Robotica (2022), **40**, pp. 1125–1150
doi:10.1017/S0263574721000941

CAMBRIDGE UNIVERSITY PRESS

RESEARCH ARTICLE

Global motion planning and redundancy resolution for large objects manipulation by dual redundant robots with closed kinematics

Yongxiang Wu, Yili Fu* and Shuguo Wang

State Key Laboratory of Robotics and System, Harbin Institute of Technology, Harbin, China
*Corresponding author. Email: meylfu_hit@126.com

Received: 6 February 2021; **Revised:** 17 May 2021; **Accepted:** 14 June 2021; **First published online:** 9 August 2021

Keywords: closed kinematic chain; dual arm coordinated manipulation; motion planning; redundancy resolution; quadratic programming

Abstract

The multi-arm robotic systems consisting of redundant robots are able to conduct more complex and coordinated tasks, such as manipulating large or heavy objects. The challenges of the motion planning and control for such systems mainly arise from the closed-chain constraint and redundancy resolution problem. The closed-chain constraint reduces the configuration space to lower-dimensional subsets, making it difficult for sampling feasible configurations and planning path connecting them. A global motion planner is proposed in this paper for the closed-chain systems, and motions in different disconnected manifolds are efficiently bridged by two type regrasping moves. The regrasping moves are automatically chosen by the planner based on cost-saving principle, which greatly improve the success rate and efficiency. Furthermore, to obtain the optional inverse kinematic solutions satisfying joint physical limits (e.g., joint position, velocity, acceleration limits) in the planning, the redundancy resolution problem for dual redundant robots is converted into a unified quadratic programming problem based on the combination of two diff erent-level optimizing criteria, i.e. the minimization velocity norm (MVN) and infinity norm torque-minimization (INTM). The Dual-MVN-INTM scheme guarantees smooth velocity, acceleration profiles, and zero final velocity at the end of motion. Finally, the planning results of three complex closed-chain manipulation task using two Franka Emika Panda robots and two Kinova Jaco2 robots in both simulation and experiment demonstrate the effectiveness and efficiency of the proposed method.

1. Introduction

Complex coordinated robotic tasks, such as moving large or heavy objects, are making multi-arm robotic systems necessary. In addition, redundant robots are more flexible and suitable for complex tasks because that the redundant degrees of freedom (DoFs) can be utilized to avoid joint limits or obstacles. Thus, the multi-redundant robotic systems undoubtedly can complete complex tasks more flexibly. However, the additional complexity leads to difficulties in planning and control, which basically boils down to the closed kinematic chain (CKC) constraint and redundancy-resolution problem.

The greatest challenge in the planning is the CKC constraint, which reduces the system's configuration space to sub-manifolds of lower dimension [1, 2]. As a result, the probability of directly sampling a valid configuration tends to be zero, which makes it difcult to directly utilize existing sampling-based planners, such as PRM [3] and RRT [2]. Previous solution is projecting configurations onto the constraint surface [4, 5, 6, 7, 8, 9]. However, such methods introduce undesirable restrictions or computation costs [10]. Moreover, the global path is generated only when feasible configurations can be connected sequentially. Due to the joint limits, one configuration is only connectable to a certain set of adjacent configurations. For a complex closed-chain task, there could be several connectable subsets, or named

connected components (CCs) [11], between the start and the target. Most previous methods focus on solving motion planning in a single CC for relatively simple tasks. In this paper, we solve the closed-chain motion planning at the global level. The $SE(3)$ object trajectory is interpolated first and dual-robot trajectories are then computed using a proposed quadratic programs (QP) based inverse kinematic (IK) solver. Regrasping moves, IK-switch, and Grasp-switch are integrated in a unified regrasping framework to connect different CCs.

In order to obtain the joint trajectory of the redundant robot in the planning, the IK problem or termed redundancy-resolution problem needed to be addressed. The traditional solution is the pseudoinverse formulation [12]. Optimization methods, especially QP-based methods, are preferred in recent years for their fast computation speed [13]. Many optimization criteria [14, 15, 16, 17, 18, 19, 20, 21, 22, 23] have been proposed to obtain optional IK solutions satisfying the physical limits. Nevertheless, these methods mainly aim at solving the redundancy resolution problem for a single robot [14, 15, 16, 17, 18, 19, 20, 21] or simple 3D position trajectory tracking tasks using 6 DOF robots [22, 23]. In this paper, a bi-criteria QP-based Dual-MVN-INTM scheme is proposed and integrated in the motion planning for closed-chain manipulation task with dual-redundant robots. The results are verified on 6D end-effector trajectory tracking tasks. The QPs of the two robots are unified into one single QP and solved by a simplified dual neural network (DNN) efficiently. The proposed scheme can remedy the discontinuity phenomenon and guarantee zero final velocities.

In summary, the main contributions of this paper are three-folds: 1) a global dual-redundant-robot closed-chain motion planner is proposed to solve difficult planning tasks. IK-switch or Grasp-switch regrasping moves are flexibly chosen to connect disconnected components. 2) A Dual-MVN-INTM redundancy resolution scheme is presented for complex 6D trajectory tracking in the planning, which computes the smooth joint trajectory that satisfy physical limits with a real-time speed. 3) The integration of Dual-MVN-INTM scheme for dual-redundant robots and the efficient regrasping mechanism greatly improve the planning performance. Our planner outperforms the previous planner [11] by 27% on success rate with a $1.78\times$ faster speed and 56% less required regrasping moves. The remainder of this paper is organized as follows. Section 2 discusses related works. Section 3 formulates the closed-chain manipulation task. Section 4 illustrates the global motion planner in detail. The trajectory generation and Dual-MVN-INTM scheme is presented in Section 5. Simulation and experiment results are given and discussed in Section 6. Section 7 concludes this paper.

2. Related works

2.1. Motion planning for closed-chain manipulation

The motion planning problem for CKC systems is to find a path that satisfies robot limits, CKC constraint, and collision avoidance to connect the start and goal configuration [10]. The low dimensionality of the constraint manifold makes this problem hard to solve [3]. Common approach for sampling valid configurations is through projection. For instance, [5, 6] break the CKC into open sub-chains, and sample the configuration for the active chain and compute for the passive chain under CKC constraint. However, this approach introduces artificial singularities, and the passive chain need to be non-redundant [10]. Newton-Raphson projection method [24] iteratively reduces the projecting residual, which achieves better success rate but slower speed. Furthermore, configurations are sampled on a tangent space of the constraint manifold to improve projection performance in recent works [7, 8, 9]. After feasible configurations are obtained, projection method [2, 25] or differential IK [5, 26] is used for local connections. However, these methods aim to address planning in a single connected component. Global-level planning, i.e., planning paths that connect different components, is required for solving complex closed-chain tasks. Gharbi et al. [27] solve this problem via singular configurations, but full knowledge of IK classes

https://doi.org/10.1017/S0263574721000941 Published online by Cambridge University Press

characterization is need. Koga et al. [28] first use regrasping to get away the local minimum in the randomized potential field but joint limits are ignored. More recent work [11] utilizes regrasping moves between different IKs of a same grasp pose (IK-switch) to bridge different components. Although the result is promising, only a single grasp pose greatly limits the flexibility of manipulating objects and thus reduces the success rate. Moreover, the planner in [11] is designed for nonredundant robots, and the IK is obtained by closed-form solution using IK fast [29], which restricts its extension to redundant robots. In this paper, we extend the global planner by integrating both IK-switch and Grasp-switch moves in the planning. Our planner can flexibly choose the two approaches efficiently to improve the success rate of planning. Furthermore, we use dual-redundant robots in view of their better efficacy and flexibility, and the IKs that satisfy joint limits are obtained by a proposed QP-based IK solver quickly.

There exist more recent works solving motion planning of multi-arm robot systems. Preda et al. [30] develop a motion planning architecture for dual-arm surgical robot completing a suturing task. Xu et al. [31] present a coordinated motion planning approach for a dual-arm space robot capturing the target. However, both suturing task in [30] and capturing task in [31] involve separate motions for the two arms only, and thus closed-chain motions are not considered. Similar problem exists in [32], where the authors use a dual-arm robot to perform human-demonstrated tasks without closed-chain motions. Optimization-based approaches are also proposed to address motion planning with closed kinematics. For example, the planning problem is formulated and solved as an optimization problem in [33] and [34], but these methods lack the ability to solve planning at a global level. A recent interesting work presented in [35] utilizes grasp-hold changes to address human-robot collaboration tasks (joint actions for rotating the object), where the key is adapting motions according to the partner's intension. Whereas we use grasp changes to bridge different constraint components in the global dual-robot closed-chain motion planning, which results in an improved planning performance.

2.2. Inverse kinematics (redundancy resolution) of redundant robot

A fundamental issue in motion planning and control of redundant robots is the IK (redundancy resolution) problem, since that infinite IK solutions exist due to the redundancy [13, 36]. Conventional solution is the pseudo-inverse-based formulation. However, the computation is time consuming, and joint inequality constraints are difficult to be formulated [24]. Therefore, to overcome the shortcomings, many works formulate the redundancy resolution as a QP problem built on different optimization criteria, such as minimum velocity norm (MVN) [14], infinity norm velocity minimization (INVM) [37], minimum acceleration norm [38], infinity-norm acceleration minimization (INAM) [39], infinity-norm torque minimization (INTM) [40], and so on. In order to meet multiple requirements in more complex applications, multi-criteria optimization methods that combine different schemes are proposed, such as MVN-INVM [41], bi-criteria torque optimization [18], two-infinity norm switching [42], MVN-MTN [16], and bi-criteria pseudoinverse minimization [43]. However, although proved effective in simple tasks using a single manipulator, these methods lack the ability to solve the multi-robot planning and control problem. Some recent works consider the coordination motion of dual-redundant-robot [22, 23], but their methods are only designed for predefined 3D position trajectory tracking tasks using two 6 DoF robots in a simulation environment. As a contrast, optimization-based redundancy resolution for dual-robot in random 6D motion planning is rarely studied. In this paper, in order to make a more effective utilization of input power in the 6D manipulation task of large object and remedy joint torque instability of INTM, two QPs for two robots are formulated combining MVN and INTM scheme, which is further unified into one QP for simultaneous control of dual arms. Our optimization scheme, termed Dual-MVN-INTM, is organically integrated in the closed-chain motion planning and guarantees smooth planned joint trajectories that satisfy joint limits. Escande et al. [44] present a hierarchical QP solution for generating humanoid-robot motions with both equality and inequality constraints considered. However, the start and target configurations can be locally connected in their close-chain task, which means their method is probably not suitable for the global-level tasks discussed in this paper.

https://doi.org/10.1017/S0263574721000941 Published online by Cambridge University Press

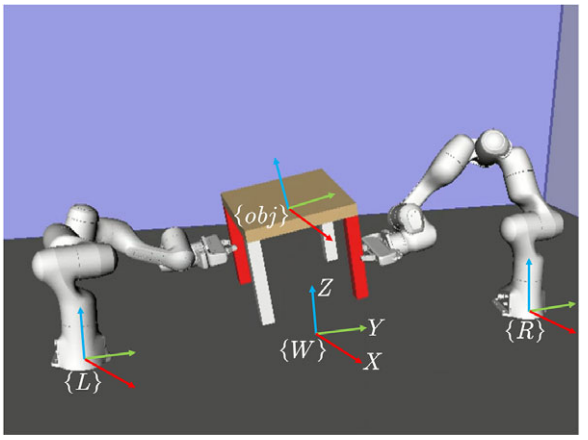

Figure 1. *A CKC system consisting of two redundant robots and an object, where* $\{L\}$ *and* $\{R\}$ *denote the base frame of the left robot and right robot, respectively,* $\{W\}$ *denotes the world frame, and* $\{obj\}$ *denotes the object frame.*

The formulated QPs that subject to both robot equality and inequality constraints are complicated and large-scale, which makes it difficult and inefficient for traditional numerical QP solvers to solve [13, 18]. Therefore, recurrent neural networks (RNNs) are widely applied due to their efficient parallel computational nature [13]. Different types of RNNs that enable real-time applications are utilized to solve the QP-formed IK problem, such as the Lagrangian neural network (LNN) [45], the primal-dual neural network (PDNN) [46], the DNN [47], linear variational inequalities-based PDNN (LVI-PDNN) [19], simplified LVI-PDNN (S-LVI-PDNN) [22], and simplified DNN (S-DNN) [18]. Among these RNNs, DNN uses the dual decision variables only and directly uses Karush-Kuhn-Tucker (KKT) condition [48] and projection operator to reduce network complexity and increase efficiency. S-DNN further reduces the complexity of DNN while preserves the desirable convergence property. Thus, we formulate an S-DNN model in this paper for the real-time solving of our Dual-MVN-INTM scheme, which makes sure the computation efficiency of our method.

3. Closed-chain manipulation task modeling

In this section, we formulate the motion planning problem of closed-chain systems. Moreover, we present two regrasping moves, i.e., IK-switch and Grasp-switch, to efficiently and flexibly connect constraint components caused by the closed-chain constraint, such that the global motion planning can be addressed.

3.1. Motion planning problem of closed-chain systems

Consider a CKC system (Fig. 1) consisting of two redundant robots and a moveable object. Let $C^i_{\text{robot}} \subseteq \mathbb{R}^{n_i}$ denote the configuration space of the i^{th} robot, where n_i is the number of DoF. If consider the position and orientation of the end-effector, the dimension of task space $m = 6$, and the redundancy for a 7 DoF robot is 1. Let $C_{\text{obj}} \subseteq SE(3)$ be the object's configuration space, then the system's composite configuration space can be expressed as $C_{\text{composite}} = C^{\text{o}}_{\text{robot}} \times C^{1}_{\text{robot}} \times C_{\text{obj}}$. A composite configuration is denoted as $c = (\theta_{\text{o}}, \theta_1, T_{\text{obj}}) \in C_{\text{composite}}$, where $\theta_i \in C^i_{\text{robot}}$ is the configuration of the i^{th} robot, and $T_{\text{obj}} \in C_{\text{obj}}$ is the configuration of the object.

