Continuous trajectory planning based on learning optimization in high dimensional input space for serial manipulators

Shiyu Zhang, Shuling Dai and Yongjia Zhao

State Key Laboratory of Virtual Reality Technology and Systems, Beihang University, Beijing, P.R. China

ABSTRACT
In order to generate trajectories continuously for serial manipulators with high dimensional degrees of freedom (DOFs) in a dynamic environment, a real-time trajectory planning method based on optimization and machine learning aimed at high dimensional inputs is presented. A learning optimization (LO) framework is established. Multiple criteria are defined to evaluate the performance quantitatively, and implementations with different sub-methods are discussed. In particular, a database generation method based on input space mapping is proposed for generating valid and representative samples. The methods presented are applied on a practical application—haptic interaction in virtual reality systems. The results show that the input space mapping method significantly elevates the efficiency and quality of database generation and consequently improves the performance of the LO. With the LO method, real-time trajectory generation with high dimensional inputs is achieved, which lays the foundation for robots with high dimensional DOFs to execute complex tasks in dynamic environments.

ARTICLE HISTORY
Received 15 September 2020
Accepted 28 June 2021

KEYWORDS
Real-time trajectory planning; global optimization; machine learning; human–robot interaction; serial manipulators

1. Introduction
When working in a dynamic environment, for example playing ball games (Ren, Zhu, and Xiong 2015), catching flying objects (Kim, Shukla, and Billard 2014; Salehian, Khoramshahi, and Billard 2016) or performing haptic feedback (Zhang and Dai 2018a), robots are required to react quickly to the continuous changes of environment and targets, for which continuous trajectory planning is needed. In frequently changing environments, trajectories need to be generated in a very short period.

Trajectory planning methods based on nonlinear optimization are adopted to improve motion efficiency by optimizing criteria such as motion time, energy and power consumption. However, the objective functions and constraints are always very complex in practical problems, leading to nonlinear and non-convex models, which are likely to become stuck in local minima and to be time-consuming to solve. Finding global minima of nonlinear optimization problems in real time is still a challenge.

Learning from former data is a promising way to reduce online calculation time. First, plenty of nonlinear optimization problems are solved offline to establish a database of optimal trajectories. Then, new optimal trajectories are generated online by regression and prediction. Two of the key factors in the learning framework are the database and the learning algorithm.

Databases are the foundation of learning, and they largely determine learning performance. Inappropriate databases may lead to poor learning performance, thence to low robot motion efficiency,
and even to task failure. The traditional method for generating databases resorts to the random choice of variables, or choosing sample variables evenly in the motion range, and calculating the global minimum by brute force methods. When it comes to complex environments or robots with high dimensional Degrees of Freedom (DOFs), the dimension of input space tends to be high. Selecting high dimensional sample variables that have a good distribution in the input space of the optimization problem as well as being valid and representative for the application is a critical problem.

Online learning is another important part of learning, which is the key to making full use of the database. A range of problems need to be tackled in online learning. First, traditional learning algorithms only deal with problems with single dimensional output. Regression model establishment and feature selection for problems with multiple dimensional outputs and high dimensional inputs are needed. In addition, optimization problems in different applications have different requirements of performance, which requires selecting appropriate regression algorithms in specific applications. Another major shortcoming of ordinary learning methods for solving constrained optimization problems is that they cannot guarantee the satisfaction of constraints. Thus, further refinements for the output of learning algorithms are required.

In this article, a learning optimization (LO) framework is constructed for solving continuous trajectory planning problems for serial manipulators. The main contributions are as follows (as illustrated in Figure 1).

- A general LO framework for solving real-time trajectory planning is formulated (Section 3.2). It consists of three adjustable sub-methods that can adapt to different requirements. Performance evaluation indices are defined for quantitative evaluations.
- A database generation method based on input space mapping is presented that efficiently generates high quality databases for problems with high dimensional inputs (Section 4).
- An online learning method based on multivariable-multiple-regression is introduced that targets problems with multiple inputs and multiple outputs and finds near-optimal solutions online (Section 5).
- The method is implemented and validated on the haptic feedback application in virtual reality systems (Section 6).

2. Related work

The continuous trajectory planning problem for robots, which is to re-plan the trajectory for robots with high frequency according to the changes of obstacles and the objectives in dynamic
environments, is widely used in practice, such as playing ball games (Ren, Zhu, and Xiong 2015) and catching flying objects (Kim, Shukla, and Billard 2014; Salehian, Khoramshahi, and Billard 2016). It plays an important role as the foundation for human–robot interaction, like collaborative manufacturing (Mainprice, Hayne, and Berenson 2016; Iqbal, Rack, and Riek 2016) and playing juggling (Kober, Glisson, and Mistry 2012). The continuous trajectory planning problem can be split into a series of point-to-point trajectory planning problems, which are highly demanding in real-time performance.

The study of point-to-point trajectory planning started from interpolation-based methods, such as polynomial interpolation (Kim et al. 1997) and B-spline interpolation (Lampariello and Hirzinger 2013). In general, pure interpolation-based methods are able to accomplish required tasks, but have difficulty achieving optimal performance in specific aspects. To obtain optimal trajectories, nonlinear optimization problems are constructed with optimal objectives based on time, energy and power consumption and constraints of mechanical limitation, time limitation and obstacle avoidance (Schulman et al. 2014). von Stryk and Schlemmer (1994) investigated nonlinear optimization with three separate criteria of minimum time, minimum energy and minimum power consumption and solved it by a numerical method combining direct collocation and an indirect multiple shooting method. Chettibi et al. (2004) presented the optimal planning problem trying to find a compromise between time, energy and power consumption and solved it by the Sequential Quadratic Programming (SQP) method. However, none of the aforementioned optimization-based methods are real-time, owing to the complex computation of nonlinear optimization.

