Design Strategy of Serial Manipulators With Certified Constraint Satisfaction

Denny Oetomo, *Member, IEEE*, David Daney, and Jean-Pierre Merlet, *Member, IEEE*

***Abstract*—This paper presents the design strategy of serial manipulators with constraint satisfaction. The algorithm provides certified solutions to the range of values of the manipulator design parameters that satisfy the given constraints for all points inside a desired workspace. Alternatively, it can also be used to obtain the achievable workspace of a particular manipulator topology within which a set of given constraints are satisfied. This strategy can therefore be applied to the general case of a serial manipulator design problem, robots of adjustable parameters, or even reconfigurable robot strategy to obtain a suitable topology. The interval-based algorithm was implemented on an example serial anthropomorphic manipulator with joint displacement constraints and obtains the possible variations to the manipulator topology that allow the required workspace to be achievable under the given joint displacement constraints. Results are presented and discussed.**

***Index Terms*—Certified workspace, interval analysis, kinematics, mechanism design, medical robots and systems.**

I. Introduction

THIS PAPER addresses the problem of determining the suitable serial-chain topologies such that a set of given constraints is guaranteed to be satisfied within all points in the defined workspace. The algorithm can also be utilized to obtain the certified workspace of a serial-chained mechanism of specific topology within which all given constraints are satisfied.

It is important to be able to certify whether or not a set of required constraints is satisfied in the design of a robotics manipulator. This is especially important as requirements for robotic tasks get more stringent in the face of many precise, sensitive, critical, and costly applications today. It is also necessary to provide the high-level intelligence to determine the required topology of a kinematic chain such that a given set of requirements are satisfied.

In this paper, the term topology is used to mean the arrangements of joints and linkages forming the mechanism. The term manipulator configuration is sometimes used in other papers to mean the same thing; however, configuration can also be confused with the joint configuration of a manipulator, which refers to the pose of the manipulator expressed in joint space. To avoid the confusion, the term topology is used in this paper to mean the design/arrangement of joints and linkages of the manipulator.

The problem of design and kinematic workspace of a manipulator has been extensively investigated in the past. Studies on serial manipulator designs and the resulting workspace were presented in [1] and [2]. Workspace definition procedures for closed-chained mechanisms have been explored through the geometric approach [3]–[5], screw theory [6], and interval analysis [7]–[11]. However, design strategies of a manipulator to satisfy a set of given objectives are less explored. Most of the design strategies proposed are optimization-based [12]–[14]. Drawbacks associated with optimization-based approaches include the difficulties in constructing a suitable cost function, especially when several measures of different physical properties are involved and the physical meanings of variables are lost in the construction of the function. Optimization approaches also often push the solutions to one side of the extreme without considering whether certain thresholds of the other constraints are violated.

In addition to designing a kinematic chain, automated generation of suitable topology is also useful in the area of reconfigurable robots [15]. In this topic, various aspects of the problem have also been studied extensively in the past, such as module design [16], [17], docking-and-release strategies [18], [19], reconfiguration strategy [17], as well as various gait and locomotion strategies [20]. However, there are usually a set of predefined topologies to which the reconfigurable robot can rearrange to. An automated generation strategy of suitable topology considering the task at hand requires the high-level intelligence that is still not well established at the moment.

In this paper, an interval-based method is proposed to provide an automated search in the design parameter space to determine the ranges of values in the robot design parameters that would yield an achievable end-effector workspace where all given constraints are satisfied for all poses inside the desired workspace. The design parameters are the parameters such as link lengths and offset angles that completely describe a serial chain topology and configuration, such as the Denavit–Hartenberg parameters. In interval-based method, these parameters, as well as the joint and task space variables, are expressed as bounded ranges of continuous values (intervals) instead of as real variables. This renders the algorithm with the ability to accommodate a continuous range of possible values for each parameter and evaluate the serial-chain performance for all points within these bounds, not just on point sampling basis. This also allows the algorithm to take into account any uncertainties, such as modeling

Manuscript received April 2, 2008. First published November 18, 2008; current version published February 4, 2009. This paper was recommended for publication by Associate Editor F. Thomas and Editor F. Park upon evaluation of the reviewers' comments. This work was supported by Assembly of Reconfigurable Endoluminal System (ARES) Project, an NEST-Adventure European Union Funded Project, under EU Contract 015653.

D. Oetomo was with the Constraints Solving, Optimization, Robust Interval Analysis (COPRIN) Project, Institut National de Recherche en Informatique et en Automatique (INRIA) Sophia-Antipolis, Nice FR-06902, France. He is now with the Department of Mechanical Engineering, University of Melbourne, Melbourne, VIC 3010, Australia (e-mail: doetomo@unimelb.edu.au).