The CKC is formed when two robots manipulating a single object and can be expressed as nonlinear equations $g(c) = 0$. Hence, the set of all composite configurations that satisfy CKC is a subset C_{cc}:

$$C_{cc} = \{c | c \in C_{\text{composite}}, g(c) = 0\} \tag{1}$$

https://doi.org/10.1017/S0263574721000941 Published online by Cambridge University Press

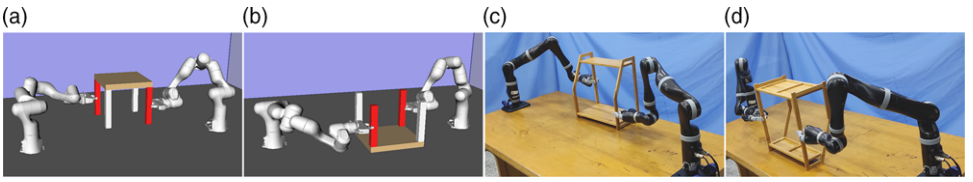

Figure 2. *Start and goal configurations in two CKC motion planning problems.* (a, c) *: Start configurations.* (b, d) *: Goal configurations. There are no continuous feasible paths without breaking the closed chains.*

Note that in this case, the grasping pose T_{Grasp} of the robots, i.e., the relative transformations between the robot end-effectors and the object, can be uniquely determined by the composite configuration c:

$$T_{\text{Grasp}_i} = T_{\text{obj,ef}_i} = T_{\text{obj}}^{-1} \cdot T_{\text{ef}_i}(\theta_i) \tag{2}$$

where $T_{\text{ef}_i}(\theta_i)$ is the transformation of the i^{th} robot's end-effector. Define a projection $f{:}C_{\text{composite}} \rightarrow C_{\text{obj}}$ that maps composite configurations to object configurations, such that given $c = (\theta_{\text{o}}, \theta_1, T_{\text{obj}}), f(c) = T_{\text{obj}}$. Moreover, let $C_{\text{b}} \subset C_{\text{composite}}$ denote the restricted region due to obstacles and joint position limits, then the free configuration space can be expressed as $C_{\text{free}} = C_{\text{composite}} \backslash C_{\text{b}}$. Accordingly, the set of feasible composite configurations that respect the CKC constraint, joint position limits, and are collision-free can be defined as

$$C_{\text{feasible}} = \{c | c \in C_{\text{cc}} \cap C_{\text{free}}\} \tag{3}$$

With the definitions determined above, the motion planning problem of a CKC system can be described as follows. Given a start composite configuration $c_{\text{start}} \in C_{\text{feasible}}$ and a goal object configuration $T_{\text{obj}_{\text{goa1}}} \in C_{\text{obj}}$, find a continuous path $P{:}[0, 1] \rightarrow C_{\text{feasible}}$ such that $P(0) = c_{\text{start}}, f(P(1)) = T_{\text{obj}}$.

3.2. Constraint manifolds and regrasping moves

The CKC constraint reduces the composite configuration space to a lower-dimensional constraint manifold in the ambient space. Due to the physical limits, the robots sometimes cannot move from one configuration to the next one while satisfying the CKC constraint. Thus, the definition of *essentially mutually disconnected* (EMD) [11] is introduced: Given a composite configuration $c \in C_{\text{feasible}}$, the set of all feasible configurations that can be reached from c by continuous and feasible paths is referred to as the *connected component* $S(c)$. Two component components, $S(c_1)$ and $S(c_2)$, are EMD if they are indeed disconnected ($S(c_1) \cap S(c_2) = \emptyset$) or they could not be connected in a reasonable amount of time. For the planning queries shown in Fig. 2, the components containing the start and the goal are disconnected, thus the system cannot realize the required motion without breaking the CKC.

The key of solving the planning at global level is how to plan motions crossing different EMD components. As shown in Fig. 3, there exists two ways to connect different EMD components: a. IK- switch: regrasping moves between different IK solutions of a same grasping pose. b. Grasp-switch: regrasping moves between different grasping poses for the object. Different IK solutions or grasping poses can correspond to different EMD components, hence the conversion between different IK solutions or grasping poses can realize the required connection. However, for a specific object configuration, IK solving may fail due to the limits such as obstacle avoidance or robot physical limits. In this case, the Grasp-switch method and the utilization of redundant robots become necessary. In addition, Grasp-switch can also be an optional choice for more flexible connections of EMDs in the planning. Therefore, the two methods are integrated into a unified regrasping framework, more details are introduced in Section 4.3.

https://doi.org/10.1017/S0263574721000941 Published online by Cambridge University Press

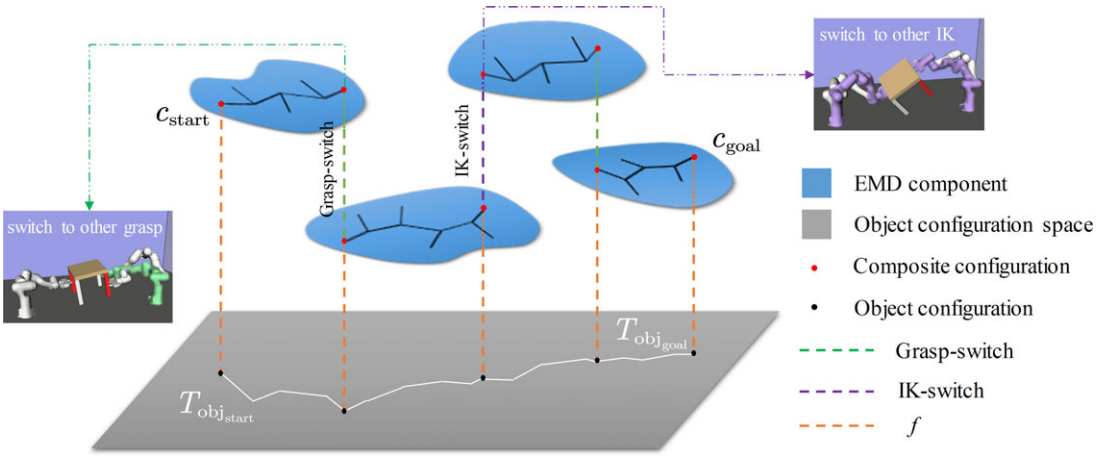

Figure 3. *Connecting different EMD components using IK-switch moves and Grasp-switch moves.*

4. Global motion planning for closed-chain manipulation

In this section, we illustrate the proposed global closed-chain motion planner. The brief overview of the algorithm is given first. Then the two stages of the planner, i.e., closed-chain motion planning and regrasping motion planning are presented in details. The closed-chain motions generated by the first stage are flexibly connected by IK-switch or grasp-switch moves to solve the task.

4.1. Overview of planning algorithm

The planner follows the classical sampling-based RRT architecture. As shown in Fig. 3, in the first stage (Section 4.2), the planer samples and plans the $SE(3)$ path for the object, and the object trajectory is computed via *Quintic Polynomial Interpolation* (Section 5.1). The QP-based differential IK algorithm (Section 5.2) is then used to map the $SE(3)$ trajectory to the joint space quickly. When the IK solution fails at some $SE(3)$ configurations due to the CKC constraint, the planner computes the valid regrasping configuration for IK-switch or Grasp-switch moves at these nodes. The output of the first stage will be several closed-chain motion segments with regrasping requests. The second stage of the planner (Section 4.3) deals with these requests and connects the motion segments via IK-switch or Grasp-switch moves to generate the global trajectory.

4.2. Closed-chain motion planning

Different from [11] that only a single grasp pose is utilized, for the start configuration c_{start} and the object goal configuration T_{goa1}, the goal configuration c_{goa1} is computed using the grasp pose of the last node in this paper. The Graspit! simulator is used to generate valid grasp poses for a certain manipulation object, and the top ranked m grasps are saved in a predefined grasp pose list Grasplist (4) after testing and selecting.

$$\text{Grasplist} = [[T^0_{\text{Grasp}_0}, T^0_{\text{Grasp}_1}], \cdots, [T^{m-1}_{\text{Grasp}_0}, T^{m-1}_{\text{Grasp}_1}]] \tag{4}$$

The pseudo-code of the closed-chain planning algorithm is shown in Algorithm 1. The algorithm takes the initial node $\mathcal{V}_{\text{start}}$ and the target node $\mathcal{V}_{\text{goa1}}$ as root nodes to grow the forward tree $\mathcal{T}_{\text{f}}$. and the backward tree $\mathcal{T}_b$, where $\mathcal{V}_{\text{start}}$ and $\mathcal{V}_{\text{goa1}}$ are endowed with the initial grasping pose Grasplist [0]. Brief descriptions of some key functions in Algorithm 1 are given below.

https://doi.org/10.1017/S0263574721000941 Published online by Cambridge University Press

Algorithm1: Closed-Chain Motion Planner

ClosedChainPlanner($c_{\text{start}}, c_{\text{goal}}, N_{\max}, R_{\max}$)

```
1.  V_start.c ← c_start; V_goal.c ← c_goal
2.  V_start .Grasp ← Grasplist[0]; V_goal .Grasp ← Grasplist[0]
3.  T_f.init(V_start); T_b.init(V_goal)
4.  for i=1 to N_max do
5.      T_rand ← SampleSE3Config()
6.      V_new ← Extend(T_f, T_rand, R_max)
7.      if V_new is not None then
8.          P_connect ← Connect(V_new, T_b, R_max)
9.          if P_connect is not None then
10.             if PlanRegrasp(T_f, T_b, V_fail) then
11.                 return GeneratePath(T_f, T_b, P_connect)
12.             else
13.                 Reorganize(T_f, T_b, P_connect)
14.     Swap(T_f, T_b)
15. return Failure
```

- SampleSE3Config() This function samples the object configuration $T_{\text{rand}} = (R, p)$ in $SE(3)$, where R is uniformly sampled in SO (3), and p is uniformly sampled within a allowed range for the task.
- Extend() This function expands $\mathcal{T}_{\text{f}}$ toward T_{rand} and generates new nodes. In Algorithm 2, NearestNeighbor() searches over all nodes in $\mathcal{T}_{\text{f}}$ and finds the node with an object configuration nearest to T_{rand} and a regrasping number less than $R_{\max}$. For $T_0 = (R_0, p_0)$ and $T_1 = (R_1, p_1)$, the minimal distance between R_0 and R_1 in SO (3) is the Euclidean length $\|r\|$, where

$$[r] = \log\left(R_0^{\text{T}} R_1\right) \tag{5}$$

$[r]$ denotes the skew-symmetric of $r \in \mathbb{R}^3$. The distance between T_0 and T_1 in $SE(3)$ is then defined as the combination of $\|r\|$ and the Euclidean distance between the translation vectors:

$$d = \left(\|r\|^2 + \|p_0 - p_1\|^2\right)^{1/2} \tag{6}$$

NewSE3Config() generates the new object configuration at a predetermined step size in the direction from $f(\mathcal{V}_{\text{near}}.c)$ to T_{rand}. InterpolateSE3Traj() interpolates a collision-free object $SE(3)$ trajectory $\mathcal{P}_{\text{SE3}}$ connecting $f(\mathcal{V}_{\text{near}}.c)$ and T_{new}, then ComputeCTraj() is called to compute a closed-chain composite trajectory $\mathcal{P}_{\text{composite}}$ tracking $\mathcal{P}_{\text{SE3}}$. When the end of $\mathcal{P}_{\text{composite}}$ needs regrasping, the regrasping configuration is assigned to $\mathcal{V}_{\text{new}}$. Finally, $\mathcal{V}_{\text{new}}$ is added to the tree $\mathcal{T}_{\text{f}}$.