Nonlinear optimization problems are likely to become stuck in local minima. To obtain the global minimum, they are generally solved with multiple initial guesses, which is significantly costly and hard to operate in real time. Finding the global minimum quickly is still challenging. A promising idea is learning from former data to reduce the online calculation time (Ude et al. 2010; Cassioli et al. 2012; Pan, Chen, and Abbeel 2014). Lampariello et al. (2011) and Werner et al. (2015) solved real time nonlinear optimization by combining offline optimal database generation and online regression prediction, which they verified on the applications of catching flying objects and bipedal walking. However, these methods are limited to specific cases having low dimensional inputs. Jetchev and Toussaint (2013) proposed a trajectory prediction method employing previous data to speed up the process of generating new trajectories. In addition, high dimensional inputs were considered by means of a high dimensional situation descriptor. Hauser (2017) extended this method by considering a more general context of nonlinear optimization problems and presented a general Learning Global Optima (LGO) framework.

The database for learning can be obtained by either recording former data or artificially generating data. Lampariello et al. (2011) chose variables evenly in a motion range to generate samples and drew a comparison of databases of different size. Hauser (2017) sampled an axis-aligned range of variables uniformly to generate a database. Additionally, a lifelong learning mode was presented to generate examples continuously with a separate background thread. In addition, the sensitivity of the required database size to the input dimension and the requirement for the database to guarantee the quality of solutions were discussed. However, for cases with high dimensional inputs, the way to generate databases efficiently with good distributions in both the input space of the optimization problem and the valid workspace of the application was not discussed.

This article extends the LGO framework and constructs a learning optimization (LO) model that can be adjusted to adapt to different applications by adopting different databases, regression methods and refinement methods. To improve the efficiency and quality of database generation for high dimensional input space, a database generation method based on input space mapping is presented. This method is validated on a continuous trajectory planning problem for haptic feedback manipulators in virtual reality systems.
3. Continuous trajectory planning based on learning optimization

3.1. Continuous trajectory planning

Considering trajectory planning problems for serial manipulators in joint space, the trajectory parameters are represented as $C \in \mathbb{R}^{NC}$, then the position, velocity and acceleration are represented as $q(C, t)$, $\dot{q}(C, t)$ and $\ddot{q}(C, t)$, respectively. The input of the trajectory problem is represented as $X \in \mathbb{R}^{NX}$, including the initial and target configuration, the initial velocity, the environmental variables, etc. In essence, the trajectory planning problem is the projection from input variables $X$ to trajectory parameters $C: X \rightarrow C = f(X)$.

When serial manipulators work in a dynamic environment in which the target configurations or the environments change all the time, $X$ changes frequently. It is required to re-plan the trajectory according to the new $X$. In the $i$th planning period, the environment is detected and the input is updated to obtain $X_i$, then the trajectory is re-planned and the according parameters $C_i$ obtained, until the robot reaches the actual target configuration, shown as Figure 2. To guarantee that the robots respond to the frequently changing factors, the planning period $T_p$ must be very small. The trajectory planning is required to be done in real time, which means that the calculation time must be shorter than $T_p$.

3.2. Real-time trajectory planning based on learning optimization

3.2.1. Learning optimization framework

To guarantee the safety and improve the efficiency of robots, a nonlinear optimization based method is employed in which the objective function is defined to optimize the efficiency and the constraints are determined by considering both mechanism limitations and task-related limitations. The constrained optimization problem is denoted as

$$C^* = O(X),$$

where $C^*$ represents the optimal parameters.

In general, the above nonlinear optimization problem is non-convex, which makes the global optimization time-consuming. Machine learning methods are employed to improve computational efficiency and achieve real-time performance. The complex original optimization problem is approximated by a regression model established from an optimal trajectory database. The database is
generated offline. In the online procedure, the solution is predicted by the regression model, which requires little calculation. The procedure of the learning optimization method is as follows.

1. **Database generation**
   Multiple input variables \(X^i (i = 1, 2, \ldots, N_D)\) in the input space are selected and the global minimum \(C^* \) is solved by the multi-restart method. The optimal database \(D = \{X^i, C^* \mid i = 1, 2, \ldots, N_D\}\) is built.

2. **Regression and prediction**
   The regression model \(R\) is established by exploiting the optimal database, which can be performed offline or online. For a new input variable \(X'\), an initial parameter can be predicted with the regression model: \(C' = R(D, X')\).

3. **Parameter refinement**
   The initial parameter \(C'\) is not guaranteed to satisfy the constraints. Thus, \(C'\) is taken as the initial guess for solving an optimization problem, which has the same constraints as the original optimization problem, to refine the parameters: \(C^* = O_{C'}(X')\).

By extracting three essential elements, the database \(D\), the regression and prediction method \(R\), and the parameter refinement method \(O\), the above trajectory planning method based on optimization and machine learning is formulated in a general form. The Learning Optimization (LO) model is constructed, which is modularized with the three sub-methods \(D, R\) and \(O\):

\[
\mathcal{L}(D, R, O).
\]

By integrating with different sub-methods, \(\mathcal{L}\) with different performances can be obtained to satisfy different application requirements. The effects of each sub-method on the LO performance are evaluated and analysed in Sections 7.2 and 7.3.

### 3.2.2. Performance evaluation indices

To evaluate the performance of the LO model formally and quantitatively, three performance evaluation indices are defined that indicate the feasibility, real-time performance and accuracy, respectively.

