Mechanism and Machine Theory 144 (2020) 103664

Contents lists available at ScienceDirect

Mechanism and Machine Theory

journal homepage: www.elsevier.com/locate/mechmachtheory

Research paper

Real-time trajectory planning based on joint-decoupled optimization in human-robot interaction

Shiyu Zhang[a,*], Andrea Maria Zanchettin[b], Renzo Villa[b], Shuling Dai[a]

[a] State Key Laboratory of Virtual Reality Technology and Systems, Beihang University, Beijing, China
[b] Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Milan, Italy

ARTICLE INFO

Article history:
Received 2 July 2019
Revised 11 September 2019
Accepted 7 October 2019

Keywords:
Real-time trajectory planning
Human-robot interaction
Non-linear optimization
Machine learning
Serial robot

ABSTRACT

In order to perform safe and natural interactions with humans, robots are required to adjust their motions quickly according to human behaviors. Performing the complex calculation and updating the trajectories in real-time is a particular challenge. In this paper, we present a real-time optimization-based trajectory planning method for serial robots. We encode the trajectory planning problem into a series of optimization problems. To solve the high-dimensional complex non-linear optimization problems in real-time, we provide a joint-decoupling method that transforms the original joint-coupled optimization problem into multiple joint-independent optimization problems, with much lower computational complexity. We implement and validate our method in a specific human-robot interaction case. Experimental results show that the computational feasibility and efficiency of optimization solution were greatly improved by the joint-decoupling transformation. Smooth, safe, and rapid motion of the robot was generated in real-time, establishing a basis for safe and reactive human-robot interactions.

1. Introduction

Robots are increasingly commonplace in human life and now interact with humans in many circumstances, including collaborative manufacturing [1–4], rehabilitation [5–8], and entertainment [9–11]. In applications with human-robot interaction (HRI), robots must react promptly to unexpected human behavior to avoid hurting users. Thus, generating safe and efficient trajectories in real-time is an essential matter [12].

Trajectory planning using non-linear optimization is an effective way to generate trajectories compromising among specific indices, encoding the combined requirements of safety and speed of motion into an objective function and constraints. However, as the objective functions and constraints are always complex in practical cases, the solution process is computationally demanding. Moreover, the motion of all of the robotic joints must be coordinated, meaning that the joint-related components in the optimization parameters are coupled with each other. As the number of degrees of freedom (DOFs) of the robot increases, the number of dimensions in the optimization problem increases as well. This leads to even greater computational complexity and makes it difficult to guarantee real-time performance. Additionally, higher numbers of DOFs lead to less flexibility in the solution from the additional constraints. The optimization solver may fail to output feasible solutions.

* Corresponding author.
E-mail address: zhangshiyu@buaa.edu.cn (S. Zhang).

https://doi.org/10.1016/j.mechmachtheory.2019.103664
0094-114X/

In this paper, we address this complex problem. First, we encode the HRI trajectory planning problem into a series of non-linear optimization problems considering HRI safety and motion rapidity. To improve the computational efficiency and flexibility of solving the high-dimensional optimization problems, especially for cases with many DOFs, we decouple the joint-coupled optimization problem and transform it into multiple joint-independent optimization problems, each of which has fewer dimensions and faster solutions. We then study and implement our approach for a specific HRI case and experimentally verify its performance.

We organize the remainder of this paper as follows (Fig. 1). We first discuss state of the art approaches to optimization-based trajectory planning in Section 2. Then we describe the main problem discussed in this paper, real-time serial trajectory planning for HRI applications, in Section 3. We present the general optimization model for this problem and the joint-decoupling model transformation method in Section 4. We detail our implementation of our method in a specific case in Section 5. In Section 6, we evaluate the feasibility and efficiency of the presented method using numerical experiments and validate the real-time performance. Finally, we discuss our conclusions and future work in Section 7.

2. Related work

In HRI applications, robots must react to changing environments and human behaviors smoothly, quickly, safely, and efficiently. Thus, real-time trajectory planning is an essential procedure.

Optimization-based trajectory planning methods, in which the indices are encoded in a constrained optimization problem [13–19], are widely used to ensure safety and improve the motion efficiency. As the objective function and constraints are complex in practical applications, the optimization problems are generally non-linear, making them very time-consuming to solve.

Some researchers adopt parallel computing to reduce the calculation time. Bäuml et al. [20] perform computations in parallel with distributed computing resources and 32 CPU cores to cope with the high computational demands. Liu and Tomizuka [21] design a parallel controller combined with a baseline controller to compute the basic optimization problem offline with an online safety controller dealing with the non-linear, non-convex, and temporal safety constraints. Real-time performance can be achieved by these methods. However, they require significant hardware resources. Moreover, when it comes to more complex situations, such as ones with higher dimensions or more sophisticated models, the requirements of hardware resources will be even higher, which are not always be satisfied. Thus, it is hardly to generalize to more complicated problems.

Other researchers resort to machine learning methods to select an approximate initial value for the iterative solver to speed up the calculation [22–25]. Lampariello et al. [26] learn an approximate initialization by generating optimal database and constructing the regression model. Jetchev and Toussaint [27] proposed a high-dimensional feature selection method and a task space transfer, which can improve the efficiency of adapting the samples to new situations. Hauser [28] discussed theoretically the requirements of the database and the algorithm for the data-driven method to guarantee the performance of the solution. However, the initial solutions obtained by learning are not guaranteed to be feasible. Thus, the iterative solving process is still necessary. Although the calculations could be speed up in most of situations, real-time performance and feasibility could not be ensured.

Another method approximates the original problem using another form that can be solved efficiently. Kröger and Wahl [29] calculate the time-optimal trajectory with decision trees and non-linear equations that can be solved analytically. Tsai et al. [30] approximate the non-convex optimization problem to a convex optimization problem with a quadratic objective function and linear constraints to significantly reduce the calculation time. Liu et al. [31] transform the non-convex optimization problem into a sequence of convex subproblems and then iteratively solve the subproblems until convergence, improving the computational efficiency for reaching the local optima.

In this paper, we present a new optimization problem approximation method from a joint decoupling perspective. We decompose the original integrated problem into several parallel subproblems, achieving better computational efficiency and flexibility than the integrated form and the sequential subproblems, especially when scaling to higher-dimensional problems. Additionally, it can be combined with machine learning methods, in which the learned initial solutions can be refined in an analytical way rather than numerical iteration and thus the calculation time restriction and the feasibility can be ensured. Moreover, owing to the very low computational complexity, it can be implemented in hardware with relatively low performance.

3. Real-time trajectory planning for HRI

3.1. Human-robot interaction applications

In this paper, we consider a specific class of HRI applications: those in which the robot and human share a workspace and interact with each other to perform tasks such as hand-over [32], collaborative assembly [33,34], and haptic feedback [25].

We consider a reaching process in which the robot moves from a current state to a target state. The target state is determined according to the human motion. To adapt to variable tasks, rather than programming the trajectory in a fixed

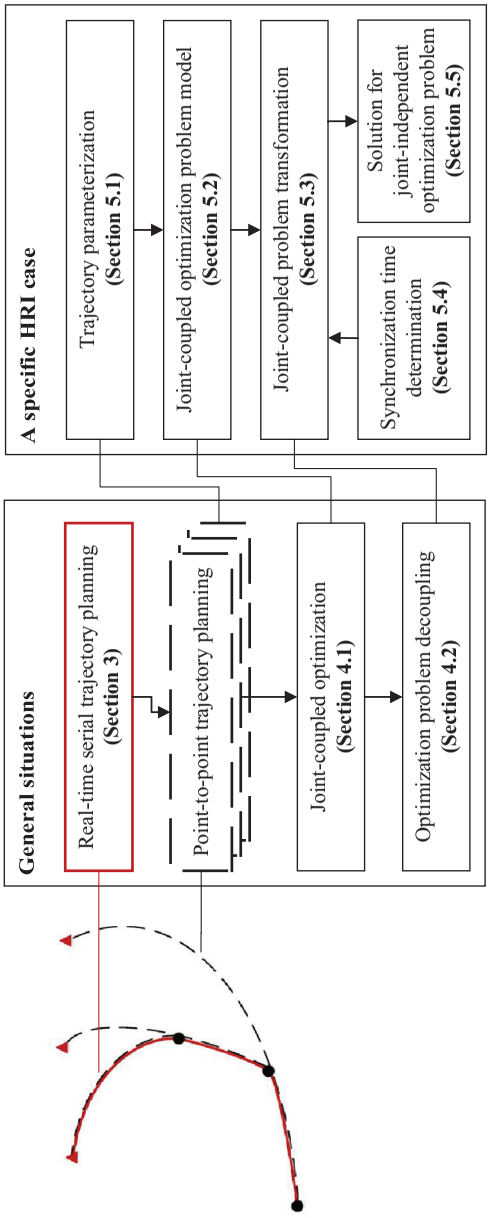


Fig. 1. Organization of the paper.

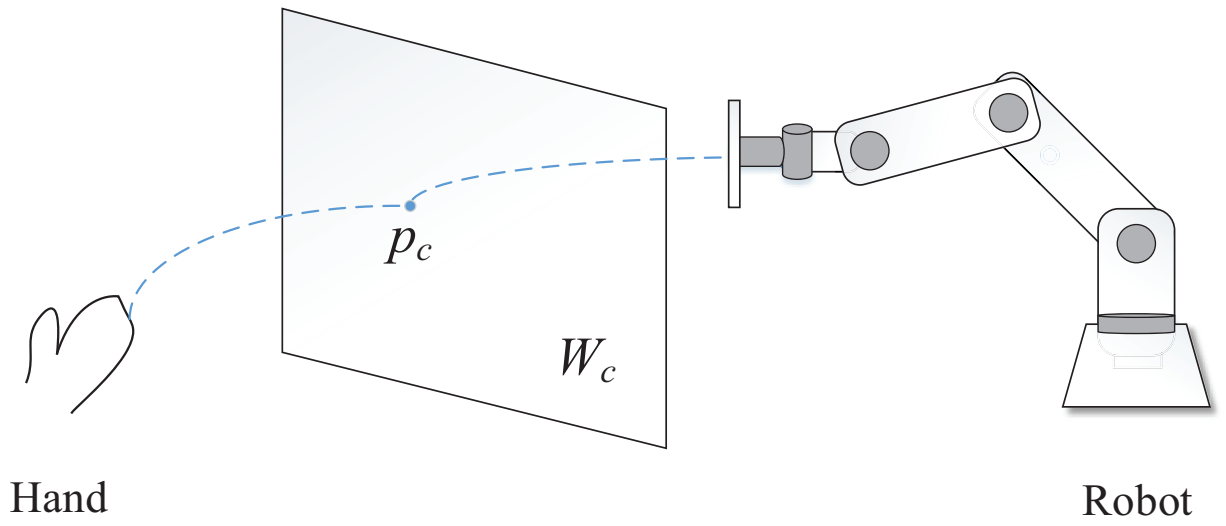


Fig. 2. A human-robot interaction case.

pattern, it is adjusted according to the detected human motion and the corresponding predicted target during runtime. To achieve this, we present a serial trajectory planning framework in Section 3.2.

For HRI trajectory planning applications, we need to consider criteria such as HRI safety and motion efficiency to generate ideal trajectories. Thus, we model the trajectory planning problem as an optimization problem in which the criteria are encoded into the objective function and constraints as detailed in Section 4.1.

Real-time performance is one of the most important criteria in this application. The unpredictability of human motion means that the target is changing all of the time. The motion state and trajectory need to be updated frequently to ensure the accuracy, which requires quick solutions to the trajectory planning problem. Improving the computational efficiency and flexibility of the solution for complex optimization problems to generate safe trajectories in real-time is the main aim of this paper. We describe our general method in Section 4.2.