- Connect() This function tries to connect the two trees via $\mathcal{V}_{\text{new}}$ and returns the connected trajectory $\mathcal{P}_{\text{connect}}$. The procedure is similar with the function Extend() and shown in Algorithm 3. The difference is that the extend goal is the given node instead of a new sampled one.
- InterpolateSE3Traj() This function computes a smooth $SE(3)$ trajectory connecting the object configuration $f(\mathcal{V}_{\text{nem}}.c)$ and T_{new}. Details are illustrated in Section 5.1.
- ComputeCTraj(): As shown in Algorithm 4, this function computes the closed-chain composite trajectory $\mathcal{P}_{\text{composite}}$ to track $\mathcal{P}_{\text{SE3}}$. DifferentialIK() (Section 5.2) is called to compute the IK solutions for dual redundant robots following $\mathcal{P}_{\text{SE3}}$. When the IK solution fails at certain T_{obj}, GetRegraspConfig()

https://doi.org/10.1017/S0263574721000941 Published online by Cambridge University Press

Algorithm 2: Extend Tree

Extend($\mathcal{T}_{\mathrm{f}}, T_{\mathrm{rand}}, R_{\max}$**)**

$\mathcal{V}_{\mathrm{near}} \leftarrow$ NearestNeighbor($\mathcal{T}_{\mathrm{f}}, T_{\mathrm{rand}}, R_{\max}$)
$T_{\mathrm{new}} \leftarrow$ NewSE3Config($f(\mathcal{V}_{\mathrm{near}}.c), T_{\mathrm{rand}}$)
if CollisionFree(T_{new}) and IsReachable(T_{new}) **then**
    $\mathcal{P}_{\mathrm{SE3}} \leftarrow$ InterpolateSE3Traj($f(\mathcal{V}_{\mathrm{near}}.c), T_{\mathrm{new}}$)
    if $\mathcal{P}_{\mathrm{SE3}}$ **is not** None **then**
        $\{status, \mathcal{P}_{\mathrm{composite}}, c_{\mathrm{regrasp}}, c_{\mathrm{orig}}, index\} \leftarrow$ ComputeCTraj ($\mathcal{V}_{\mathrm{near}}, \mathcal{P}_{\mathrm{SE3}}$)
        $Grasp \leftarrow \mathcal{V}_{\mathrm{near}}.Grasp$
        if *status* == REACHED **then**
            $\mathcal{V}_{\mathrm{new}} \leftarrow$ Vertex($T_{\mathrm{new}}, Grasp, \mathcal{P}_{\mathrm{composite}}$)
            $\mathcal{V}_{\mathrm{new}}.parent \leftarrow \mathcal{V}_{\mathrm{near}}$
            $\mathcal{T}_f$.AddVextex($\mathcal{V}_{\mathrm{new}}$)
            return $\mathcal{V}_{\mathrm{new}}$
        elif $\mathcal{V}_{\mathrm{near}}$.RegraspCount $< R_{\max}$ **and** *status* **is not** TRAPPED **then**
            if *status* == NEED_IKSWITCH **then**
                $\mathcal{V}_{\mathrm{new}}.NeedIKswitch \leftarrow$ True
            elif *status* == NEED_GRASPSWITCH **then**
                $Grasp \leftarrow Grasplist[index]$
                $\mathcal{V}_{\mathrm{new}}.NeedGraspswitch \leftarrow$ True
            $\mathcal{V}_{\mathrm{new}} \leftarrow$ Vertex($f(c_{\mathrm{regrasp}}), Grasp, \mathcal{P}_{\mathrm{composite}}$)
            $\mathcal{V}_{\mathrm{new}}.parent \leftarrow \mathcal{V}_{\mathrm{near}}$
            $\mathcal{V}_{\mathrm{new}}.RegraspCount \leftarrow \mathcal{V}_{\mathrm{new}}.Parent.RegraspCount + 1$
            $\mathcal{V}_{\mathrm{new}}.c_{\mathrm{orig}} \leftarrow c_{\mathrm{orig}}$
            $\mathcal{T}_f$.AddVertex($\mathcal{V}_{\mathrm{new}}$)
            return $\mathcal{V}_{\mathrm{new}}$
return None

is used to compute the valid regrasping configuration. The final output of ComputeCTraj() will be the trajectory starting from $\mathcal{V}_{\mathrm{start}}$ and following longest possible along $\mathcal{P}_{\mathrm{SE3}}$.

4.3. Regrasping motion planning

When the first stage successfully connects $\mathcal{T}_{\mathrm{f}}$ and $\mathcal{T}_{\mathrm{b}}$, and returns the trajectory $\mathcal{P}_{\mathrm{connect}}$, the global path is found. At this point, the algorithm turns to plan the regrasping motion for the nodes with regrasping requests. The whole regrasping process is divided into three stages: a). Two robots transfer the object from the current configuration T_{cur} to the placement configuration T_{place}. b). The robot that needs regrasping performs the regrasping motion. c). Two robots move the object back to T_{cur}. The pseudo-code is shown in Algorithm 5. Some key functions are briefly described as follows.

- ExtractGlobalVertices(): Starting from the intersection node of $\mathcal{T}_{\mathrm{f}}$ and $\mathcal{T}_{\mathrm{b}}$, this function extracts the parent node of the node until $\mathcal{V}_{\mathrm{start}}$ and $\mathcal{V}_{\mathrm{goa1}}$, and finally returns all nodes in the global path.

https://doi.org/10.1017/S0263574721000941 Published online by Cambridge University Press

Algorithm 3: Connect Trees

Connect($\mathcal{V}_{\text{new}}, \mathcal{T}_{\text{b}}, R_{\max}$)

$\mathcal{V}_{\text{near}} \leftarrow \text{NearestNeighbor}\left(\mathcal{T}_{\text{b}}, f(\mathcal{V}_{\text{new}}.c), R_{\max} - \mathcal{V}_{\text{new}}.\text{RegraspCount}\right)$
$\mathcal{P}_{\text{SE3}} \leftarrow \text{InterpolateSE3Traj}\ (f(\mathcal{V}_{\text{near}}.c), f(\mathcal{V}_{\text{new}}.c))$
if $\mathcal{P}_{\text{SE3}}$ **is not** None **then**
  $\{status, \mathcal{P}_{\text{connect}}, c_{\text{regrasp}}, c_{\text{orig}}\} \leftarrow$ ComputCTraj ($\mathcal{V}_{\text{near}}, \mathcal{P}_{\text{SE3}}$)
  if $status ==$ REACHED **then**
    if $\mathcal{P}_{\text{connect}}.c_{\text{end}} == \mathcal{V}_{\text{new}}.c$ **then**
      return $\mathcal{P}_{\text{connect}}$
    elif $\mathcal{V}_{\text{new}}$.RegraspCount $+ \mathcal{V}_{\text{near}}$.RegraspCount $< R_{\max}$ **then**
      if $\mathcal{V}_{\text{new}}.Grasp$ **is not** $\mathcal{V}_{\text{near}}.Grasp$ **then**
        $\mathcal{P}_{\text{connect}}.NeedGraspswitch \leftarrow$ True
      else
        $\mathcal{P}_{\text{connect}}.NeedIKswitch \leftarrow$ True
      return $\mathcal{P}_{\text{connect}}$
return None

- SamplePlacementConfig(): This function samples a valid placement configuration T_{place} that close to the object configuration $f(\mathcal{V}_C)$, which depends on the environment and is similar to the one in [11].
- ComputeCTraj(): In stage a), this function computes the closed-chain trajectory $\mathcal{P}_{\text{before}}$ from current configuration $f(\mathcal{V}_C)$ to T_{place}. The grasping poses are set to the ones corresponding to the current node's parent node. In stage c), the grasping poses are same with the current node (has been changed by GetRegraspConfig()). The returned trajectory moves the object from $f(\mathcal{V}_C)$ to T_{place}, then the order of the path points is reversed to get the correct trajectory (from T_{place} to $f(\mathcal{V}_C)$).
- GetRegraspConfig(): This function is the key of integrating both IK-switch and Grasp-switch into a unified regrasping framework. As shown in Algorithm 6, when regrasping is needed, it first tries to compute the regrasping configuration using IK-switch (with the grasping poses unchanged), then the grasping pose is changed to each one in Grasplist in turn. The configuration farthest from the joint position limits is selected. All the configurations corresponding to IK-switch and Grasp-switch are compared and the one closest to the original configuration is chosen as the regrasping configuration. Finally, the regrasping configuration, the regrasping type, and the grasping pose index are returned.
- BiRRT() In stage c), this function plans the regrasping path. The start point is the robot configuration the end of $\mathcal{P}_{\text{before}}$, and the goal is the robot configuration corresponding to the start of $\mathcal{P}_{\text{after}}$.

5. Trajectory generation for dual redundant robot

In this section, we illustrate details of trajectory generation in the planning. The 6D $SE(3)$ trajectory is first interpolated using quintic polynomial interpolation for both rotation part and translation part. Then we present the Dual-MVN-INTM scheme to convert the $SE(3)$ trajectory to the joint space for both redundant robots. Two QPs are formulated, unified, and solved to obtain smooth joint trajectories that satisfy robot physical limits with a real-time speed.

https://doi.org/10.1017/S0263574721000941 Published online by Cambridge University Press

Algorithm 4: Track Object Trajectory

ComputeCTraj($\mathcal{V}_{\text{start}}, \mathcal{P}_{\text{SE3}}, c_{\text{inter}} \leftarrow \text{None}, Grasp_0 \leftarrow \text{None}$**)**

$L_{\text{T}} \leftarrow$ Discretize($\longleftarrow$)
$L_{\text{c}} \leftarrow$ EmptyList()
$c_{\text{prev}} \leftarrow \mathcal{V}_{\text{start}}.c$
$Grasp \leftarrow \mathcal{V}_{\text{start}}.Grasp$
if c_{inter} **is not** None **then**
    $c_{\text{prev}} \leftarrow c_{\text{inter}}$
if $Grasp_0$ **is not** None **then**
    $Grasp \leftarrow Grasp_0$
for each T_{obj} **in** L_{T} **do**
    $c_{\text{next}}.T_{\text{obj}} \leftarrow T_{\text{obj}}$
    for $i \leftarrow 1$ **to 2 do**
        $c_{\text{next}}.\theta_i \leftarrow \text{DifferentialIK}(c_{\text{prev}}.\theta_i, T_{\text{obj}}, \text{Grasp})$
        if $c_{\text{next}}.\theta_i == \text{None}$ **then**
            $\{c_{\text{regrasp}}, \text{RegraspType}, \text{index}\} \leftarrow \text{GetRegraspConfig}(\mathcal{V}_{\text{start}}, c_{\text{prev}}, i)$
            if c_{regrasp} **is not None then**
                $c_{\text{orig}} \leftarrow c_{\text{prev}}$
                $\mathcal{P}_{\text{composite}} \leftarrow \text{CompositePath}(L_{\text{c}})$
                if $\text{RegraspType} == \text{IKswitch}$ **then**
                    return NEED_IKSWITCH, $\mathcal{P}_{\text{composite}}, c_{\text{regrasp}}, c_{\text{orig}}, \text{index}$
                elif $\text{RegraspType} == \text{Graspswitch}$ **then**
                    return NEED_GRASPSWITCH, $\mathcal{P}_{\text{composite}}, c_{\text{regrasp}}, c_{\text{orig}}, \text{index}$
            else
                return TRAPPED, None, None, None
        L_c.Append(c_{next})
        $c_{\text{prev}} \leftarrow c_{\text{next}}$
$\mathcal{P}_{\text{composite}} \leftarrow \text{CompositePath}(L_{\text{c}})$
return REACHED, $\mathcal{P}_{\text{composite}}$, None, None

5.1. Trajectory interpolation in SE(3)

The rotational motion and translation motion are decoupled in the interpolation. For the rotation part, for two rotation matrices $R_{\text{o}}, R_f \in SO(3)$, the corresponding angular velocity vectors $\omega_0, \omega_f \in \mathbb{R}^3$ and the angular acceleration vectors $\dot{\omega}_{\text{o}}, \dot{\omega}_f \in \mathbb{R}^3$, the smooth trajectory $[R(t)]_{t\in[0,t_f]}$ in SO(3) connecting R_{o} and R_f is computed through *quintic polynomial interpolation* for the axis-angle

$$R(t) = R_0 e^{[r]}, r = \sum_{i=0}^{5} a_i t^i \tag{7}$$

where $a_0, a_1, a_2, a_3, a_4, a_5 \in \mathbb{R}^3$ are constant vectors and can be computed through the following boundary conditions

https://doi.org/10.1017/S0263574721000941 Published online by Cambridge University Press

Algorithm 5: Regrasping Motion Planner

PlanRegrasp($\mathcal{T}_{\mathrm{f}}, \mathcal{T}_{\mathrm{b}}, \mathcal{V}_{\mathrm{fail}}$)

for each $\mathcal{V}$ **in** ExtractGlobalVertices($\mathcal{T}_{\mathrm{f}}, \mathcal{T}_{\mathrm{b}}$) **do**
  if $\mathcal{V}.NeedIKswitch$ **or** $\mathcal{V}.NeedGraspswitch$ **then**
    if not $\mathcal{V}.HasRegrasp$ **then**
      success $\leftarrow$ False
      for $c_{\mathrm{next}}.\theta_i \leftarrow \mathrm{DifferentialIK}(c_{\mathrm{prev}}.\theta_i, T_{\mathrm{obj}}, \mathrm{Grasp})$ **to** $N_{\max}$ **do**
        $T_{\mathrm{place}} \leftarrow \mathrm{SamplePlacementConfig}(f(\mathcal{V}.c))$
        if T_{place} **is** None **then**
          $\mathcal{V}_{\mathrm{fail}} \leftarrow \mathcal{V}$
          return False
        $\mathcal{P}_{\mathrm{SE3}} \leftarrow \mathrm{InterpolateSE3Path}\ (f(\mathcal{V}.c), T_{\mathrm{place}})$
        $\{status, \mathcal{P}_{\mathrm{before}}\} \leftarrow \mathrm{ComputeCTraj}(\mathcal{V}, \mathcal{P}_{\mathrm{SE3}}, \mathcal{V}.c_{\mathrm{orig}}, \mathcal{V}.Parent.\mathrm{Grasp})$
        if *status* **is not** reached **then**
          continue
        $\{status, \mathcal{P}_{\mathrm{after}}\} \leftarrow \mathrm{ComputeCTraj}(\mathcal{V}, \mathcal{P}_{\mathrm{SE3}}, \mathcal{V}.c)$
        if *status* **is not** reached **then**
          continue
        $\mathcal{P}_{\mathrm{after}} \leftarrow \mathcal{P}_{\mathrm{after}}[::-1]$
        for $c_{\mathrm{next}}.\theta_i \leftarrow \mathrm{DifferentialIK}(c_{\mathrm{prev}}.\theta_i, T_{\mathrm{obj}}, \mathrm{Grasp})$ **to** 2 **do**
          $\mathcal{P}_{\mathrm{regrasp}}.\mathrm{append}(\mathrm{BiRRT}(\mathcal{P}_{\mathrm{before}}[-1].c.\theta_i, \mathcal{P}_{\mathrm{after}}[0].c.\theta_i))$
        if $\mathcal{P}_{\mathrm{regrasp}}$ **is** None **then**
          continue
        $\mathcal{V}$.HasRegrasp $\leftarrow$ True
        $\mathcal{V}$.AddRegraspAction($\mathcal{P}_{\mathrm{before}}, \mathcal{P}_{\mathrm{after}}, \mathcal{P}_{\mathrm{regrasp}}$)
        success $\leftarrow$ True
        break
      if not *success* **then**
        $\mathcal{V}_{\mathrm{fail}} \leftarrow \mathcal{V}$
        return False
return True

$$
\begin{aligned}
r_0 &= a_0 = 0_{3\times 1} \\
r_f &= a_0 + a_1 t_f + a_2 t_f^2 + a_3 t_f^3 + a_4 t_f^4 + a_5 t_f^5 \\
\dot{r}_0 &= a_1 = \omega_0 \\
\dot{r}_f &= a_1 + 2a_2 t_f + 3a_3 t_f^2 + 4a_4 t_f^3 + 5a_5 t_f^4 = \Lambda^{-1}(r_f)\omega_f \\
\ddot{r}_0 &= 2a_2 = \dot{\omega}_0 \\
\ddot{r}_f &= 2a_2 + 6a_3 t_f + 12a_4 t_f^2 + 20a_5 t_f^3 = \Lambda^{-1}(r_f)(\dot{\omega}_f - \Omega(r_f, \dot{r}_f))
\end{aligned} \tag{8}
$$

https://doi.org/10.1017/S0263574721000941 Published online by Cambridge University Press