1. **Feasible success rate**
   According to Section 3.2.1, for a new input \(X'\), an initial parameter \(C'\) is obtained by regression and prediction, which needs further refinement by solving the nonlinear optimization problem. Assuming \(N_r\) \(X''\)’s are input to \(\mathcal{L}\), among which only \(N_{r1}\) groups output feasible parameters in the refinement stage, the feasible success rate \(R_s\) of \(\mathcal{L}\) can be approximated as

\[
R_s = N_{r1} / N_r.
\]

The feasible success rate is the most crucial index in LO, which indicates the feasibility of the LO model and reflects whether the task can be done successfully. In continuous trajectory planning, if an infeasible solution is obtained in any planning period, i.e. the output parameter does not satisfy the constraints of the nonlinear optimization problem, the task may fail.

2. **Online learning time**
   The online learning time \(T_L\) is the average time for \(\mathcal{L}\) to perform online learning, including the regression prediction time \(T_{PR}\) and the refinement time \(T_{RO}\):

\[
T_L = T_{PR} + T_{RO}.
\]

It reflects whether trajectory planning can be performed in real time. According to Section 3.1, in continuous trajectory planning applications, to achieve real-time performance, an online trajectory planning procedure is required to be finished within a prediction period \(T_p\), i.e. \(T_L < T_p\).
(3) **Average cost increase**

Assume $C_a$ is the accurate optimal parameter, i.e. the solution of the global optimization problem, and $F_a$ is the corresponding objective function value. Similarly, $C_p$ is the parameter derived through $L$ and $F_p$ is the corresponding objective function value. Suppose that LO are run with $N_r$ groups of $(X, C_a, F_a)$, and $C_p$ and $F_p$ are obtained. The average cost increase is approximated as

$$e_F = \frac{1}{N_r} \sum_{i=1}^{N_r} \frac{|F_p^i - F_a^i|}{F_a^i}.$$  \hspace{1cm} (5)

This indicates the deviation of the solution from the optima, which does not influence the feasibility of the task, but reflects whether the task will be done with high quality and efficiency.

4. **Database generation method based on input space mapping**

4.1. **Input space mapping**

In the LO framework, ample samples are required to establish high quality databases. The construction of a database is essentially selecting various $X$ and solving the corresponding optimization problems specified in Equation (1). To make the sample variables locate in a valid workspace for a specific application and be meaningful for learning, the domain of $X$ depends on the application-specified variable $X_A$. The traditional method for generating sample variables is either randomly or evenly choosing values in the range of each component and combining them to obtain sample vectors. Supposing $d$ (defined as variable density) values are selected for each component, $N_v$ is the dimension of the vector, then $d^{N_v}$ samples are calculated. One option is to generate samples in the space of $X$, i.e. randomly or evenly selecting values for each component of $X$. However, many sample variables selected by this way are not in the valid workspace of the specific application. These sample variables are less significant for learning. In addition, some of the sample variables may be meaningless in Equation (1), which cannot converge to a feasible solution. To deal with these problems, another option is to generate sample variables in the space of $X_A$. However, $X_A$ usually has a higher dimension than $X$, which results in a much larger size of samples. Moreover, as $X_A$ is not directly used in the optimization problem, the sample variables tend to gather in the input space of the optimization problem.

To generate sample variables with high efficiency and quality, an input space mapping method is presented for selecting them that jointly considers the space of $X$ and $X_A$. The general properties of sample generation in the two spaces are analysed, including the size and the distribution of the sample variables.

Two types of mapping are presented. One is **merging mapping**, which maps a high dimensional application-specified space to the lower dimensional input space of the optimization problem:

$$X = F_M(X_{A1}).$$  \hspace{1cm} (6)

The other is **transforming mapping**, which maps the input space of the optimization problem to an application-specific space, where whether a sample is valid for the application can be easily determined:

$$X_{A2} = F_T(X).$$  \hspace{1cm} (7)

4.1.1. **Merging mapping**

A merging mapping $F_M$ is to transform a higher dimensional dependent component vector $x_d \in \mathbb{R}^{N_d}$ to a lower dimensional independent component vector $x_{id} \in \mathbb{R}^{N_{id}}$

$$x_{id} = F_M(x_d),$$

$$F_M : V_d \subset \mathbb{R}^{N_d} \rightarrow V_{id} \subset \mathbb{R}^{N_{id}} (N_d > N_{id}).$$  \hspace{1cm} (8)
Merging mappings combine the dependent redundant elements and merge them into an independent component vector in the input space of the optimization problem, which reduces the dimension of the variables. The samples are generated as follows.

(1) Derive the domain of \( x_{id} \) (represented as \( V_{id} \)) from the domain of \( x_d \) (represented as \( V_d \)) with the mapping \( \mathcal{F}_M \).
(2) Randomly or evenly select variable vectors in \( V_{id} \) with variable density \( d \), and a set of sample variables \( S = \{ x^i_{id} \in V_{id} | i = 1, 2, \ldots, d^{N_{id}} \} \) is obtained.

Suppose \( S_1 \) is a sample variable set obtained by merging mapping, and \( S_2 \) is a sample variable set obtained by randomly or evenly selecting variable combinations of \( x_d \) in \( V_d \), then mapping them into \( \mathbb{R}^{N_{id}} \). For variable density \( d \in \mathbb{N}^+ \), the sizes of \( S_1 \) and \( S_2 \) are \( d^{N_{id}} \) and \( d^{N_d} \), respectively. As \( N_{id} < N_d \), in a finite range of sample size \( (d^{N_{id}} < n, \ d^{N_d} < n, \ n \in \mathbb{N}^+) \), the feasible sizes of \( S_2 \) are far less than \( S_1 \). For example, considering \( N_d = 6 \) and \( N_{id} = 2 \), the feasible sizes of \( S_1 \) and \( S_2 \) less than \( n = 16,000 \) are shown in Figure 3. Additionally, the sample variables in \( S_1 \) are always evenly distributed in \( \mathbb{R}^{N_{id}} \) while the sample variables in \( S_2 \) tend to gather, which is less representative. For example, consider that \( N_d = 6, N_{id} = 2 \), and the variable densities of \( S_1 \) and \( S_2 \) are \( d_1 = 27 \) and \( d_2 = 3 \), respectively. They have the same size \( d^{N_{id}}_1 = d^{N_{id}}_2 \). Performing linear merging mapping \( \mathcal{F}_{M1} \) and nonlinear merging mapping \( \mathcal{F}_{M2} \),

