Version of Record: https://www.sciencedirect.com/science/article/pii/S0094114X23001453
Manuscript_1b7ac82a0f4c2813b64ad7ac193232ab

Task Priority Based Design Optimization of a Kinematic Redundant Robot

Angelica Ginnante[a,b,c,*], Enrico Simetti[a], Stéphane Caro[b], François Leborne[c]

[a] *University of Genova, DIBRIS, 16145 Genova, Italy*
[b] *Nantes Université, École Centrale Nantes, CNRS, LS2N, UMR 6004, F-44000 Nantes, France*
[c] *Nimbl'Bot, 7 Av. de Guitayne, 33610 Canéjan, France*

Abstract

This paper presents and defines a new design optimization method for kinematic redundant robot manipulators based on their applications. Kinematic redundant manipulators can reach a pose with an infinite number of postures. So, identifying the best robot design and configuration for a set of desired tasks is a highly complex non-linear problem. This approach employs a task priority control algorithm to perform a task oriented robot design optimization. The design parameters are replaced by controllable prismatic or revolute virtual joints and controlled by the algorithm to accomplish the desired tasks. Therefore, this new method finds an optimal robot design for a set of tasks taking advantage of the robot kinematic redundancy. This method is evaluated on a highly kinematic redundant manipulator, which tracks a set of paths with its end-effector while maintaining good kinetostatic performance.

Keywords:
Kinematic redundancy, Design optimization, Kinetostatic performance, Task priority kinematic control

1. Introduction

Redundant manipulators are increasingly being used in many applications thanks to their ability to perform secondary tasks, which improves the robot performance [1]. This paper focuses on the design optimization of such robots. When building a new robot, an important step is optimizing its design with respect to relevant performance indices. Angeles, in [2], proposed an approach to design isotropic redundant manipulators by minimizing the condition number of the robot kinematic Jacobian matrix. However, a complete way to optimize redundant robots is to consider global indices. In [1], the authors optimized a

*Corresponding author
Email addresses: aginnante@nimbl-bot.com (Angelica Ginnante), Enrico.Simetti@unige.it (Enrico Simetti), Stephane.Caro@ls2n.fr (Stéphane Caro), fleborne@nimbl-bot.com (François Leborne)

redundant serial manipulator using the global conditioning index. The design can also be optimized through both kinematic and dynamic global indices to obtain a robot with a high global dexterity through its workspace, ensuring high dynamic performance and energy efficiency [3].

It is important to consider the main robot tasks during the robot design optimization process. An example limited to non-redundant manipulators is presented in [4]. In [5], the authors worked with re-configurable modular manipulator systems to address the problem of task-based robot design. The work presented in [6] explored a method to optimize the design parameters and the desired path together, considering the desired task with interesting results.

A critical issue in these task-oriented robot optimization processes is the complexity caused by the problem non-linearity. There are many ways to reduce the complexity, for example, breaking down the problem into multiple easier steps [5]. In [7], the authors adopted a particle swarm optimization algorithm to determine the design parameter values of two collaborative robots. Other solutions employed a grid method [8] or a complex direct search method [9]. However, none of these techniques benefits from the robot redundancy to perform an optimization as a function of multiple tasks. In [10], the authors developed a novel method for optimizing manipulator design using kinematic redundancy resolution. The authors use a combination of kinematic redundancy resolution and a multi-objective optimization algorithm. Kinematic redundancy resolution involves adding additional degrees of freedom in the place of the design parameters to optimize in the robot architecture. This allows performing the same task with different robot configurations. Through the Jacobian null-space projection, the optimization algorithm modifies the chosen design parameters together with the robot configuration solving the main task and some additional performance optimization sub-tasks. This algorithm gave promising results being able to identify some optimal parameter values. However, the algorithm was tested only on a two degrees of freedom non-redundant robot to which other two degrees of freedom were added for redundancy resolution optimization.

This paper stands from the idea of [10] and extends the design optimization concept. As in [10], virtual prismatic or revolute joints replace the design parameters to be optimized by a kinematic redundancy resolution algorithm. The first main novelty consists in the chosen kinematic control framework that can solve both equality and inequality tasks simultaneously. For example, an inequality task employed in the optimization process could be the joint limit avoidance. The joint limit task can be applied on the both real and virtual joints resulting fundamental for respecting the robot and design parameter constraints. Another important feature of the kinematic control framework is the task activation/deactivation ability to prevent over-constraining the system without causing discontinuities in the velocity generation. The second significant difference between the two design algorithms lies in the optimization methodology implementation. In [10], the arm design is optimized for singular poses in its workspace. Here, the proposed optimization algorithm is based on a trajectory planning task for machining application. The robot end-effector moves along some trajectories that characterize a workspace area. The results are guidelines for building optimal robots for that desired workspace area. The design optimization algorithm here

presented is then applied on a 21 degrees of freedom manipulator tracing a set of squared paths. The performance of the robot are improved using three kinetostatic indices.

This paper is outlined as follows. Section 2 describes the employed task-priority-based kinematic control algorithm and illustrates the tasks used to optimize the manipulator kinetostatic performance. Section 3 explains the new method for design optimization of robotic manipulators, which is the central topic. Section 4 discusses and compares the obtained results. Section 5 presents the conclusions and future work.

2. Background and methods

In this section, the kinematic control algorithm used to perform the task-oriented design optimization and the tasks employed to improve the robot kinetostatic performance are described.

2.1. Kinematic control algorithm

In the literature, many kinematic control algorithms exist to deal with redundant robots and exploit the redundancy to solve simultaneous tasks. In [11], the authors present a hierarchical quadratic programming control algorithm used to find a solution to multiple and antagonistic objectives for humanoid robot motion generation. Another interesting multiple tasks control framework is presented in [12], called Set-Based Multi-Task Priority Inverse Kinematics Framework. In [13], the so-called Saturation in the Null Space (SNS) algorithm implements a predictive prioritizing technique for multiple tasks. However, these task priority algorithms generate discontinuities in the null space projector when activating or deactivating one or more tasks. Here, the new optimization method employs a task-priority based control algorithm named Task Priority Inverse Kinematic (TPIK), presented in [14, 15, 16, 17]. The distinctive aspect of this algorithm is its ability to activate and deactivate control tasks without causing discontinuities in the control variables. In [17], the TPIK algorithm was employed in the control of a highly redundant robot demonstrating its ability of optimizing the robot configuration.

Some definitions are taken from [15] that are necessary for the introduction of the kinematic control algorithm. The vector $\mathbf{q} \in \mathbb{R}^n$ is the joint variable vector, describing the arm configuration, where n is the number of joints. The joint velocities are collected in the vector $\dot{\mathbf{q}} \in \mathbb{R}^n$.

The control objectives are used to define the robot goals. A control objective corresponds to a scalar variable $x(\mathbf{q})$ computed as a function of the robot configuration vector $\mathbf{q}$ and represents the state of one task. There exists two types of control objectives, equality and inequality. Equality control objectives aim to satisfy the relationship $x(\mathbf{q}) = x_0$. Inequality control objectives take the form $x(\mathbf{q}) \leq x_M$, or $x(\mathbf{q}) \geq x_m$, or both simultaneously, where x_m and x_M are the lower and upper bounds of the variable $x(\mathbf{q})$. A conceptual division can be done to divide the control objectives into categories depending on their scope, simplifying the identification of their priority levels. These categories are system safety objectives, e.g. joint limits or obstacle avoidance, action oriented objectives, e.g. reaching a pose or following a trajectory, and optimization objectives, e.g. minimizing the joint velocities or optimizing

the kinetostatic performance indices. Then, each scalar control objective is associated with a feedback reference rate $\dot{\bar{x}}$. A closed-loop rate control law drives the actual variable $x(\mathbf{q})$ to the desired point x^* with the associated feed-forward changing rate $\dot{x}^*$ and is defined as

$$\dot{\bar{x}} = \lambda(x^* - x(\mathbf{q})) + \dot{x}^*, \tag{1}$$

where λ is a positive gain related to the target convergence rate. The actual derivative of x is defined as a function of the joint velocity vector $\dot{\mathbf{q}}$ as follows:

$$\dot{x}(\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{J}_{\text{task}}(\mathbf{q})\dot{\mathbf{q}} = \begin{bmatrix} \frac{\partial x}{\partial q_1} & \cdots & \frac{\partial x}{\partial q_n} \end{bmatrix} \dot{\mathbf{q}}, \tag{2}$$

where $\mathbf{q} = [q_1 \ \dots \ q_n]$.

An activation function $a^i(x) \in [0, 1]$ is associated with each control objective $x(\mathbf{q})$ indicating whether or not the objective has to be taken into account at a specific time instant. The tasks associated with inequality control objectives are relevant only when the scalar variable $x(\mathbf{q})$ is close or outside its validity region bounded by x_m and x_M. Thus, the activation function is equal to zero within the validity region of its associated inequality objective. When it is near or out of the validity region, its value is set to one with a smooth transition. For tasks associated with equality control objectives, the activation function is always equal to $a^i(x) \triangleq 1$ because these control objectives never become inactive.