Algorithm 6: Generate Regrasping Configuration

GetRegraspConfig($\mathcal{V}_{\text{near}}$, c, i)

RegraspType $\leftarrow$ IKswitch
$c_{\text{regrasp}}.T_{\text{obj}} \leftarrow f(c)$
$\theta_{\text{IK}} \leftarrow \text{SelectIK}(\text{IKfast}(f(c), i, \text{Grasp}))$
$\theta_{\text{best}} \leftarrow \theta_{\text{IK}}$
Graspindex $\leftarrow$ None
for $k \leftarrow 0$ **to** m **do**
  $Grasp \leftarrow Grasplist[k]$
  $\theta_{\text{Grasp}} \leftarrow \text{SelectIK}(\text{IKfast}(f(c), i, \text{Grasp}))$
  $\theta_{\text{best}} \leftarrow \text{mindistance}(\theta_{\text{best}}, \theta_{\text{Grasp}}, c.\theta_i)$
$c_{\text{regrasp}}.\theta_i \leftarrow \theta_{\text{best}}$
Graspindex $\leftarrow k$
if $\theta_{\text{best}} == \text{None}$ **then**
  return None, None, None
if $\theta_{\text{IK}} == \text{None}$ **or** $\theta_{\text{best}} == \theta_{\text{Grasp}}$ **then**
  RegraspType $\leftarrow$ Graspswitch
return $c_{\text{orig}} \leftarrow c_{\text{prev}}$, RegraspType, Graspindex

The exponential coordinates r_f, the matrix Λ, and the vector Ω can be computed as follows

$$[r_f] = \log(R_0^T R_f) \tag{9}$$

$$\Lambda(r) = I - \frac{1 - \cos||r||}{||r||^2}[r] + \frac{||r|| - \sin||r||}{||r||^3}[r]^2 \tag{10}$$

$$\Omega(r, \dot{r}) = \frac{\|r\| - \sin\|r\|}{\|r\|^3}\dot{r} \times (r \times \dot{r}) - \frac{2\cos\|r\| + \|r\|\sin\|r\| - 2}{\|r\|^4} r^\top \dot{r}(r \times \dot{r}) + \frac{3\sin\|r\| - \|r\|\cos\|r\| - 2\|r\|}{\|r\|^5} r^\top r(r \times (r \times \dot{r})) \tag{11}$$

For the translation part, given the translation vectors $p_0, p_f \in \mathbb{R}^3$, the linear velocity vectors $v_0, v_f \in \mathbb{R}^3$ and the linear acceleration vectors $\dot{v}_o, \dot{v}_f \in \mathbb{R}^3$, a *quintic polynomial* is used to interpolate the position trajectory $[p(t)]_{t\in[0,t_f]}$ between p_0 and p_f:

$$p(t)_{t\in[0,t_f]} = \sum_{i=0}^{5} k_i t^i \tag{12}$$

Similar with the rotation part, the coefficients $k_5, k_4, k_3, k_2, k_1, k_o$ can be computed by the boundary condition $p(t) = p_t$, $v(t) = v_t$, $\dot{v}(t) = \dot{v}_t$, $t = 0, t_f$. As shown in Section 6.2, such interpolation method has many advantages. First, decoupling the rotational motion from the translational motion yields a straight-line motion in Cartesian space, where the frame origin follows a straight line while the axis of rotation is constant in the body frame. Second, the interpolated trajectory is short and smooth, and the start and final velocity and acceleration conditions can be considered. Finally, the first and second derivatives are smooth and easy to compute.

https://doi.org/10.1017/S0263574721000941 Published online by Cambridge University Press

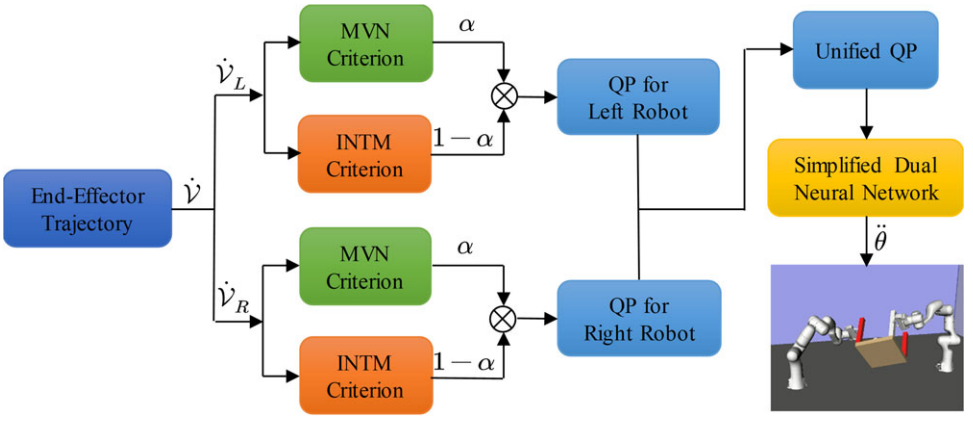

Figure 4. *System structure of Dual-MVN-INTM scheme for the dual-redundant robot.*

5.2. Quadratic program-based redundant resolution

To minimize the input power of redundant robots in the large object manipulation task, the joint torque minimization criteria are used for the redundancy resolution. And to remedy the joint torque instability and divergence phenomenon exists in the INTM scheme, a Dual-MVN-INTM different-level minimization scheme is proposed in this paper. As shown in Fig. 4, MVN scheme and INTM scheme are integrated via a weighting factor in the form of a QP for each robot, then the two QPs are unified into a standard QP and formulated to an S-DNN model for fast solving. Note that the proposed scheme is solved at acceleration level with joint position, velocity, and acceleration limits considered.

Given an end-effector trajectory in terms of position, orientation and velocity, acceleration twist, and the joint acceleration $\ddot{\theta}(t)$ computed through Dual-MVN-INTM scheme, suppose the initial joint position and velocity $\theta(0), \dot{\theta}(0)$ is known, joint positions can then be computed by Euler integration:

$$\theta(t_{k+1}) = \theta(t_k) + \dot{\theta}(t_k)\Delta t \tag{13}$$

$$\dot{\theta}(t_{k+1}) = \dot{\theta}(t_k) + \ddot{\theta}(t_k)\Delta t \tag{14}$$

5.2.1. QP formulation and unification

For an n DoF robot, the relationship between the end-effector velocity twist $\mathcal{V}_b = [\omega_b, \nu_b] \in \mathbb{R}^6$ in the end-effector (or body) frame and the joint velocity $\dot{\theta} \in \mathbb{R}^n$ can be expressed as

$$\mathcal{V}_b = J_b(\theta)\dot{\theta} \tag{15}$$

where $J_b(\theta) \in \mathbb{R}^{m\times n}$ is the Jacobian expressed in end-effector frame. Note that $\mathcal{V}, J$ are used to replace $\mathcal{V}_b, J_b$ for simplicity in remainder of this paper.

Take the derivative of (15) with respect to time t:

$$J(\theta)\ddot{\theta} = \dot{\mathcal{V}} - \dot{J}(\theta)\dot{\theta} = \dot{\mathcal{V}}_\tau \tag{16}$$

Without considering the friction force at the joint, the dynamic equation of the n DOF robot is

$$\tau = M(\theta)\ddot{\theta} + c(\theta, \dot{\theta}) + g(\theta) \tag{17}$$

where $M(\theta) \in \mathbb{R}^{n\times n}$ is the symmetric positive-definite mass matrix, $c(\theta, \dot{\theta}) \in \mathbb{R}^n$ is a vector of Coriolis and centripetal torques, and $g(\theta) \in \mathbb{R}^n$ is a vector of gravitational torques.

https://doi.org/10.1017/S0263574721000941 Published online by Cambridge University Press

The bi-criteria scheme that combined MVN and INTM for a single robot can be formulated as:

$$\begin{aligned}
&\min_{\ddot{\theta},\tau} \frac{1}{2}\{\alpha||\ddot{\theta}||_2^2 + (1-\alpha)||\tau||_\infty^2\} \\
&\text{s.t. } J(\theta)\ddot{\theta} = \dot{\mathcal{V}}_\tau \\
&\tau = M(\theta)\ddot{\theta} + c(\theta,\dot{\theta}) + g(\theta) \\
&\theta^- \leqslant \theta \leqslant \theta^+ \\
&\dot{\theta}^- \leqslant \dot{\theta} \leqslant \dot{\theta}^+ \\
&\ddot{\theta}^- \leqslant \ddot{\theta} \leqslant \ddot{\theta}^+
\end{aligned} \tag{18}$$

where $\|\cdot\|_2$ and $\|\cdot\|_\infty$ represent the two-norm and infinity-norm of a vector. $\alpha \in [0,\ 1]$ is the weighting factor to balance the two schemes. $\theta^+, \dot{\theta}^+, \ddot{\theta}^+ \in \mathbb{R}^n$ and $\theta^-, \dot{\theta}^-, \ddot{\theta}^- \in \mathbb{R}^n$ denote the upper and lower limits of joint position, velocity, and acceleration, respectively.

The MVN-INTM scheme is built on different levels (velocity and torque). To solve the optimization problem at the same level, the MVN scheme and INTM scheme are unified at the acceleration level. More precisely, the velocity-level minimization of $||\dot{\theta}||_2^2/2$ is approximately equivalent to the minimization of $\ddot{\theta}^{\mathrm{T}}\ddot{\theta}/2 + \lambda\dot{\theta}^T\ddot{\theta}$ [17], where $\lambda > 0 \in R$ should be set as large as the robot system would permit. The torque-level minimization of $(1-\alpha)||\tau||_\infty^2/2$ is equivalent to the minimization of $(1-\alpha)s^2/2$ [15], where $s = \|_{\mathcal{T}}\|_\infty$, such that

$$\begin{bmatrix} M & -1_{7\times 1} \\ -M & -1_{7\times 1} \end{bmatrix} \begin{bmatrix} \ddot{\theta} \\ s \end{bmatrix} \leqslant \begin{bmatrix} -c-g \\ c+g \end{bmatrix} \tag{19}$$

To solve the MVN-INTM scheme at acceleration level, the joint position and velocity limits should also be converted in the form of joint acceleration [38]:

$$\eta^- \leqslant \ddot{\theta} \leqslant \eta^+ \tag{20}$$

where the ith element of the new upper and lower bound constraints are

$$\begin{aligned}
\eta_i^- &= \max\{\kappa_p(\mu\theta_i^- - \theta_i), \kappa_v(\dot{\theta}_i^- - \dot{\theta}_i), \ddot{\theta}_i^-\} \\
\eta_i^+ &= \min\{\kappa_p(\mu\theta_i^+ - \theta_i),\ \kappa_v(\dot{\theta}_i^+ - \dot{\theta}_i), \ddot{\theta}_i^+\},\ 0 < \mu < 1,\ \kappa_p > 0,\ \kappa_v > 0
\end{aligned}$$

When the joints enter into the dangerous zone $[\theta^-, \mu\theta^-]$ and $[\mu\theta^+, \theta^+]$, the adjustment coefcient κ_p prevents the joints from getting closer to the limits. Coefcients $\kappa_p > 0, \kappa_v > 0$ determine the deceleration magnitude and should be selected such that the feasible region of $\ddot{\theta}$ converted by the joint position and velocity limits is not smaller than the original one. However, without elaborate selection of the coefficients in practice, such conversion may encounter the phenomenon that the converted lower bounds are beyond the converted upper bounds. Therefore, modifications have been made to the aforementioned conversion in this paper to make it more robust for different robots:

$$\eta_i^- = \min\{\max\{\kappa_p(\mu\theta_i^- - \theta_i),\ \kappa_v(\dot{\theta}_i^- - \dot{\theta}_i), \ddot{\theta}_i^-\}, \ddot{\theta}_i^+\} \tag{21}$$

$$\eta_i^+ = \max\{\min\{\kappa_p(\mu\theta_i^+ - \theta_i),\ \kappa_v(\dot{\theta}_i^+ - \dot{\theta}_i), \ddot{\theta}_i^+\}, \ddot{\theta}_i^-\} \tag{22}$$