An HRI example is shown in Fig. 2. The human hand performs manipulations at specific points in the workspace. As the hand approaches a specific point, the robot system continuously predicts the position of the target point according to the hand motion and then generates new trajectories to bring the end-effector to the same point, providing assistance for the manipulation task. This pattern is extremely common in the HRI tasks of interest here: hand-over, collaborative assembly, and haptic feedback.

To avoid the robot injuring the user, we restrict the working range of the robot. The user and the robot work in different areas separated by the interaction area W_C, and contact between the human hand and the robotic end-effector happens only in the interaction area. In other words, the predicted interaction point $\boldsymbol{p_c}$ should be in the interaction area

$$\boldsymbol{p_c} \in W_C. \tag{1}$$

Also, the robot must stop moving when it reaches the final interaction point.

Additionally, as the end-effector approaches the interaction point, the motion should be smooth. The acceleration should be as low as possible.

Section 5 details the implementation of our method for this example.

3.2. Real-time serial trajectory planning

We select joint space to perform trajectory planning. Since the axis low-level controller is fed with position and velocity references in joint space, the point-to-point trajectory have to be planned in joint space as well. Also, kinematical feasible trajectories are obtained only if position, velocity and acceleration constraints are satisfied which are easy to implement in joint space.

Considering point-to-point trajectory planning for serial robots in joint space, we parameterize and represent the trajectory as parameter $\boldsymbol{C} \in \mathbb{R}^{N_C}$. We then denote the joint position, velocity, and acceleration as $\boldsymbol{q}(\boldsymbol{C}, t) \in \mathbb{R}^{N_J}$, $\dot{\boldsymbol{q}}(\boldsymbol{C}, t) \in \mathbb{R}^{N_J}$, $\ddot{\boldsymbol{q}}(\boldsymbol{C}, t) \in \mathbb{R}^{N_J}$, respectively, where $t \in \mathbb{R}$ represents time and N_J represents the number of DOFs for the robot. Trajectory parameter $\boldsymbol{C}$ is determined by the input variable $\boldsymbol{X} \in \mathbb{R}^{N_X}$, which includes factors of the initial and final motion state, and variables related to the environment and the human behavior.

The trajectory planning problem is essentially the mapping from input variable $\boldsymbol{X}$ to trajectory parameter $\boldsymbol{C}$:

$$\boldsymbol{X} \rightarrow \boldsymbol{C} = f(\boldsymbol{X}). \tag{2}$$

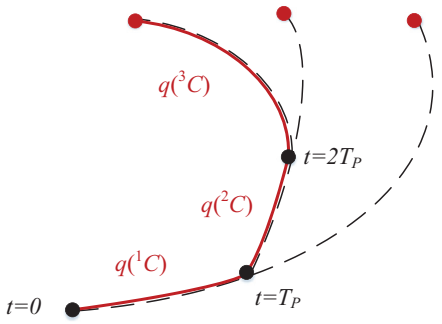


Fig. 3. Serial trajectory planning.

In the HRI applications mentioned above, in which $\boldsymbol{X}$ is time varying, the robot must react to the changes quickly, which requires re-planning the trajectory at high frequency according to the updated $\boldsymbol{X}$. In the ith sampling and planning interval, we detect the environment and update the input variable to obtain ${}^{\boldsymbol{i}}\boldsymbol{X}$ and then re-plan the trajectory to obtain the corresponding parameter ${}^{\boldsymbol{i}}\boldsymbol{C}$. This occurs until the robot reaches the final target as shown in Fig. 3. Trajectories calculated in each interval are depicted as the black dashed lines, and the actual trajectory of the whole process is depicted as the red solid line. The trajectory of the whole procedure is

$$\{\boldsymbol{q}({}^{\boldsymbol{i}}\boldsymbol{C}, t), t \in [(i-1)T_p, iT_p), i = 1, 2, \ldots, N_T\}, \tag{3}$$

where N_T is the total number of periods, and T_p represents the duration of one period.

To ensure that the robot will make a complete stop at the exact target position without exceeding the acceleration bounds, in each period we always plan from the current instant (shown as black points in Fig. 3) to the predicted final instant in which the robot will stop moving (shown as red points in Fig. 3) rather than to the next sampling time.

This process consists of a series of point-to-point trajectory planning problems. To guarantee that the robot responds to the frequently changing environment, T_p must be very small (less than 10 milliseconds in general [29,35–37]). Thus, high real-time performance is required, which means the calculation time for each planning period should not exceed T_p.

4. Trajectory planning based on decoupled optimization

4.1. Joint-coupled optimization problem

In this section, we consider the point-to-point trajectory planning problem for each period. Optimization-based trajectory planning is widely used to improve the required performance and balance all of the indices. For this reason, we specify Eq. (2) as an optimization problem

$$\boldsymbol{X} \rightarrow \boldsymbol{C} = O_{cpl}(\boldsymbol{X}), \tag{4}$$

which is always non-linear in practice.

The optimization parameter $\boldsymbol{C}$ consists of the trajectory parameters of the robotic joints, such as velocity, acceleration, and motion time. As the motion of a joint is affected by other joints, each component in $\boldsymbol{C}$ depends on all of the joints of the robot.

Assuming a robot with N_J DOFs, we classify the trajectory parameter $\boldsymbol{C}$ as

$$\boldsymbol{C} = (\boldsymbol{C_J}^{\boldsymbol{1}}, \ldots, \boldsymbol{C_J}^{\boldsymbol{N_J}}, \boldsymbol{C_{cpl}}), \tag{5}$$

where $\boldsymbol{C_{cpl}} \in \mathbb{R}^{N_{CP}}$ is the coupling parameter, and $\boldsymbol{C_J}^{\boldsymbol{i}} \in \mathbb{R}^{N_{JP}} (i = 1, \ldots, N_J)$ is the joint-related parameter of the ith joint.

The coupling parameter is the set of components in the trajectory expression that applies to all of the joints in the robot simultaneously. For example, since all of the joints move and stop simultaneously, the motion time is a shared parameter and, therefore, a component of the coupling parameter $\boldsymbol{C_{cpl}}$.

The joint-specific parameter $\boldsymbol{C_J}^{\boldsymbol{i}}$ is the component that only appears in the trajectory expression of the ith joint. In the coupled optimization problem, $\boldsymbol{C_J}^{\boldsymbol{i}}$ is not solely related with the ith joint in actuality, as it is affected by other joints indirectly through the coupling parameter $\boldsymbol{C_{cpl}}$. However, when $\boldsymbol{C_{cpl}}$ is fixed, $\boldsymbol{C_J}^{\boldsymbol{i}}$ can be considered to be determined by the ith joint alone. The method to determine coupling parameter and make decoupling transformation will be detailed in Section 4.2.

The coupled optimization problem has $N_{JP} \times N_J + N_{CP}$ optimization parameters. In general, the computational complexity of the optimization problem increases in a super linear way as the dimensionality of the optimization parameter increases. For example, solving one optimization problem with $m \times n$ parameters always takes much more time than solving m optimization problems with n parameters. Thus, when the number of joints N_J increases, the dimensionality of $\boldsymbol{C}$ increases, and the computational complexity rises rapidly.

Further, the coupling solver only solves problems with fixed N_J. For example, a new robot with the different number of DOFs has a different number of dimensions and requires modifying the problem model with new optimization parameters and developing a new solver accordingly. Additionally, the coupling between joints means that the coupling solver may fail to find a feasible solution for the whole robot (e.g., due to the unsatisfactory conditions of certain joints), causing it to stop working.

4.2. Optimization problem decoupling

4.2.1. Overview

HRI applications demand high real-time performance, with calculation efficiency as a significant criterion. To reduce the complexity and improve the flexibility of the solver, we make an approximation by decoupling the joint-coupled optimization problem and transforming it into N_J joint-independent optimization problems.

With the parameter classification described in Eq. (5), we can rewrite the coupled optimization problem defined in Eq. (4) as:

$$\boldsymbol{X} \rightarrow (\boldsymbol{C_J^1}, \ldots, \boldsymbol{C_J^{N_J}}, \boldsymbol{C_{cpl}}) = O_{cpl}(\boldsymbol{X}), \ \ i = 1, \ldots, N_J. \tag{6}$$

By this way, all the coupling elements are encoded into the coupling parameter $\boldsymbol{C_{cpl}}$. Thus, when $\boldsymbol{C_{cpl}}$ is determined to a certain value, which means the way of coupling has been determined and fixed, the original joint-coupled optimization problem can be transformed into N_J independent optimization problems:

$$(\boldsymbol{X}, \boldsymbol{C_{cpl}}) \rightarrow \boldsymbol{C_J^i} = {O_{ind}}^i(\boldsymbol{X}, \boldsymbol{C_{cpl}}), \ \ i = 1, \ldots, N_J. \tag{7}$$

The dimensionality of the optimization parameter is N_{JP}, which is much lower than that of the original problem. Consequently, the computing complexity is greatly reduced.

4.2.2. Coupling parameter determination using machine learning

According to Section 4.2.1, the coupling parameter $\boldsymbol{C_{cpl}}$ is the key factor of joint decoupling transformation. Thus, finding an appropriate $\boldsymbol{C_{cpl}}$ is the precondition and most crucial step.

The coupling parameter determination is the key to coordinate all of the joints. It affects the solutions of independent optimization problems. First of all, the feasibility of the solution should be guaranteed. That is, with the selected coupling parameter, we are able to find a $\boldsymbol{C_J^i}$ that satisfies all of the constraints for any joint i. Thus, we need to find the feasible range for $\boldsymbol{C_{cpl}}$ considering all of the joints.

Note that in some cases, the feasible set of $\boldsymbol{C_{cpl}}$ is empty with respect to joint i, which means that there is no feasible $\boldsymbol{C_{cpl}}$ satisfying all of the constraints for joint i. Accordingly, there is no feasible solution for the coupling problem. In these circumstances, we separate the infeasible joints from the coupling problem and change the value of N_J accordingly, then solve the remaining N_J-DOF coupling planning problems. At the end, we deal with the infeasible joints independently.

Additionally, in the feasible range of the coupling parameters, we try to obtain a solution close the optimal one of the original joint-coupled optimization problem. To select a near-optimal coupling parameter $\boldsymbol{C_{cpl}}^\star$ without determining $\boldsymbol{C_J^i}(i = 1, \ldots, N_J)$, we adopt a machine learning method. We learn from the previous optimal data collected by solving the original coupled optimization problems in order to obtain an approximately optimal initial value. We then refine it to ensure the feasibility.

The detailed procedure to determine a near-optimal coupling parameter is as follows.

1. Determine the feasible range of the coupling parameter with respect to each joint.
   The feasible range of the coupling parameter satisfying all of the constraints of joint i is
   $$S_{\boldsymbol{C_{cpl}}^i} = \{\boldsymbol{C_{cpl}}^i | \boldsymbol{C_J^i} \in S_{\boldsymbol{C_J}^i}, \boldsymbol{Constraints^i}(\boldsymbol{C_{cpl}}^i, \boldsymbol{C_J^i}) < \boldsymbol{0}\}, \tag{8}$$
   where $S_{\boldsymbol{C_J}^i}$ is the domain for $\boldsymbol{C_J^i}$, and $\boldsymbol{Constraints^i}(\boldsymbol{C_{cpl}}^i, \boldsymbol{C_J^i})$ represents all the inequality constraints for the ith joint.
2. Determine feasible and infeasible joints.
   If the feasible sets for the coupling parameter for some joints are empty, i.e.,
   $$S_{\boldsymbol{C_{cpl}}^i} = \emptyset, \tag{9}$$
   then these joints are denoted as infeasible joints.
   We remove infeasible joints and change the value of N_J accordingly.
3. Determine the feasible range of the coupling parameter with respect to all of the joints.
   Considering all of the feasible joints, the feasible range of the coupling parameter is
   $$S_{\boldsymbol{C_{cpl}}} = \cap_{i=1}^{N_J} S_{\boldsymbol{C_{cpl}}^i}. \tag{10}$$
4. Learn a near-optimal coupling parameter.
   We select a near-optimal $\boldsymbol{C_{cpl}}$ in $S_{\boldsymbol{C_{cpl}}}$ using machine learning.