\[
\begin{align*}
x_{id} & = \mathcal{F}_{M1}(x_d) = \begin{pmatrix} 1 & 2 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.5 & 3 & 10 \end{pmatrix} x_d, \\
x_d & \rightarrow x_{id} = \mathcal{F}_{M2}(x_d) : \nonumber \\x_{id}(1) & = x_d^3(1) + x_d^3(2) + x_d^3(3), \\
x_{id}(2) & = x_d^3(4) + x_d^3(5) + x_d^3(6).
\end{align*}
\]

A comparison of distributions of \( x_{id} \) obtained by randomly and uniformly selecting sample variables before and after the input space mapping is shown in Figure 4.

### 4.1.2. Transforming mapping

A transforming mapping \( \mathcal{F}_T \) is to transform an independent component vector \( x_{d_1} \in \mathbb{R}^{N_{d_1}} \) to another vector \( x_{d_2} \in \mathbb{R}^{N_{d_2}} \):

\[
x_{d_2} = \mathcal{F}_T(x_{d_1}),
\]

\[
\mathcal{F}_T : V_{d_1} \subset \mathbb{R}^{N_{d_1}} \rightarrow V_{d_2} \subset \mathbb{R}^{N_{d_2}} (N_{d_1} \geq N_{d_2}).
\]

The main aim is to map the component of the input variables of the optimization problem into the space in which whether the input variable is in the valid workspace of specific applications can be determined.

In contrast to merging mapping, the variables used as the input of the optimization problem are the variables before the transforming mapping \( x_{d_1} \), while \( x_{d_2} \) is only used to evaluate whether the variable is in the valid workspace. The variables are first selected in \( V_{d_1} \) and then mapped to \( V_{d_2} \) for verification. The sample variables are generated as follows.

(1) Randomly or evenly select multiple vectors in \( V_{d_1} \) with the variable density \( d \) to generate a set of variables \( S = \{ x^i_{d_1} \in V_{d_1} | i = 1, 2, \ldots, d^{N_{d_1}} \} \).
(2) Map each \( x^i_{d_1} \) in \( S \) through \( \mathcal{F}_T \) to obtain the corresponding \( x^i_{d_2} \) and check if it is in the workspace.
(3) Delete all the \( x^j_{d_1} \) in \( S \) to which the according \( x^j_{d_2} \) is not in the workspace, to obtain the new set of sample variables \( S' \).
Figure 3. Feasible sizes of the sample variable sets before and after the input space mapping.

(a) Linear mapping, uniform selection  (b) Non-linear mapping, uniform selection

(c) Linear mapping, random selection  (d) Non-linear mapping, random selection

Figure 4. Sample variables distribution before and after the input space mapping. (a) Linear mapping, uniform selection. (b) Non-linear mapping, uniform selection. (c) Linear mapping, random selection. (d) Non-linear mapping, random selection.
If the following conditions are satisfied:

1. $V_{d2}$ is known,
2. the inverse mapping of $F_T$ exists and is an injective mapping,
3. $N_{d1} > N_{d2}$,

another way can be employed that selects sample variables in $V_{d2}$ and maps them back to $V_{d1}$:

1. randomly or evenly select multiple vectors in $V_{d2}$ with the variable density $d$ to generate a set of variables $S = \{x_{d2}^i \in V_{d2} | i = 1, 2, \ldots, d^{N_{d2}}\}$;
2. map each $x_{d2}^i$ by the inverse mapping of $F_T$ to obtain the corresponding $x_{d1}^i$ and get the sample variable set $S = \{x_{d1}^i \in V_{d1} | i = 1, 2, \ldots, d^{N_{d2}}\}$.

Invalid and unrepresentative samples are removed by performing the transforming mapping. First, $S'$ contains fewer sample variables than $S$, which results in less calculation time. In addition, the feasible success rate of nonlinear optimization for sample calculation is improved. Consequently, the efficiency and quality of database generation can be significantly improved.

### 4.2. Database generation

The procedure for database generation based on input space mapping is as follows.

1. Classify the components of the input variable $X$. The high dimensional dependent components $\{x_{id}^i | i = 1, 2, \ldots, n_M\}$ and the independent components that need to be transformed $\{x_{d1}^j | j = 1, 2, \ldots, n_T\}$ are extracted, where $n_M$ and $n_T$ represent the numbers of $x_{id}$ and $x_{d1}$ in $X$, i.e. the amount of merging mapping and transforming mapping needing to be done, respectively. The remainder is denoted as $x_c$. Then $X$ is denoted as

$$X = \{x_{id}^1, x_{id}^2, \ldots, x_{id}^{n_M}, x_{d1}^1, x_{d1}^2, \ldots, x_{d1}^{n_T}, x_c\}.$$  \hspace{1cm} (12)

2. Perform merging mapping for each $x_{id}^i$ and transforming mapping for each $x_{d1}^j$:

$$x_{d1}^i = F_{Mi}(x_{id}^i), \quad i = 1, 2, \ldots, n_M,$$

$$x_{d2}^j = F_{Tj}(x_{d1}^j), \quad j = 1, 2, \ldots, n_T.$$  \hspace{1cm} (13)

The new variable after the input space mapping is

$$X_L = \{x_{d1}^1, x_{d1}^2, \ldots, x_{d1}^{n_M}, x_{d2}^1, x_{d2}^2, \ldots, x_{d2}^{n_T}, x_c\}.$$  \hspace{1cm} (14)

3. If there are any new components in $X_L$ that can be mapped further, let $X = X_L$ and repeat steps (1) and (2). Otherwise, move to step (4).