With the conversions made above, by defining the decision variable $x = [\ddot{\theta}^T, s]^T \in R^{n+1}$, the MVN-INTM scheme for a single robot can be expressed as the following QP :

$$\begin{aligned}
&\min_x\ x^TQx/2 + p^Tx \\
&\text{s.t. } Ax = b \\
&Cx \leqslant d \\
&x^- \leqslant x \leqslant x^+
\end{aligned} \tag{23}$$

https://doi.org/10.1017/S0263574721000941 Published online by Cambridge University Press

where the coefficient matrices and vector are defined as

$$Q=\begin{bmatrix}\alpha I & 0\\ 0 & (1-\alpha)\end{bmatrix}\in R^{(n+1)\times(n+1)},\ p=\begin{bmatrix}\alpha\lambda\dot{\theta}\\ 0\end{bmatrix}\in R^{n+1}$$

$$A=[J\ 0]\in R^{m\times(n+1)},\ b=\dot{\mathcal{V}}_\tau\in R^m$$

$$C=\begin{bmatrix}M & -1_{n\times 1}\\ -M & -1_{n\times 1}\end{bmatrix}\in R^{2n\times(n+1)},\ d=\begin{bmatrix}-c-g\\ c+g\end{bmatrix}\in R^{2n}$$

$x^-=[\eta^-,\ 0]^T\in R^{n+1}, x^+=[\eta^+,w]^T\in R^{n+1}$, here w is a large constant used to replace $+\infty$. In order to improve the computational efficiency, by defining the decision variable $u=[x_L,x_R]^T$, the two QPs of left and right robot can be unified into a standard QP:

$$\begin{aligned}&\min_u\ u^TWu/2+P^Tu\\ &\text{s.t. } Eu=f\\ &Gu\leqslant h\\ &u^-\leqslant u\leqslant u^+\end{aligned}\tag{24}$$

where the coefficient matrices and vector are defined as

$$W=\begin{bmatrix}Q_L & 0\\ 0 & Q_R\end{bmatrix}\in R^{2(n+1)\times 2(n+1)},\ P=\begin{bmatrix}p_L\\ p_R\end{bmatrix}\in R^{2(n+1)}$$

$$E=\begin{bmatrix}A_L & 0\\ 0 & A_R\end{bmatrix}\in R^{2m\times 2(n+1)},\ f=\begin{bmatrix}b_L\\ b_R\end{bmatrix}\in R^{2m}\ G=\begin{bmatrix}C_L & 0\\ 0 & C_R\end{bmatrix}\in R^{4n\times 2(n+1)},\ h=\begin{bmatrix}d_L\\ d_R\end{bmatrix}\in R^{4n}$$

$$u^-=\begin{bmatrix}x_L^-\\ x_R^-\end{bmatrix}\in R^{2(n+1)},\ u^+=\begin{bmatrix}x_L^+\\ x_R^+\end{bmatrix}\in R^{2(n+1)}$$

coefficient matrices and variables $Q_{L/R}, p_{L/R}, A_{L/R}, b_{L/R}, C_{L/R}, d_{L/R}\ x_{L/R}^-, x_{L/R}^+$ are the same as those defined in (23), and the subscript L, R correspond to left and right robot, respectively.

5.2.2. QP solver based on simplified dual neural network

At this point, the IK problem of dual-redundant-robot is transformed into a unified time-varying QP problem. The matrix W is positive definite, so the objective function in (24) is strictly convex. And the feasible region of the constraints in (24) is a closed convex set. Therefore, the optimal solution of the QP problem (24) is unique and satisfies the KKT conditions [48]. Since traditional numerical QP solver is inefficient for solving of the large-scale QP-formed IK like (24) [13], many researchers use the parallel computational RNNs instead for real-time computations [13, 14, 15, 16, 17, 18, 19]. In this paper, in order to solve our Dual-MVN-INTM scheme in real time, we formulate (24) into an architecture-efficient RNN model S-DNN [18], which further reduces the compute complexity than DNN while preserving the desirable convergence property.

By defining $K:=[G,I]^T\in R^{(6n+2)\times(2n+2)}, \zeta^-=[-\infty,\ \ u^-]^T\in R^{6n+2}, \zeta^+=[h,u^+]^T\in R^{6n+2}$ the inequality constraints in (24) can be reformulated as

$$\zeta^-\leqslant Ku\leqslant\zeta^+\tag{25}$$

Consider (24) as the primal problem P and D as its dual problem. The Lagrangian function of P is

$$L(u,\ \lambda,\nu)=u^TWu/2+g^Tu+\lambda_1^T(\zeta^--Ku)+\lambda_2^T(Ku-\zeta^+)+\nu^T\ (Eu-f)\tag{26}$$

where $\lambda_1,\lambda_2\in\mathbb{R}^p$ is the dual variables for the inequality constraints, $\nu\in\mathbb{R}^m$ is the dual variable for the equality constraints. According to KKT conditions, if u^* is the optimal solution of P, and $(\lambda_1^*,\ \lambda_2^*,\nu^*)$

https://doi.org/10.1017/S0263574721000941 Published online by Cambridge University Press

is the optimal solution of D, then $(u^*,\ \lambda_1^*,\ \lambda_2^*, \nu^*)$ should satisfy

$$Wu + P + E^T\nu - K^T\lambda_1 + K^T\lambda_2 = 0 \tag{27}$$

$$Eu = f \tag{28}$$

$$\zeta^- \leqslant Ku \leqslant \zeta^+ \tag{29}$$

$$\lambda_1,\ \lambda_2 \geqslant 0 \tag{30}$$

$$\lambda_1^T(\zeta^- - Ku) = \lambda_2^T(Ku - \zeta^+) = 0 \tag{31}$$

Define $\psi = \lambda_1 - \lambda_2$, then the constraints (29)$\sim$(31) are equivalent to

$$Ku = g(Ku - \psi) \tag{32}$$

where $g_i(v_i) = \begin{cases} \zeta_i^- & \text{if } v_i < \zeta_i^- \\ v_i & \text{if } \zeta_i^- \leq v_i \leq \zeta_i^+, \\ \zeta_i^+ & \text{if } v_i < \zeta_i^+ \end{cases} \quad i = 1, ..., p$

Since W is invertible, (27) can be reformulated as

$$u = -W^{-1}(P + E^T\nu - K^T\psi) \tag{33}$$

Substituting (33) into (28) yields

$$EW^{-1}(P + E^T\nu - K^T\psi) = -f \tag{34}$$

Because $EW^{-1}E^T$ is invertible, ν can be expressed by ψ:

$$\nu = -(EW^{-1}E^T)^{-1}(f + EW^{-1}P - EW^{-1}K^T\psi) \tag{35}$$

Substituting (35) into (33) yields

$$\begin{aligned} u = &(- W^{-1}E^T(EW^{-1}E^T)^{-1}EW^{-1} + W^{-1})(K^T\psi - P) \\ &+ W^{-1}E^T(EW^{-1}E^T)^{-1}f \end{aligned} \tag{36}$$

Since W is symmetric, by defining $T = W^{-1}E^T$ and $J = T(T^TWT)^{-1}$, (36) can be reformulated as

$$u = (- JT^T + W^{-1})(K^T\psi - P) + Jf \tag{37}$$

Therefore, the optimal solution to the primal problem P is equivalent to the solution to (32) and (37), which yield a S-DNN model [18] whose governing differential and output equations are

$$\begin{aligned} \dot{\psi} = &\varepsilon(K(JT^T - W^{-1})(K^T\psi - P) - K^TJf) \\ &+ \varepsilon g(K(- JT^T + W^{-1})(K^T\psi - P) + K^TJf - \psi) \\ u = &(- JT^T + W^{-1})(K^T\psi - P) + Jf \end{aligned} \tag{38}$$

where ϵ is the positive scaling parameter to control the convergence rate of the neural network.

The S-DNN defined by (38) is composed of only one layer of $6n + 2$ neurons (number of inequality constraints), which is much fewer than $6n + 2m + 2$ neurons of the DNN, $18n + 2m + 8$ neurons of LNN, and $10n + 2m + 2$ neurons of PDNN for solving the same QP (24). As shown in the experiments, the formulated S-DNN model converges within less than 5×10^{-6}s, which is very fast for computing the required IK solutions.

6. Experiment

In order to illustrate our algorithm's effectiveness and applicability, we design three difficult manipulation tasks involving two different kinds of robots and three manipulation objects. We first illustrate the experiment set-up and descriptions of three tasks in Section 6.1. In Section 6.2, one segment of the

https://doi.org/10.1017/S0263574721000941 Published online by Cambridge University Press

closed-chain motion in Task 1 is taken as an example to show the process and result of $SE(3)$ trajectory interpolation. Then the proposed Dual-MVN-INTM scheme is used to convert the $SE(3)$ trajectory to the joint space for both robots in Section 6.3. The results are further analyzed and compared to other IK algorithms. Section 6.4 shows the simulated planning results of all three tasks, where our planner and the planner in [11] are compared in term of planning success rate, required regrasping amount, and planning time. In Section 6.5, we implement a physical experiment to demonstrate the effectiveness in real world by solving one task with two 7 DoF Kinova Jaco2 robot.

6.1. Experiment set-up and task description

As shown in Fig. 2, the experiment platform consists of two 7 DOF robots and a manipulated object. The motion planner is implemented in Python and the QP-based IK solver was integrated as subfunctions. The collision checking is implemented using FCL [49]. All tasks were simulated on OpenRAVE [29] and run on a PC platform with a 3. 6GHz Intel Core i7-9700K CPU. Two different kinds of robots-7DoF Franka Emika Panda (Panda) and 7 DoF Kinova Jaco2 (Jaco2) are used. Each robot is equipped with a parallel gripper, which can provide a clamping force ranging from 40N to 140N. The distance between robot bases is set to 1.6 m. Three objects, including a 40 cm × 40cm × 37 cm square table, a 56 cm × 44cm × 50 cm stool, and a 43 cm × 22cm × 43 cm book shelf, are manipulated in three different tasks.

In Task 1, two Panda robots are required to transfer and turn the square table upside down (Fig. 11). In Task 2, the stool is moved and flipped back and forth by two Panda robots (Fig. 12). Task 3 is similar with Taskl, where two Jaco2 robots are required to transfer and reverse the book shelf (Fig. 13). Our planner is used in all these tasks to plan smooth global composite trajectories. The initial configurations and corresponding target configurations are essentially mutually disconnected, which makes these three tasks difficult and unsolvable for local planners.

6.2. SE(3) trajectory interpolation

Take a segment of the closed-chain motion in Task 1 as an example, we first show the process and result *of SE* (3) trajectory interpolation. $T_{\text{obj0}}, T_{\text{obj1}} \in SE(3)$ are the start and the end configurations of this segment, and $T_{\text{Grasp}_l}, T_{\text{Grasp}_r}$ are the corresponding grasping poses of the left and the right robot. For the left robot, the start configuration $T_{l0} = (R_{l0}, p_{l0}) \in SE(3)$ and the end configuration $T_{l1} = (R_{l1}, p_{l1}) \in SE(3)$ of the end-effector expressed in the coordinate frame $\{L\rangle$ can be computed through the closed-chain constraint. The time duration of this trajectory $t_f = 1.5s$. The start and end velocity twists $\mathcal{V}_0,\ \mathcal{V}_1 \in \mathbb{R}^6$, and the start and end acceleration twist $\dot{\mathcal{V}}_0, \dot{\mathcal{V}}_1 \in \mathbb{R}^6$ expressed in end-effector frame are set to zero:

$$\mathcal{V}_0 = (\omega_0, v_0) = 0_{6\times1}, \mathcal{V}_1 = (\omega_1, v_1) = 0_{6\times1}$$
$$\dot{\mathcal{V}}_0 = (\dot{\omega}_0, \dot{v}_0) = 0_{6\times1}, \dot{\mathcal{V}}_1 = (\dot{\omega}_1, \dot{v}_1) = 0_{6\times1}$$

We interpolate the $SO(3)$ trajectory between R_0 and R_1 according to (7)

$$R_l(t) = R_{l0} e^{[a_5 t^5 + a_4 t^4 + a_3 t^3 + a_2 t^2 + a_1 t + a_0]}, t \in [0, t_f]$$

Then we interpolate the translational trajectory between p_0 and p_1 according to (12)

$$p_l(t) = k_5 t^5 + k_4 t^4 + k_3 t^3 + k_2 t^2 + k_1 t + k_0, t \in [0, t_f]$$

So far, the $SE(3)$ trajectory $(R_l(t), p_l(t)), t \in [0, t_f]$ of the left robot end-effector is ready, and the corresponding velocity and acceleration twists $\mathcal{V}_l(t)$, $\dot{\mathcal{V}}_l(t)$ expressed in the end-effector coordinate frame can

https://doi.org/10.1017/S0263574721000941 Published online by Cambridge University Press