A specific priority level is assigned to each task based on its objective importance. The highest priority tasks are solved as first using the available robot degrees of freedom and are not affected by the lower priority ones. Hence, lower priority tasks are handled with the remaining robot degrees of freedom. When two or more tasks have the same priority level, they are grouped into a multidimensional control task. A specific list of prioritized tasks is called control action $\mathscr{A}$. A control action $\mathscr{A}$ collects the vectors and matrices associated with each priority level These vectors and matrices are defined in [17] as:

- $\dot{\bar{\mathbf{x}}}_k = [\dot{\bar{x}}_{1,k}, \ \dot{\bar{x}}_{2,k}, \ \dots \ , \ \dot{\bar{x}}_{m_k,k}]^\top$ is the vector containing all the reference rates of the scalar control tasks, where m_k is the number of scalar tasks for the priority level k.
- $\mathbf{J}_k$ is the Jacobian matrix associated with the k^{th} task vector $[\dot{x}_{1,k}, \ \dots \ , \ \dot{x}_{m_k,k}]^\top$ with respect to the joint velocity vector $\dot{\mathbf{q}}$.
- $\mathbf{A}_k = \text{diag}(a_{1,k}, \ \dots \ , \ a_{m_k,k})$ is a diagonal matrix of the activation functions.

The TPIK algorithm solves a sequence of nested minimization problems to compute the system velocity reference vector $\dot{\bar{\mathbf{q}}}$ that meets all the priority requirements inside a given control action $\mathscr{A}$. The solution S_k to the k^{th} priority level can be written as

$$S_k = \arg \underset{\dot{\bar{\mathbf{q}}} \in S_{k-1}}{\text{R} - \min} ||\mathbf{A}_k(\dot{\bar{\mathbf{x}}}_k - \mathbf{J}_k\dot{\bar{\mathbf{q}}})||^2, \tag{3}$$

where S_{k-1} is the manifold of all the previous priority level solutions. The notation $\text{R} - \min$ highlights that each minimization is performed through specific regularized space projections to implement priorities among the tasks defined in [14]. In addition to Eq. (3), other

regularization costs are included. These regularization costs avoid discontinuities in the system velocity vector due to kinematic and algorithmic singularities. In [14], the authors fully describe these regularization costs that are not analyzed in this paper.

A significant advantage of the TPIK algorithm is the use of the activation functions to handle inequality control objectives without over-constraining the system. Both equality and inequality objectives require a certain amount of robot degree-of-freedom to be handled. When an inequality task is inside its validity region, the activation function is set to zero to avoid locking any degrees of freedom. So, safety tasks, like joint limits, can be placed at the top of the hierarchy without over-constraining the optimization problem.

Finally, the TPIK algorithm adopts another continuous sigmoidal function $a^p(p)$ that includes the previous and current executed actions and the time elapsed in the current step to perform a smooth transition between two actions. This function $a^p(p)$ is employed in the algorithm execution together with $a^i(x)$. More details are presented in [14].

2.2. Tasks for optimization of robot kinematic performance

The TPIK algorithm employs different kinetostatic performance tasks to optimize the robot design, namely dexterity, manipulability and robot transmission ratio, to optimize the robot design. Some of these tasks were used with the TPIK algorithm in [17]. To introduce the definition of these indices, let us consider the kinematic Jacobian matrix $\mathbf{J}_e$ related to the end-effector velocity:

$$\mathbf{t} = \begin{bmatrix} \dot{\mathbf{p}} \\ \boldsymbol{\omega} \end{bmatrix} = \mathbf{J}_e(\mathbf{q})\dot{\mathbf{q}} = \begin{bmatrix} \mathbf{J}_l(\mathbf{q}) \\ \mathbf{J}_a(\mathbf{q}) \end{bmatrix} \dot{\mathbf{q}}, \tag{4}$$

where $\mathbf{t} = \left[\dot{\mathbf{p}}^\top, \boldsymbol{\omega}^\top\right]^\top \in \mathbb{R}^6$ is the robot end-effector twist, with $\dot{\mathbf{p}} \in \mathbb{R}^3$ and $\boldsymbol{\omega} \in \mathbb{R}^3$ the linear and angular velocity vectors of the end-effector, respectively. Since the kinematic Jacobian matrix $\mathbf{J}_e$ contains non-homogeneous terms, namely linear and angular, it needs to be weighted to compute the kinetostatic performance indices correctly. The weighting of $\mathbf{J}_e$ employs the characteristic length L that was introduced in [2] to solve the absence of dimensional homogeneity in the kinematic Jacobian matrix entries and is computed in [18]. To weight $\mathbf{J}_e$, the revolute joint columns of the linear kinematic Jacobian matrix part are divided by L. The weighted kinematic Jacobian matrix is written as $\mathbf{J}_w$. The weighting is a critical issue when analyzing the kinetostatic performance of $\mathbf{J}_e$ [19, 20].

2.2.1. Manipulability

The manipulability is an index that measures the kinematic abilities of the robotic system through its Jacobian matrix $\mathbf{J}_w$ [21]. The manipulability of a manipulator is defined as

$$\mu = \sqrt{\det(\mathbf{J}_w \mathbf{J}_w^\top)}, \tag{5}$$

and amounts to the product of all the singular values of $\mathbf{J}_w$. The higher the manipulability value, the larger the manipulability hyper-ellipsoid and the better the kinematic performance of the mechanism [22]. It should be noted that the manipulator reaches a kinematic singularity when μ vanishes.

The derivative of the manipulability as a function of the joint variables is explained in [23] and used in [24]:

$$\frac{\partial \mu}{\partial q_i} = \mu \ \text{trace}\left\{ \frac{\partial \mathbf{J}_w}{\partial q_i} \mathbf{J}_w^+ \right\}, \tag{6}$$

where the matrix $\mathbf{J}_w^+$ is the pseudo-inverse of the weighted kinematic Jacobian matrix.

Hence, the manipulability Jacobian matrix $\mathbf{J}_\mu$ as a function of the joint variables is:

$$\mathbf{J}_\mu = \begin{bmatrix} \frac{\partial \mu}{\partial q_1} & \cdots & \frac{\partial \mu}{\partial q_n} \end{bmatrix}, \tag{7}$$

where n, which represents the number of columns of $\mathbf{J}_w$, is the dimension of joint space.

2.2.2. Dexterity

The dexterity $\eta(\mathbf{J}_w)$ characterizes the kinematic performance of a manipulator in a given configuration and is defined as the inverse of the conditioning number $\kappa(\mathbf{J}_w)$ of its Jacobian matrix [25]:

$$\kappa(\mathbf{J}_w) = ||\mathbf{J}_w|| \, ||\mathbf{J}_w^{-1}|| \ \text{ and } \ \eta(\mathbf{J}_w) = 1/\kappa(\mathbf{J}_w). \tag{8}$$

The index η is bounded by 0 and 1. The higher η, the better the manipulator dexterity. The manipulator reaches an isotropic posture when $\eta = 1$. The smaller η, the worse the manipulator dexterity and the closer to a singularity. Moreover, η can be defined as the ratio between the smallest and largest singular values of $\mathbf{J}_w$ indicating how close the manipulability hyper-ellipsoid is to being a hyper-sphere [26].

In order to obtain an analytical expression of η, the Frobenius norm of $\mathbf{J}_w$ can be used [27]:

$$\eta(\mathbf{J}_w) = \frac{m}{\sqrt{\text{trace}(\mathbf{J}_w \mathbf{J}_w^\top) \ \text{trace}[(\mathbf{J}_w \mathbf{J}_w^\top)^{-1}]}}, \tag{9}$$

where m, which represents the number of rows of $\mathbf{J}_w$, is the dimension of the task space. The following definitions are introduced to increase the readability of the equations:

$$\gamma_1 \triangleq \sqrt{\text{trace}(\mathbf{J}_w \mathbf{J}_w^\top)} \ \text{ and } \ \gamma_2 \triangleq \sqrt{\text{trace}[(\mathbf{J}_w \mathbf{J}_w^\top)^{-1}]}, \tag{10}$$

where $\triangleq$ is the define operator. With these definition, it follows that

$$\eta(\mathbf{J}_w) = \frac{m}{\gamma_1(\mathbf{J}_w) \, \gamma_2(\mathbf{J}_w)}. \tag{11}$$

Then, the dexterity Jacobian matrix is determined to relate the velocity rate of η with respect to the joint velocity vector $\dot{\mathbf{q}}$. The Frobenius formula used in Eq. (9) expresses η as a function of joint position vector $\mathbf{q}$ in an analytical way allowing its derivation. So, the derivative of Eq. (9) with respect to each joint position $q_i \in \mathbf{q}$ is

$$\frac{\partial \eta}{\partial q_i} = -\eta \left(\frac{\partial \gamma_1}{\partial q_i} \frac{1}{\gamma_1} + \frac{1}{\gamma_2} \frac{\partial \gamma_2}{\partial q_i} \right), \tag{12}$$

where

$$\frac{\partial \gamma_1}{\partial q_i} = \frac{1}{\gamma_1}\text{trace}\left\{ \mathbf{J}_w \frac{\partial \mathbf{J}_w^\top}{\partial q_i} \right\} \text{ and } \frac{\partial \gamma_2}{\partial q_i} = \frac{1}{\gamma_2}\text{trace}\left\{ -\mathbf{J}_w \frac{\partial \mathbf{J}_w^\top}{\partial q_i} (\mathbf{J}_w \mathbf{J}_w^\top)^2 \right\}. \tag{13}$$

In conclusion, the dexterity Jacobian matrix $\mathbf{J}_\eta$ as a function of the joint variables is:

$$\mathbf{J}_\eta = \begin{bmatrix} \frac{\partial \eta}{\partial q_1} & \cdots & \frac{\partial \eta}{\partial q_n} \end{bmatrix}, \tag{14}$$

where n, which represents the number of columns of $\mathbf{J}_w$, is the dimension of joint space.