In detail, the procedure for learning a near-optimal coupling parameter is as follows.

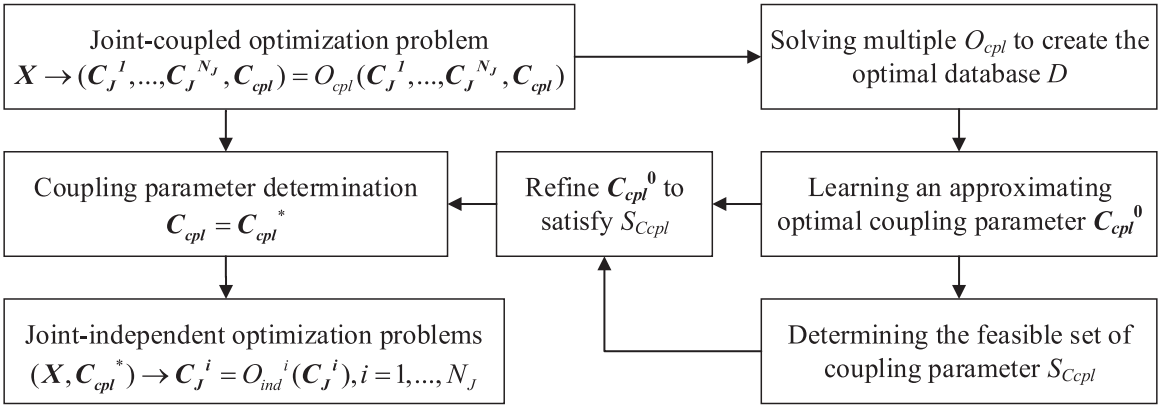


Fig. 4. Joint-coupled optimization problem transformation.

1. Generate the database.
   We select N_D groups of input variables ${}^i\boldsymbol{X}(i = 1, 2, \ldots, N_D)$ and solve the coupled optimization problem by Eq. (4):

   $$ {}^i\boldsymbol{C}_{\boldsymbol{opt}} = O_{cpl}({}^i\boldsymbol{X}). \tag{11}$$

   Then we extract the coupling parameter and construct an optimal database D:

   $$ D = \{{}^i\boldsymbol{X}, {}^i\boldsymbol{C}_{\boldsymbol{cpl\_opt}} | i = 1, 2, \ldots, N_D\}. \tag{12}$$

   The database is generated offline.
2. Establish the regression model.
   By analyzing and processing the data in D, we obtain an approximating mapping from input variable to the optimal coupling parameter as

   $$ \boldsymbol{X} \to \boldsymbol{C}_{\boldsymbol{cpl}} = R(\boldsymbol{X}). \tag{13}$$

   The regression model can be established both online and offline depending on the specific regression method.
3. Learn an approximate coupling parameter.
   According to the new input variable $\boldsymbol{X}'$, we obtain an initial approximating coupling parameter $\boldsymbol{C}'_{\boldsymbol{cpl}}$ using the regression model in Step 2:

   $$ \boldsymbol{C}'_{\boldsymbol{cpl}} = R(\boldsymbol{X}'). \tag{14}$$

4. Refine the coupling parameter.
   The initial $\boldsymbol{C}'_{\boldsymbol{cpl}}$ may not satisfy all of the constraints, so we adjust it to ensure it is belonging to the feasible set $S_{\boldsymbol{C}_{\boldsymbol{cpl}}}$. If $\boldsymbol{C}'_{\boldsymbol{cpl}} \in S_{\boldsymbol{C}_{\boldsymbol{cpl}}}$, then

   $$ \boldsymbol{C}_{\boldsymbol{cpl}}{}^* = \boldsymbol{C}'_{\boldsymbol{cpl}}. \tag{15}$$

   Otherwise, if $\boldsymbol{C}'_{\boldsymbol{cpl}} \notin S_{\boldsymbol{C}_{\boldsymbol{cpl}}}$,

   $$ \boldsymbol{C}_{\boldsymbol{cpl}}{}^* = \boldsymbol{C}_{\boldsymbol{cpl\_b}}, \tag{16}$$

   where $\boldsymbol{C}_{\boldsymbol{cpl\_b}}$ is the nearest boundary point to $\boldsymbol{C}'_{\boldsymbol{cpl}}$.

In summary, Fig. 4 shows the whole process of optimization decoupling.

5. Implementation of a HRI case

In this section, we consider a special case of the real-time serial trajectory planning problem for HRI applications described in Sections 3 and 4. The trajectory for each period is parameterized by the trapezoidal velocity profile (Section 5.1). We construct the coupled optimization problem by considering the speed of motion, HRI safety, and the mechanical limitations of the robot (Section 5.2). We then describe the implementation of the joint-decoupling method for this model, including coupling optimization transformation (Section 5.3), coupling parameter selection (Section 5.4), and solutions for the independent optimization problems (Section 5.5).

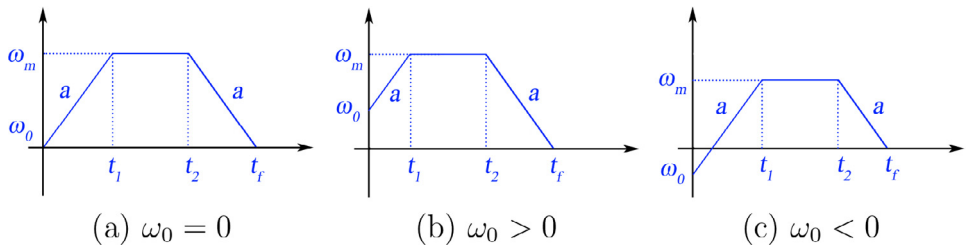


Fig. 5. Trapezoidal velocity profile.

5.1. Trajectory parameterization

Considering kinematical trajectory planning for the serial robot in joint space, the trajectory for each joint is encoded by the trapezoidal velocity profile [20,38] (Fig. 5), which consists of three phases: the uniform acceleration phase $(0-t_1)$, the maximum velocity phase (t_1-t_2), and the uniform deceleration phase (t_2-t_f). The uniform acceleration and deceleration phase have the same absolute value of acceleration, denoted as a. The velocity of the maximum velocity phase is ω_m.

For each joint, the initial velocity ω_0 and the position offset $q_f = q_c - q_0$, in which q_c is the goal joint position and q_0 is the initial joint position, are known. We can then determine the velocity profile with five variables a, ω_m, t_1, t_2, and t_f, which must satisfy the following constraints:

$$0 < a \leq a_{max} \tag{17}$$

$$\max(0, \omega_0) \leq \omega_m \leq \omega_{max} \tag{18}$$

$$0 \leq t_1 \leq t_2, \tag{19}$$

$$t_1 \leq t_2 \leq t_f, \text{ and} \tag{20}$$

$$0 \leq t_f \leq t_{max}. \tag{21}$$

These are linked by three equations:

$$\frac{1}{2}\omega_m(t_f + t_2 - t_1) + \frac{1}{2}\omega_0 t_1 = q_c - q_0, \tag{22}$$

$$\omega_m = \omega_0 + at_1, \text{ and} \tag{23}$$

$$(\omega_m - \omega_0)(t_f - t_2) = t_1\omega_m. \tag{24}$$

where $a_{\max}$, $\omega_{\max}$ and $t_{\max}$ are the maximum allowable acceleration, velocity, and motion time, respectively.

We then represent the trajectory of each joint as

$$\ddot{q}(t) = a\hat{r}(t, t_1, t_2), \tag{25}$$

$$\dot{q}(t) = \omega_0 + a\hat{r}(t, t_1, t_2)t, \text{ and} \tag{26}$$

$$q(t) = q_0 + \omega_0 t + \frac{1}{2}a\hat{r}(t, t_1, t_2)t^2, \tag{27}$$

where

$$\hat{r}(t, t_1, t_2) = r(t_1 - t) - r(t - t_2), \tag{28}$$

and $r(t)$ is the step function.

Note that in this model, we only consider the situation where $q_f > 0$. ω_0 can be positive, negative, or zero (shown as Fig. 5). In the cases of $q_f < 0$, we can use the same model to calculate the trajectory parameters by changing the signs of q_f, ω_0, and ω_m accordingly.

5.2. Joint-coupled optimization problem model

5.2.1. Input variable and optimization parameter

As described in Section 3.1, the movement of the robot from a standstill state to the final goal consists of a series of point-to-point trajectory planning problems. The series of trajectory planning problems are specified with initial configuration $\boldsymbol{q_0} \in \mathbb{R}^{N_J}$, initial velocity $\boldsymbol{\omega_0} \in \mathbb{R}^{N_J}$, final configuration $\boldsymbol{q_c} \in \mathbb{R}^{N_J}$, and null final velocity. Among these, null final velocity indicates that the robot must stop moving when it reaches the interaction point and would not harm the humans while interacting with them.

At each sampling moment, the initial configuration $\boldsymbol{q_0}$ and the initial velocity $\boldsymbol{\omega_0}$ are detected as the current motion state. The final configuration $\boldsymbol{q_c}$ is set corresponding to the predicted interaction point $\boldsymbol{p_c}$ by the inverse kinematics. If more than one feasible solutions of the inverse kinematics exist, we select the one with the smallest deviation from the current configuration. As detailed in Section 5.1, the kinematical trajectory planning concerns only the offset $\boldsymbol{q_f} = \boldsymbol{q_c} - \boldsymbol{q_0}$, rather than the individual $\boldsymbol{q_c}$ and $\boldsymbol{q_0}$. Therefore, the input variable is defined as $\boldsymbol{X} = (\boldsymbol{q_f}, \boldsymbol{\omega_0}) \in \mathbb{R}^{2N_J}$.

As in Section 5.1, we denote the trajectory variables of the ith joint as a^i, ${\omega_m}^i$, ${t_1}^i$, ${t_2}^i$, and ${t_f}^i$, which are linked by Eqs. (22)–(24).

For the ith joint, we select ${\omega_m}^i$, and ${t_f}^i$ as the independent variables. We can then determine a^i, ${t_1}^i$ and ${t_2}^i$:

$$a^i = \frac{({\omega_m}^i)^2 + ({\omega_0}^i)^2/2 - {\omega_0}^i{\omega_m}^i}{{\omega_m}^i{t_f}^i - {q_f}^i}, \tag{29}$$

$${t_1}^i = \frac{{\omega_m}^i - {\omega_0}^i}{a^i}, \text{and} \tag{30}$$

$${t_2}^i = {t_f}^i - \frac{{\omega_m}^i}{a^i}. \tag{31}$$

For each joint, the motion time is the same:

$${t_f}^1 = {t_f}^2 = \ldots = {t_f}^{N_J} = t_f. \tag{32}$$

This means the motion of different joints is coupled with the shared motion time, which is known as the synchronization time.

Thus, the joint-coupled optimization parameter is

$$\boldsymbol{C} = ({\omega_m}^1, \ldots, {\omega_m}^{N_J}, t_f) \in \mathbb{R}^{N_J+1}. \tag{33}$$

5.2.2. Objective function

According to the HRI application requirements described in Section 3.1, the robot is expected to work smoothly by keeping the acceleration as small as possible to ensure the human's safety. However, the robot is required to react to user behavior promptly to ensure a brief motion time. Unfortunately, there is a conflict between safety and rapidity of motion. Therefore, we make a compromise by weighing the two aspects, defining the objective function as

$$F(\boldsymbol{C}) = \sum_{i=1}^{N_J} \alpha^i \left(\frac{a^i}{{a_{max}}^i}\right)^2 + \alpha^{N_J+1}\left(\frac{t_f}{t_{max}}\right)^2, \tag{34}$$

where α^i is the weighing coefficient representing the weighting of acceleration and motion time, which satisfies:

$$\alpha^i > 0, \text{and} \tag{35}$$

$$\sum_{i=1}^{N_J+1} \alpha^i = 1. \tag{36}$$

We can select different combinations of α^i according to the demands on motion efficiency and safety of certain applications. For example, for an application in which a robot is expected to reach the goal as soon as possible, a relatively larger α^{N_J+1} should be selected. Otherwise, in an application highly demanding on safety, such as HRI, we select a relatively smaller α^{N_J+1}.