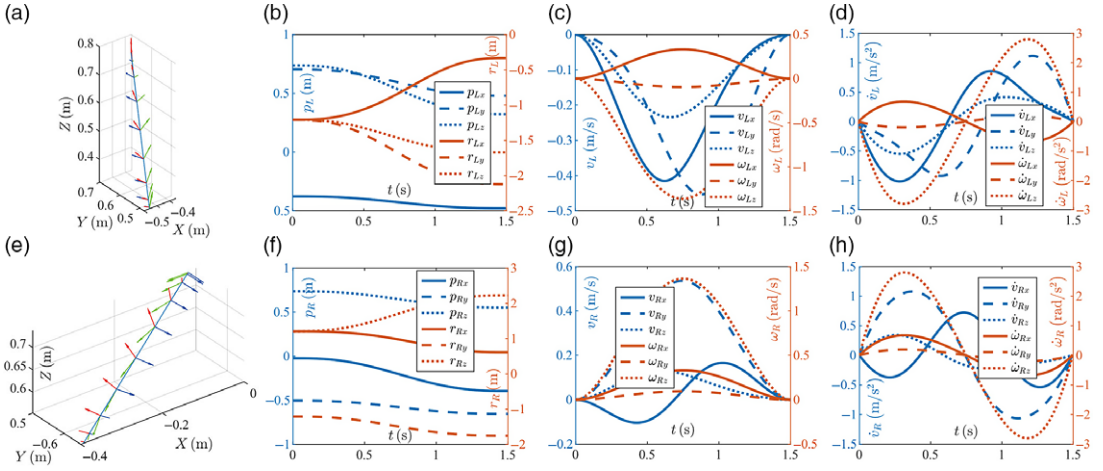

Figure 5. *Interpolated SE(3) trajectories for two robots. (a)-(d) are for the left robot and (e)-(h) are for the right robot. (a) and (e) are end-effector motions in Cartesian space. (b) and (f) are SE(3) configuration profiles, where the orientations are denoted as axis angles. (c) and (g) are velocity profiles. (d) and (h) are acceleration profiles.*

be computed as follows:

$$\mathcal{V}_l(t) = \begin{bmatrix} w_l(t) \\ v_l(t) \end{bmatrix} = \begin{bmatrix} \Lambda(r(t))\dot{r}(t) \\ R^T(t)\dot{p}(t) \end{bmatrix}$$

$$\dot{\mathcal{V}}_l(t) = \begin{bmatrix} \dot{w}_l(t) \\ \dot{v}_l(t) \end{bmatrix} = \begin{bmatrix} \Lambda(r(t))\ddot{r}(t) + \Omega(r, \dot{r}) \\ \dot{R}^T(t)\dot{p}(t) + R^T(t)\ddot{p}(t) \end{bmatrix}$$

where $\Lambda(r(t))$ and $\Omega(r, \dot{r})$ can be computed via (10) and (11), respectively.

The $SE(3)$ trajectory and the velocity twists of the right robot can be computed through the same process. The final $SE(3)$ trajectory of both end-effectors and the velocity and acceleration twist profiles are shown in Fig. 5, where the rotational trajectory is shown in the form of axis-angle $r(t)$. Note that the segment shown in the figure is extracted for illustration convenience, while the whole closed-chain trajectory consists of many motion segments like this. As can be seen form Fig. 5, the interpolated method provides a straight-line motion in Cartesian space and the velocity and acceleration curves are smooth, which make the interpolated trajectory ideal for the bimanual manipulation task discussed in this paper.

6.3. Joint space trajectory conversion and comparing

In this section, the trajectory of the end-effector in task space is converted to joint space using different IK algorithms for comparing and verifying the effectiveness of the proposed Dual-MVN-INTM redundancy resolution scheme. The initial state of the left and right robot is

$$\theta_L(0) = [-0.26,\ -1.02, 1.50,\ -1.10, 0.72, 2.82, 1.16]^T \text{ rad}$$

$$\theta_R(0) = [-1.51,\ -0.56,\ -0.16,\ -2.31,\ -0.38, 3.34, 1.08]^T \text{ rad}$$

$$\dot{\theta}_L(0) = \dot{\theta}_R(0) = [0, 0, 0, 0, 0, 0, 0]^T \text{ rad/s}$$

The kinematic and dynamic parameters, joint limits of Panda robot can be obtained from [50]. Moreover, the design parameters used in INTM scheme and Dual-MVN-INTM $\lambda = 30$, $\mu = 0.9$, $\kappa_p = \kappa_v = 15$.

https://doi.org/10.1017/S0263574721000941 Published online by Cambridge University Press

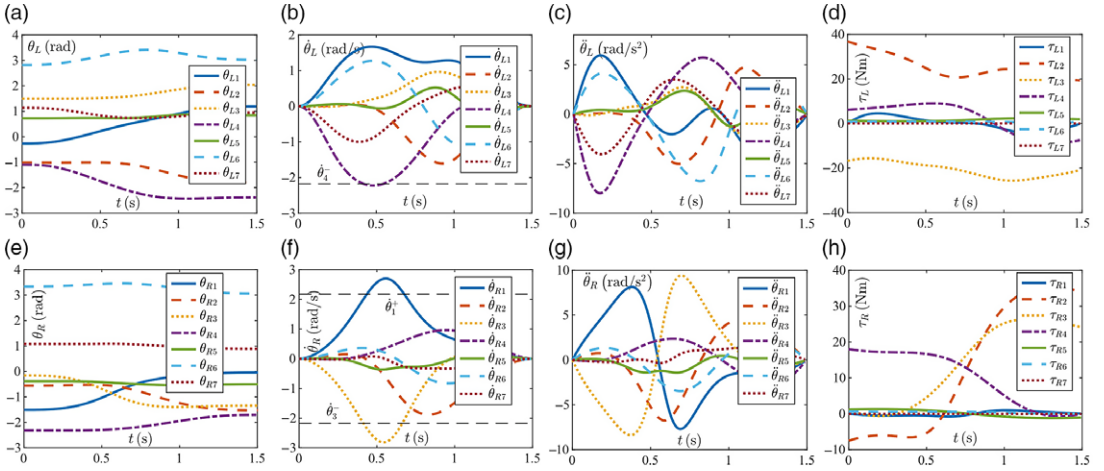

Figure 6. *Dual Panda end-effectors tracking the interpolated straight-line trajectory synthesized by the pseudoinverse-based method. (a)-(d) are joint position, velocity, acceleration, torque profiles of the left robot. (e)-(h) are corresponding profiles of the right robot, respectively.*

In order to compare with the proposed Dual-MVN-INTM scheme, the pseudoinverse-based method [12] is used first for the conversion: $\dot{\theta} = J^{+}\mathcal{V} + k_0(I_n - J^{\dagger}J)\partial w(\theta)/\partial\theta$, where the secondary objective function $\omega(\theta)$ is chosen as $\omega(\theta) = -1/2n\sum_{i=1}^{n}((\theta_i - \overline{\theta}_i)/(\theta_i^{+} - \theta_i^{-}))^2$, which measures the distance from joint position limits and is frequently-used. By maximizing this distance, redundancy is exploited to keep the joint positions as close as possible to the middle value of the joint range $\overline{\theta}_i$. The computed joint position, velocity, acceleration, and torque profiles are shown in Fig. 6. As can be seen, the joint positions are kept within the limits, and the joint variables are changed smoothly. However, such method only considers the joint position limits. While dragging the joint positions to the middle, the joint velocities, accelerations, or torques might exceed their limits. For example, for the left robot, at the time interval [0.4,0.6], the velocity of joint 7 exceeds the limit. And for the right robot, at time interval [0.4,0.7], the velocities of joint 1 and joint 3 exceed their limits, respectively, which is unacceptable in practice.

Fig. 7(a) and Fig. 7(d) show the joint trajectories of dual Panda robots synthesized by INTM scheme [40] (i.e. $\alpha = 0$) with joint position, velocity, acceleration limits considered. The corresponding joint velocity, acceleration, torque profiles of dual robots are shown in Fig. 7(b)–(d) and Fig. 7(f) − (h) , respectively. As seen from Fig. 7, the joint variables are kept within their limits by incorporating joint physical limits. However, some of the joint accelerations, joint velocities are so large that limits are reached, such as joint 5 of the left robot and joint 5, 7 of the right robot. And abrupt increase or decrease in these joint variables exhibited, which is because that INVM scheme minimizes the largest element of the joint velocity or acceleration vector in magnitude. This phenomenon is not desirable in practice and would cause damages to the robot. Furthermore, the final velocities of some joints are too large in the real robotic applications. Therefore, the INVM scheme is unsuitable in practical applications.

Fig. 8(a) and Fig. 8(d) illustrates the joint trajectories of dual Panda robots synthesized by the proposed Dual-MVN-INTM scheme ($\alpha = 0.6$) with joint position, velocity, acceleration limits considered. The corresponding joint velocity, acceleration, and torque profiles of dual robots are shown in Fig. 8(b)–(d) and Fig. 8(f) − (h) , respectively. As seen from Fig. 8, the joint variables are all kept within the feasible regions, and the curves are quite smooth without abrupt changes or chattering phenomena, which illustrates the discontinuous problem occurred in the INTM scheme is remedied by the Dual-MVN-INTM scheme. And the joint velocities, accelerations, and torques are smaller comparing with INTM scheme (some variables in Fig. 7 are too large even reach the limits). To further verify the effectiveness of trajectory tracking, the translational and rotational tracking errors of the dual robots are shown in Fig. 9, where the rotational tracking errors are measured in the form of axis-angle, e_r and e_p

https://doi.org/10.1017/S0263574721000941 Published online by Cambridge University Press

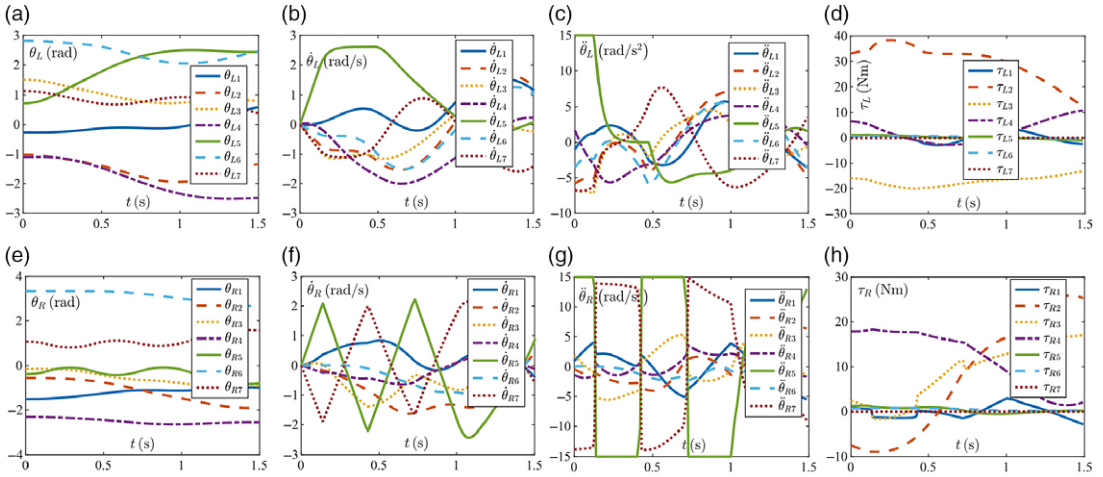

Figure 7. *Dual Panda end-effectors tracking the interpolated straight-line trajectory synthesized by INTM scheme ($\alpha = 0$). (a)-(d) are joint position, velocity, acceleration, torque profiles of the left robot. (e)-(h) are corresponding profiles of the right robot, respectively.*

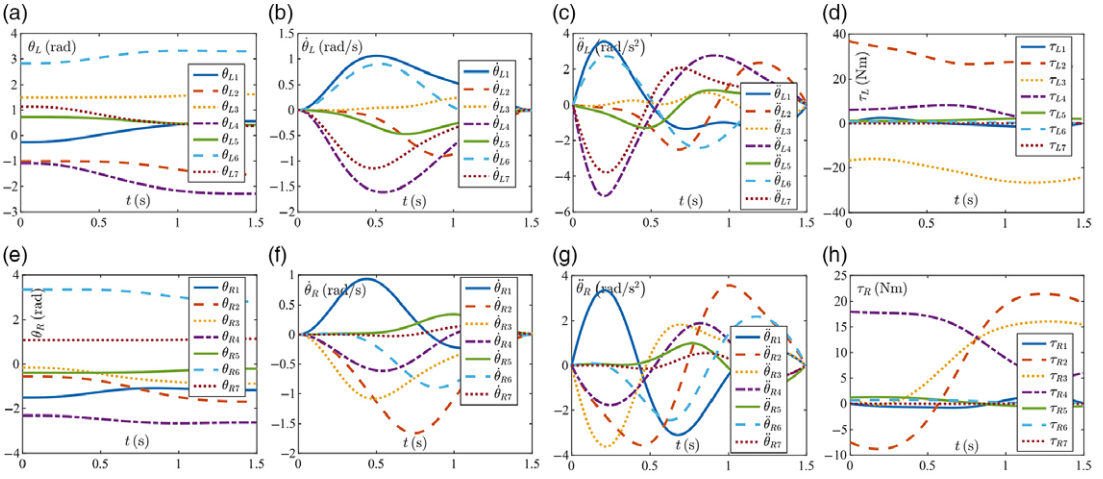

Figure 8. *Dual Panda end-effectors tracking the interpolated straight-line trajectory synthesized by Dual-MVN-INTM scheme ($\alpha = 0.6$). (a)–(d) are joint position, velocity, acceleration, torque profiles of the left robot. (e)–(h) are joint position, velocity, acceleration, torque profiles of the right robot, respectively.*

represent the rotational and translational tracking error, respectively. As can be seen, the converted trajectories track the end-effectors trajectories well with the maximal tracking error less than 1.6×10^{-4} for the left robot and 1.2×10^{-4} for the right robot, respectively. In addition, Fig. 8(b) and Fig. 8(f) show the Dual-MVN-INTM scheme guarantee the joint velocity to approach zero at the end of the motion.