2.2.3. Robot transmission ratio

The robot transmission ratio (RTR) $\rho(\mathbf{J}_w)$ quantifies the effectiveness of the actuator force in producing a prescribed robot motion [20]. It corresponds to the angle between the joint velocity $\dot{\mathbf{q}}$ and torque $\boldsymbol{\tau}$ vectors in the joint space and is defined as

$$\rho = \frac{|\boldsymbol{\tau}^\top \dot{\mathbf{q}}|}{||\boldsymbol{\tau}||\,||\dot{\mathbf{q}}||} = |\cos \angle(\boldsymbol{\tau}, \dot{\mathbf{q}})|. \tag{15}$$

This metric is bounded between 0 and 1. In case of kinetostatic redundancy, ρ can be expressed in terms of the end-effector twist $\mathbf{t}$ and the wrench $\mathbf{w}$ applied to it, leading to

$$\rho = \frac{|\mathbf{w}^\top \mathbf{t}|}{||\mathbf{J}_w^\top \mathbf{w}||\,||\mathbf{J}_w^+ \mathbf{t}||}, \tag{16}$$

where $\mathbf{t}$ is the robot end-effector twist defined in Eq. (4) and $\mathbf{w} = [\mathbf{f}^\top, \mathbf{m}^\top]^\top$ is the wrench that collects the forces $\mathbf{f}$ and moments $\mathbf{m}$ exerted by the environment on the end-effector. To ensure that ρ is dimensionless, the linear part $\dot{\mathbf{p}}$ in $\mathbf{t}$ and the moment $\mathbf{m}$ in $\mathbf{w}$ are divided by the characteristic length L.

The RTR Jacobian matrix is obtained upon differentiation of Eq. (16) with respect to each joint position $q_i \in \mathbf{q}$:

$$\frac{\partial \rho}{\partial q_i} = \rho\, \frac{\mathbf{w}^\top \mathbf{J}_w \frac{\partial \mathbf{J}_w^\top}{\partial q_i} \mathbf{w} ||\mathbf{J}_w^+ \mathbf{t}||^2 - ||\mathbf{J}_w^\top \mathbf{w}||^2 \mathbf{t}^\top \mathbf{J}_w^{+\top} \frac{\partial \mathbf{J}_w^+}{\partial q_i} \mathbf{t}}{(||\mathbf{J}_w^\top \mathbf{w}||\,||\mathbf{J}_w^+ \mathbf{t}||)^2}, \tag{17}$$

where the values of the end-effector twist $\mathbf{t}$ and wrench $\mathbf{w}$ are known from the trajectory planning. The derivative of the pseudo-inverse weighted kinematic Jacobian matrix $\partial \mathbf{J}_w^+ / \partial q_i$ is defined in [28].

The RTR Jacobian matrix $\mathbf{J}_\rho$ as a function of the joint variables is:

$$\mathbf{J}_\rho = \begin{bmatrix} \frac{\partial \rho}{\partial q_1} & \cdots & \frac{\partial \rho}{\partial q_n} \end{bmatrix}, \tag{18}$$

where $n = \text{columns}(\mathbf{J}_w)$ is the dimension of joint space.

3. Design and optimization method

This paper proposes a new optimization method intended for redundant robots. The main idea is to optimize the design of a robot exploiting the same algorithm used during its real-time control. In particular, the idea is to benefit from the fact that the TPIK algorithm was specifically designed to control highly redundant robot manipulators. This section presents the problem formulation and all the phases that compose this optimization method. In the considered case study, the robot main application requires following a trajectory inside a defined workspace while maintaining satisfactory kinetostatic performance. However, this optimization methodology can be applied to any robot and task.

3.1. Problem formulation

The application considered is the tracking of a set of trajectories describing a workspace area while maintaining high kinetostatic performance. So, two principal inputs are defined and given to the optimization algorithm. The first one is a series of p trajectories with desired orientations and velocities. These trajectories describe the workspace area where the robot will work. The second input is the list of tasks that the robot should achieve while tracking the trajectories. The main tasks are related to the end-effector pose and velocity to track the desired trajectories. A safety task is added for joint limit compliance. Three tasks based on dexterity, manipulability and RTR evolution are used to improve the robot kinetostatic performance. While the dexterity and the manipulability are kinematic performance indices, maximizing them forces the manipulability hyper-ellipsoid to be as big as possible and close to a hyper-sphere. So, ideally, the robot will be able to move with the same high velocity amplification factor in all directions while reducing actuator velocity limits. For this reason, the optimization method uses both dexterity and manipulability in design optimization. A final task based on the robot center of mass position is included. This task aims to to maintain the robot center of mass position as close as possible to the base and, as a result, to maximize the robot compactness.

Table 1 reports all the information about the tasks used during the optimization. The dexterity, manipulability and RTR tasks have the same priority level since they have the same relevance in the design optimization process. The optimization algorithm uses a convex combination $\epsilon(\eta, \nu, \rho)$ of the three kinetostatic performance indices to rate and compare the kinetostatic performance of the obtained designs. It should be noted that η and ρ are bounded between $[0, 1]$ whereas μ is not bounded, $[0, \infty)$. So, it is necessary to bound μ in the range $[0, 1]$ before writing the convex combination. A new index called bounded manipulability ν is defined as:

$$\nu = 1 - \frac{1}{1 + \mu}. \tag{19}$$

When $\mu = 0$ then $\nu = 0$, and when $\mu = \infty$ then $\nu = 1$. Now, η, ν and ρ are all bounded between 0 and 1 and can then be used in a convex combination:

$$\epsilon(\eta, \nu, \rho) = \lambda_1 \eta + \lambda_2 \nu + \lambda_3 \rho, \tag{20}$$

Table 1: Details about the task names, control objective types, and hierarchy levels. Symbol (E) represents the equality control objective tasks and (I) the inequality ones. The last two columns list the task hierarchies for actions $\mathscr{A}_1$ (Reach Pose) and $\mathscr{A}_2$ (Follow Trajectory). The symbol "/" means that a task is not present in an action.

			Hierarchy level	
Task	Category	Type	$\mathscr{A}_1$	$\mathscr{A}_2$
Joint Limit	system safety	I	1^{st}	1^{st}
End-Effector Pose	action oriented	E	2^{nd}	/
End-Effector Velocity	action oriented	E	/	2^{nd}
Dexterity	optimization	I	3^{rd}	3^{rd}
Manipulability	optimization	I	3^{rd}	3^{rd}
RTR	optimization	I	3^{rd}	3^{rd}
Center of Mass Position	optimization	I	4^{th}	4^{th}

where λ_1, λ_2 and λ_3 are scaling factors. Since all the kinetostatic performance indices are valid in the same range, the weighting factors are selected as $\lambda_1 = \lambda_2 = \lambda_3 = 1/3$.

3.2. Preliminary phase

The optimization algorithm is divided into two main phases. The first is the candidate generation phase, where several design candidates are generated. The second phase is the candidate selection one, where the robot designs obtained from the optimization process are evaluated and compared. Before these phases, some preliminary steps are necessary. The robot design parameters $\boldsymbol{\zeta}$ to be optimized need to be selected, for example, the link length or the angular offset amplitude between two consecutive joints. Then, controllable virtual joints are inserted in the robot architecture to substitute these design parameters. A link is substituted by a prismatic joint to control its length, while an angular offset by a revolute joint to modify its angle. So, the vector of the design parameters $\boldsymbol{\zeta}$ is converted in the virtual joint vector $\mathbf{q}_v$. During the candidate generation phase, the vector of the joint variables contains both the position of the real and virtual joints $\mathbf{q} = [\mathbf{q}_r, \mathbf{q}_v] \in \mathbb{R}^n$, with $\mathbf{q}_r \in \mathbb{R}^r$, $\mathbf{q}_v \in \mathbb{R}^v$ and the robot degrees of freedom is $n = r + v$. The joint limit task constrains the virtual joints $\mathbf{q}_v$ value between the desired limits and so the resulting value of the design parameters $\boldsymbol{\zeta}$ is limited.