5.2.3. Constraints

We construct constraints according to the mechanical limitations of the robot and the intrinsic properties of the trapezoidal profile.

According to Eqs. (17)–(21), we have

$$0 < a^i \leq {a_{max}}^i \tag{37}$$

$$\max(0, {\omega_0}^i) \le {\omega_m}^i \le {\omega_{max}}^i \tag{38}$$

$$0 \le {t_1}^i \le {t_2}^i, \tag{39}$$

$${t_1}^i \le {t_2}^i \le t_f, \text{ and} \tag{40}$$

$$0 \le t_f \le t_{max}. \tag{41}$$

According to Eqs. (30) and (31), ${t_1}^i \ge 0$ and ${t_2}^i \le t_f$ are always true. Also, ${t_1}^i \le {t_2}^i$ is equivalent to

$$2{\omega_m}^i - {\omega_0}^i - a^i t_f \le 0. \tag{42}$$

According to Eq. (29), $a^i > 0$ is equivalent to

$${q_f}^i - {\omega_m}^i t_f < 0. \tag{43}$$

5.2.4. Joint-coupled optimization problem

In summary, we represent the coupled optimization problem as follows.

Optimization parameter and the domain:

$$\boldsymbol{C} = ({\omega_m}^1, \ldots, {\omega_m}^{N_J}, t_f) \in \mathbb{R}^{N_J+1}, \tag{44a}$$

$${\omega_m}^i \in [\max(0, {\omega_0}^i), {\omega_{max}}^i], \text{ and} \tag{44b}$$

$$t_f \in [0, t_{max}]. \tag{44c}$$

Objective function:

$$F(\boldsymbol{C}) = \sum_{i=1}^{N_J} \alpha^i \left(\frac{a^i}{{a_{max}}^i}\right)^2 + \alpha^{N_J+1}\left(\frac{t_f}{t_{max}}\right)^2. \tag{44d}$$

Inequality constraints:

$${q_f}^i - {\omega_m}^i t_f < 0, \tag{44e}$$

$$a^i - {a_{max}}^i \le 0, \text{ and} \tag{44f}$$

$$2{\omega_m}^i - {\omega_0}^i - a^i t_f \le 0, \tag{44g}$$

where

$$a^i = \frac{({\omega_m}^i)^2 + ({\omega_0}^i)^2/2 - {\omega_0}^i {\omega_m}^i}{{\omega_m}^i t_f - {q_f}^i}. \tag{44h}$$

5.3. Joint-coupled problem transformation

For the coupled optimization problem (Eq. (44)), we first classify the optimization parameter according to Section 4.1. Using Eq. (32), we select t_f as the coupling parameter while ${\omega_m}^i$ as the joint-related parameters, which yields

$$C_{cpl} = t_f \in \mathbb{R}, \text{ and} \tag{45}$$

$$C_J^i = {\omega_m}^i \in \mathbb{R}. \tag{46}$$

We then decouple the joint-coupled problem as described in Section 4.2. We first need to determine the coupling parameter. In this case, we must select a synchronization time, denoted as t_{syn}, as detailed in Section 5.4. Once t_{syn} is determined, we can calculate the remaining optimization parameters for specific joints independently. The independent optimization problem for the ith joint is:

$$({q_f}^i, {\omega_0}^i, t_{syn}) \rightarrow {\omega_m}^i = {O_{ind}}^i({\omega_m}^i), i = 1, \ldots, N_J. \tag{47}$$

The formulas and the corresponding solutions for feasible and infeasible joints are detailed in Sections 5.5.1 and 5.5.2, respectively.

5.4. Synchronization time determination

5.4.1. Overview

From Section 4.2.2, the procedure for determining t_{syn} is:

1. Determine the feasible range of motion time with respect to each joint.
   The motion time of the ith joint is represented as ${t_f}^i$. The feasible range of ${t_f}^i$, denoted as $S_{{t_f}^i}$, is determined by considering all of the constraints of joint i.
2. Determine feasible joints and infeasible joints.
   A joint that satisfies $S_{{t_f}^i} = \emptyset$ is an infeasible joint. We remove infeasible joints in the coupled optimization problem and change the value of N_J accordingly.
3. Determine feasible range of motion time with respect to all of the joints.
   Considering all of the joints, the feasible range of t_f is determined as
   $$S_{t_f} = \cap_{i=1}^{N_J} S_{{t_f}^i}. \tag{48}$$
4. Learn a near-optimal synchronization time
   We select a near-optimal synchronization time t_{syn} in $S_{{t_f}^i}$ by machine learning.

Steps 1–3 are detailed in Section 5.4.2, and Step 4 is detailed in Section 5.4.3.

5.4.2. Feasible range determination

First, for the ith joint, we determine the feasible range of the motion time, denoted as ${t_f}^i$.

The constrains for $({\omega_m}^i, {t_f}^i)$ are given by

$$0 \le {t_f}^i \le {t_{max}}, \tag{49a}$$

$$max(0, {\omega_0}^i) \le {\omega_m}^i \le {\omega_{max}}^i, \tag{49b}$$

$$0 < a^i \le {a_{max}}^i, \text{ and} \tag{49c}$$

$$2{\omega_m}^i - {\omega_0}^i - a^i {t_f}^i \le 0 \ ({t_1}^i \le {t_2}^i). \tag{49d}$$

The feasible range of ${t_f}^i$ is determined according to Constraints (49a)–(49d). We first consider Constraints (49c) and (49d) to estimate the shape of the feasible range of $({\omega_m}^i, {t_f}^i)$.

Eq. (49c) is equivalent to

$${t_f}^i \ge \frac{1}{{a_{max}}^i}\left\{{\omega_m}^i - {\omega_0}^i + \left[\frac{({\omega_0}^i)^2}{2} + {a_{max}}^i {q_f}^i\right]({\omega_m}^i)^{-1}\right\}. \tag{50}$$

Eq. (49d) is equivalent to

$${t_f}^i \le 2{q_f}^i \frac{2{\omega_m}^i - {\omega_0}^i}{2({\omega_m}^i)^2 - ({\omega_0}^i)^2}. \tag{51}$$

We now define

$${f_1}^i({\omega_m}^i) = \frac{1}{{a_{max}}^i}\left\{{\omega_m}^i - {\omega_0}^i + \left[\frac{({\omega_0}^i)^2}{2} + {a_{max}}^i {q_f}^i\right]({\omega_m}^i)^{-1}\right\}, \text{and} \tag{52}$$

$${f_2}^i({\omega_m}^i) = 2{q_f}^i \frac{2{\omega_m}^i - {\omega_0}^i}{2({\omega_m}^i)^2 - ({\omega_0}^i)^2}. \tag{53}$$

The monotonicity of ${f_1}^i({\omega_m}^i)$ and ${f_2}^i({\omega_m}^i)$ is determined by calculating the derivatives. ${f_1}^i({\omega_m}^i)$ is decreasing when ${\omega_m}^i < \sqrt{\frac{({\omega_0}^i)^2}{2} + {a_{\max}}^i {q_f}^i}$ and increasing when ${\omega_m}^i > \sqrt{\frac{({\omega_0}^i)^2}{2} + {a_{\max}}^i {q_f}^i}$. ${f_2}^i({\omega_m}^i)$ is decreasing. The detailed derivations are shown in Appendix A.

Then we examine if ${f_1}^i({\omega_m}^i)$ and ${f_2}^i({\omega_m}^i)$ intersect. If $\exists {\omega_1}^i$ satisfying ${f_1}^i({\omega_1}^i) = {f_2}^i({\omega_1}^i)$, we have

$$({\omega_1}^i)^2 = \frac{({\omega_0}^i)^2}{2} + {a_{max}}^i {q_f}^i. \tag{54}$$

Since ${\omega_1}^i > 0$, we obtain the intersection point

$${\omega_1}^i = \sqrt{\frac{({\omega_0}^i)^2}{2} + {a_{max}}^i {q_f}^i}. \tag{55}$$

Thus, when $\max(0, {\omega_0}^i) \le {\omega_m}^i < {\omega_1}^i$, we have ${f_1}^i({\omega_1}^i) < {f_2}^i({\omega_1}^i)$. Otherwise, when ${\omega_m}^i > {\omega_1}^i$, we have ${f_1}^i({\omega_1}^i) > {f_2}^i({\omega_1}^i)$. The shape of the feasible set of $({\omega_m}^i, {t_f}^i)$ that satisfies (49c) and (49d) is shown as Fig. 6.

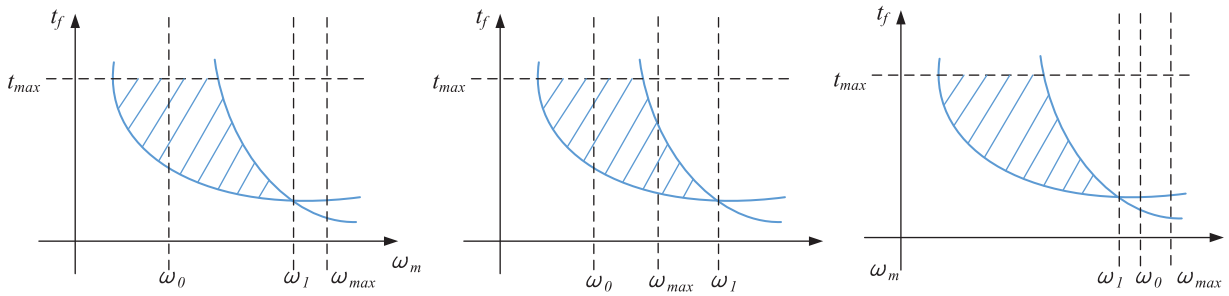


Fig. 6. Feasible range of each joint.

Table 1
Feasible range of ${t_f}^i$ with different ${\omega_1}^i$.

${\omega_1}^i$	${t_{lb}}^i$	${t_{ub}}^i$
${\omega_1}^i \geq {\omega_{\max}}^i$	$\frac{{q_f}^i}{{\omega_{\max}}^i} + \frac{1}{{a_{\max}}^i}\left[{\omega_{\max}}^i - {\omega_0}^i + \frac{({\omega_0}^i)^2}{2{\omega_{\max}}^i}\right]$	$\min\left(t_{\max}, \frac{2{q_f}^i}{{\omega_0}^i}\right)$
${\omega_0}^i \leq {\omega_1}^i < {\omega_{\max}}^i$	$\frac{1}{{a_{\max}}^i}\left(2\sqrt{\frac{({\omega_0}^i)^2}{2} + {a_{\max}}^i{q_f}^i} - {\omega_0}^i\right)$	$\min\left(t_{\max}, \frac{2{q_f}^i}{{\omega_0}^i}\right)$
${\omega_1}^i < {\omega_0}^i$	–	–

Then we consider the constraints (49a) and (49b) to determine the feasible range of ${t_f}^i$. First, we select an upper bound $t_{\max}$ to ensure that the feasible set for all of the joints is non-empty:

$$t_{max} \geq \max_i({f_1}^i({\omega_1}^i)). \tag{56}$$

According to Eq. (55), ${\omega_1}^i$ depends on ${\omega_0}^i$ and ${q_f}^i$. There are three cases depending on the value of ${\omega_m}^i$, as shown in Fig. 6.