Moreover, the final joint velocities and tracking errors with different α used in Dual-MVN-INTM scheme are reported in Tables I and II. As shown in Table I, the joint velocities at the end of the motion are very close to zero (10^{-3}rad/s) , which is preferable in applications. Furthermore, the tracking errors shown in Table II are quite small (10^{-4}m), which illustrates that the proposed scheme can track the end-eff ector trajectory precisely. Fig. 10 shows the joint energy cost $\tau^T M^{-1} \tau$ synthesized with different α. As can be seen, when Dual-MVN-INTM scheme ($\alpha \in (0, 1)$) is utilized, the joint energy costs of

https://doi.org/10.1017/S0263574721000941 Published online by Cambridge University Press

Table I. *Final joint velocities* ($10^{-3} rad/s$) *with different* α *used in Dual-MVN-INTM scheme.*

	$\alpha = 0.2$		$\alpha = 0.4$		$\alpha = 0.6$		$\alpha = 0.8$	
Joint i	$\lvert\dot{\theta}_{Li}\rvert$	$\lvert\dot{\theta}_{Ri}\rvert$	$\lvert\dot{\theta}_{Li}\rvert$	$\lvert\dot{\theta}_{Ri}\rvert$	$\lvert\dot{\theta}_{Li}\rvert$	$\lvert\dot{\theta}_{Ri}\rvert$	$\lvert\dot{\theta}_{Li}\rvert$	$\lvert\dot{\theta}_{Ri}\rvert$
1	1.7926	3.6012	1.7120	5.7863	1.6907	7.0431	1.6804	6.5792
2	2.0229	4.2831	1.9263	0.0055	1.9007	1.1433	1.8882	2.3610
3	0.9320	2.1295	0.9850	2.6069	1.0013	2.3609	1.0086	2.5640
4	0.3465	0.3756	0.2658	0.3784	0.2390	0.0101	0.2256	0.0097
5	7.0609	4.1255	6.9349	1.7653	6.9037	2.8513	6.8866	4.2597
6	0.7554	4.2417	0.5759	8.2184	0.5188	8.0878	0.4907	7.4185
7	4.9721	1.5156	4.8864	3.2828	4.8651	4.5451	4.8533	5.2748

Table II. *Tracking errors* ($10^{-4} m$) *with different* α *used in Dual-MVN-INTM scheme.*

α	e_{Lr}	e_{Lp}	e_{Rr}	e_{Rp}
0.2	3.2024	2.3534	5.5947	3.6064
0.4	2.1469	1.5410	3.0256	2.5840
0.6	1.6620	1.4012	1.1335	2.2215
0.8	2.1985	1.8310	2.9417	1.1137

both robots are much smaller than that of INTM scheme ($\alpha = 0$). As mentioned before, the Dual-MVN-INTM scheme is solved by a simplified dual network in this paper, and the scaling parameter of the network is $\epsilon = 3 \times 10^5$, which leads to a very fast convergence of the dual network and the equilibrium point is reached in less than 5×10^{-6}s. In summary, the comparison results verify the effectiveness, accuracy, and high efficiency of the proposed Dual-MVN-INTM scheme used for dual-redundant-robot cooperatively tracking the end-effector trajectories in bimanual operation tasks.

6.4. Planning results and comparing

In this section, we display and analyze the planning results for three tasks. As illustrated in Section 3, the key of solving the global motion planning of CKC systems is connecting the closed-chain motions through regrasping motions. Therefore, for comparing the performance of the proposed planner and the IK-switch planner in [11], contrast experiments are conducted by sending the same planning query of three tasks to two planners. Each planner runs 50 times to solve one task, and we set the allowed total number of regrasping moves for two robots as 6, considering the complexity of the tasks and quality of planned trajectory. Note that since the planner in [11] is designed for nonredundant robots, we replace the IKfast solver with our Dual-MVN-INTM solver for a fair comparison.

Table III reports the comparisons between our planner and the IK-switch planner for all three tasks in terms of the averages of planning success rate, planning time, and number of required regrasping moves.

From the results in Table III, we can see that 1) with flexible grasp-switch and IK-switch connections, our planner achieves much better success rates than IK-switch planner (95%/75%). 2) Regrasping moves are essential for the success of global planning and also take most of the total planning time. IK-switch planner solely depends on regrasping moves between different IKs of a same grasping pose, which undoubtedly increases the computation time for finding a valid regrasping configuration and also the number of required regrasping moves. While in our planner, the regrasping moves have much more options, thus it is easier for our planner to find a valid regrasping configuration. As a result, our planer is $1.78 \times$ (7.89 s/14.05 s) faster than IK-switch planner with 56% (3/4.7) less regrasping moves needed. 3) The planning success rate and required time are task-dependent, involving robots, objects, and the start and goal configurations. Fig. 11, Fig. 12, and Fig. 13 display the snapshots of composite motions

https://doi.org/10.1017/S0263574721000941 Published online by Cambridge University Press

Table III. *Results of three manipulation tasks using two planners.*

Tasks	Planners	Regrasp count	Success rate	Global planning time/s	Regrasp planning time/s	Total time/s
Task 1	Our planner	4	0.96	2.54	10.17	12.71
	IK-switch planner	5	0.74	4.23	15.46	19.69
Task 2	Our planner	2	0.93	1.65	6.36	8.01
	IK-switch planner	4	0.69	2.81	12.42	15.23
Task 3	Our planner	3	0.95	2.32	7.15	9.47
	IK-switch planner	5	0.82	3.94	14.27	18.21

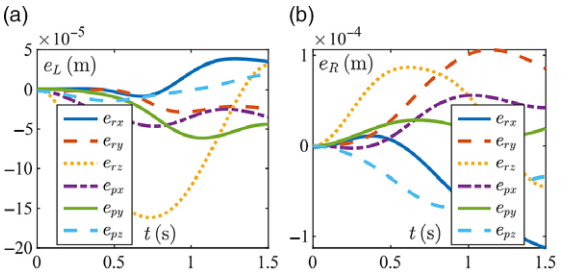

Figure 9. *Tracking errors of dual Panda synthesized by Dual-MVN-INTM scheme* ($\alpha = 0.6$). *(a) . The tracking error of the left robot. (b). The tracking error of the right robot.*

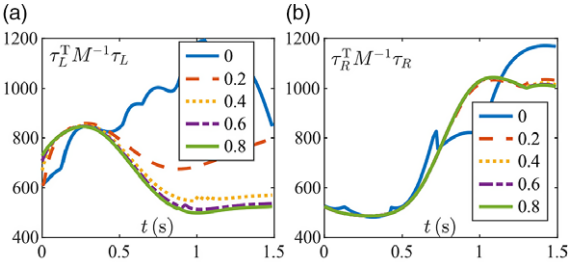

Figure 10. *Energy costs $\tau^T M^{-1} \tau$ profiles of dual Panda synthesized by Dual-MVN-INTM scheme with different α. (a). The energy cost profiles of the left robot. (b) The energy cost profiles of the right robot.*

of Task 1, Task 2, and Task 3 that planned by our planner, where our planner is able to plan smooth composite trajectories for different robots and objects that include closed-chain motions and Grasp-switch or IK-switch moves, which demonstrates the great effectiveness and applicability of the proposed framework.

6.5. Real-world robot experiment

In order to further verify the effective of our method in real world, we conduct the robot experiment using two real 7 DoF Jaco2 robots to complete Task 3. The experiment setup is shown in Fig. 13, and the start and goal configurations are shown in Fig. 2(c) – (d). Note that the rigid contact between the book shelf and grippers should be considered. Even though the tracking error of our Dual-MVN-INTM scheme is very small as shown in Table II, the discrepancies of the simulation model and robot location errors could cause damages to the object and robots. Therefore, we past a syntactic foam on each fingertip to enhance the passive compliance and the grasping stability, as shown in Fig. 14. Such measure ensures that the

https://doi.org/10.1017/S0263574721000941 Published online by Cambridge University Press

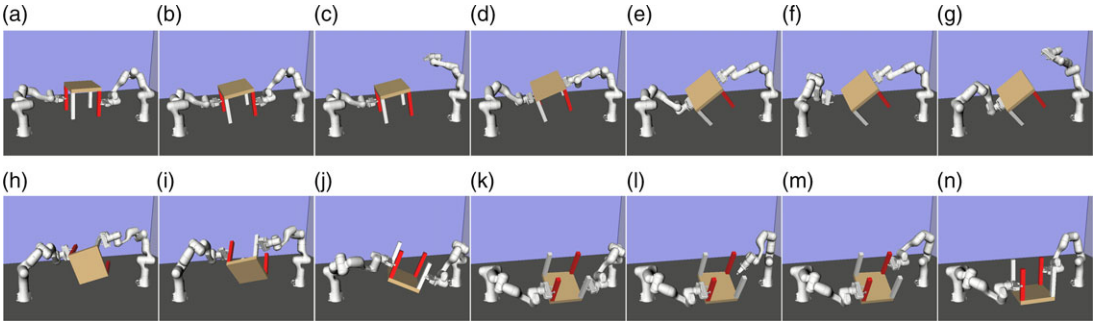

Figure 11. *Snapshots of planned composite motion for Task 1 with Panda robots.* (a, b, d, e, h−k, m, n)*: Closed-chain motions.* (c, l)*: Grasp-switch motions.* (f, g)*: IK-switch motions.*

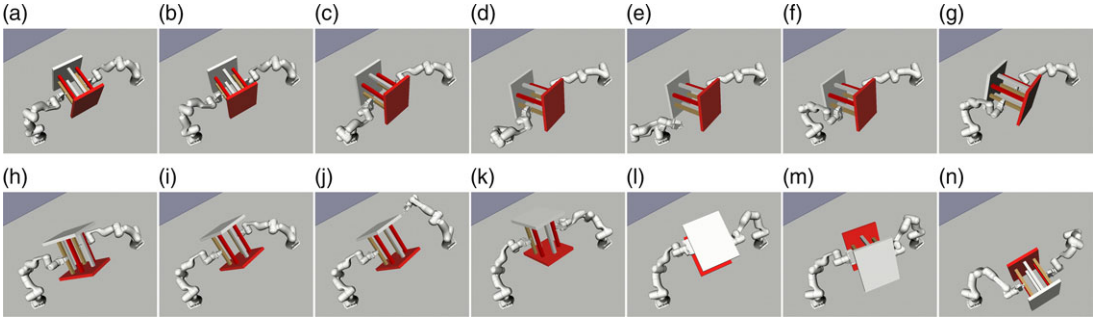

Figure 12. *Snapshots of planned composite motion for Task 2 with Panda robots.* (a−d, f−i, k−n)*: Closed- chain motions.* (e)*: IK-switch motion.* (j)*: Grasp-switch motion.*

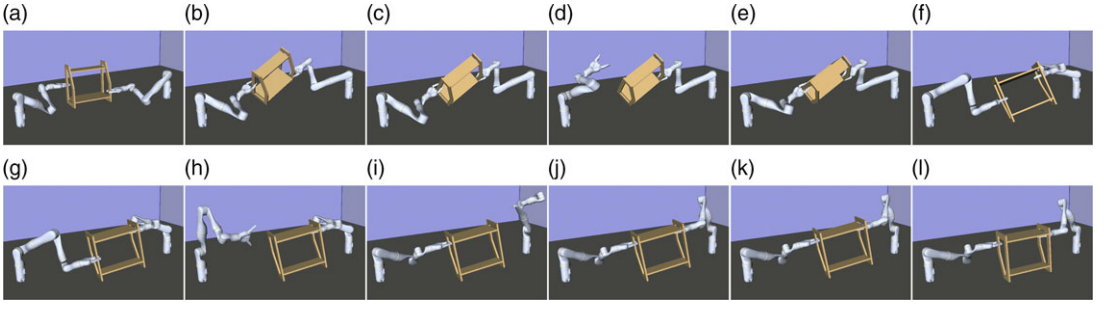

Figure 13. *Snapshots of planned composite motion for Task 3 with Jaco2 robots.* (a−e, e−g, j−l)*: Closed- chain motions.* (d, i)*: IK-switch motions.* (h)*: Grasp-switch motion.*

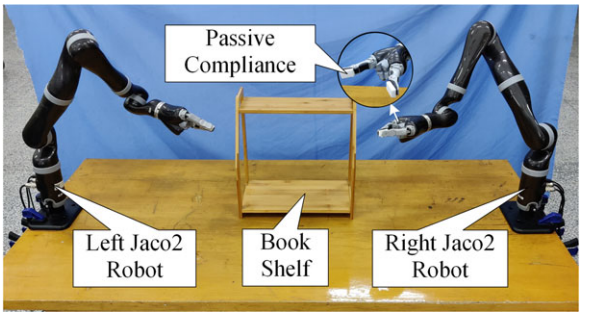

Figure 14. *Hardware system setup.*

https://doi.org/10.1017/S0263574721000941 Published online by Cambridge University Press

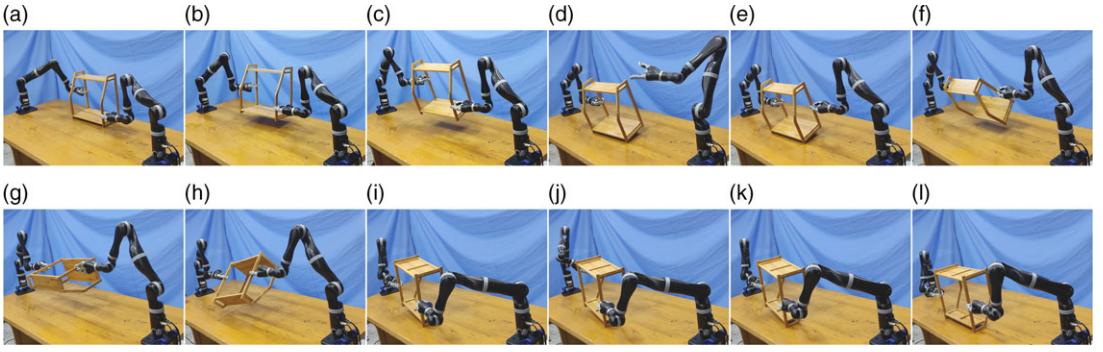

Figure 15. *Snapshots of Jaco2 robots completing Task 3 in real world.* (a−c, e−i, k, l)*: Closed-chain motions.* (d)*: Grasp-switch motion.* (j)*: IK-switch motion.*

experiment can carry out with an allowable pose error. We first send the plan query to our planner and use the hardware system to implement the planned composite trajectory. As shown in Fig. 15, our system executes the planned trajectory smoothly and completes the task successfully, which further demonstrate that our method is effective to solve the complex closed-chain manipulation task in real world.

