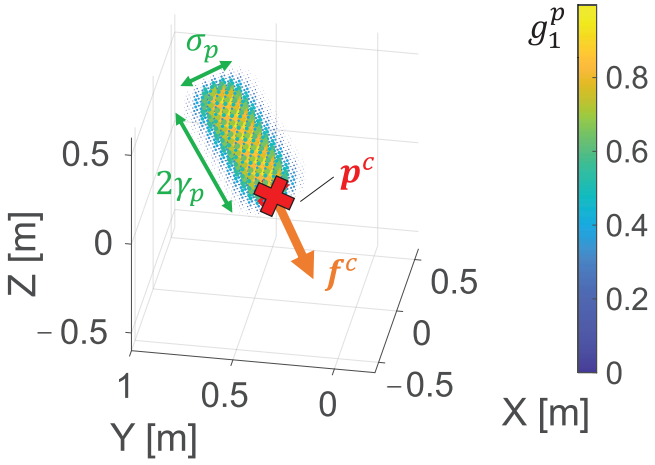


Figure 4. Cost of position.

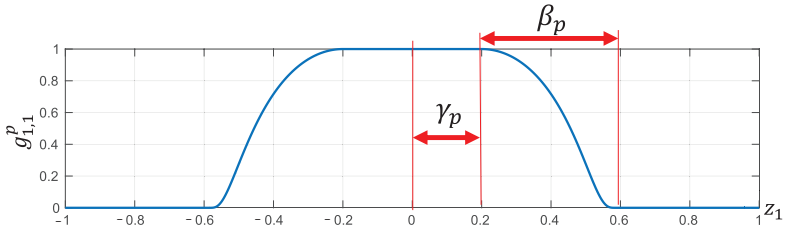


Figure 5. Bump function.

Cost for Orientation Correction

We describe the orientation-related cost g_1^r of the cost function g_1. The cost g_1^r is designed such that, when the end-effector contacts an obstacle, the cost takes a large value if the orientation is rotated toward the direction where the obstacle exists, with the contact orientation as a reference in the vicinity of the contact position. To this end, the following three cost terms are designed and used for orientation correction, where the rotation axis

and rotation angle of the orientation change from the contact orientation $\boldsymbol{r}^c$ to the evaluated orientation $\boldsymbol{r}^q$ are denoted by $\boldsymbol{v}^q$ and ψ, respectively:

1. h^v: A cost that decreases as the rotation axis of the orientation change approaches the axis of the moment.
2. h^ψ: A cost that decreases as the magnitude of the orientation change increases.
3. h^p: A cost that takes a large value in the vicinity of the contact position.

First, we define the rotation-axis-related cost h^v. In order to rotate the orientation in a desirable direction to avoid contact with the obstacle from the contact orientation, a preferred rotation axis is required. The cost h^v is designed to encourage rotation about this preferred axis. Here, the preferred rotation axis $\boldsymbol{v}^c$ is defined using the moment $\boldsymbol{n}^c$ as follows:

$$\boldsymbol{v}^c = \frac{\boldsymbol{n}^c}{\|\boldsymbol{n}^c\|}. \tag{16}$$

Next, we define the cost h^ψ for evaluating the magnitude of the orientation change. Let ψ denote the rotation angle between the contact orientation and the evaluation orientation, computed using quaternions. To promote rotation about the axis encouraged by h^v, the cost h^ψ is defined such that it decreases as the rotation angle increases:

$$h^\psi(\boldsymbol{r}^q) = \frac{1}{1+\exp(a_\psi \psi)}, \tag{17}$$

where a_ψ is a gain (constant). Furthermore, to promote orientation changes in the vicinity of the contact position, the cost h^p is defined as follows:

$$h^p(\boldsymbol{p}^q) = \frac{1}{1+\exp\bigl(-a_p(z_o - b_p)\bigr)}, \tag{18}$$

where a_p is a gain (constant) and b_p is a parameter. The scalar variable z_o is defined as follows:

$$z_o = \|\boldsymbol{p}^q - \boldsymbol{p}^c\|. \tag{19}$$

Based on the above definitions, the orientation correction cost g_1^r, which promotes orientation changes in the vicinity of the contact position, is defined as follows:

$$g_1^r(\boldsymbol{x}^q) = h^p(\boldsymbol{p}^q)\bigl(h^v(\boldsymbol{r}^q) + h^\psi(\boldsymbol{r}^q)\bigr). \tag{20}$$

4.4.2. Cost Function for Path Shape Correction

Next, we describe the cost functions for correcting the path shape based on evaluation criteria 2–4. In these cost functions, a weighting matrix $\mathbf{W}$ is applied in distance calculations to adjust the relative scales of position and orientation. Let w_p and w_o denote the weights for position and orientation, respectively. The weighting matrix is defined as follows:

$$\mathbf{W} = \mathrm{diag}(w_p, w_p, w_p, w_o, w_o, w_o). \tag{21}$$

Since the objective of the proposed automatic path correction is to perform local modifications of the path, large deviations from the initial path may hinder path generation. Therefore, the cost function G_2 is defined such that the cost increases as the difference between the initial control point sequence $\boldsymbol{X}^{\mathrm{init}}$ and the generated control point sequence $\boldsymbol{X}$ becomes larger:

$$G_2(\boldsymbol{X}) = \sum_{i=2}^{N-1} (\boldsymbol{x}_i - \boldsymbol{x}_i^{\mathrm{init}})^{\mathrm{T}} \mathbf{W} (\boldsymbol{x}_i - \boldsymbol{x}_i^{\mathrm{init}}). \tag{22}$$