In the cases of ${\omega_1}^i \geq {\omega_{\max}}^i$ (Fig. 6a) and ${\omega_0}^i \leq {\omega_1}^i < {\omega_{\max}}^i$ (Fig. 6b), $S_{{t_f}^i} \neq \emptyset$ and joint i is feasible. The range of ${t_f}^i$ is a continuous set:

$$S_{{t_f}^i} = \{{t_f}^i | {t_{lb}}^i \leq {t_f}^i \leq {t_{ub}}^i\}, \tag{57}$$

in which ${t_{lb}}^i$ and ${t_{ub}}^i$ are shown in Table 1. The solving process is detailed in Appendix B.

In the joint-coupled problem, the lower and upper bounds of t_f are

$$t_{lb} = \max_i({t_{lb}}^i), \text{and} \tag{58}$$

$$t_{ub} = \min_i({t_{ub}}^i). \tag{59}$$

Thus, the feasible range of t_f is

$$S_{t_f} = \cap_{i=1}^{N_J} S_{{t_f}^i} = \{t_f | t_{lb} \leq t_f \leq t_{ub}\}. \tag{60}$$

In the case of ${\omega_1}^i < {\omega_0}^i$ (Fig. 6c), the feasible set of ${t_f}^i$ is empty. In this situation, we exclude this joint from the coupling problem and change the value of N_J accordingly. We select the coupling parameter t_f considering the rest of joints and calculate the parameters of the infeasible DOFs independently. The motion time of this joint may be different from the synchronization time t_{syn}. We discuss the details in Section 5.5.2.

5.4.3. Near-optimal synchronization time selection

We select a near-optimal synchronization time using machine learning. The detailed procedure is as follows.

1. Generate the database.
   We select N_D groups of input variables ${}^i\boldsymbol{X}(i = 1, 2, \ldots, N_D)$ and solve the coupled optimization problem

   $${}^i\boldsymbol{C} = O_{cpl}({}^i\boldsymbol{X}). \tag{61}$$

   We then construct the optimal database:

   $$D = \{{}^i\boldsymbol{X}, {}^i t_{f\_opt} | i = 1, 2, \ldots, N_D\}. \tag{62}$$

2. Establish the regression model.
   We establish the approximating mapping from input variable to the optimal motion time using

$$\boldsymbol{X} \rightarrow t_f = R(\boldsymbol{X}). \tag{63}$$

3. Learn an approximate coupling parameter.
   According to the new input variable $\boldsymbol{X'}$, we obtain an initial t_f with the regression model in Step 2:

$$t'_f = R(\boldsymbol{X'}). \tag{64}$$

4. Refine the initial parameter to obtain a near-optimal synchronization time.
   If the initial synchronization time obtained in (3) does not satisfy the constraints, we refine it to make sure it is in S_{t_f}.
   If $t_{lb} \le t'_f \le t_{ub}$,

$$t_{syn} = t'_f. \tag{65}$$

   If $t'_f < t_{lb}$,

$$t_{syn} = t_{lb}. \tag{66}$$

   If $t'_f > t_{ub}$,

$$t_{syn} = t_{ub}. \tag{67}$$

5.5. Solution for joint-independent optimization problem

5.5.1. Solution for feasible joints

With the synchronized time t_{syn}, the joint-coupled optimization problem is transformed into N_j joint-independent problems.

To simplify the notation, we omit the superscript that represents the joint number.

For each joint, the independent optimization problems is

$$(q_f, \omega_0, t_{syn}) \rightarrow \omega_m = O_{ind}(\omega_m). \tag{68a}$$

The objective function is

$$F(\omega_m) = a = \frac{{\omega_m}^2 + {\omega_0}^2/2 - \omega_0\omega_m}{\omega_m t_{syn} - q_f}. \tag{68b}$$

The inequivalent constraints are

$$\max(0, \omega_0) \le \omega_m \le \omega_{max}, \tag{68c}$$

$$q_f - \omega_m t_{syn} < 0 \ \ (a > 0), \tag{68d}$$

$$a - a_{max} \le 0, \ \text{and} \tag{68e}$$

$$2\omega_m - \omega_0 - a t_{syn} \le 0 \ \ (t_1 \le t_2). \tag{68f}$$

By analyzing the derivation, we can determine $F(\omega_m)$ is non-increasing in the domain (Appendix C). Thus, we obtain the solution of the optimization problem:

$$argminF(\omega_m) = \min(\omega_3, \omega_{max}). \tag{69}$$

5.5.2. Solution for infeasible joints

As there is no feasible solution for infeasible joints, we need to loosen some constraints to reach an acceptable solution.

According to the trapezoidal profile property, Constraint (49d) is a hard constraint which cannot be violated. Thus, when $\omega_m > \omega_1$, (49c) is not satisfied, which means that the acceleration will exceed the maximal allowed value. Additionally, according to Section 5.4.2, when $\omega_m > \sqrt{\frac{(\omega_0)^2}{2} + a_{\max} q_f}$, $f_1(\omega_m)$ is increasing, and $f_2(\omega_m)$ is decreasing. Thus, as ω_m increases, the violation becomes larger.

To avoid violating the mechanical limitations and damaging the robot, we choose to loosen the equality Constraint (22), which will introduce a small error of q_f. Also, this situation indicates a relatively large ω_0 and small q_f, which always happens when the robot is close to the target and needs to decelerate. Thus, we choose

$$\omega_m = \omega_0, \ \text{and} \tag{70}$$

$$t_1 = t_2 = 0. \tag{71}$$

Then we adjust the trapezoidal profile described in Section 5.1. The remaining two variables a and t_f are linked by the equality constraint

$$t_f = \frac{\omega_0}{a} \tag{72}$$

and constrained by Eqs. (17) and (21).

The objective is to make the position violation as small as possible

$$\min\left[\frac{1}{2}\omega_m(t_f + t_2 - t_1) + \frac{1}{2}\omega_0 t_1 - q_f\right]. \tag{73}$$

By selecting a as the optimization parameter, we construct the independent optimization problem as follows:

Objective function:

$$F(a) = \min\left[\frac{(\omega_0)^2}{2a} - q_f\right] \tag{74a}$$

Inequality constraints:

$$0 < a \le a_{max}, \text{ and} \tag{74b}$$

$$0 < t_f \le t_{max} \tag{74c}$$

The optimal solution is

$$a = a_{max}, \text{ and} \tag{75}$$

$$t_f = \frac{\omega_0}{a_{max}}. \tag{76}$$

6. Results

6.1. Coupled optimization and decoupled optimization

We implemented our algorithm in Matlab on a laptop equipped an Intel Core i7-6700HQ 2.6 GHz CPU with 16 GB RAM. We adopted the joint-decoupling method presented here to solve the optimization-based trajectory planning problem and compared it with the coupled optimization method (with arbitrary initializations and learned initializations) by evaluating the online calculation time, feasibility, and accuracy. First, we solved the coupled optimization problems by the sequence quadratic program (SQP) method with arbitrary initializations [14]. Then, we adopted the 5-Nearest Neighbor (5-NN) method with a 1000-sample optimal database D1 to learn an initial guess for the SQP solver and solve the coupled problems [26]. Finally, we solved the optimization problems by our decoupling method, determining the coupling parameter with the same database D1.

We first evaluated the online calculation time by solving 1000 optimization problems with different DOF dimensions using the three methods mentioned above. The average calculation times are shown in Fig. 7. The calculation time of the coupled optimization problem increased in a super linear trend as the number of DOFs increases. Learning a good initial value sped up the solving process of coupled optimization problems, but the time reduction was limited. In contrast, by transforming the coupled optimization problem into multiple independent optimization problems with fewer parameters, the calculation time was reduced by more than an order of magnitude with respect to the coupled problems.

Then we evaluated the feasibility by replanning trajectories for a 6-DOF robot according to the movement of a hand reaching a target. The success rate is shown in Fig. 8. The iterative solver starting from randomly selected initial values failed to find a feasible solution in many cases. Selecting a good initial value by learning was helpful, but still did not guarantee a feasible solution. Using our decoupling method, we always obtained a feasible solution, indicating that this algorithm had better solution flexibility.

However, achieving a better success rate and faster solution times came at a price: some optimality was sacrificed. We got only near optimal solutions rather than the accurate one. We calculated the deviation between the solutions obtained by the presented method and the accurate optimal solution:

$$e_F = \left|\frac{F^* - F_{opt}}{F_{opt}}\right|, \tag{77}$$

$$e_C{}^i = \left|\frac{C^{*i} - C_{opt}{}^i}{C_{opt}{}^i}\right|, \tag{78}$$

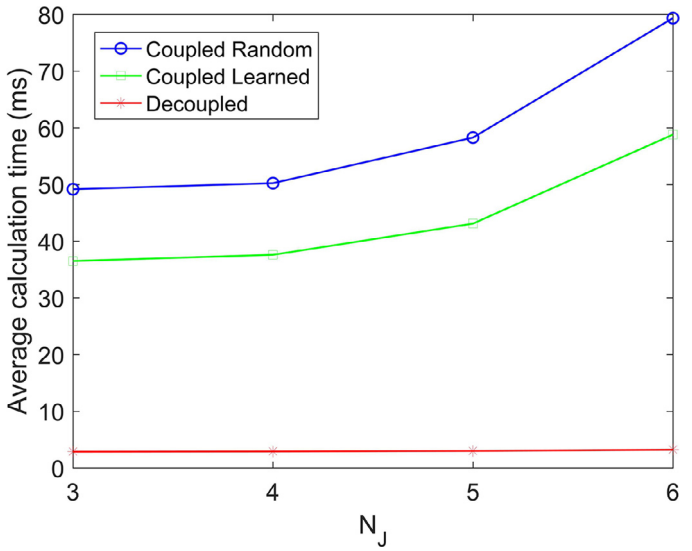


Fig. 7. Online calculation time for different solving methods and different DOFs.

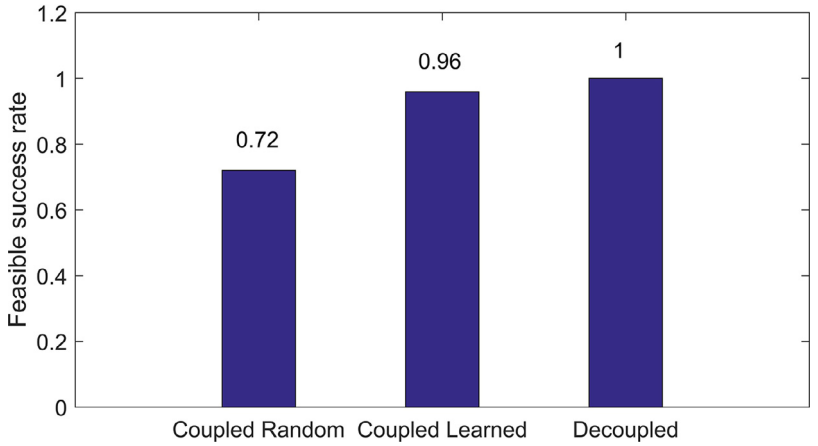


Fig. 8. Feasibility with different solving methods.

Table 2
Cost increase and parameter error of the joint-decoupling method.

e_F	$\boldsymbol{e_C}$
0.1758	0.1694 0.2579 0.1649 0.1652 0.2518 0.2134 0.1717

where $\boldsymbol{C_{opt}}$ and F_{opt} are the accurate optimal parameter and the corresponding objective function value, respectively, while $\boldsymbol{C}^*$ and F^* are the near optimal solution obtained by the joint-decoupled method.