4. According to Sections 4.1.1 and 4.1.2, $x_{d1}^i$ and $x_{d2}^j$ are used to solve the nonlinear optimization. Construct the sample variable:

$$X_S = \{x_{d1}^1, x_{d1}^2, \ldots, x_{d1}^{n_M}, x_{d2}^1, x_{d2}^2, \ldots, x_{d2}^{n_T}, x_c\}.$$  \hspace{1cm} (15)

Select $N_D$ groups of sample variables by the method introduced in Sections 4.1.1 and 4.1.2 and generate a set of sample variables:

$$S = \{X_S^i | i = 1, 2, \ldots, N_D\}.$$  \hspace{1cm} (16)
(5) Solve the nonlinear optimization problem specified by each $X^i_S$:

$$C^* = O(X^i_S).$$

(17)

The optimal database is established:

$$D = \{(X^i_S, C^*_{i}) | i = 1, 2, \ldots, N_D\}. \quad (18)$$

5. Online learning based on the multivariable-multiple-regression

5.1. Regression and prediction

The multivariable-multiple-regression model is constructed (Zhang et al. 2020) to deal with multiple-dimensional inputs and multiple means multiple-dimensional outputs:

$$R = \{r^{(i)} | i = 1, 2, \ldots, N_C\}, \quad C^{(i)} = R^{(i)}(D, X_F^{(i)}).$$

(19)

This can be regarded as a set of regression models in which each regression model $R^{(i)}$ is built independently for one output parameter $C^{(i)}$ with its own feature vector $X_F^{(i)}$.

Then general regression methods can be adopted to build maps between $X_F^{(i)}$ and $C^{(i)}$. Three methods, $k$-Nearest Neighbours ($k$-NN) regression (Cover and Hart 1967), Support Vector Machine Regression (SVR) (Wu and Der-Tsai Lee 2004) and Gaussian Process Regression (GPR) (Seeger 2004), are discussed in Section 7.2.

5.2. Parameter refinement

The parameter output by the regression and prediction $C'$ may not satisfy the constraints, which requires further refinement. $C'$ is taken as an initial guess to solve the original optimization problem

$$C^* = O_C(X).$$

(20)

As $C'$ is close to the global minimum, the calculation time is likely to be smaller than using a random initial guess. However, when the objective function and constraints are complex, Equation (20) is still hard to solve in real time. To reduce the calculation time further, another way is to simplify the objective function, such as performing an optimization without an objective function:

$$C^* = \hat{O}_C(X).$$

(21)

This method improves the calculation speed, but compromises the optimality. The quality of the solution depends on the property of the original objective function. When $F(C)$ changes sharply around the global optimum, a small deviation of $C$ can result in a high cost increase. Assuming $F(C)$ is differentiable around the global minimum $C^*_{g}$, the total differential of $F(C)$ at $C^*_{g}$ is

$$dF(C^*_{g}) = \sum_{i=1}^{N_C} \frac{\partial F}{\partial C_i}(C^*_{g}) \, dC_i,$$

(22)

which indicates the changing rate of $F(C)$ around $C^*_{g}$. The relative rate of change of the objective function around $C^*_{g}$ is defined as

$$dF_{r}(C^*_{g}) = \frac{dF(C^*_{g})}{F(C^*_{g})}.$$

(23)

When $dF_{r}(C^*_{g})$ is small, a small deviation $dC$ will cause a low cost increase. Otherwise, the cost will increase rapidly.
The refinement method is taken as an unfixed sub-method of LO. In practice, the refinement method is chosen according to the specific requirement. For example, Equation (21) is chosen in situations that are sensitive to computation speed but have a lower requirement for optimality.

6. Application to kinematic trajectory planning for haptic feedback manipulators

6.1. Continuous trajectory planning of haptic feedback systems

A practical case, trajectory planning for haptic feedback manipulators, is introduced to demonstrate the practicality and efficiency of the methods presented.

In virtual reality systems, the real world is simulated by constructing a virtual environment that provides visual sense to users. However, the corresponding haptic feedback is lacking when users try to touch the objects in the virtual environment, which significantly limits the interaction and immersion performance. To achieve haptic feedback in virtual reality systems, a haptic feedback system based on the servo serial manipulator of (Zhang and Dai 2018a) is constructed, as shown as Figure 5. A serial manipulator serves as the Haptic Feedback Manipulator (HFM). Various kinds of tools or buttons are linked to the end-effector. As the user’s hand moves, the hand motion data is tracked and the manipulation intention is predicted. Specifically, the interacting time $t_c$ and the interacting position $p_c$ when and where the human hand and the specific tool or button will contact are determined. Then trajectory planning and control for the HFM are performed to bring the tool or button to $p_c$ at $t_c$. When the user’s hand finally reaches the actual target position and conducts manipulation, the user sees the virtual hand operating the tool or button in the virtual environment, while the corresponding tool or button reaches the target position to provide haptic feedback.

To avoid injury from robots to humans when working in a shared workspace, the motion ranges of the HFM and the user’s hand are restricted. The reachable workspace of the HFM is divided into the interacting area $W_C$, the effective workspace $W_E$ and the prohibited area $W_P$ (Zhang and Dai 2018b), which satisfy

$$W_C \subset W_E, \quad W_E + W_P = W. \quad (24)$$

The motion range of the HFM and the hand are restricted to the side of $W_E$ and $W_P$, respectively. All the contacts are performed in $W_C$.