To reduce unnecessary motions of the manipulator, the motion path should be as short as possible. Accordingly, the cost function G_3 is defined such that the cost increases as the path length becomes larger:

$$G_3(\boldsymbol{X}) = \sum_{k=1}^{N_T-1} (\boldsymbol{x}^q_{(k+1)} - \boldsymbol{x}^q_{(k)})^{\mathrm{T}} \mathbf{W} (\boldsymbol{x}^q_{(k+1)} - \boldsymbol{x}^q_{(k)}). \tag{23}$$

When control points are modified through path correction, adjacent control points may become too close to each other or overlap, which can hinder path generation. Therefore, the cost function G_4 is defined such that the cost becomes smaller as the spacing between adjacent control points becomes more uniform:

$$G_4(\boldsymbol{X}) = \sum_{i=1}^{N-1} |d_{i(i+1)} - d_{\mathrm{ave}}|. \tag{24}$$

Here, the distances are defined as follows:

$$d_{ij} = (\boldsymbol{x}_{(j)} - \boldsymbol{x}_{(i)})^{\mathrm{T}} \mathbf{W} (\boldsymbol{x}_{(j)} - \boldsymbol{x}_{(i)}), \tag{25}$$

$$d_{\mathrm{ave}} = \frac{1}{N-1} \sum_{j=1}^{N-1} d_{j(j+1)}. \tag{26}$$

4.5. Automatic Control-Point Insertion

As contact information accumulates, the directions of the forces assigned to nearby contact positions may contradict each other. In such cases, the number of control points may be insufficient to appropriately represent the desired path shape. Therefore, this subsection describes an evaluation criterion for determining whether additional control points are required based on the accumulated contact information, as well as an automatic control-point insertion method based on this criterion. The insertion decision is applied to control points excluding the start and end points, and at most one control point is added in a single path correction process.

For the control-point insertion decision, only the position information of the control points and contact information—namely, the force and the contact position—are used. As illustrated in Figure 6a, focusing on a control point $\boldsymbol{p}_i$, contact positions $\boldsymbol{p}^c$ located within a distance d_r from $\boldsymbol{p}_i$ are extracted, and their set is defined as $\boldsymbol{C}'$. Next, a method for evaluating inconsistencies among the contact information included in $\boldsymbol{C}'$ is described. As illustrated in Figure 6b, the gradient $\boldsymbol{\gamma} \in \mathbb{R}^3$ of the control point $\boldsymbol{p}_i$ is computed for each contact information instance and used for the evaluation. For each contact information instance, the position cost is evaluated on grid points arranged around the control point $\boldsymbol{p}_i$. For the linear approximation of cost in the position space $\mathbb{R}^3$, 27 grids on the surface and center (equivalent to $\boldsymbol{p}_i$) of a cube are specified. The gradient vector $\boldsymbol{\gamma}$ of the position cost at the control point is then obtained by the least-squares method. Using the directional differences between the obtained gradient vectors, the degree of inconsistency among correction directions is evaluated. As an index for this evaluation, the following evaluation function E is defined:

$$E = \frac{1}{2}\left(1 - \frac{\boldsymbol{\gamma}_j \cdot \boldsymbol{\gamma}_k}{\|\boldsymbol{\gamma}_j\| \, \|\boldsymbol{\gamma}_k\|}\right), \qquad 0 \le E \le 1. \tag{27}$$

A larger value of E indicates a greater inconsistency between correction directions derived from different contact information. For each control point, the maximum value E_i is computed, and the maximum value among all control points is defined as follows:

$$E_{\max} = \max_i E_i. \tag{28}$$

The control point that attains $E_{\max}$ is denoted by $\boldsymbol{p}_{i_{\max}}$. If $E_{\max}$ exceeds a predefined threshold E_{th}, it is determined that an additional control point is required, and a new control point is inserted based on $\boldsymbol{p}_{i_{\max}}$. The values $E = 0, 1/2$, and 1 correspond to the cases where the angle between the two vectors is 0 deg, 90 deg, and 180 deg, respectively. In our implementation, E_{th} is set as $\frac{1}{2}$ to regard a 90 deg difference in angle as the condition to add a new control point.

In this study, each control point is treated as a six-dimensional point consisting of position and orientation. As illustrated in Figure 6c, a new control point is inserted at one of the midpoints of the path segments adjacent to the control point $\boldsymbol{p}_{i_{\max}}$ identified in the insertion decision, namely, either the midpoint $\boldsymbol{p}^-$ between $\boldsymbol{p}_{i_{\max}-1}$ and $\boldsymbol{p}_{i_{\max}}$, or the midpoint $\boldsymbol{p}^+$ between $\boldsymbol{p}_{i_{\max}}$ and $\boldsymbol{p}_{i_{\max}+1}$. An overview of the control point position selection method is shown in Figure 6d. Let $\boldsymbol{p}^c_{\text{mid}}$ denote the midpoint of the contact positions that yielded the maximum inconsistency. The insertion position is determined as either the forward or backward midpoint based on the inner product between the vector from the control point $\boldsymbol{p}_{i_{\max}}$ to $\boldsymbol{p}^c_{\text{mid}}$ and the tangent vector $\boldsymbol{u}$ of the path at the control point. The orientation corresponding to the selected position is then assigned by taking average orientations of of the control points (generally, spherical linear interpolation is used for interpolating two orientations based on quaternions, but here we use the Euclidean average as a linear approximation), and the resulting six-dimensional control point is added to the path.