3.3. Candidate generation phase

This phase employs the robot with real and virtual joints $\mathbf{q} = [\mathbf{q}_r, \mathbf{q}_v]$. The robot configuration vector $\mathbf{q}$ is randomly initialized. From here, the robot is moved to track all the p trajectories in a random order to ensure more general results. The robot reaches the starting pose of the trajectories and tracks it entirely under the kinematic control of the TPIK algorithm. When the robot finishes tracking a trajectory, it is moved to another one until it follows all the p trajectories. At equidistant time steps on each trajectory, the optimization algorithm saves the values of the virtual joints $\mathbf{q}_v$, which will be used as design parameters $\boldsymbol{\zeta}$, and the ϵ value in that configuration. Once the robot has tracked

Algorithm 1 Candidate Generation Phase

Require: Actions $\mathscr{A}_1$ = *Reach Pose* and $\mathscr{A}_2$ = *Follow Path* and trajectory vector $\mathbf{p}$.

for $r := 1 \rightarrow$ number of repetitions **do**
  Random initialization of $\mathbf{q} = [\mathbf{q}_r, \mathbf{q}_v]$.
  Random shuffle of $\mathbf{p}$.
  for $k := 1 \rightarrow p$ **do**
    Reach starting pose of $\mathbf{p}(k)$ using $\mathscr{A}_1$.
    while Trajectory $\mathbf{p}(k)$ not finished **do**
      Move to next step on $\mathbf{p}(k)$ using $\mathscr{A}_2$.
      if t_i = equidistant time step **then**
        Save $\mathbf{q}_v$ as $\boldsymbol{\zeta}^{t_i}$.
        Save ϵ^{t_i}.
      end if
    end while
  end for
  Compute weighted average $\overline{\boldsymbol{\zeta}}$ for all $[t_0, \ldots, t_\mathrm{f}]$.
end for

all the p trajectories, the optimization algorithm makes the weighted average $\overline{\boldsymbol{\zeta}}$ of $\boldsymbol{\zeta}$ using the values already stored at equidistant steps along each trajectory and the associated ϵ as weighting factor:

$$\overline{\boldsymbol{\zeta}} = \frac{\epsilon^{t_0}\boldsymbol{\zeta}^{t_0} + \cdots + \epsilon^{t_\mathrm{f}}\boldsymbol{\zeta}^{t_\mathrm{f}}}{\epsilon^{t_0} + \cdots + \epsilon^{t_\mathrm{f}}}, \tag{21}$$

where $[t_0, \ldots, t_\mathrm{f}]$ are the equidistant time steps in which the design parameter vector $\boldsymbol{\zeta}$ was saved. The average result $\overline{\boldsymbol{\zeta}}$ is also stored. The process described above is repeated several times. At the end of this phase, the optimization algorithm has collected a list of candidate designs described by $\boldsymbol{\zeta}$. Algorithm 1 sums up all the steps of the candidate generation phase.

During this phase, the optimization algorithm computes the kinetostatic performance indices η, ν and ρ of the real robot architecture without virtual joints. To correctly compute these metrics and their derivatives, the columns corresponding to the virtual joints $\mathbf{q}_v$ in $\mathbf{J}_w$ and its derivative $\partial\mathbf{J}_w/\partial q_i$, $\forall q_i \in \mathbf{q}$ are set to zero.

3.4. Candidate selection phase

The candidate selection phase tests the designs obtained from the candidate generation phase to identify the best one. Here, the virtual joints are removed from the robot architecture vector and replaced by constant links and offsets whose value was stored inside $\boldsymbol{\zeta}$. Hence, from now on $\mathbf{q} = [\mathbf{q}_r]$ and $v = 0$, and the total amount of degrees of freedom $n = r$. The robot configuration vector $\mathbf{q}$ is randomly initialized for each optimized design. Again, the robot is moved to the starting pose of one trajectory and tracks it entirely under the kinematic control of the TPIK algorithm. When the robot finishes tracking one trajectory, it is moved to the next one until it has followed all the p trajectories. At each step on each trajectory, the optimization algorithm stores the value of ϵ for that robot configuration.

Algorithm 2 Candidate Selection Phase

Require: Actions $\mathscr{A}_1$ = *Reach Pose* and $\mathscr{A}_2$ = *Follow Path* and trajectory vector $\mathbf{p}$.

for $d := 1 \rightarrow$ number of optimized designs **do**
  for $r := 1 \rightarrow$ number of repetitions **do**
    Random initialization of $\mathbf{q} = [\mathbf{q}_r]$.
    for $k := 1 \rightarrow p$ **do**
      Reach starting pose of $\mathbf{p}(k)$ using $\mathscr{A}_1$.
      while Trajectory $\mathbf{p}(k)$ not finished **do**
        Move to next step on $\mathbf{p}(k)$ using $\mathscr{A}_2$.
        Save ϵ^{t_i} at the time step t_i.
      end while
    end for
    Compute $\hat{\epsilon} = 0.5\ \epsilon_{\min} + 0.25\ (\overline{\epsilon} + \epsilon_{\max})$.
  end for
end for
Select the best design as $\max(\hat{\epsilon})$.

When the robot has tracked all the trajectories, the optimization algorithm computes $\hat{\epsilon}$ as

$$\hat{\epsilon} \triangleq 0.5\ \epsilon_{\min} + 0.25\ (\overline{\epsilon} + \epsilon_{\max}), \tag{22}$$

where $\epsilon_{\min}$, $\overline{\epsilon}$ and $\epsilon_{\max}$ are respectively the minimum, mean and maximum of ϵ values along all the p trajectories. The $\hat{\epsilon}$ value is used to compare the designs and identify the best one. The process is repeated for all the designs obtained from the previous phase. The design that has the highest $\hat{\epsilon}$ is identified as the best one. Algorithm 2 sums up all the steps of the candidate selection phase.

4. Application and results

The following section introduces the robot used to test the proposed optimization method and discusses the obtained results.

4.1. Robot under study

A new highly redundant robot actuated by a mechanism recently developed and patented by the company Nimbl'Bot [29] is used as a case study in this section. Both the mechanism and the robot are fully introduced in [17, 30]. The mechanism that actuates the robot has two degrees of freedom, and ten mechanisms are arranged in a serial way assembling the Nimbl'Bot robot, shown in Fig. 1. The robot is divided into three regions attached by two links: a shoulder composed of three mechanisms, an elbow consisting of four mechanisms, and a wrist composed of three mechanisms. At the end of the robot, a revolute joint is attached to adjust the wrist orientation. In total, the robot has 21 degrees of freedom, i.e., $r = 21$. Thus, this robot is hyper-redundant, which means that the end-effector can reach a given pose in an infinite number of configurations [31].

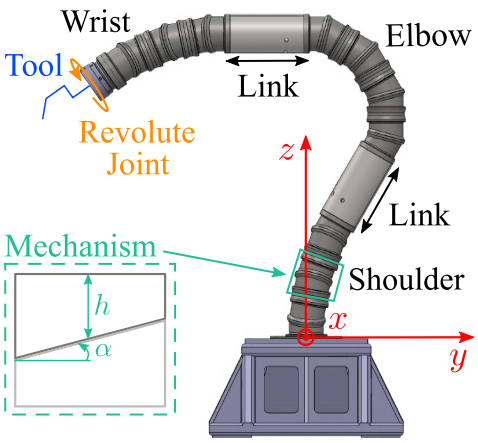


Figure 1: The 21 degrees of freedom robot under optimization; the shoulder and the wrist are composed of three mechanisms, and the elbow is made of four mechanisms. In the bottom left corner, the Nimbl'Bot mechanism is shown with its design parameters half height h and slope α.

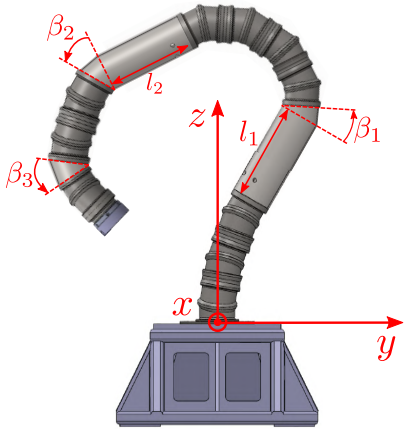


Figure 2: Robot design parameters under optimization: link lengths l_1 and l_2, angular offsets β_1, β_2 and β_3.