The trajectory of the HFM is continuously re-planned. As the hand moves, the predicted interacting point slightly changes and gradually converges to the actual interacting point. A sequence of

Figure 5. Structure of the haptic feedback system.
predicted interacting points \((p_0, p_1, \ldots, p_n) \subset S_E\) is generated with the updating period \(T_p\). In the \(i\)th period, the interacting configuration corresponding to \(p_i\) is taken as the target configuration of trajectory planning, denoted as \(q_c \in \mathbb{R}^{N_j}\). For motion continuity, the current configuration and velocity are taken as the initial configuration and velocity of the trajectory planning.

**6.2. Trajectory planning based on nonlinear optimization**

An \(N_j\)-DOF serial robot is adopted as the HFM. Considering the kinematical trajectory planning for the HFM, the trajectory of each joint is encoded by the Trapezoidal Velocity Profile (TVP). The input of the trajectory planning based on nonlinear optimization is selected as \(X = (q_0, q_c, \omega_0) \in \mathbb{R}^{3N_j}\), where \(q_0 \in \mathbb{R}^{N_j}\), \(q_c \in \mathbb{R}^{N_j}\), \(\omega_0 \in \mathbb{R}^{N_j}\) are the initial configuration, the target configuration and the initial velocity, respectively. The optimization parameter is selected as \(C = (\omega_m^{(1)}, \omega_m^{(2)}, \ldots, \omega_m^{(N_j)}, t_f) \in \mathbb{R}^{N_j+1}\) (Zhang et al. 2020), where \(\omega_m^{(i)}\) is the velocity for the maximum velocity phase in TVP and superscript \((i)\) represents the \(i\)th joint. All the joints are required to move simultaneously and have the same motion time \(t_f\). The optimization problem described in Equation (1) is be specified with the input variable \(X\) and the output parameter \(C\).

The objective function is constructed to balance safety and motion efficiency. The constraints are constructed considering the mechanism limits, TVP and the requirements of HFM (Zhang and Dai 2018a; Zhang et al. 2020).

**6.3. Database generation based on input space mapping**

According to Section 4.2, the elements of the input variable \(X = (q_0, q_c, \omega_0)\) are classified into different types and handled respectively. First, while the range of \(q_c\) and \(q_0\) are determined by the workspace restriction shown as Equation (24), the optimization problem is related to \(q_c - q_0\) rather than the individual \(q_c\) and \(q_0\). Therefore, \((q_0, q_c)\) is a dependent component and can be merged:

\[
q_f = F_M(q_0, q_c) = q_c - q_0,
\]

where \(V_q\) is the range of the configuration, and \(V_{q_f}\) is the range of the difference between two configurations.

Then the sample variable can be selected as \(X_S = (q_f, \omega_0)\). In addition, according to Section 6.1, the sample variables should be in the workspace of the HFM, i.e. the initial and target positions of the end-effector should be located in the effective workspace and the interacting area, respectively:

\[
p_0 \in W_E, \quad p_c \in W_C.
\]

To determine whether an input variable is feasible for the HFM, \(q_0\) and \(q_c\) are transformed into the task space by the forward kinematics:

\[
p_0 = F_{TFK}(q_0),\]

\[
F_{TFK} : V_q \subset \mathbb{R}^{N_j} \rightarrow W \subset \mathbb{R}^{N_j},
\]

\[
p_c = F_{TFK}(q_c),
\]

\[
F_{TFK} : V_q \subset \mathbb{R}^{N_j} \rightarrow W \subset \mathbb{R}^{N_j}.
\]

A set of sample variables \(X_S = (q_f, \omega_0)\) are selected to build the database. In order to obtain more representative sample variables, the combinations of \(q_f\) and \(\omega_0\) are selected according to the practical motion curves. The haptic feedback process consists of a sequence of trajectory planning problems. At the beginning of this process, the initial state of the first trajectory planning problem is
the static state with null velocity. As the process proceeds, the initial state of the next trajectory planning problem is one of the intermediate points of the last trajectory. Thus, first $\omega_0 = 0$ and multiple $q_f$’s are selected to solve $C = O(q_f, 0)$. The resulting parameters $C = (\omega_m, t_f)$ and the corresponding trapezoidal velocity curves $\dot{q}(C, t)$ are obtained. Then some intermediate points of each curve are selected:

$$t' = \alpha t_f,$$

where $\alpha \in V_\alpha = [0, 1)$. The new $\omega'_0$ and $q'_f$ are generated and serve as the sample variables:

$$\omega'_0 = \dot{q}(C, t'),$$

$$q'_f = q(t_f) - q(t') - q_0.$$

The above process is another transforming mapping

$$F_T : (V_{q_f}, V_{\omega}) \subset \mathbb{R}^{2N_j} \rightarrow (V_{q_f}, V_\alpha) \subset \mathbb{R}^{N_j+1},$$

where $V_\omega$ is the range of the joint velocity.

By combining Equations (25), (27) and (31), the final mapping of the input variables is

$$X_L = (q_f, \alpha) = F(q_0, q_c, \omega_0),$$

$$F : V_X \subset \mathbb{R}^{3N_j} \rightarrow V_{X_L} \subset \mathbb{R}^{N_j+1}.$$

By randomly or evenly choosing $N_1$ $q_f$ and $N_2$ $\alpha$, $N_1 \times N_2$ sample variables $X_S = (q_f, \omega_0)$ are obtained. Finally, the database including $N_1 \times N_2$ samples is built by solving the global optimization problems according to all $X_S$’s.

6.4. Feature selection for multivariable-multiple-regression

In multivariate-multiple-regression, the feature vector for each element is constructed:

$$t_f = f_t(X), \quad \omega_m^{(i)} = f_\omega^{(i)}(q_f^{(i)}, \omega_0^{(i)}, t_f) \quad (i = 1, 2, \ldots, N_j).$$

The synchronized motion time $t_f$ takes $X$ as the feature as it is affected directly by all the elements in $X$. The maximum velocity for the $i$th joint $\omega_m^{(i)}$ takes only the components of the corresponding joint and the motion time as feature, since the other elements in $X$ affect it only through the common motion time (Zhang et al. 2020).