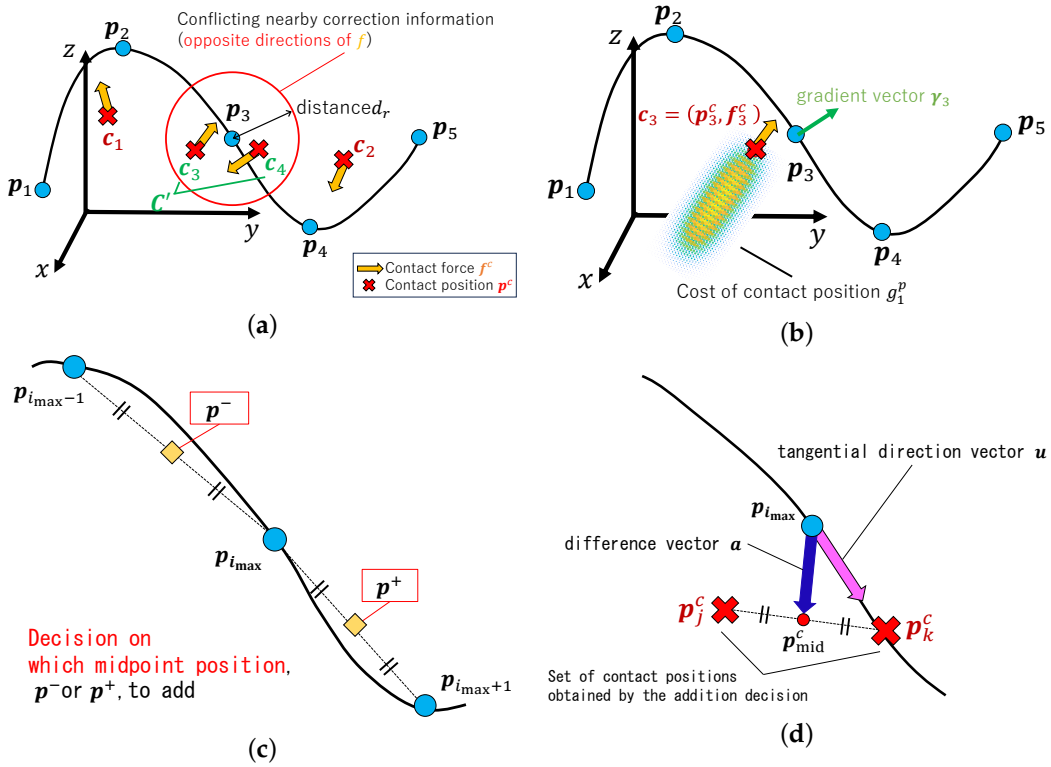


Figure 6. Procedure for control-point insertion based on contact information: (**a**) Extraction of conflicting contact information. (**b**) Gradient vector of control points induced by the cost of contact information c_3. (**c**) Two candidate positions for control-point insertion. (**d**) Determination of the control-point insertion position.

5. Simulation

5.1. Objectives and Conditions

To verify the applicability of the proposed method as a path generation approach for the end-effector position and orientation of a manipulator, motion path-following simulations of a three-dimensional six-degree-of-freedom manipulator (detailed in Appendix A) were conducted using the open-source physics engine MuJoCo [25]. Although experimental validation on a physical system is generally desirable, this study was evaluated solely through simulation. This is because the assessment of contact-related behavior strongly depends on implementation-specific factors such as actuator performance, control strategies, and force-sensing mechanisms. These elements should be individually designed according to the required task accuracy and characteristics, and including such case-specific implementations may obscure the fundamental properties of the proposed method.

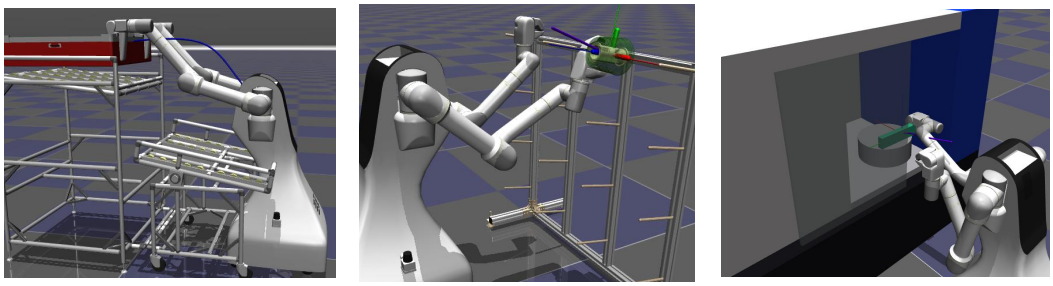
Figure 7 show three different simulation environments and their corresponding taught paths. Five simulation trials were performed for each environment. In the part box unloading task, the cart was designed to move when the distance between the end-effector and the cart became smaller than a predefined threshold. In the inverse kinematics calculation, the orientation error was expressed as an axis–angle vector obtained from the rotation matrix using the method proposed in [26].