As shown in Table 2, our method showed a 17.58% increase in costs and around 20% average error in the parameters. While the obtained solution was suboptimal, it was still feasible, which satisfied all the hard constraints. Thus, an acceptable solution can always be obtained, as long as we construct appropriate objective function and constraints according to the requirements. For example, if some indices are required to be strictly within the certain range, we encode them in hard constraints. While if some indices are expected to be higher or lower, we encode them in the objective function. Then a suboptimal solution will have lower performance, but still will be an acceptable solution.

6.2. Learning synchronization time

For the case when $N_J = 6$, we selected $\boldsymbol{q_0} = (1.5708, 0, 0.2618, 0, -0.2618, 0)$ (rad) and $\boldsymbol{\omega_0} = (0, 0, 0, 0, 0, 0)$ (rad/s) as the initial state of the whole process. The first objective configuration was $\boldsymbol{q_c} = (1.8466, 1.3631, -0.4817,$

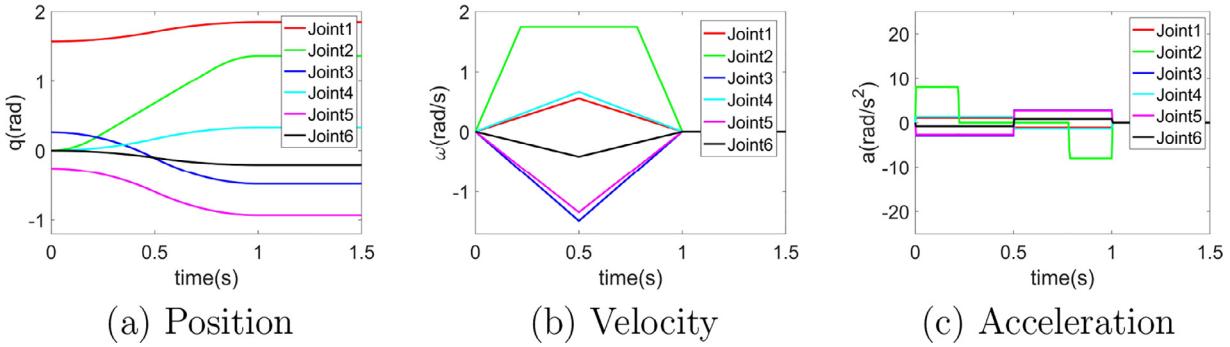


Fig. 9. Trajectory with the learned t_{syn}.

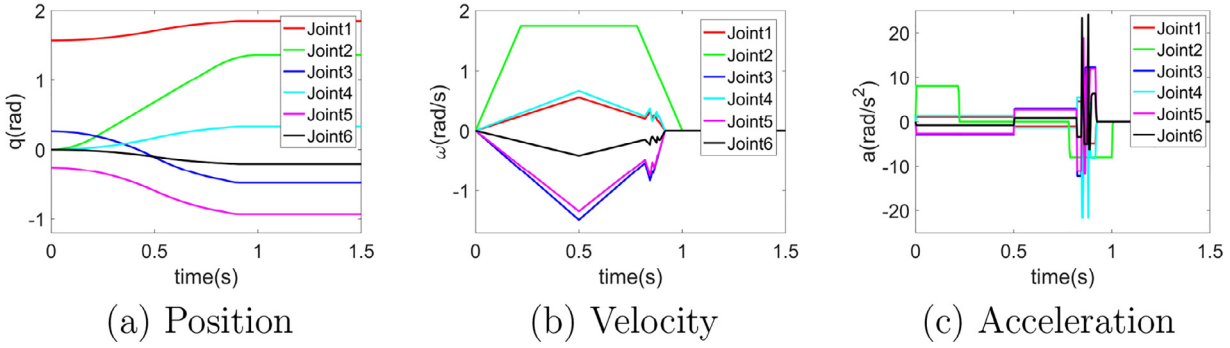


Fig. 10. Trajectory with the lowest t_{syn}.

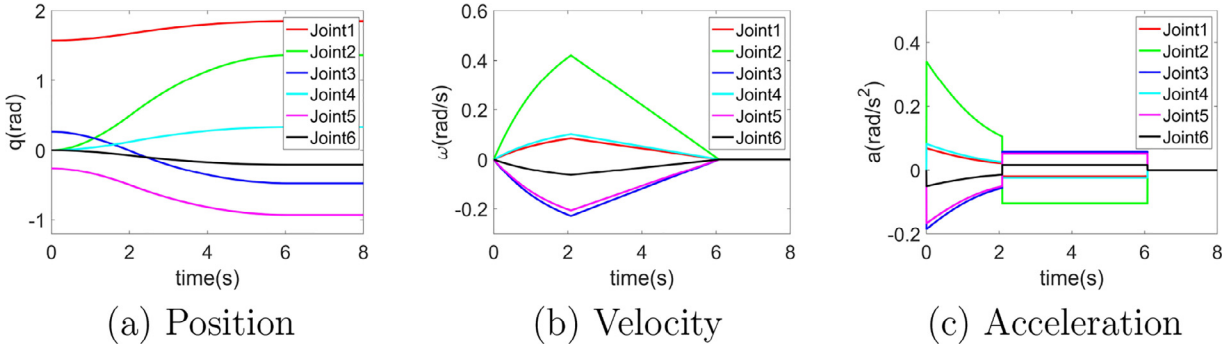


Fig. 11. Trajectory with the largest t_{syn}.

0.3299, −0.9338, −0.2077) (rad) with an update period of $T_p = 4ms$. Employing the 5-NN method with database D1, we learned the synchronization time. The trajectory of the whole process is shown in Fig. 9.

We compared this method with other synchronization time selection methods. We selected the synchronization time first as the lower bound and then as the upper bound. The corresponding trajectories are shown as Figs. 10 and 11.

When selecting the lower bound as the synchronization time, the motion of the robot was not very smooth, especially in the latter half. The velocity and acceleration did not change smoothly. The acceleration changed directions and tended to be very large in many time instants, which would be harmful to the robot structure and weaken HRI safety. Moreover, joint 2 did not synchronize with other joints.

When selecting the upper bound as the synchronization time, the motion time of the whole process was about six times as long as the learning synchronization time.

In contrast, by selecting the learned synchronization time, smooth motion with moderate acceleration was achieved. Moreover, since the motion time was much smaller than the upper bound, motion rapidity could also be guaranteed.

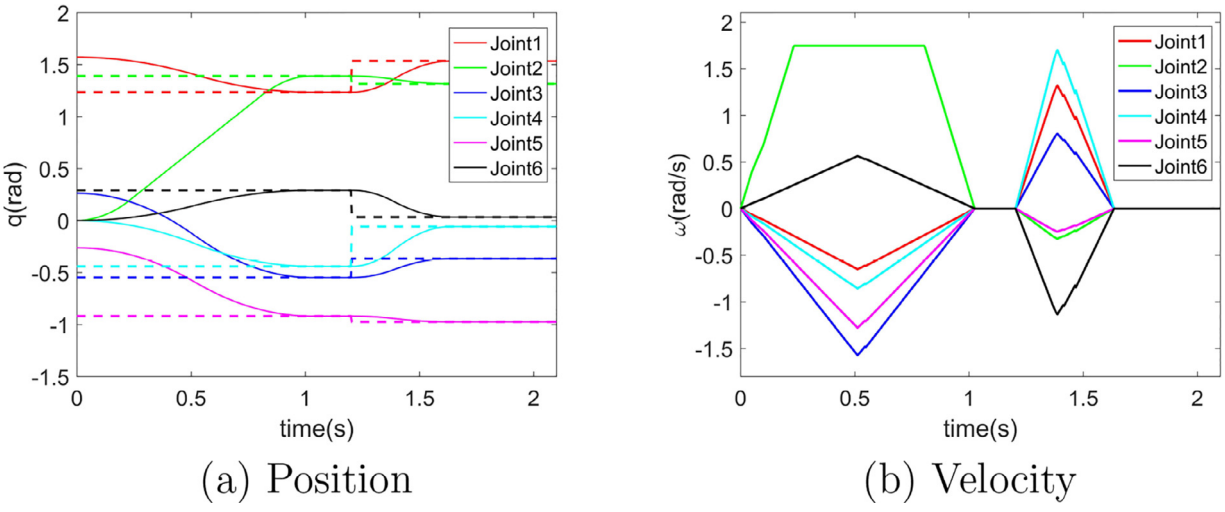


Fig. 12. Robot motion curve with target changing in the static state.

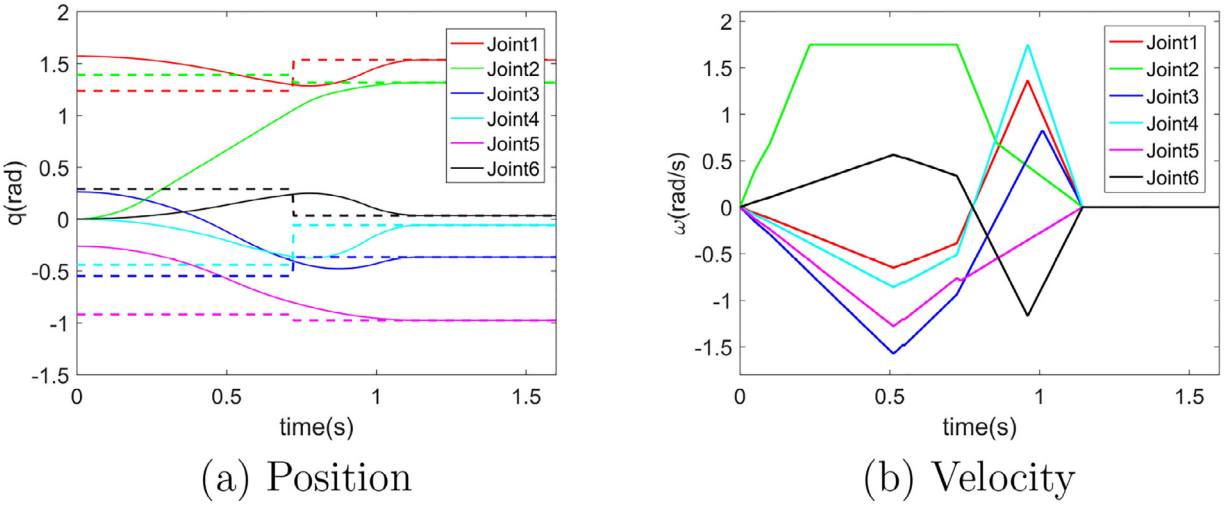


Fig. 13. Robot motion curve with target changing in the middle of movement.

6.3. Real-time trajectory planning with the changing goal

In HRI applications, the robot must react to the changing goal promptly. To verify the ability of the algorithm to deal with a changing goal, we studied two cases.

In the first case, the target changed after the robot reached the first goal. The trajectory was re-planned from the stand-still state. The position and velocity curves in joint space of the whole process are shown in Fig. 12. In Fig. 12 a, the target position and the measured position are represented by dashed and solid lines, respectively.

In the second case, the target changed while the robot was moving to the first goal. Then the trajectory was re-planned from the current state, as shown in Fig. 13.

In both cases, the robot reacted to the changing targets promptly and finally reached the actual goals.

6.4. Human robot interaction experiment

Considering the HRI case described in Section 3.1, the user moved a hand in the workspace to perform some manipulations in the interaction area. The system was required to predict the interaction point and move the end-effector there.

The interaction area was a panel perpendicular to the Y-axis of the base frame as shown in Fig. 14. As the hand moved, we detected the wrist position and selected the interaction point as the point in the panel which had shortest distance to the wrist position in the current state.