7. Results

7.1. Database generation by input space mapping

A 3-DOF manipulator is employed as the HFM, whose effective workspace $W_E$ and interacting area $W_C$ are illustrated with the asterisks and points in Figure 6, respectively. With the input space mapping introduced in Equations (25)–(27), a set of $q_f$ is generated by selecting $N_1 = 1000$ groups of $(p_0, p_c)$ that satisfy $p_0 \in W_E$ and $p_c \in W_C$ and mapping them into the joint space, shown as the points in Figure 7(a). For comparison, another set of $q_f$ are generated by randomly choosing 1000 groups of $(p_0, p_c)$ in the motion range, shown as the asterisks in Figure 7(a). The asterisks have a smaller coverage and a more concentrated distribution than the points, which indicates that the valid workspace of the HFM does not cover the whole reachable workspace of the robot. Transforming mapping reduces the selection range of input variables and results in more dense samples with the same number of samples.
According to the sample selection method described in Section 6.3, selecting $N_1 = 1000$ and $N_2 = 10$, $N_1 \times N_2 = 10,000$ sample variables are obtained. The database $D$ is generated by running global optimizations solved with multiple re-start SQP, in which the maximum re-start number $N_{RO}$ is set to 10. For comparison, two databases $D_1$ and $D_2$ are generated by randomly selecting values in the motion range of each joint with $d = 5$ and solving optimizations by multiple re-start SQP with $N_{RO} = 10$ and $N_{RO} = 100$, respectively. The feasible success rates for optimization in the sample calculations for the above three databases are shown in Figure 7(b). The feasible success rate is greatly improved by the input space mapping.

Figures 7(a) and 7(b) show that the input space mapping method can help to select more valid and representative samples, thereby increasing the success rate of sample calculation and improving the database quality.

7.2. Regression methods

In this section, different regression methods for LO are tested. $k$-NN ($k = 1, 5, 10$), SVR with linear kernel and Gaussian kernel and GPR are employed, and LO models $\mathcal{L}(D_{ND}, R_{k-NN}, O)$,
Figure 8. Learning results with different regression methods. (a) Feasible success rate. (b) Average cost increase. (c) Model-building time. (d) Online learning time.

Figure 9. Online learning time with different regression methods. (a) Regression prediction time. (b) Refinement time.

For all the regression methods tested, small cost increases (less than 1%) and online motion time (less than 50 ms) are achieved, and high feasible success rates are attained with large database sizes. In
7.3. Parameter refinement

In this section, refinement methods with and without an objective function (introduced in Equations [20] and [21]) are tested. The database \( D_{5000} \) and the regression methods of 10-NN, Gaussian-SVR and GPR are employed to build the LO models. The performances evaluated with \( D_T \) are shown in Figure 10, which indicates that refinement with an objective function leads to shorter calculation time for parameter refinement but larger increases in objective function values.

By examining the cost increase of each datum (Figure 11(a)), it is found that the cost increases by refinement without an objective function are tiny in most cases, but are very large for several inputs, which leads to the high average cost increase. The relative change rate at each \( C_g^* \) is calculated and plotted in Figure 11(b) as the bars, which show a consistent trend with the cost increase. Refinement with an objective function can eliminate the cases with large errors, which significantly reduces the average error with a slight increase of refinement time.
7.4. Continuous trajectory planning

Now the above presented methods are integrated and applied for continuously planning trajectories for the HFM. The LO model $L(D, R_{GPR}, O)$ is employed. The update period is $T_p = 60$ ms. The velocity and position curve of the whole process is shown as Figure 12. The moving distance of the joints and the predicted interacting position in each period are shown by the solid line and the stars in the second figure, respectively. By the LO method, smooth velocity curves are obtained and HFM finally reaches the interacting configuration within the interacting time.

8. Conclusions

A real-time trajectory generation method based on optimization and machine learning is presented in this article. An LO framework is constructed in which different sub-methods, including databases, regression methods and refinement methods, are implemented and evaluated by performance evaluation indices indicating feasibility, real-time performance and accuracy. In particular, aiming at high dimensional inputs, an input space mapping method is presented that achieves a higher feasible sample calculation success rate, a better sample distribution and less calculation time for database generation. In addition, an online learning method based on multivariable-multiple-regression is employed that can flexibly integrate different regression methods and improve learning efficiency further. The method is implemented and validated on a practical application—haptic interaction in virtual reality systems—where trajectories are generated continuously in real time.

In the future, a possible direction would be to extend this work to more variations of the LO model and various specific optimization problems. Also, technologies for further improving the calculation efficiency for both offline sample calculation and online regression could be explored.

Disclosure statement

No potential conflict of interest was reported by the author(s).

Funding

This work was supported by the Major Project of the New Generation of Artificial Intelligence, China [grant number 2018AAA0102900].
Data availability statement

The data that support the findings of this study are available from Shiyu Zhang (zhangshiyu@buaa.edu.cn), upon reasonable request.

ORCID

Shiyu Zhang http://orcid.org/0000-0003-2474-7451

References

Cassioli, Andrea, David Di Lorenzo, Marco Locatelli, Fabio Schoen, and Marco Sciandrone. 2012. “Machine Learning for Global Optimization.” *Computational Optimization and Applications* 51 (1): 279–303.

Chettibi, T., H. E. Lehtihet, M. Haddad, and S. Hanchi. 2004. “Minimum Cost Trajectory Planning for Industrial Robots.” *European Journal of Mechanics—A/Solids* 23 (4): 703–715.