In this case, the dimensions of the mechanism are constant and not included in the optimization. The optimized robot design parameters are the link lengths l_1 and l_2 and the amplitude of three angular offsets β_1, β_2 and β_3, shown in Fig. 2. The angular offsets are respectively inserted between the first link and first mechanism of the elbow, the second link and first mechanism of the wrist, and between the second-to-last and last mechanisms of the wrist. The angles β_1, β_2 and β_3 rotate about the axis x, depicted in red in Fig. 2. An ending tool is mounted on the last revolute joint of the robot. The design parameters of the tool are inserted in the optimized variables, namely its length l_t and orientation β_t, shown in Fig. 3. So, the number of virtual joints is $v = 7$ and the robot with virtual joints has 28 degrees of freedom, i.e. $r + v = 28$. Table 2 gives the main parameter values and limits of the robot.

The design parameters under optimization are limited, but the limits are large enough not to over-constrain the optimization process. The link lengths l_1 and l_2 are constrained in the range $[0, 3]$ m, and the angular offsets amplitude β_1 in $[-3\pi/4, \pi/4]$ and β_2 and β_3 in $[-3\pi/4, 3\pi/4]$. The tool length l_t is valid in the range $[0, 0.5]$ m, and the tool orientation offsets β_t in $[0, 3\pi/4]$. These ranges are used by the joint limits task to constrain the virtual joints. The Nimbl'Bot mechanism actuators can rotate infinitely and have no limits.

4.2. Workspace area

The robot application requires tracking horizontal and vertical paths in a cube whose sides are 0.7 m × 0.7 m long and centered in $(x, y, z) = (0.0, 0.75, 0.65)$ m. The orientation of the end-effector is expressed in terms of roll ϕ, pitch θ and yaw ψ. The angle

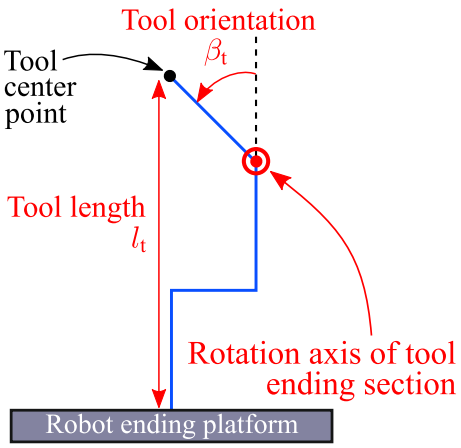


Figure 3: Ending tool design parameters under optimization: tool length l_t and orientation β_t.

Table 2: Robot details.

Mechanism half height h	0.07 m
Mechanism slope α	15°
Max/min revolute joint velocities	± 1.0 rad/s
Max module revolute joint accelerations/decelerations	2.5 rad/s^2
Max/min prismatic joint velocities	± 1.0 m/s
Max module prismatic joint accelerations/decelerations	2.5 m/s^2

values for the horizontal paths are $(\phi, \theta) = (\pi, 0)$, $\forall\ \psi \in [-\pi, \pi]$, and for the vertical paths $(\phi, \theta) = (\pi/2, 0)$, $\forall\ \psi \in [-\pi, \pi]$.

Figure 4 shows the $p = 4$ paths used during the candidate generation and selection phases of the optimization process. Two paths are horizontal, i.e. the green and the magenta ones, and the other two are vertical, i.e. the blue and the black ones. The horizontal paths have the corners placed in $x = (-0.35, 0.35)$ m and $y = (0.4, 1.1)$ m at the height of $z = 0.3$ m, green path, and $z = 1.0$ m, magenta path. The vertical paths have the corners placed in $x = (-0.35, 0.35)$ m and $z = (0.3, 1.0)$ m at the depth of $y = 0.4$ m, blue path, and $y = -1.1$ m, black path. The small arrows along the paths express the z-axis orientation of the end-effector frame while following the paths. These four paths were chosen for the design optimization because they describe the cube where the robot is supposed to work in the real world. Then, the kinetostatic performance of the best and worst designs are compared on two new paths, one horizontal and one vertical, placed inside the cubic workspace area to confirm the optimization process ability to identify the best design for kinetostatic performance. Figure 5 shows the two paths used for testing the design obtained from the optimization process. The horizontal path has the corners placed in $x = (-0.35, 0.35)$ m and $y = (0.4, 1.1)$ m at the height of $z = 0.65$ m, red path. The vertical path has the corners placed in $x = (-0.35, 0.35)$ m and $z = (0.3, 1.0)$ m at the depth of $y = 0.75$ m, cyan path. Table 3 gives the trajectory details.

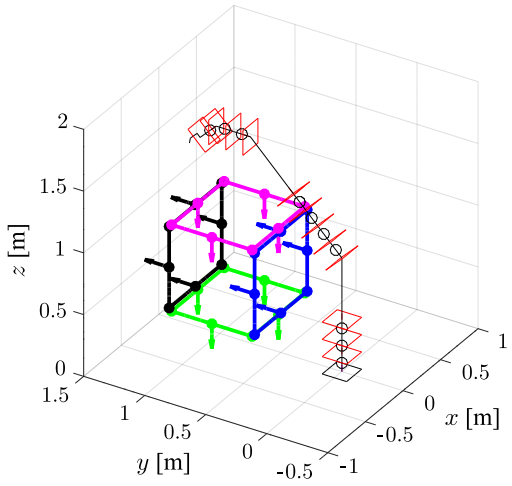


Figure 4: The four paths used in the optimization process and a version of Nimbl'bot robot of size: $l_1 = 0.5$ m, $l_2 = 0.5$ m, $\beta_1 = \pi/4$, $\beta_2 = \pi/4$, $\beta_3 = \pi/4$, $l_t = 0.1$ m, , $\beta_t = \pi/4$.

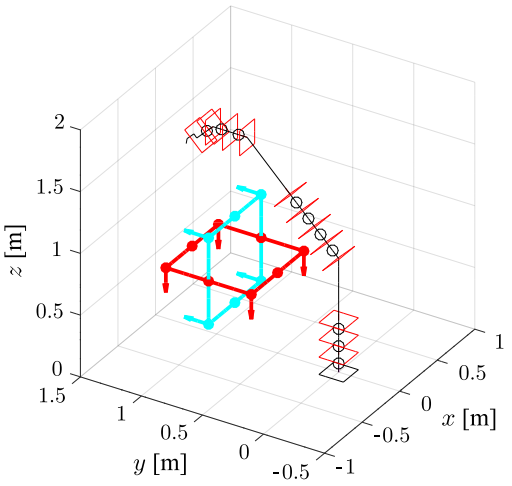


Figure 5: The two paths used to validate the designs obtained from the optimization process and a version of Nimbl'bot robot of size: $l_1 = 0.5$ m, $l_2 = 0.5$ m, $\beta_1 = \pi/4$, $\beta_2 = \pi/4$, $\beta_3 = \pi/4$.

Table 3: Trajectory details.

Steps	560
Magnitude velocity vector $\|\|\vec{\mathbf{v}}\|\|$	0.002 m/s
Magnitude tangential force vector $\|\|\vec{\mathbf{f}}_t\|\|$	60 N
Magnitude radial force vector $\|\|\vec{\mathbf{f}}_r\|\|$	20 N
Time	1400 s

4.3. Computational time evaluation and comparison

Before optimizing all the design variables of Figs. 2 and 3, a brute force optimization process is run to have a baseline for the comparison with the proposed methodology. In this case, only the design variables $l_2 \in [0, 1]$ m and $\beta_2 \in [-3\pi/4, 3\pi/4]$ are optimized as a function of the green and blue paths shown in Fig. 4. The other design variables are set to $l_1 = 0.2$ m, $\beta_2 = \beta_3 = 0$, $l_t = 0.1$ m and $\beta_t = \pi/4$. Several robot designs are generated combining the discretization of l_2 and β_2 over their allowed ranges. Namely, l_2 is bounded between 0 and 1 with a step equal to 0.1 m, leading to 11 possible values. The parameter β_2 is bounded between $-3\pi/4$ and $3\pi/4$ with a step equal to 5°, leading to 55 values. Combining l_2 and β_2 generates 605 designs that are tested on the green and blue trajectories. The goal is to identify the design with the highest performance in terms of $\hat{\epsilon}$. Each design is tested ten times on each trajectory to obtain general results. If a design is not able to follow one or more trajectories, it is discarded.

Then, the design optimization process presented in this paper is run optimizing l_2 and β_2 as a function of the green and blue trajectories and the kinetostatic performance. The candidate generation phase is repeated four times and produces only 76 designs. So, the

Table 4: Design parameter and kinetostatic performance values $\hat{\epsilon}$, $\hat{\eta}$, $\hat{\nu}$ and $\hat{\rho}$ for the best designs obtained from discretization and optimization, respectively.

Method	l_2	β_2	$\hat{\epsilon}$	$\hat{\eta}$	$\hat{\nu}$	$\hat{\rho}$	Time
Discretization	0.1 m	-105°	0.269	0.222	0.087	0.558	120 min
Optimization	0.06 m	-102°	0.281	0.221	0.086	0.595	20 min

Table 5: Machine details.