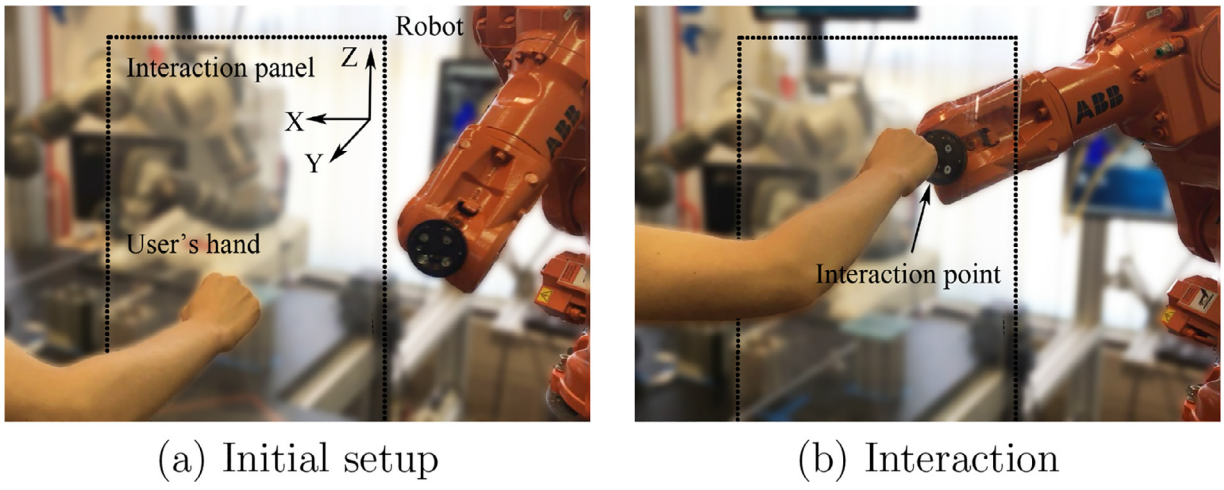


Fig. 14. Experiment setup.

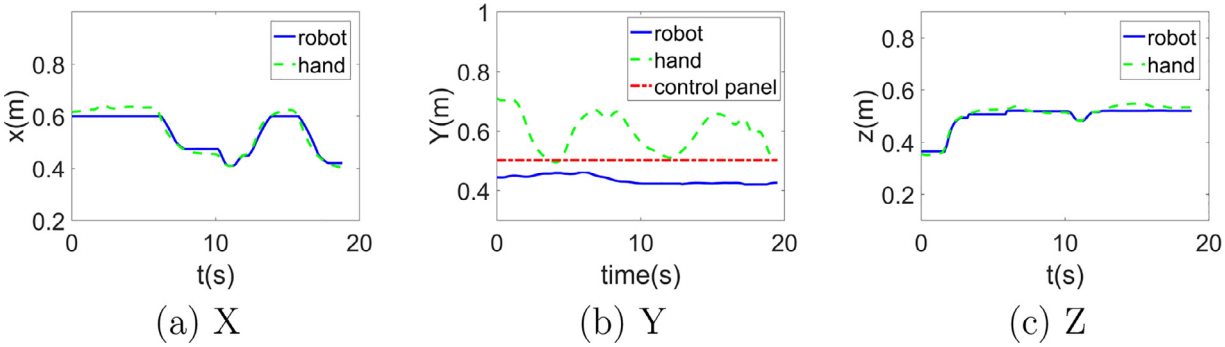


Fig. 15. Task space trajectory in HRI case.

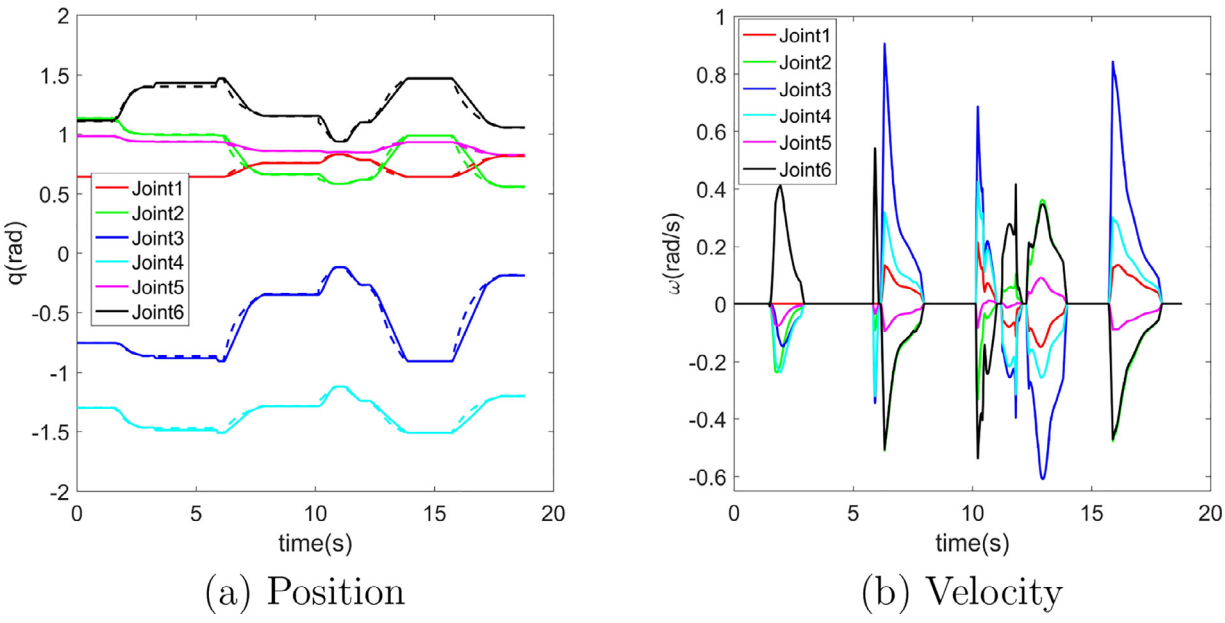


Fig. 16. Joint space trajectory in HRI case.

We converted the algorithm into C code and ran it on a real-time system. The system consisted of an ABB IRB140 6-DOF robot [37], a Microsoft Kinect as the human behavior detection sensor, and a Linux real-time PC used to gather information from all of the devices and calculate the trajectory. The sampling and planning period was $T_p = 4ms$.

When the hand moved, the predicted interaction point kept changing while the trajectory of the robot was re-calculated in real-time accordingly. The position of the end-effector and user's hand in task space are shown in Fig. 15, which indicates that the robot tracked the goal rapidly and precisely. Joint position and velocity are shown in Fig. 16. Smooth motion was achieved throughout the process.

7. Conclusion

In this paper, we present a real-time optimization-based trajectory planning method for HRI along with an HRI case study. Our joint-decoupled optimization method transforms a high-dimensional coupled optimization problem into several lower-dimensional, independent optimization problems. We validated our method in a HRI case study using a 6-DOF robot. The computational efficiency, feasibility, and flexibility of the solution were greatly improved. Not only was the calculation time reduced by more than an order of magnitude, our approach was able to return an acceptable solution even when the coupled problem approach could not. Also, our solver scales easily for problems with different numbers of joints. Stable online operation of the HRI case was achieved in a 250 Hz real-time system.

We have placed the emphasis on the implementation of this method in a specific case, but we believe this is a general method. Thus, we would like to work on the implementation for more specific applications and optimization models. We also would like to exploit the scalability potential of this method to extend it to higher dimensional situations, such as multiple robot systems.

Appendix A. Monotonicity of feasible range boundary

In order to determine the shape of the feasible range, we calculate the derivatives of ${f_1}^i({\omega_m}^i)$ and ${f_2}^i({\omega_m}^i)$ and get the monotonicity.

The derivatives of ${f_1}^i({\omega_m}^i)$ and ${f_2}^i({\omega_m}^i)$ are:

$$\frac{d{f_1}^i}{d{\omega_m}^i} = \frac{({\omega_m}^i)^2 - [({\omega_0}^i)^2/2 + {a_{max}}^i {q_f}^i]}{{a_{max}}^i({\omega_m}^i)^2}, \text{ and} \tag{A.1}$$

$$\frac{d{f_2}^i}{d{\omega_m}^i} = -{q_f}^i \frac{({\omega_m}^i - {\omega_0}^i)^2 + ({\omega_m}^i)^2}{({\omega_m}^i - {\omega_0}^i/2)^2}. \tag{A.2}$$

Since ${\omega_m}^i > 0$, when

$${\omega_m}^i < \sqrt{\frac{({\omega_0}^i)^2}{2} + {a_{max}}^i {q_f}^i}, \tag{A.3}$$

we have

$$\frac{d{f_1}^i}{d{\omega_m}^i} < 0. \tag{A.4}$$

Otherwise when

$${\omega_m}^i > \sqrt{\frac{({\omega_0}^i)^2}{2} + {a_{max}}^i {q_f}^i}, \tag{A.5}$$

we have

$$\frac{d{f_1}^i}{d{\omega_m}^i} > 0. \tag{A.6}$$

Thus, when ${\omega_m}^i < \sqrt{\frac{({\omega_0}^i)^2}{2} + {a_{\max}}^i {q_f}^i}$, ${f_1}^i({\omega_m}^i)$ is decreasing. When ${\omega_m}^i > \sqrt{\frac{({\omega_0}^i)^2}{2} + {a_{\max}}^i {q_f}^i}$, ${f_1}^i({\omega_m}^i)$ is increasing.

Further,

$$\frac{d{f_2}^i}{d{\omega_m}^i} < 0 \tag{A.7}$$

is always true.

Thus, ${f_2}^i({\omega_m}^i)$ is decreasing.

Appendix B. Motion time range of feasible joints

We calculate the upper and lower bound of $t_f{}^i$ for the feasible cases that ${\omega_1}^i$ locates in different ranges (shown in Fig. 6a and 6b).

(1) ${\omega_1}^i \geq {\omega_{\max}}^i$

As shown in Fig. 6a, when

$${\omega_m}^i = {\omega_{max}}^i, \tag{B.1}$$

$$a^i = {a_{max}}^i, \tag{B.2}$$

we obtain the lower bound of $t_f{}^i$ as

$${t_{lb}}^i = \frac{{q_f}^i}{{\omega_{max}}^i} + \frac{1}{{a_{max}}^i}\left[{\omega_{max}}^i - {\omega_0}^i + \frac{({\omega_0}^i)^2}{2{\omega_{max}}^i}\right]. \tag{B.3}$$

When

$${\omega_m}^i = {\omega_0}^i, \text{ and} \tag{B.4}$$

$${t_1}^i = {t_2}^i, \tag{B.5}$$

we have

$${t_f}^i = \frac{2{q_f}^i}{{\omega_0}^i}. \tag{B.6}$$

Then the upper bound is

$${t_{ub}}^i = \min\left(t_{max}, \frac{2{q_f}^i}{{\omega_0}^i}\right). \tag{B.7}$$

(2) ${\omega_0}^i \leq {\omega_1}^i < {\omega_{\max}}^i$

As shown in Fig. 6b, when

$${t_1}^i = {t_2}^i, \text{ and} \tag{B.8}$$

$$a^i = {a_{max}}^i, \tag{B.9}$$

we obtain the lower bound of $t_f{}^i$ as

$${t_{lb}}^i = \frac{1}{{a_{max}}^i}\left(2\sqrt{\frac{({\omega_0}^i)^2}{2} + {a_{max}}^i{q_f}^i} - {\omega_0}^i\right). \tag{B.10}$$

When

$${\omega_m}^i = {\omega_0}^i, \text{ and} \tag{B.11}$$

$${t_1}^i = {t_2}^i, \tag{B.12}$$

we have

$${t_f}^i = \frac{2{q_f}^i}{{\omega_0}^i}, \tag{B.13}$$

Then the upper bound is

$${t_{ub}}^i = \min\left(t_{max}, \frac{2{q_f}^i}{{\omega_0}^i}\right). \tag{B.14}$$

Appendix C. Solution for feasible joint-independent optimization

We calculate the monotonicity of the objective function F and get the solution for the feasible joint-independent optimization problem.