The manipulator has torque sensors at all of its joints, and the information is converted as a wrench at the end-effector position for collision detection. The torque sensors provide information at 1 kHz. We used a Nelder–Mead algorithm implemented in the Matlab function (fminsearch), and its parameters for optimization were set as MaxFunctionEvaluation: $2000 \times D$, where D denotes the number of control points excluding the initial and target points, and both TolFun and TolX set as 1×10^{-4}. The parameters used in each task are summarized in Table 1.

Table 1. Parameters used in each task.

(a) Part Box Unloading					
Parameter	Value	Parameter	Value	Parameter	Value
$\boldsymbol{\alpha}$	$(1,1,2,2)$	a_ψ	20	w_p	1
β_p	0.1	b_ψ	0.1	w_o	1
γ_p	0.1	a_p	10	d_r	0.3
σ_p	0.5	b_p	0.1	d_g	0.01
a_v	5	w_1^p	1	E_{th}	0.5
b_v	$\pi/12$	w_1^o	0	N_q	101
(b) Yarn Cone Removal					
Parameter	Value	Parameter	Value	Parameter	Value
$\boldsymbol{\alpha}$	$(1,30,30,10)$	a_ψ	20	w_p	30
β_p	0.1	b_ψ	0.1	w_o	0.5
γ_p	0.1	a_p	10	d_r	0.05
σ_p	0.05	b_p	0.08	d_g	0.01
a_v	5	w_1^p	1	E_{th}	0.5
b_v	$\pi/12$	w_1^o	1	N_q	101
(c) Part Extraction					
Parameter	Value	Parameter	Value	Parameter	Value
$\boldsymbol{\alpha}$	$(1,10,10,5)$	a_ψ	20	w_p	20
β_p	0.1	b_ψ	0.1	w_o	0.1
γ_p	0.1	a_p	10	d_r	0.1
σ_p	0.05	b_p	0.08	d_g	0.01
a_v	5	w_1^p	1	E_{th}	0.5
b_v	$\pi/12$	w_1^o	1	N_q	101

The implementation of the proposed method, including online path correction, consists of a path generator and a simulator. The path generator was implemented in MATLAB 2025a, and the generated trajectories were passed to a MuJoCo (Ver. 2.2.1) simulator implemented in C++ (GCC 11.4.0). All experiments were conducted on a system equipped with 16 GB RAM, an Intel Core i7-12700 CPU, and an NVIDIA GeForce GTX 1650 GPU. The GPU was used only for visualization and not for computation.



(**a**) (**b**) (**c**)

Figure 7. Simulation environments: (**a**) Part box unloading. (**b**) Yarn cone removal. (**c**) Part extraction.

5.2. Baseline Method

As described in Section 2, optimization-based path planning approaches such as CHOMP and STOMP are not applicable to our problem, where model information is not available. To show this in comparative way, we built a baseline method where contact information is used just as the proof of obstacle existence around the contact position in the joint space. In the baseline method, $g_1(\boldsymbol{x}^q, \boldsymbol{c}_i)$ in (9) is replaced by $\hat{g}_1(\boldsymbol{\theta}^q, \boldsymbol{\theta}_i)$, where $\boldsymbol{\theta}^q$ and $\boldsymbol{\theta}_i$ denote the joint angle corresponding to $\boldsymbol{x}^q$ and contact $\boldsymbol{c}_i$, respectively. To reflect the distance between evaluation configuration $\boldsymbol{\theta}^q$ and contact configuration $\boldsymbol{\theta}_i$, the replaced cost is defined using a Gaussian function as follows:

$$\hat{g}_1(\boldsymbol{\theta}^q, \boldsymbol{\theta}_i) = \exp\left(-\frac{\|\boldsymbol{\theta}^q - \boldsymbol{\theta}_i\|^2}{\hat{\sigma}^2}\right), \tag{29}$$

where $\hat{\sigma}$ denotes the width parameter for the Gaussian, which was set as 0.1 rad. The baseline method can be regarded as a variant of the proposed approach in which the cost term based on the contact force direction is removed and replaced with a distance-based cost in the joint space, as commonly used in optimization-based planning methods. In this sense, it can be interpreted as an indirect approximation of how model-based methods such as CHOMP would behave when only collision information is available without an explicit environment model.

5.3. Results

Table 2 shows the average number of path corrections for each task. Examples of path corrections can be seen in movies in Supplementary Materials. Figure 8 presents the simulation results for the part extraction task. These results confirm that the proposed method can generate paths that avoid obstacles by modifying both position and orientation. The 3D vectors indicated in each snapshot denote the orientation of the end-effector with $(\phi v_x, \phi v_y, \phi v_z)$. It can be seen from the figure that it starts from and ends at orientations with zero rotation, while rotating the object around the z axis in the intermediate postures.

As shown in Table 2, tasks that require corrections in both position and orientation tend to exhibit a larger number of path corrections. This is because the appropriate amounts of position and orientation corrections are information that can be acquired by accumulating contact information. Therefore, to achieve more efficient path correction, it is necessary to design a cost function that can adjust the correction magnitudes for position and orientation

according to the magnitudes of the force and moment in the contact wrench. The last column in the table shows the lengths of the paths finally obtained. Compared with the initial lengths, the paths became longer as the task complexity increased. However, the standard deviations indicate that the final path lengths did not vary significantly across different trials.

Table 2. Number of corrections, computation time, and path length: Mean (SD).