Operating System	Linux
Distribution	Ubuntu 20.04
CPUs number	4
CPU model	Intel Core i7 10th Gen, 1.30GHz
Language	C++
Control frequency	10Hz
Time to track one trajectory	5 s

candidate selection phase takes less time in testing the 76 designs on the desired trajectories. Each design is tested ten times along each path to obtain general results. If one design is not able to follow one or more trajectories, it is discarded.

Table 4 shows the designs obtained from the two tests described above and their kinetostatic performance $\hat{\epsilon}$, $\hat{\eta}$, $\hat{\nu}$ and $\hat{\rho}$. The values of $\hat{\eta}$, $\hat{\nu}$ and $\hat{\rho}$ are computed in the same way of Eq. (22) using the η, ν and ρ values collected at path step. The best designs identified by the two methods are highly similar. However, the optimization process reached a higher $\hat{\epsilon}$ because the identified values for l_2 and β_2 were not included in the discretized values. Moreover, the time required by the optimization process, 20 minutes, is lower than testing all the designs obtained from the discretization, more than 120 minutes. Therefore, the optimization method is six times faster in this case. In this case, only two design variables were considered. Testing all the possible combinations for seven design parameters, which correspond to more than two billion designs, would be highly time consuming.

4.4. Optimization test and results

Here, the optimization process is applied to all the design variables of Figs. 2 and 3. The optimization is performed with respect to the four desired trajectories of Fig. 4, and the kinetostatic performance indices, dexterity η, bounded manipulability ν and RTR ρ. The candidate generation phase is run ten times generating 370 designs. Then, the candidate selection phase tests these designs on the four trajectories collecting ϵ, η, ν and ρ at each step. In total, 282 designs passed the candidate selection phase. The others could not properly track all the trajectories and were discarded. During the candidate selection phase, each design follows each trajectory ten times starting from a random configuration to obtain general results. Table 5 shows the machine and implementation details.

Table 6 reports the correlation coefficients between the kinetostatic performance indices $\hat{\epsilon}$, $\hat{\eta}$, $\hat{\nu}$ and $\hat{\rho}$ and the optimized design parameters for all the solutions obtained from the optimization process. The higher the absolute value of the correlation coefficient, the

Table 6: Correlation coefficients between kinetostatic performance indices $\hat{\epsilon}$, $\hat{\eta}$, $\hat{\nu}$ and $\hat{\rho}$ and design parameters, bottom left, and their standard deviation along the diagonal, gray. The higher the correlation coefficient absolute value, the darker the blue shade. Correlation coefficients $\in [-1, 1]$.

	$\hat{\epsilon}$	$\hat{\eta}$	$\hat{\nu}$	$\hat{\rho}$	l_1	l_2	β_1	β_2	β_3	l_t	β_t
$\hat{\epsilon}$	0.06										
$\hat{\eta}$	0.947	0.09									
$\hat{\nu}$	0.654	0.579	0.06								
$\hat{\rho}$	0.810	0.770	0.176	0.05							
l_1	0.394	0.310	0.823	-0.026	0.41 m						
l_2	0.326	0.206	0.753	-0.035	0.743	0.4 m					
β_1	0.090	0.048	-0.081	0.280	-0.104	0.100	27.2°				
β_2	-0.326	-0.336	-0.225	-0.328	0.049	-0.008	-0.261	116.7°			
β_3	-0.141	-0.148	-0.132	-0.098	0.076	-0.089	-0.081	0.426	52.5°		
l_t	0.155	0.187	0.182	0.082	0.105	0.105	-0.118	0.111	-0.083	0.08 m	
β_t	-0.016	-0.018	0.092	-0.165	-0.012	-0.079	-0.107	-0.069	-0.213	-0.098	13.7°

Standard deviation

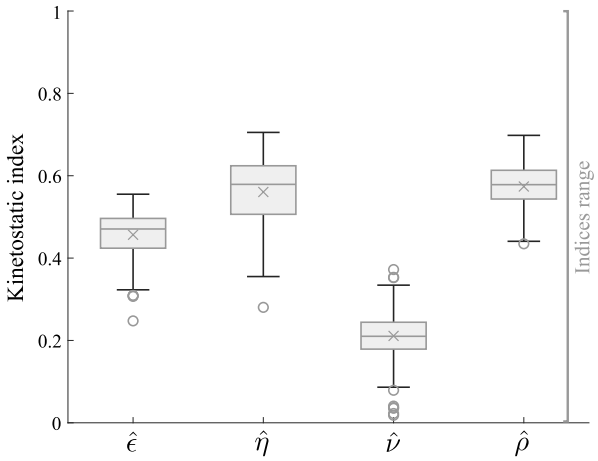
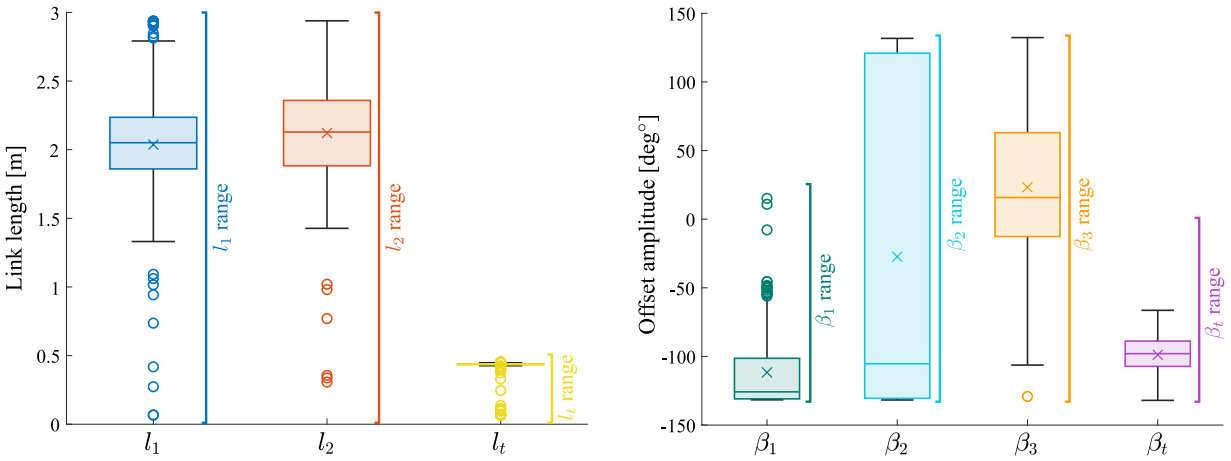


Figure 6: Box plot of the values assumed by the kinetostatic indices $\hat{\epsilon}$, $\hat{\eta}$, $\hat{\nu}$ and $\hat{\rho}$ for the selected candidates. Symbol $\times$ represents the mean and $\circ$ indicates the outliers. The kinetostatic index range is displayed next to the plots.

darker the shade of blue associated. A positive correlation coefficient indicates a directly proportional relation between two parameters. A negative correlation coefficient means a indirectly proportional relationship. The value of $\hat{\eta}$, $\hat{\nu}$ and $\hat{\rho}$ is computed in the same way of Eq. (22) using the η, ν and ρ values collected at each step on each trajectory. The diagonal of Table 6 collects the standard deviation of the kinetostatic performance indices and design parameters between all the solutions identified by the optimization process. Figures 6 and 7 show the box plots of all the values assumed by the kinetostatic indices $\hat{\epsilon}$, $\hat{\eta}$, $\hat{\nu}$ and $\hat{\rho}$ and the design parameters for the selected candidates, respectively. From Table 6, it is clear that $\hat{\nu}$ is directly linked to the size of l_1 and l_2. The longer the links, the higher $\hat{\nu}$. Moreover, a longer link l_1 tends to be associated with a longer l_2. Another interesting aspect concerns $\hat{\eta}$. The higher $\hat{\eta}$, the higher both $\hat{\nu}$ and $\hat{\rho}$. This means that the same design parameters influence



(a) Box plot for the links l_1, l_2 and l_t.

(b) Box plot for the offset β_1, β_2, β_3 and β_t.