D. Daney and J.-P. Merlet are with the Constraints Solving, Optimization, Robust Interval Analysis (COPRIN) Project, Institut National de Recherche en Informatique et en Automatique (INRIA) Sophia-Antipolis, Nice FR-06902, France (e-mail: david.daney@sophia.inria.fr; jean-pierre.merlet@sophia.inria.fr).

Digital Object Identifier 10.1109/TRO.2008.2006867

uncertainties, fabrication tolerances, and roundoff errors, among others, to produce the certified solutions to the constraints.

In contrast to optimization-based approaches, interval analysis techniques maintain the physical meanings of various parameters and quantitative measures. The techniques are also able to verify that all these functions meet their required performance within the given bounds of uncertainties. The result is a set of windows in the manipulator design parameter space, within which it is guaranteed that each point defines a manipulator topology that satisfies all the required constraints for all points in the desired workspace. If desired, optimization technique can be performed afterward on the solution boxes found by the interval techniques. This would yield an optimized solution out of the set of design parameters that are already certified to satisfy all the given constraints.

The study presented in this paper is carried out under a European Union project Assembly of Reconfigurable Endoluminal Surgical System (ARES) that aims to perform surgical procedures within the gastrointestinal tract through reconfigurable robotics modules. Therefore, it is necessary for the algorithm to consider the reachable workspace of the resulting manipulators under given constraints. One of the most common constraints in robotics is the joint displacement limit. It should be noted that the algorithm proposed in this paper addresses any general constraints involved in a design problem, as long as they can be expressed in mathematical equalities and inequalities. In this paper, an interval-based verification algorithm of constraint satisfaction is defined first (Section II). This is placed inside a nested branch-and-bound loop that searches through (all points in) the end-effector workspace and the design parameter space, respectively (Section III). The joint displacement limits are presented as the constraints in obtaining the reachable workspace and the design parameters of a serial manipulator. As the manipulator topology is not initially known in the design process, an interval inverse kinematic process is proposed for a serial chain without any explicitly defined topology (Section IV). The design process and workspace verification of an anthropomorphic serial manipulator are presented as examples in the design exercise (Section V). The results of the algorithm on this manipulator are presented and discussed.

II. Performance Constraints in a Serial Manipulator

In the design exercise, the desired workspace is the end-effector (task-space) range of poses (translations and orientations) that is required for the given task. Within this workspace, all constraints are to be satisfied; for example, it is required that all points within the desired workspace be reachable given the joint limits of the manipulator, the workspace does not contain singular configurations, a required amount of force at the end-effector is achievable given the available joint torques, etc. The set of constraints is defined as $\mathbf{C}(\mathbf{x},\mathbf{h}) = \{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_{\mathrm{NC}}\}$, where $\mathbf{x}$ is the end-effector pose, $\mathbf{h}$ represents the design parameters, $\mathcal{C}_i(\mathbf{x},\mathbf{h})$ represents a single constraint, and $_{\mathrm{NC}}$ denotes the total number of constraints being imposed on the system. Each constraint $\mathcal{C}_i(\mathbf{x},\mathbf{h})$ could be defined as an inequality or equality constraint. It should be noted that an equality constraint $\mathcal{C}_i(\mathbf{x},\mathbf{h}) = 0$ is more difficult to accommodate compared to an inequality constraint $\mathcal{C}_i(\mathbf{x},\mathbf{h}) \leq 0$. Fortunately, many of the workspace constraints can be expressed as inequality constraints.

In this paper, interval analysis method [21] is used as a means to express a variable as a continuous range of values. An interval extension of variable x is denoted by $X = [\underline{x}, \overline{x}]$, where $\underline{x}$ and $\overline{x}$ are the lower and upper bounds of the interval variable, respectively. Similarly, an interval extension of a function $f(x)$ is denoted as $F(X) = [\underline{f}, \overline{f}]$. Introduction to interval analysis can be found in the literature, such as [22].

A specific interval box X can therefore be defined as an *inner* box when

$$\forall x \in X, \qquad f(x) = 0 \tag{1}$$

and as an *outer* box when

$$\forall x \notin X, \qquad f(x) \neq 0. \tag{2}$$

In the case of our manipulator design problem, each constraint $\mathcal{C}_i(\mathbf{x},\mathbf{h})$ can be expressed as the required range of value, bounded within $[\underline{c}_i, \overline{c}_i]$. They represent the minimal and maximal values that a quantitative property (the constraint) is allowed to have. An interval box $(\mathbf{X},\mathbf{H})$ is therefore an *inner box* when the interval extension of a quantifier $F_i(\mathbf{X},\mathbf{H}) = [\underline{f}_i, \overline{f}_i]$ evaluates to an interval value such that

$$\{\forall \mathbf{x} \in \mathbf{X}, \forall \mathbf{h} \in \mathbf{H}; \underline{c}_i \leq \underline{f}_i \text{ and } \overline{f}_i \leq \overline{c}_i\} \tag{3}$$

for all constraints, $(i = 1, \ldots, \mathrm{NC})$. An interval box $(\mathbf{X},\mathbf{H})$ is an *outer box* when

$$\{\forall \mathbf{x} \in \mathbf{X}, \forall \mathbf{h} \in \mathbf{H}; \overline{f}_i \leq \underline{c}_i \text{ or } \underline{f}_i \geq \overline{c}_i\} \tag{4}$$

for any of the constraints. When $(\mathbf{X},\mathbf{H})$ does not yield an interval $\mathbf{F}(\mathbf{X},\mathbf{H})$ that is completely contained within the limits of constraints (inner box) or completely outside these limits (outer box), it is designated as a *boundary box*.