Simulation	Corrections	Computation Time per Iteration [s]	Total Computation Time [s]	Path Length [m] (/Initial Path Length)
Part Box Unloading	1.6 (0.89)	1.04 (0.25)	1.66 (1.05)	1.42 (0.13)/1.04
Yarn Cone Removal	4.2 (0.84)	2.80 (3.11)	11.77 (8.27)	0.49 (0.11)/0.31
Part Extraction	6 (3.24)	2.48 (2.72)	14.90 (15.72)	3.24 (0.50)/0.52

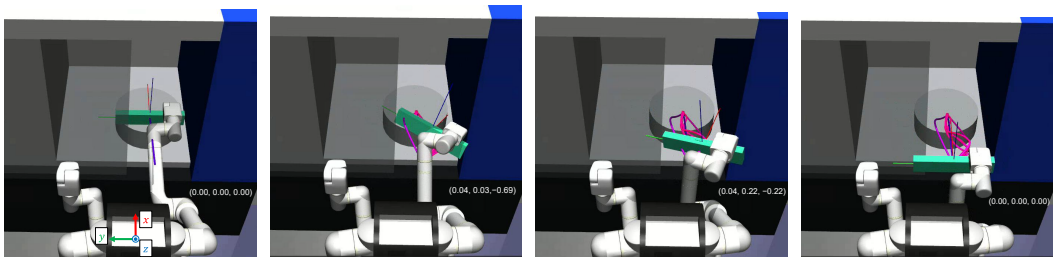


Figure 8. Simulation process (Start → Intermediate 1 → Intermediate 2 → End). Each snapshot includes the orientation of the end-effector in $(\phi v_x, \phi v_y, \phi v_z)$.

Figure 9 compares the performance of the proposed method and the baseline in the part extraction task. The red and blue plots correspond to the proposed and baseline methods, respectively. Circles indicate successful completion without collision, while crosses denote contact; each point represents the distance traveled until collision or goal completion. Across three independent runs, the baseline method failed to find a collision-free trajectory within 10 trials. Although the optimizer attempts to modify the trajectory to avoid collisions, the absence of directional information in the cost function prevents it from identifying a feasible avoidance direction. This result highlights the importance of distance-based information in conventional planning frameworks, demonstrating that the proposed contact-wrench-based cost can effectively serve as an alternative source of guidance.

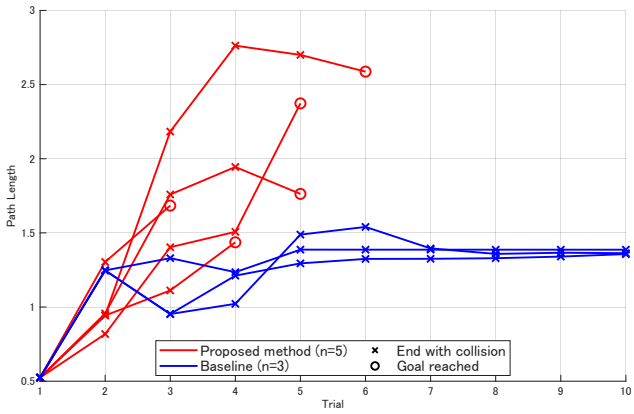


Figure 9. Motion generation processes with proposed and baseline methods (part extraction task).

To evaluate the noise tolerance of the proposed method, we conducted experiments under force measurement noise conditions. Table 3 summarizes the performance of the proposed method in the presence of force measurement noise. Gaussian noise following the distribution $\mathcal{N}(0, \sigma)$ was added to the wrench $\boldsymbol{F}^C$. The noise level σ was determined based on the contact forces measured in each task, corresponding to 1–25% of each force

and torque component in units of N and N·m, respectively. If a collision-free path could not be found within 20 attempts, the trial was terminated and regarded as a failure. In such cases, the number of corrections was recorded as 20 in the table. Depending on the task and noise level, there were cases in which the robot failed to find a path to the target pose within 20 attempts. Nevertheless, once a feasible path was found, the resulting path lengths were almost equivalent to those obtained under noise-free conditions.

Table 3. Number of corrections and path length with noise: Mean (SD).

Simulation	Noise σ	Corrections	Success Ratio	Path Length [m]
Part Box Unloading	0.1	4.4 (5.8)	9/10	1.42 (0.16)
Yarn Cone Removal	0.01	3.0 (1.6)	10/10	0.51 (0.17)
Part Extraction	0.1	9.6 (7.6)	7/10	2.72 (0.43)

6. Discussion

The simulation results confirmed that collision-free paths can be generated using only contact information. However, if users require an optimal path—for example, in terms of path length—the robot must perform additional contact interactions to more precisely estimate the collision boundaries even after a feasible collision-free path has been found. This limitation arises from the model-free nature of the proposed approach. Integrating local visual information around collision regions could improve path efficiency by refining the estimated boundaries. Such an extension would be effective in situations without severe occlusions.

In the evaluation under noisy force measurement conditions, more attempts were required, and in some cases the robot failed to find collision-free paths. Since the injected noise covered a wide range of both translational force and torque components, force measurement noise should be carefully characterized and considered in real robot applications. On the other hand, these results suggest that the proposed method requires an adaptive mechanism to promote exploration when contradictory force measurements are observed. The current implementation of the cost function is based on a deterministic contact force model. However, it could be extended to incorporate a probabilistic force measurement model, allowing the cost function to adapt according to the estimated uncertainty in the force measurements.