Cover, Thomas, and Peter Hart. 1967. “Nearest Neighbor Pattern Classification.” *IEEE Transactions on Information Theory* 13 (1): 21–27.

Hauser, Kris. 2017. “Learning the Problem-Optimum Map: Analysis and Application to Global Optimization in Robotics.” *IEEE Transactions on Robotics* 33 (1): 141–152.

Iqbal, Tariq, Samantha Rack, and Laurel D. Riek. 2016. “Movement Coordination in Human–Robot Teams: A Dynamical Systems Approach.” *IEEE Transactions on Robotics* 32 (4): 909–919.

Jetchev, Nikolay, and Marc Toussaint. 2013. “Fast Motion Planning From Experience: Trajectory Prediction for Speeding Up Movement Generation.” *Autonomous Robots* 34 (1–2): 111–127.

Kim, Kee Whan, Hyun Sik Kim, Yong Kiu Choi, and Jin Hyun Park. 1997. “Optimization of Cubic Polynomial Joint Trajectories and Sliding Mode Controllers for Robots Using Evolution Strategy.” In *Proceedings of IECON*, Vol. 3, 1444–1447. Piscataway, NJ: IEEE.

Kim, Seungsu, Ashwini Shukla, and Aude Billard. 2014. “Catching Objects in Flight.” *IEEE Transactions on Robotics* 30 (5): 1049–1065.

Koher, Jens, Matthew Glisson, and Michael Mistry. 2012. “Playing Catch and Juggling with a Humanoid Robot.” In *Proceedings of the 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012)*, 875–881. Piscataway, NJ: IEEE. doi:10.1109/HUMANOIDS.2012.6651623

Lampariello, Roberto, and Gerd Hirzinger. 2013. “Generating Feasible Trajectories for Autonomous On-Orbit Grasping of Spinning Debris in a Useful Time.” In *Proceedings of 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2013)*, 5652–5659. Piscataway, NJ: IEEE. doi:10.1109/IROS.2013.6697175

Lampariello, Roberto, Duy Nguyen-Tuong, Claudio Castellini, Gerd Hirzinger, and Jan Peters. 2011. “Trajectory Planning for Optimal Robot Catching in Real-Time.” In *Proceedings of the 2011 IEEE International Conference on Robotics and Automation (ICRA 2011)*, 3719–3726. Piscataway, NJ: IEEE. doi:10.1109/ICRA.2011.5980114

Mainprice, Jim, Rafi Hayne, and Dmitry Berenson. 2016. “Goal Set Inverse Optimal Control and Iterative Replanning for Predicting Human Movements in Shared Workspaces.” *IEEE Transactions on Robotics* 32 (4): 897–908.

Pan, Jia, Zhuo Chen, and Pieter Abbeel. 2014. “Predicting Initialization Effectiveness for Trajectory Optimization.” In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2014)*, 5183–5190. Piscataway, NJ: IEEE.

Ren, Ziwu, Qiuguo Zhu, and Rong Xiong. 2015. “Trajectory Planning of 7-DOF Humanoid Manipulator Under Rapid and Continuous Reaction and Obstacle Avoidance Environment.” *Acta Automatica Sinica* 41 (6): 1131–1144.

Salehian, Seyed Sina Mirrazavi, Mahdi Khoramshahi, and Aude Billard. 2016. “A Dynamical System Approach for Softly Catching a Flying Object: Theory and Experiment.” *IEEE Transactions on Robotics* 32 (2): 462–471.

Schulman, John, Yan Duan, Jonathan Ho, Alex Lee, Ibrahim Awwal, Henry Bradlow, Jia Pan, Sachin Patil, Ken Goldberg, and Pieter Abbeel. 2014. “Motion Planning with Sequential Convex Optimization and Convex Collision Checking.” *The International Journal of Robotics Research* 33 (9): 1251–1270.

Seeger, Matthias. 2004. “Gaussian Processes for Machine Learning.” *International Journal of Neural Systems* 14 (02): 69–106.

Ude, Aleš, Andrej Gams, Tamim Asfour, and Jun Morimoto. 2010. “Task-Specific Generalization of Discrete and Periodic Dynamic Movement Patterns.” *IEEE Transactions on Robotics* 26 (5): 800–815.

von STRyk, Oskar, and Maximilian Schlemmer. 1994. “Optimal Control of the Industrial Robot Manutec r3.” *Computational Optimal Control*, 367–382. Basel, Switzerland: Birkhäuser. doi:10.1007/978-3-0348-8497-6_30

Werner, Alexander, Dietrich Trautmann, Dongheui Lee, and Roberto Lampariello. 2015. “Generalization of Optimal Motion Trajectories for Bipedal Walking.” In *Proceedings of the 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2015)*, 1571–1577. Piscataway, NJ: IEEE. https://alexanderwerner.net/publications/iros_2015_Werner.pdf

Wu, Chun-Hsin, Jan-Ming Ho, and Der-Tsai Lee. 2004. “Travel-Time Prediction with Support Vector Regression.” *IEEE Transactions on Intelligent Transportation Systems* 5 (4): 276–281.
Zhang, Shiyu, and Shuling Dai. 2018a. “Real-Time Trajectory Generation for Haptic Feedback Manipulators in Virtual Cockpit Systems.” *Journal of Computing and Information Science in Engineering* 18 (4): Article ID 041015, 11pp. doi:10.1115/1.4041166

Zhang, Shiyu, and Shuling Dai. 2018b. “Workspace Analysis for Haptic Feedback Manipulator in Virtual Cockpit System.” *Virtual Reality* 22 (4): 321–338.

Zhang, Shiyu, Shuling Dai, Andrea Maria Zanchettin, and Renzo Villa. 2020. “Trajectory Planning Based on Non-Convex Global Optimization for Serial Manipulators.” *Applied Mathematical Modelling* 84: 89–105.