First, we calculate the first derivative of F:

$$\frac{dF}{d\omega_m} = \frac{t_{syn}{\omega_m}^2 - 2q_f\omega_m + \omega_0 q_f - {\omega_0}^2 t_{syn}/2}{(\omega_m t_{syn} - q_f)^2}. \tag{C.1}$$

Then we analyze the derivative to determine the shape of F. We define

$$A = t_{syn}{\omega_m}^2 - 2q_f\omega_m + \omega_0 q_f - {\omega_0}^2 t_{syn}/2, \text{ and} \tag{C.2}$$

$$B = (\omega_m t_{syn} - q_f)^2. \tag{C.3}$$

When $a > 0$, we have $B > 0$.
A is a quadratic function, and

$$(2q_f)^2 - 4t_{syn}\left(\omega_0 q_f - \frac{{\omega_0}^2 t_{syn}}{2}\right) = (2q_f - \omega_0 t_{syn})^2 + (\omega_0 t_{syn})^2 > 0. \tag{C.4}$$

Thus, A has two zero points. Then F has two inflection points, represented as $\omega_m = {\omega_{in}}^1$ and $\omega_m = {\omega_{in}}^2$, shown as Fig. C17.

After determining the shape of F, we need to compare the inflection points and the bounds of ω_m to find the minimum. According to Fig. 6, the feasible range of ω_m is

$$\omega_2 < \omega_m < \min(\omega_3, \omega_{max}), \tag{C.5}$$

in which

$$\omega_2 = \omega_m | a = a_{max}, \text{ and} \tag{C.6}$$

$$\omega_3 = \omega_m | t_1 = t_2. \tag{C.7}$$

According to Eq. (C.6), when $\omega_m = \omega_2$, we have $a = a_{\max}$. Thus, we obtain

$$a(\omega_2 + \triangle\omega) < a(\omega_2) = a_{max}. \tag{C.8}$$

Consequently, we obtain the range of ω_2:

$${\omega_{in}}^1 \le \omega_2 \le {\omega_{in}}^2. \tag{C.9}$$

According to Eq. (C.7), when $\omega_m = \omega_3$, we have $t_1 = t_2$. Then we obtain

$$a = \frac{2\omega_3 - \omega_0}{t_{syn}}. \tag{C.10}$$

Further, according to Eq. (44h), we have

$$a = \frac{{\omega_3}^2 + {\omega_0}^2/2 - \omega_0\omega_3}{\omega_3 t_{syn} - q_f}. \tag{C.11}$$

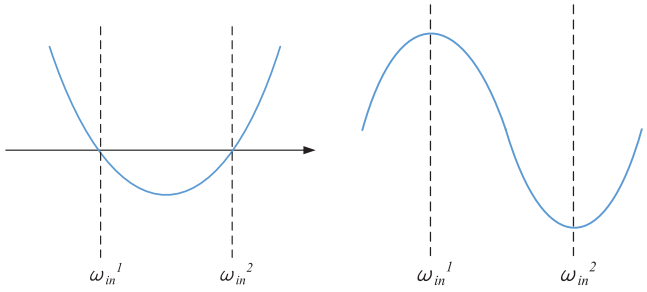
According to Eqs. (C.10) and (C.11), we obtain

$${t_{syn}}{\omega_3}^2 - 2q_f\omega_3 + \omega_0 q_f - \frac{t_{syn}{\omega_0}^2}{2} = 0. \tag{C.12}$$

Then we have

$$\frac{dF}{d\omega_m}\bigg|\omega_3 = 0. \tag{C.13}$$

Therefore, ω_3 is an inflection point.



(a) Zero points of A (b) Inflection points of F

Fig. C1. Objective function curve and inflection points.

Since $\omega_2 \geq \omega_1 \geq {}_{in}{}^1$, when the two equality signs hold at the same time, we have

$$\omega_3 = \omega_2 = \omega_{in}{}^1. \tag{C.14}$$

Otherwise,

$$\omega_3 = \omega_{in}{}^2. \tag{C.15}$$

Thus, $F(\omega_m)$ is non-increasing in the domain. We then obtain the solution of the optimization problem:

$$argminF(\omega_m) = \min(\omega_3, \omega_{max}). \tag{C.16}$$

References

[1] A. Cherubini, R. Passama, A. Crosnier, A. Lasnier, P. Fraisse, Collaborative manufacturing with physical human-robot interaction, Robot. Comput. Integr. Manuf. 40 (2016) 1–13.
[2] H. Liu, L. Wang, Human motion prediction for human-robot collaboration, J. Manuf. Syst. 44 (2017) 287–294.
[3] R.-J. Halme, M. Lanz, J. Kämäräinen, R. Pieters, J. Latokartano, A. Hietanen, Review of vision-based safety systems for human-robot collaboration, Procedia CIRP 72 (1) (2018) 111–116.
[4] A. Ajoudani, A.M. Zanchettin, S. Ivaldi, A. Albu-Schäffer, K. Kosuge, O. Khatib, Progress and prospects of the human–robot collaboration, Autonom. Rob. (2018) 1–19.
[5] M.J. Matarić, J. Eriksson, D.J. Feil-Seifer, C.J. Winstein, Socially assistive robotics for post-stroke rehabilitation, J. Neuro Eng. Rehabil. 4 (1) (2007) 5.
[6] V. Arakelian, S. Ghazaryan, Improvement of balancing accuracy of robotic systems: application to leg orthosis for rehabilitation devices, Mech. Mach. Theory 43 (5) (2008) 565–575.
[7] J.L. Pons, Rehabilitation exoskeletal robotics, IEEE Eng. Med. Biol. Mag. 29 (3) (2010) 57–63.
[8] X. Li, Y. Pan, G. Chen, H. Yu, Adaptive human-robot interaction control for robots driven by series elastic actuators, IEEE Trans. Robot. 33 (1) (2017) 169–182.
[9] J. Kober, M. Glisson, M. Mistry, Playing catch and juggling with a humanoid robot, in: Proceedings of the Humanoids, 2012, pp. 875–881.
[10] Z. Ren, Q. Zhu, R. Xiong, Trajectory planning of 7-DOF humanoid manipulator under rapid and continuous reaction and obstacle avoidance environment, Acta Automatica Sinica 41 (6) (2015) 1131–1144.
[11] L. Marchal-Crespo, M. van Raai, G. Rauter, P. Wolf, R. Riener, The effect of haptic guidance and visual feedback on learning a complex tennis task, Exper. Brain Res. 231 (3) (2013) 277–291.
[12] A. De Santis, B. Siciliano, A. De Luca, A. Bicchi, An atlas of physical human–robot interaction, Mech. Mach. Theory 43 (3) (2008) 253–270.
[13] O. von Stryk, M. Schlemmer, Optimal control of the industrial robot Manutec r3, Comput. Opt. Control (1994) 367–382.
[14] T. Chettibi, H. Lehtihet, M. Haddad, S. Hanchi, Minimum cost trajectory planning for industrial robots, Eur. J. Mech. A Solids 23 (4) (2004) 703–715.
[15] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, S. Schaal, Stomp: Stochastic trajectory optimization for motion planning, in: Proceedings of the ICRA, IEEE, 2011, pp. 4569–4574.
[16] C. Park, J. Pan, D. Manocha, Itomp: Incremental trajectory optimization for real-time replanning in dynamic environments, in: Proceedings of the ICAPS, 2012.
[17] M. Zucker, N. Ratliff, A.D. Dragan, M. Pivtoraiko, M. Klingensmith, C.M. Dellin, J.A. Bagnell, S.S. Srinivasa, Chomp: covariant hamiltonian optimization for motion planning, Int. J. Robot. Res. 32 (9–10) (2013) 1164–1193.
[18] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, P. Abbeel, Motion planning with sequential convex optimization and convex collision checking, Int. J. Robot. Res. 33 (9) (2014) 1251–1270.
[19] J. Huang, P. Hu, K. Wu, M. Zeng, Optimal time-jerk trajectory planning for industrial robots, Mech. Mach. Theory 121 (2018) 530–544.
[20] B. Bäuml, T. Wimböck, G. Hirzinger, Kinematically optimal catching a flying ball with a hand-arm-system, in: Proceedings of the IROS, 2010, pp. 2592–2599.
[21] C. Liu, M. Tomizuka, Algorithmic safety measures for intelligent industrial co-robots, in: Proceedings of the ICRA, 2016, pp. 3095–3102.
[22] A. Cassioli, D. Di Lorenzo, M. Locatelli, F. Schoen, M. Sciandrone, Machine learning for global optimization, Comput. Optim. Appl. 51 (1) (2012) 279–303.
[23] J. Pan, Z. Chen, P. Abbeel, Predicting initialization effectiveness for trajectory optimization, in: Proceedings of the ICRA, 2014, pp. 5183–5190.
[24] A. Werner, D. Trautmann, D. Lee, R. Lampariello, Generalization of optimal motion trajectories for bipedal walking, in: Proc. IROS, 2015, pp. 1571–1577.
[25] S. Zhang, S. Dai, Real-time trajectory generation for haptic feedback manipulators in virtual cockpit systems, J. Comput. Inf. Sci. Eng. 18 (4) (2018) 041015.
[26] R. Lampariello, D. Nguyen-Tuong, C. Castellini, G. Hirzinger, J. Peters, Trajectory planning for optimal robot catching in real-time, in: Proceedings of the ICRA, 2011, pp. 3719–3726.
[27] N. Jetchev, M. Toussaint, Fast motion planning from experience: trajectory prediction for speeding up movement generation, Autonom. Robots 34 (1) (2013) 111–127.
[28] K. Hauser, Learning the problem-optimum map: analysis and application to global optimization in robotics, IEEE Trans. Robot. 33 (1) (2017) 141–152.
[29] T. Kröger, F.M. Wahl, Online trajectory generation: basic concepts for instantaneous reactions to unforeseen events, IEEE Trans. Robot. 26 (1) (2010) 94–111.
[30] C.-S. Tsai, J.-S. Hu, M. Tomizuka, Ensuring safety in human-robot coexistence environment, in: Proceedings of the IROS, 2014, pp. 4191–4196.
[31] C. Liu, C.-Y. Lin, M. Tomizuka, The convex feasible set algorithm for real time optimization in motion planning, SIAM J. Control Optim. 56 (4) (2018) 2712–2733.
[32] D. Sidobre, X. Broquère, J. Mainprice, E. Burattini, A. Finzi, S. Rossi, M. Staffa, Human–Robot Interaction, in: B. Siciliano (Ed.), Advanced Bimanual Manipulation, vol 80, Springer Tracts in Advanced Robotics, 2012, pp. 123–172.
[33] J. Mainprice, R. Hayne, D. Berenson, Goal set inverse optimal control and iterative replanning for predicting human reaching motions in shared workspaces, IEEE Trans. Robot. 32 (4) (2016) 897–908.
[34] S. Pellegrinelli, A. Orlandini, N. Pedrocchi, A. Umbrico, T. Tolio, Motion planning and scheduling for human and industrial-robot collaboration, CIRP Annals 66 (1) (2017) 1–4.
[35] S. Kim, A. Shukla, A. Billard, Catching objects in flight, IEEE Trans. Robot. 30 (5) (2014) 1049–1065.
[36] M.M.G. Ardakani, B. Olofsson, A. Robertsson, R. Johansson, Real-time trajectory generation using model predictive control, in: Proceedings of the CASE, IEEE, 2015, pp. 942–948.
[37] R. Maderna, A. Casalino, A.M. Zanchettin, P. Rocco, Robotic handling of liquids with spilling avoidance: a constraint-based control approach, in: Proceedings of the ICRA, 2018, pp. 7414–7420.
[38] L. Biagiotti, C. Melchiorri, Trajectory Planning for Automatic Machines and Robots, Springer Berlin Heidelberg, 2009.