A fundamental limitation of the proposed method is that it performs local optimization of the trajectory. As a result, it may fail to find a globally optimal solution when the cost landscape contains barriers between the initial trajectory and the global optimum. Nevertheless, in many practical industrial scenarios, the search space for trajectory generation is sufficiently structured such that locally optimal solutions are adequate. Furthermore, when this assumption does not hold, a human operator can provide appropriate via points to guide the optimization and avoid undesirable local minima as needed. Another limitation is the sensitivity of the optimization to parameter settings, which are currently tuned for each task. Given their physical and geometric interpretations, developing systematic or automated parameter selection remains an important direction for future work.

7. Conclusions

In this study, we propose an automatic motion path correction method that considers end-effector orientation, with the aim of reducing the burden on operators during motion teaching for production-support robots. To verify the applicability of the proposed method, simulations of practical robot tasks were conducted. The results confirmed that the proposed method can avoid contact with obstacles by modifying both position and orientation. Future work will include improving robustness against force measurement

uncertainties that may arise in real robot applications, as well as developing a systematic approach for parameter adjustment in path optimization. Although the proposed method is based on contact information, it may be extended to incorporate proximity or local distance sensing instead of physical contact. Such an extension, incorporated with representation of proximity (e.g., [27,28]), would improve applicability to fragile or contact-sensitive objects and environments.

Supplementary Materials: The following supporting information can be downloaded at https://www.mdpi.com/article/10.3390/robotics15060108/s1, Video S1: Simulation of the part box unloading task demonstrating the proposed path correction method. Video S2: Simulation of the yarn cone removal task demonstrating the proposed path correction method. Video S3: Simulation of the part extraction task demonstrating the proposed path correction method.

Author Contributions: Conceptualization, Y.K. (Yuta Kasahara), Y.K. (Yuichi Kobayashi) and Y.N.; methodology, Y.K. (Yuta Kasahara), Y.K. (Yuichi Kobayashi), T.H. and Y.N.; software, Y.K. (Yuta Kasahara) and F.J.A.G.; validation, Y.K. (Yuta Kasahara), K.T. and Y.W.; formal analysis, Y.K. (Yuta Kasahara); investigation, Y.K. (Yuta Kasahara); resources, K.T. and Y.W.; data curation, Y.K. (Yuta Kasahara), K.Y. and F.J.A.G.; writing—original draft preparation, Y.K. (Yuta Kasahara); writing—review and editing, Y.K. (Yuichi Kobayashi) and T.H.; visualization, Y.K. (Yuta Kasahara) and F.J.A.G.; funding acquisition, Y.N. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the New Energy and Industrial Technology Development Organization (NEDO), Grant Number JPNP18002.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available upon request from the corresponding author.

Acknowledgments: During the preparation of this manuscript, the authors used ChatGPT 5.3 Instant/Thinking for the purposes of language polishing and translation. The authors have reviewed and edited the output and take full responsibility for the content of this publication.

Conflicts of Interest: Keiji Takeshita, Yuya Wada and Yoichiro Nakamura are employed by Shibaura Machine Co., Ltd. The authors declare that the research was conducted without any commercial or financial relationships that could be construed as a potential conflict of interest.

Appendix A. Configuration Settings

Detailed settings regarding the robot arm and objects are provided in this section. Figure A1a indicates the configuration of the arm. The DH parameters of the arm are shown in Table A1. Figure A1b,c indicate the size parameters of the objects for the part box unloading and yarn cone removal tasks, respectively. For the part extraction tasks, the object size was set as $0.04 \times 0.4 \times 0.08$ m. In all cases, the weight of the object was set to zero to reduce trajectory tracking control of the arm.