Figure 7: Box plots of the values assumed by the design parameters for the selected candidates. Symbol $\times$ represents the mean and $\circ$ indicates the outliers. The design variable range is displayed next to each plot.

them. On the contrary, the correlation coefficient between $\hat{\nu}$ and $\hat{\rho}$ is low meaning that different design variables influence these two metrics. However, the correlation coefficients between $\hat{\eta}$, $\hat{\nu}$ and $\hat{\rho}$ are always positive meaning that these performance indices are not antagonistic. A consideration can be done on the correlation coefficients between β_2 and the indices $\hat{\epsilon}$, $\hat{\eta}$, $\hat{\nu}$ and $\hat{\rho}$, which are negative. This means that the lower β_2 the larger $\hat{\epsilon}$, $\hat{\eta}$, $\hat{\nu}$ and $\hat{\rho}$. Moreover, there is a relevant positive correlation between β_2 and β_3, meaning that a bigger β_2 is associated to a bigger β_3. The other design parameters do not present significant correlation coefficients. However, this fact does not mean that any value can be used for these design parameters. For example, the standard deviation of the kinetostatic indices $\hat{\epsilon}$, $\hat{\eta}$, $\hat{\nu}$ and $\hat{\rho}$ is small meaning that the obtained results are closer to each other. The plots in Fig. 6 show how each index is concentrated in a smaller area compared to the available one. Considering the design parameters, the standard deviation of l_t is much lower than l_1 and l_2. This means that almost all the obtained designs have a similar value for l_t. So, the optimization algorithm identified a small optimal range of values for l_t. This can also be noticed by comparing the interquartile size, i.e. the distance between the lower and upper quartile, of the box plots in Fig. 7a. Similarly, β_1, β_3 and β_t have a standard deviation lower than β_2 meaning that the optimization algorithm found a smaller optimal range of values for these parameters. Moreover, the β_2 interquartile size in Fig. 7b is much bigger than the other ones. This can justify why the correlation coefficients between β_1, β_3 and β_t and $\hat{\epsilon}$, $\hat{\eta}$, $\hat{\nu}$ and $\hat{\rho}$ are low as well. There is no design that can be identified globally as the best one, but several combinations can lead to a good kinetostatic design for the Nimbl'Bot robot. This is mainly due to the high kinetostatic redundancy of the robot that allows infinite possible solutions for the same problem. Moreover, more than one kinetostatic performance index is considered in the optimization process. So, the designs identified by the optimization process can perform better in dexterity, manipulability or RTR. It is necessary to make a trade-off for choosing the best design.

Table 7: Design parameters and kinetostatic performance indices $\hat{\epsilon}$, $\hat{\eta}$, $\hat{\nu}$ and $\hat{\rho}$ for best and worst designs obtained during candidate selection phase. Green and red colors indicate the highest and lowest values for $\hat{\epsilon}$, $\hat{\eta}$, $\hat{\nu}$ and $\hat{\rho}$ obtained during the candidate selection and testing phases.

	l_1	l_2	β_1	β_2	β_3	l_t	β_t	Candidate selection				Testing			
								$\hat{\epsilon}$	$\hat{\eta}$	$\hat{\nu}$	$\hat{\rho}$	$\hat{\epsilon}$	$\hat{\eta}$	$\hat{\nu}$	$\hat{\rho}$
Best $\hat{\epsilon}$ $\hat{\rho}$	2.19 m	2.33 m	-61°	-131°	-3°	0.44 m	131°	0.555	0.695	0.214	0.698	0.556	0.671	0.209	0.718
Best $\hat{\eta}$	2.05 m	1.99 m	-108°	-131°	17°	0.44 m	97°	0.537	0.705	0.259	0.631	0.538	0.703	0.242	0.639
Best $\hat{\nu}$	2.94 m	2.78 m	-131°	-108°	38°	0.44 m	105°	0.509	0.599	0.372	0.564	0.449	0.526	0.320	0.527
Worst $\hat{\epsilon}$ $\hat{\eta}$ $\hat{\nu}$	0.07 m	0.31 m	11°	-105°	1°	0.44 m	73°	0.247	0.280	0.018	0.458	0.292	0.350	0.024	0.522
Worst $\hat{\rho}$	1.54 m	2.32 m	-131°	131°	-14°	0.44 m	105°	0.309	0.363	0.137	0.435	0.435	0.552	0.176	0.540

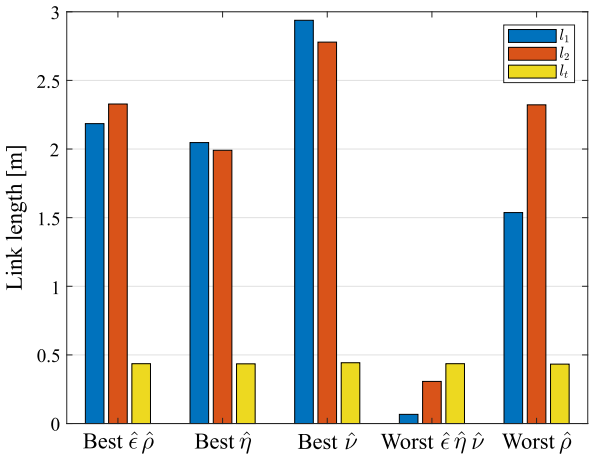


Figure 8: Link lengths of best and worst design for indices $\hat{\epsilon}$, $\hat{\eta}$, $\hat{\nu}$ and $\hat{\rho}$ during candidate selection phase.

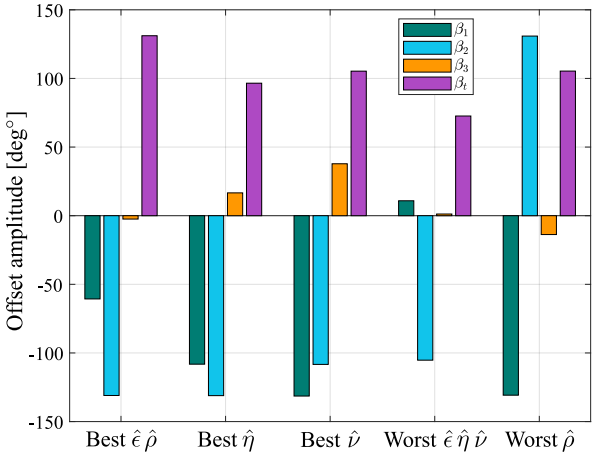


Figure 9: Offset amplitudes of best and worst design for indices $\hat{\epsilon}$, $\hat{\eta}$, $\hat{\nu}$ and $\hat{\rho}$ during candidate selection phase.

An analysis of the designs that performed the best and the worst in terms of $\hat{\epsilon}$, $\hat{\eta}$, $\hat{\nu}$ and $\hat{\rho}$ is accomplished. The right side of Table 7 shows the parameter values of the designs that performed the best and worst for each index, i.e. $\hat{\epsilon}$, $\hat{\eta}$, $\hat{\nu}$ and $\hat{\rho}$, along the four trajectories used by the optimization process, shown in Fig. 4. In some cases, one design had the best or worst performance for more than one index. Figures 8 and 9 present the link lengths and offset amplitudes collected in Table 7 on graphs. The middle of Table 7 reports the values of $\hat{\epsilon}$, $\hat{\eta}$, $\hat{\nu}$ and $\hat{\rho}$ obtained during the candidate selection phase. The green and red colors highlight the highest and lowest values for $\hat{\epsilon}$, $\hat{\eta}$, $\hat{\nu}$ and $\hat{\rho}$ during the candidate selection phase. One design has the worst performance in terms of $\hat{\epsilon}$, $\hat{\eta}$ and especially for $\hat{\nu}$. Small links characterize this design. In fact, the design that is characterized by the best $\hat{\nu}$ has the largest links. The value of $\hat{\rho}$ is generally higher for all the designs, best and worst. This means that the Nimbl'Bot robot generally has high performance in terms of $\hat{\rho}$, independently of the design parameters. Finally, β_3 is generally close to 0° while β_t is generally high and take the place of β_3. So, β_3 could be removed from the optimized design parameters. Figures 10 and 11 show the designs that reached the best and worst $\hat{\epsilon}$.

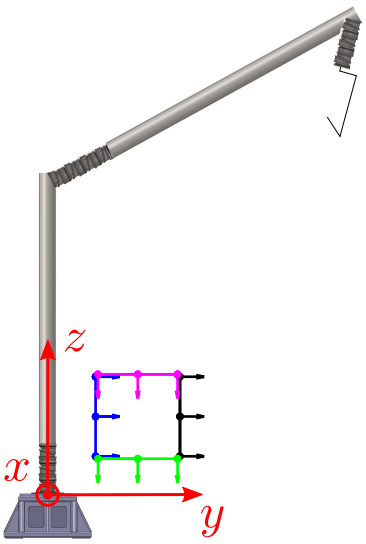


Figure 10: Representation of the design with the best $\hat{\epsilon}$ together with the four trajectories used for the optimization, its design parameter values are in the first line of Table 7.

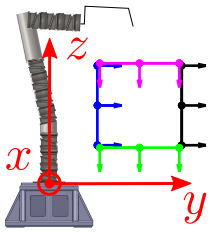


Figure 11: Representation of the design with the worst $\hat{\epsilon}$ together with the four trajectories used for the optimization, its design parameter values are in the fourth line of Table 7.