7. Conclusion

In this paper, a global-level motion planner is proposed for the closed-chain systems consisting of two redundant robots, by flexibly choosing IK-switch or Grasp-switch regrasping moves to bridge the disconnected constrained components, the proposed planner can efficiently solve the plan queries for complex bimanual operation tasks that are unsolvable for the local planners. To address the redundancy resolution problem for the dual-redundant robots, a QP-based Dual-MVN-INTM optimizing scheme is proposed and integrated in the planning process. For computational efficiency, the two QPs for two robots are unified into a single QP and solved by a simplified DNN. Such integration yields smooth motions that respect the joint physical limits, and the zero final velocities at the end of motion are guaranteed. Simulations and experiments on three difficult closed-chain manipulation tasks using two Franka Emika Robots and two Kinova Jaco2 robots verified the effectiveness and efficiency of the propose method. The current main limitation of our method is that the plan query is solved by the planner first and then executed by the robots. In the future work, we will further improve the planning speed and success rate and extend our method to online planning. Also, the introduction of other sensors, such as cameras and force sensors, is considered to improve the control performance of the robots when completing different tasks.

Declaration of Competing Interest. The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] J. Cortés and T. Siméon, "Sampling-Based Motion Planning under Kinematic Loop-Closure Constraints," in Algorithmic Foundations of Robotics VI, M. Erdmann, M. Overmars, D. Hsu, and F. van der Stappen, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 75–90.

[2] J. H. Yakey, S. M. LaValle, and L. E. Kavraki, "Randomized path planning for linkages with closed kinematic chains," *IEEE Trans Rob Autom.* **17**, 951958 (2001).

[3] S. M. LaValle, J. H. Yakey and L. E. Kavraki, "A probabilistic roadmap approach for systems with closed kinematic chains," *1999 IEEE Int ConfRob Autom.* pp. 1671–1676 (1999).

[4] D. Berenson, S. S. Srinivasa, D. Ferguson and J. J. Kuffner, "Manipulation planning on constraint manifolds," *2009 IEEE Int. Conf Rob Autom.* pp. 625–632 (2009).

https://doi.org/10.1017/S0263574721000941 Published online by Cambridge University Press

[5] D. Xie and N. M. Amato, "A kinematics-based probabilistic roadmap method for high DOF closed chain systems," *2004 IEEE Int Conf Rob Autom.* pp. 473–478 (2004).

[6] J. Cortes, T. Simeon and J. P. Laumond, "A random loop generator for planning the motions of closed kinematic chains using PRM methods," *2002 IEEE Int Conf Rob Autom.* pp. 2141–2146 (2002).

[7] L. Jaillet and J. M. Porta, "Path planning under kinematic constraints by rapidly exploring manifolds," *IEEE Trans Rob.* **29** 10–5117 (2013).

[8] B. Kim, T. T. Um, C. Suh and F. C. Park, "Tangent bundle RRT: a randomized algorithm for constrained motion planning," *Robotica*. **34**, 202–225 (2016).

[9] Z. Kingston, M. Moll and L. E. Kavraki, "Decoupling Constraints from Sampling-Based Planners," In: *Robotics Research* (Springer International Publishing, 2020), pp. 913–928.

[10] A. Sintov, A. Borum and T. Bretl, "Motion planning of fully actuated closed kinematic chains with revolute joints: a comparative analysis," *IEEE Rob Autom Lett.* **3**, 2886–2893 (2018).

[11] Z. Xian, P Lertkultanon and Q. Pham, "Closed-chain manipulation of large objects by multi-arm robotic systems," *IEEE Rob Autom Lett*. **2**, 1832–1839 (2017).

[12] B. Siciliano, L. Sciavicco, L. Villani and G. Oriolo, eds., *Differential Kinematics and Statics*. *Robotics*: *Modelling, Planning and Control* (Springer, London, 2009), pp. 105–160.

[13] A. A. Hassan, M. El-Habrouk and S. Deghedie, "Inverse kinematics of redundant manipulators formulated as quadratic programming optimization problem solved using recurrent neural networks: a review," *Robotica*. **38**, 1495–1512 (2020).

[14] D. Guo and Y Zhang, "A new inequality-based obstacle-avoidance MVN scheme and its application to redundant robot manipulators," *IEEE Trans Syst Man Cybernetics Part C* (*Appl Rev*). **42**, 1326–1340 (2012).

[15] D. Guo and Y Zhang, "Different-level two-norm and infinity-norm minimization to remedy joint-torque instability/divergence for redundant robot manipulators," *Rob Autonom Syst*. **60**, 874–888 (2012).

[16] Y Zhang, D. Guo and S. Ma, "Different-level simultaneous minimization of joint-velocity and joint-torque for redundant robot manipulators," *J. Intell Rob Syst*. **72**, 301–323 (2013).

[17] B. Cai and Y Zhang, "Equivalence of velocity-level and acceleration-level redundancy- resolution of manipulators," *Phys Lett. A* **373**, 3450–3453 (2009).

[18] S. Liu and J. Wang, "Bi-criteria torque optimization of redundant manipulators based on a simplified dual neural network," *2005 IEEE Int Joint Conf Neural Networks*. pp. 2796–2801 (2005).

[19] Y Zhang and K. Li, "Bi-criteria velocity minimization of robot manipulators using LVI-based primal-dual neural network and illustrated via PUMA560 robot arm," *Robotica*. **28**, 525–537 (2010).

[20] K. A. O'Neil, "Divergence of linear acceleration-based redundancy resolution schemes," *IEEE Trans Rob Autom*. **18**, 625–631 (2002).

[21] S. Pedrammehr, B. Danaei, H. Abdi, M. Masouleh and S. Nahavandi, "Dynamic analysis of Hexarot: axis-symmetric parallel manipulator," *Robotica*. **36**, 1–16 (2017).

[22] Z. Zhang, Y Lin, S. Li, Y Li, Z. Yu and Y Luo, "Tricriteria optimization-coordination motion of dual-redundant-robot manipulators for complex path planning," *IEEE Trans Control Syst Technol*. **26**, 1345–1357 (2018).

[23] Z. Jia, S. Chen, Z. Zhang, N. Zhong, P Zhang, X. Qu, J. Xie and F Ouyang, "Tri-criteria optimization motion planning at acceleration-level of dual redundant manipulators," *Robotica*. **38**, 983–999 (2020).

[24] J.-C. Latombe & K. Hauser, *Motion planning for legged and humanoid robots*, Ph.D. dissertation, Stanford University, 2008.

[25] M. Stilman, "Global manipulation planning in robot joint space with task constraints," *IEEE Trans Rob*. **26**, 576–584 (2010).

[26] Q. Pham and O. Stasse, "Time-optimal path parameterization for redundantly actuated robots: a numerical integration approach," *IEEE/ASME Trans Mechatron*. **20**, 3257–3263 (2015).

[27] M. Gharbi, J. Cortes and T. Simeon, "A sampling-based path planner for dual-arm manipulation," *2008 IEEE/ASME Int Conf Adv Intell Mechatron.* pp. 383–388 (2008).

[28] Y. Koga and J.-C. Latombe, "Experiments in dual-arm manipulation planning," Proceedings 1992 IEEE International Conference on Robotics and Automation, Vol. 3, pp. 2238–2245 (1992).

[29] R. Diankov, *Automated Construction of Robotic Manipulation Programs*, Ph.D. dissertation, Carnegie Mellon University, 2010.

[30] N. Preda, A. Manurung, O. Lambercy, R. Gassert and M. Bonfè, "Motion planning for a multi- arm surgical robot using both sampling-based algorithms and motion primitives," *2015 IEEE/RSJ Int Conf Intell Rob Syst.* pp. 1422–1427 (2015).

[31] W Xu, Y Liu and Y Xu, "The coordinated motion planning of a dual-arm space robot for target capturing," *Robotica*. **30**, 755–771 (2012).

[32] N. García, J. Rosell and R. Suárez, "Motion planning by demonstration with human-likeness evaluation for dual-arm robots," *IEEE Trans Syst Man Cybernetics*: *Syst*. **49**, 2298–2307 (2019).

[33] W Szynkiewicz and J. Blaszczyk, "Optimization-based approach to path planning for closed chain robot systems," *International Journal of Applied Mathematics and Computer Science* **21**, 659-670 (2011).

[34] A. Völz and K. Graichen, "An optimization-based approach to dual-arm motion planning with closed kinematics," *2018 IEEE/RSJ Int Conf Intell Rob Syst.* pp. 8346–8351 (2018).

[35] T. Stouraitis, I. Chatzinikolaidis, M. Gienger and S. Vijayakumar, "Online hybrid motion planning for dyadic collaborative manipulation via bilevel optimization," *IEEE Trans Rob*. **36**, 1452–1471 (2020).

https://doi.org/10.1017/S0263574721000941 Published online by Cambridge University Press

[36] G. Song, S. Su, Y Li, X. Zhao, H. Du, J. Han and Y Zhao, "A closed-loop framework for the inverse kinematics of the 7 degrees of freedom manipulator," *Robotica*. **39**, 572–581 (2021).
[37] H.-B. Choi, S. Lee and J. Lee, "Minimum infinity-norm joint velocity solutions for singularity- robust inverse kinematics," *Int J Precis. Eng Manuf*. **12**, 469–474 (2011).
[38] D. Guo and Y Zhang, "Acceleration-level inequality-based MAN scheme for obstacle avoidance of redundant robot manipulators," *IEEE Trans Ind Electronics* **61**, 6903–6914 (2014).
[39] Y Zhang, J. Yin and B. Cai, "Infinity-norm acceleration minimization of robotic redundant manipulators using the LVI-based primaldual neural network," *Rob. Comput Integr Manu*. **25**, 358–365 (2009).
[40] Y Zhang, "Inverse-free computation for infinity-norm torque minimization of robot manipulators," *Mechatronics*. **16**, 177–184 (2006).
[41] Y Zhang, B. Cai, L. Zhang and K. Li, "Bi-criteria velocity minimization of robot manipulators using a linear variational inequalities-based primal-dual neural network and PUMA560 example," *Adv. Rob*. **22**, 1479–1496 (2008).
[42] T. Baratcart, V Salvucci and T. Koseki, "Experimental verification of two-norm, infinity-norm continuous switching implemented in resolution of biarticular actuation redundancy" *Adv. Rob*. **29**, 1243–1252 (2015).
[43] B. Liao and W Liu, "Pseudoinverse-type bi-criteria minimization scheme for redundancy resolution of robot manipulators," *Robotica*. **33**, 2100–2113 (2015).
[44] A. Escande, N. Mansard and P-B. Wieber, "Hierarchical quadratic programming: fast online humanoid-robot motion generation," *Int JRob Res*. **33**, 1006–1028 (2014).
[45] J. Wang, Q. Hu and D. Jiang, "A Lagrangian network for kinematic control of redundant robot manipulators," *IEEE Trans Neural Networks* **10**, 1123–1132 (1999).
[46] W. S. Tang and J. Wang, "A recurrent neural network for minimum infinity-norm kinematic control of redundant manipulators with an improved problem formulation and reduced architecture complexity," *IEEE Trans Syst Man Cybernetics, Part B* (*Cybernetics*) **31**, 98–105 (2001).
[47] Y Zhang and J. Wang, "A dual neural network for constrained joint torque optimization of kinematically redundant manipulators," *IEEE Trans Syst Man Cybernetics Part B* (*Cybernetics*) **32**, 654–662 (2002).
[48] S. Boyd, L. Vandenberghe and L. Faybusovich, "Convex optimization," *IEEE Trans Autom. Control*. **51**, 1859–1859 (2006).
[49] J. Pan, S. Chitta and D. Manocha, "FCL: a general purpose library for collision and proximity queries," *2012 IEEE Int Conf Rob Autom.* pp. 3859–3866 (2012).
[50] C. Gaz, M. Cognetti, A. Oliva, P. R. Giordano and A. D. Luca, "Dynamic identification of the Franka Emika Panda robot with retrieval of feasible parameters using penalty-based optimization," *IEEE Rob Autom Lett*. **4**, 4147–4154 (2019).

Cite this article: Y. Wu, Y. Fu and S. Wang (2022). "Global motion planning and redundancy resolution for large objects manipulation by dual redundant robots with closed kinematics", Robotica **40**, 1125–1150. https://doi.org/10.1017/S0263574721000941