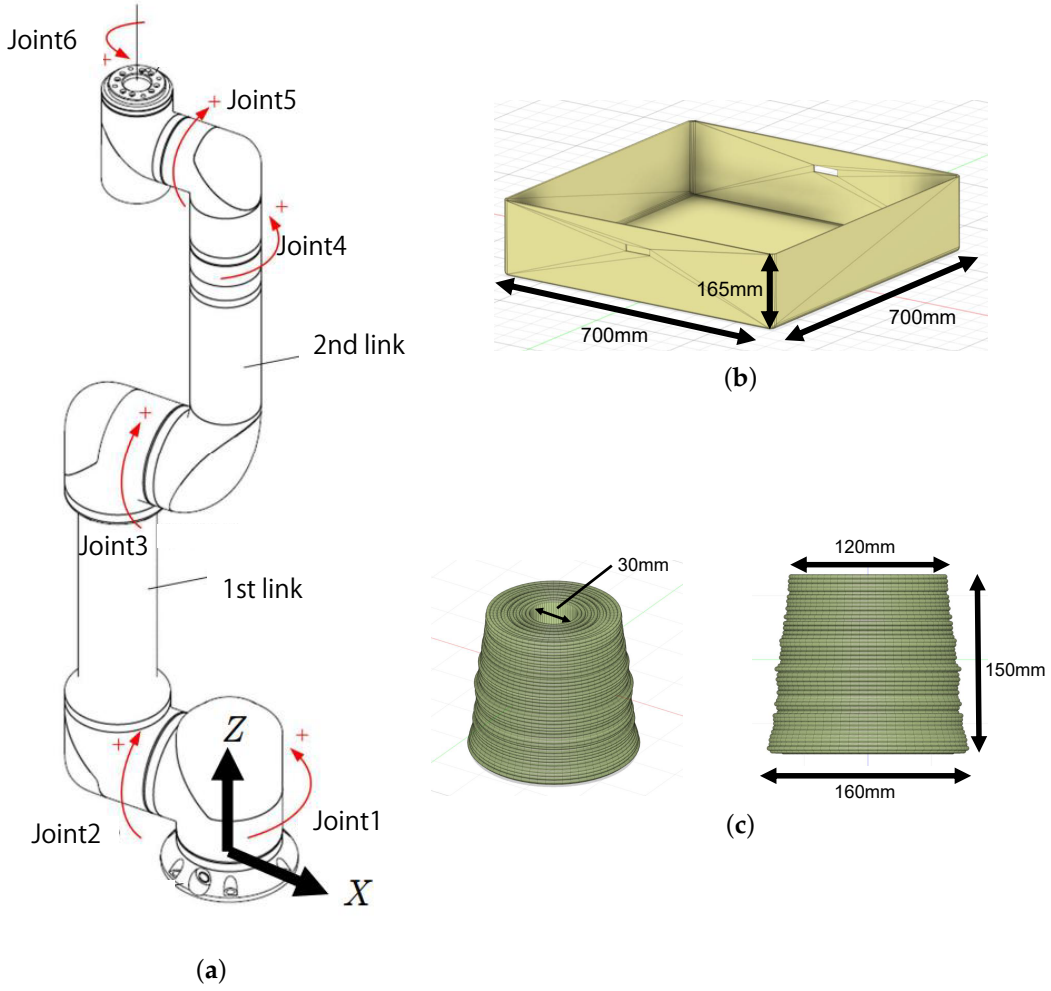


Figure A1. Arm joint configuration and object size parameters: (**a**) Arm joint configuration. (**b**) Part box. (**c**) Yarn cone.

Table A1. DH parameters for each arm.

Link	θ_i [rad]	d_i [m]	a_i [m]	α_i [rad]
1	$\theta_1 - \frac{\pi}{2}$	0.106	0	$\frac{\pi}{2}$
2	$\theta_2 + \frac{\pi}{2}$	0.140	0.428	π
3	$\theta_3 + \frac{\pi}{2}$	0.135	0	$\frac{\pi}{2}$
4	θ_4	0.428	0	$\frac{\pi}{2}$
5	θ_5	0.111	0	$-\frac{\pi}{2}$
6	$\theta_6 + \frac{\pi}{2}$	0.076	0	0

References

1. Pan, Z.; Polden, J.; Larkin, N.; Van Duin, S.; Norrish, J. Recent Progress on Programming Methods for Industrial Robots. *Robot. Comput.-Integr. Manuf.* **2012**, *28*, 87–94. [CrossRef]
2. Kawaharazuka, K.; Oh, J.; Yamada, J.; Posner, I.; Zhu, Y. Vision-Language-Action Models for Robotics: A Review Towards Real-World Applications. *IEEE Access* **2025**, *13*, 162467–162504. [CrossRef]
3. Brohan, A.; Brown, N.; Carbajal, J.; Chebotar, Y.; Dabis, J.; Finn, C.; Gopalakrishnan, K.; Hausman, K.; Herzog, A.; Hsu, J.; et al. RT-1: Robotics Transformer for Real-World Control at Scale. In Proceedings of the Robotics: Science and Systems XIX. Robotics: Science and Systems Foundation, Daegu, Republic of Korea, 10–14 July 2023. [CrossRef]