4.5. Selected designs performance inside desired workspace

The designs collected in Table 7 are tested on two trajectories inside the desired workspace area, shown in Fig. 5, which are different from the ones used during the candidate generation and selection phases. This testing phase is performed to show that the best and worst designs maintain similar performance in tracking trajectories inside the volume defined by the four ones used in the optimization process. The left side of Table 7 reports the values of $\hat{\epsilon}$, $\hat{\eta}$, $\hat{\nu}$ and $\hat{\rho}$ obtained during the testing phase. The green and red colors highlight the highest and lowest values for $\hat{\epsilon}$, $\hat{\eta}$, $\hat{\nu}$ and $\hat{\rho}$ during the testing phase. The results show the same trend during the candidate selection and testing phases meaning that each design has uniform performance in the desired workspace area. To conclude, this optimization process identified several possible optimal designs for a specific set of tasks in the desired workspace area. There is no unique global solution, but more than one design can reach high values for $\hat{\epsilon}$. For designs that share the same global index $\hat{\epsilon}$, choosing the final one is a trade-off between optimizing one of the three indices that contribute to $\hat{\epsilon}$.

5. Conclusions

This paper presented and analyzed a new method for designing and optimizing kinematic redundant manipulators as a function of their application and kinetostatic performance. The process employs a task priority kinematic control algorithm to solve several tasks simultaneously exploiting the robot high kinematic redundancy. First, the process was tested on a simple case with two design variables and two paths. The obtained best design was compared with the results obtained from testing some discretized combinations of the two design

variables. The best designs in terms of the global kinetostatic performance index $\hat{\epsilon}$ obtained from the two tests are close and the optimization process converged toward a better design. This means that the optimization process can identify the best solution for a specific problem without testing all possible combinations of design parameter values and reduce the computational time.

Then, this new method worked on more design parameters identifying several possible Nimbl'Bot manipulators with high kinetostatic performance for the desired workspace area. It also pointed out some general guidelines for building a kinetostatic efficient Nimbl'Bot robot. However, there is no specific design trend to obtain high kinetostatic performance. This is due to the high redundancy of the robot that admits infinite possible robot configurations to solve the same problem. Moreover, the kinetostatic performance index $\hat{\epsilon}$ is a linear combination of dexterity η, bounded manipulability ν and RTR ρ. Choosing the best design is a trade-off between these metrics.

Future work will involve more analysis to identify if some global guidelines exist for building a Nimbl'Bot robot design with high kinetostatic performance, giving more importance to one index than others or searching for specific relationships between the design parameters. Other design parameters will be included to study how they affect the kinetostatic performance. New tasks based on self-collision and obstacle avoidance will be added to consider more complex problems.

Acknowledgments

This paper was supported by the ANRT (Agence Nationale Recherche Technologie) grant CIFRE no 2020/1051 and Nimbl'bot company.

References

[1] V. Kumar, S. Sen, S. S. Roy, C. Har, S. Shome, Design optimization of serial link redundant manipulator: an approach using global performance metric, Procedia Technology 14 (2014) 43–50.

[2] J. Angeles, The design of isotropic manipulator architectures in the presence of redundancies, The International Journal of Robotics Research 11 (1992) 196–201.

[3] S. Hwang, H. Kim, Y. Choi, K. Shin, C. Han, Design optimization method for 7 dof robot manipulator using performance indices, International Journal of Precision Engineering and Manufacturing 18 (2017) 293–299.

[4] C. J. Paredis, P. K. Khosla, Kinematic design of serial link manipulators from task specifications, The International Journal of Robotics Research 12 (1993) 274–287.

[5] J.-O. Kim, P. K. Khosla, A formulation for task based design of robot manipulators, in: Proceedings of 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'93), volume 3, IEEE, 1993, pp. 2310–2317.

[6] J. Whitman, H. Choset, Task-specific manipulator design and trajectory synthesis, IEEE Robotics and Automation Letters 4 (2018) 301–308.

[7] M. T. Chikhaoui, J. Granna, J. Starke, J. Burgner-Kahrs, Toward motion coordination control and design optimization for dual-arm concentric tube continuum robots, IEEE Robotics and Automation Letters 3 (2018) 1793–1800.

[8] J.-Y. Park, P.-H. Chang, J.-Y. Yang, Task-oriented design of robot kinematics using the grid method, Advanced robotics 17 (2003) 879–907.

[9] H. Al-Dois, A. Jha, R. Mishra, Task-based design optimization of serial robot manipulators, Engineering Optimization 45 (2013) 647–658.
[10] O. W. Maaroof, M. İ. C. Dede, L. Aydin, A robot arm design optimization method by using a kinematic redundancy resolution technique, Robotics 11 (2022) 1.
[11] A. Escande, N. Mansard, P.-B. Wieber, Hierarchical quadratic programming: Fast online humanoid-robot motion generation, The International Journal of Robotics Research 33 (2014) 1006–1028.
[12] P. Di Lillo, S. Chiaverini, G. Antonelli, Handling robot constraints within a set-based multi-task priority inverse kinematics framework, in: 2019 International Conference on Robotics and Automation (ICRA), IEEE, 2019, pp. 7477–7483.
[13] F. Flacco, A. De Luca, O. Khatib, Prioritized multi-task motion control of redundant robots under hard joint constraints, in: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2012, pp. 3970–3977.
[14] E. Simetti, G. Casalino, A novel practical technique to integrate inequality control objectives and task transitions in priority based control, Journal of Intelligent & Robotic Systems 84 (2016) 877–902.
[15] E. Simetti, G. Casalino, M. Aicardi, F. Wanderlingh, Task priority control of underwater intervention systems: Theory and applications, Ocean Engineering 164 (2018) 40–54.
[16] E. Simetti, G. Casalino, M. Aicardi, F. Wanderlingh, A task priority approach to cooperative mobile manipulation: Theory and experiments, Robotics and Autonomous Systems 122 (2019) 103287.
[17] A. Ginnante, S. Caro, E. Simetti, F. Leborne, Kinetostatic optimization for kinematic redundancy planning of nimbl'bot robot, ASME Journal of Mechanisms and Robotics (2023) 1–20.
[18] W. A. Khan, J. Angeles, The kinetostatic optimization of robotic manipulators: The inverse and the direct problems, Journal of Mechanical Design 128 (2005) 168–178.
[19] S. Khan, K. Andersson, J. Wikander, Jacobian matrix normalization - A comparison of different approaches in the context of multi-objective optimization of 6-dof haptic devices, Journal of Intelligent & Robotic Systems 79 (2015) 87–100.
[20] S. Zargarbashi, W. Khan, J. Angeles, Posture optimization in robot-assisted machining operations, Mechanism and Machine Theory 51 (2012) 74–86.
[21] T. Yoshikawa, Manipulability of robotic mechanisms, The international journal of Robotics Research 4 (1985) 3–9.
[22] J. Angeles, Fundamentals of robotic mechanical systems: theory, methods, and algorithms, Springer, 2003.
[23] J. Park, Analysis and control of kinematically redundant manipulators: An approach based on kinematically decoupled joint space decomposition, PhD Thesis, POSTECH (2000).
[24] G. Marani, J. Kim, J. Yuh, W. K. Chung, A real-time approach for singularity avoidance in resolved motion rate control of robotic manipulators, in: Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292), volume 2, IEEE, 2002, pp. 1973–1978.
[25] J. Angeles, C. S. López-Cajún, Kinematic isotropy and the conditioning index of serial robotic manipulators, The International Journal of Robotics Research 11 (1992) 560–571.
[26] G. Pond, J. A. Carretero, Formulating jacobian matrices for the dexterity analysis of parallel manipulators, Mechanism and Machine Theory 41 (2006) 1505–1519.
[27] N. Rakotomanga, D. Chablat, S. Caro, Kinetostatic performance of a planar parallel mechanism with variable actuation, in: Advances in robot kinematics: Analysis and design, Springer, 2008, pp. 311–320.
[28] G. H. Golub, V. Pereyra, The differentiation of pseudo-inverses and nonlinear least squares problems whose variables separate, SIAM Journal on numerical analysis 10 (1973) 413–432.
[29] L. Dufau, Articulated robot arm, March 23, 2021. US patent 10,953,554.
[30] A. Ginnante, F. Leborne, S. Caro, E. Simetti, G. Casalino, Design and kinematic analysis of a novel 2-dof closed-loop mechanism for the actuation of machining robots, in: International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, volume 85444, American Society of Mechanical Engineers, 2021.
[31] G. S. Chirikjian, J. W. Burdick, Kinematically optimal hyper-redundant manipulator configurations, IEEE transactions on Robotics and Automation 11 (1995) 794–806.

Graphical Abstract

Task Priority Based Design Optimization of a Kinematic Redundant Robot

Angelica Ginnante, Enrico Simetti, Stéphane Caro, François Leborne

Highlights

Task Priority Based Design Optimization of a Kinematic Redundant Robot

Angelica Ginnante, Enrico Simetti, Stéphane Caro, François Leborne

- A new method for the design of kinematic redundant robots.
- A task priority control algorithm is used to perform the design optimization.
- The design variables are replaced by controllable prismatic or revolute virtual joints.
- The method has a low computational time requirement while being accurate.
- Illustration of good and bad Nimbl'Bot designs.