4. Zitkovich, B.; Yu, T.; Xu, S.; Xu, P.; Xiao, T.; Xia, F.; Wu, J.; Wohlhart, P.; Welker, S.; Wahid, A.; et al. RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control. In *Proceedings of the 7th Conference on Robot Learning*; Tan, J., Toussaint, M., Darvish, K., Eds.; PMLR: New York, NY, USA, 2023; Volume 229, pp. 2165–2183.
5. Ichter, B.; Brohan, A.; Chebotar, Y.; Finn, C.; Hausman, K.; Herzog, A.; Ho, D.; Ibarz, J.; Irpan, A.; Jang, E.; et al. Do As I Can, Not As I Say: Grounding Language in Robotic Affordances. In *Proceedings of the 6th Conference on Robot Learning*; PMLR: New York, NY, USA, 2023; pp. 287–318.
6. Driess, D.; Xia, F.; Sajjadi, M.S.M.; Lynch, C.; Chowdhery, A.; Ichter, B.; Wahid, A.; Tompson, J.; Vuong, Q.; Yu, T.; et al. PaLM-E: An Embodied Multimodal Language Model. In *Proceedings of the 40th International Conference on Machine Learning*; PMLR: New York, NY, USA, 2023; pp. 8469–8488.
7. Wu, Z.; Zhou, Y.; Xu, X.; Wang, Z.; Yan, H. MoManipVLA: Transferring Vision-language-action Models for General Mobile Manipulation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 11–15 June 2025; pp. 1714–1723.
8. Kober, J.; Bagnell, J.A.; Peters, J. Reinforcement Learning in Robotics: A Survey. *Int. J. Robot. Res.* **2013**, *32*, 1238–1274. [CrossRef]
9. Chen, L.; Jiang, Z.; Cheng, L.; Knoll, A.C.; Zhou, M. Deep Reinforcement Learning Based Trajectory Planning under Uncertain Constraints. *Front. Neurorobot.* **2022**, *16*, 883562. [CrossRef] [PubMed]
10. Levine, S.; Finn, C.; Darrell, T.; Abbeel, P. End-to-End Training of Deep Visuomotor Policies. *J. Mach. Learn. Res.* **2016**, *17*, 1–40.
11. Zhao, W.; Queralta, J.P.; Westerlund, T. Sim-to-Real Transfer in Deep Reinforcement Learning for Robotics: A Survey. In Proceedings of the 2020 IEEE Symposium Series on Computational Intelligence (SSCI), Canberra, Australia, 1–4 December 2020; pp. 737–744. [CrossRef]
12. Zhang, Z.; Hong, J.; Enayati, A.M.S.; Najjaran, H. Using Implicit Behavior Cloning and Dynamic Movement Primitive to Facilitate Reinforcement Learning for Robot Motion Planning. *IEEE Trans. Robot.* **2024**, *40*, 4733–4749. [CrossRef]
13. LaValle, S.M. *Rapidly-Exploring Random Trees: A New Tool for Path Planning*; Technical Report TR 98-11; Iowa State University: Ames, IA, USA, 1998.
14. Karaman, S.; Frazzoli, E. Sampling-Based Algorithms for Optimal Motion Planning. *Int. J. Robot. Res.* **2011**, *30*, 846–894. [CrossRef]
15. Khatib, O. Real-Time Obstacle Avoidance for Manipulators and Mobile Robots. *Int. J. Robot. Res.* **1986**, *5*, 90–98. [CrossRef]
16. Ratliff, N.; Zucker, M.; Bagnell, J.A.; Srinivasa, S. CHOMP: Gradient Optimization Techniques for Efficient Motion Planning. In Proceedings of the 2009 IEEE International Conference on Robotics and Automation, Kobe, Japan, 12–17 May 2009; pp. 489–494. [CrossRef]
17. Kalakrishnan, M.; Chitta, S.; Theodorou, E.; Pastor, P.; Schaal, S. STOMP: Stochastic Trajectory Optimization for Motion Planning. In Proceedings of the 2011 IEEE International Conference on Robotics and Automation, Shanghai, China, 9–13 May 2011; pp. 4569–4574. [CrossRef]
18. Schulman, J.; Ho, J.; Lee, A.; Awwal, I.; Bradlow, H.; Abbeel, P. Finding Locally Optimal, Collision-Free Trajectories with Sequential Convex Optimization. In Proceedings of the Robotics: Science and Systems IX. Robotics: Science and Systems Foundation, Berlin, Germany, 24–28 June 2013. [CrossRef]
19. Mukadam, M.; Dong, J.; Yan, X.; Dellaert, F.; Boots, B. Continuous-Time Gaussian Process Motion Planning via Probabilistic Inference. *Int. J. Robot. Res.* **2018**, *37*, 1319–1340. [CrossRef]
20. Xidias, E.K.; Zacharia, P.T.; Aspragathos, N.A. Time-optimal task scheduling for articulated manipulators in environments cluttered with obstacles. *Robotica* **2010**, *28*, 427–440. [CrossRef]
21. Laha, R.; Sun, R.; Wu, W.; Mahalingam, D.; Chakraborty, N.; Figueredo, L.F.; Haddadin, S. Coordinate Invariant User-Guided Constrained Path Planning with Reactive Rapidly Expanding Plane-Oriented Escaping Trees. In Proceedings of the 2022 International Conference on Robotics and Automation (ICRA), Philadelphia, PA, USA, 23–27 May 2022; pp. 977–984. [CrossRef]
22. Losey, D.P.; O'Malley, M.K. Trajectory Deformations From Physical Human–Robot Interaction. *IEEE Trans. Robot.* **2018**, *34*, 126–138. [CrossRef]
23. Liu, Q.; Liu, Y.; Li, Y.; Zhu, C.; Meng, W.; Ai, Q.; Xie, S.Q. Path Planning and Impedance Control of a Soft Modular Exoskeleton for Coordinated Upper Limb Rehabilitation. *Front. Neurorobot.* **2021**, *15*, 745531. [CrossRef] [PubMed]
24. Nelder, J.A.; Mead, R. A Simplex Method for Function Minimization. *Comput. J.* **1965**, *7*, 308–313. [CrossRef]
25. Todorov, E.; Erez, T.; Tassa, Y. MuJoCo: A Physics Engine for Model-Based Control. In Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, Vilamoura-Algarve, Portugal, 7–12 October 2012; pp. 5026–5033. [CrossRef]
26. Sekiguchi, M.; Takesue, N. Numerical Method for Inverse Kinematics Using an Extended Angle-Axis Vector to Avoid Deadlock Caused by Joint Limits. *Adv. Robot.* **2021**, *35*, 919–926. [CrossRef]

27. Kobayashi, Y.; Matsui, R. Manifold Learning Approach Toward Constructing State Representation for Robot Motion Generation. In *Transactions on Computational Collective Intelligence XXIV*; Nguyen, N.T., Kowalczyk, R., Filipe, J., Eds.; Springer: Berlin/Heidelberg, Germany, 2016; pp. 101–116. [CrossRef]
28. Somei, T.; Kobayashi, Y.; Shimizu, A.; Kaneko, T. Clustering of image features based on contact and occlusion among robot body and objects. In Proceedings of the 2013 IEEE Workshop on Robot Vision (WORV), Clearwater Beach, FL, USA, 15–17 January 2013; pp. 203–208. [CrossRef]