In Section III, the solution search algorithm in the design parameter space and the end-effector workspace are elaborated. The algorithms are constructed within branch-and-bound loop(s) to facilitate the solution search process. The algorithms are set to yield the inner, outer, and boundary boxes to the given constraints, in either design parameter space or the end-effector workspace.

In Section IV, the interval-based inverse kinematic solution is presented. As the topology of the resulting mechanism is not known, explicit inverse kinematics cannot be constructed. The joint limit constraints are presented in obtaining the reachable workspace and the design parameters of a serial manipulator. These constraints need to be satisfied for all points within a defined workspace of the manipulator. The kinematic relationship between the joint and task space displacement of a serial manipulator can always be expressed in the forward kinematic form of

$$\mathbf{x} = F_{FK}(\mathbf{q}) = {}^0\mathbf{T}_1 \cdot {}^1\mathbf{T}_2 \cdots {}^{N-1}\mathbf{T}_N \tag{5}$$

where $\mathbf{x}$ is the task space pose of the end-effector, $\mathbf{q}$ is the vector containing joint space displacement $(q_1, q_2, \ldots, q_N)$, ${}^{i-1}\mathbf{T}_i$ is the transformation matrix from link $i-1$ to link i, and N denotes the total number of joints in the serial manipulator.

Solving for $\mathbf{q}$ through interval analysis yields the N-element joint displacement (interval) vector $\mathbf{Q}$ for the M-dimensional interval box containing the range of the end-effector pose variable $\mathbf{X}$. M is the number of degrees of freedom in task space that the manipulator possesses. An example of the algorithm is implemented on a workspace analysis and design algorithm of a 6-DOF serial manipulator with the results and discussion given in Section V.

III. Interval Analysis Algorithm for Serial Kinematic Chains

To construct the interval-based algorithm for serial kinematic chains, a function $S(\mathbf{x}, \mathbf{h})$ is defined such that it evaluates the interval extension of the kinematic functions that are constrained in the design of the system and compares it with the given constraints and returns 1, 0, -1, if interval $(\mathbf{X}, \mathbf{H})$ forms an inner, boundary, and outer box to the constraints, respectively. Specifically, when given a set of joint limit constraints $[q_{\min}, q_{\max}]$, the algorithm $S(\mathbf{x}, \mathbf{h})$ returns

1) -1 if $\underline{q}_i > q_{\max}$ or $\overline{q}_i < q_{\min}$ for at least one of the joints and at least one pose in $(\mathbf{X}, \mathbf{H})$;
2) 1 if $\underline{q}_i \geq q_{\min}$ and $\overline{q}_i \leq q_{\max}$ for all joints and all poses of $(\mathbf{X}, \mathbf{H})$;
3) 0 if $(\underline{q}_i < q_{\min}$ and $\overline{q}_i > q_{\min})$ or $(\underline{q}_i < q_{\max}$ and $\overline{q}_i > q_{\max})$ for at least one of the joints and at least one pose in $(\mathbf{X}, \mathbf{H})$.

A pseudocode example of a function $S(\mathbf{X}_i, \mathbf{H})$ given a set of joint limit constraints is the $IK(\mathbf{X}_i, \mathbf{H}, \mathcal{F})$ function, used in this paper to calculate the inverse kinematics of a given workspace and then compared them to the joint limits $[q_{\min}, q_{\max}]$ and decide whether the inverse kinematics solutions fall within or outside of the allowed joint limits, or it is not decided as the solution spans across the inner and outer regions of the allowed joint limits. Function $IK(\mathbf{X}_i, \mathbf{H}, \mathcal{F})$ is elaborated on Section IV.

When there are more than one constraint involved, function $S(\mathbf{x}, \mathbf{h})$ will evaluate and enforce all the constraints and return

1) -1 if $(\mathbf{X}, \mathbf{H})$ forms an outer box to one or more constraints;
2) 0 if $(\mathbf{X}, \mathbf{H})$ forms a boundary box to one or more constraints and forms no outer box to any constraints;
3) 1 if $(\mathbf{X}, \mathbf{H})$ forms inner boxes to all constraints.

The function $S(\mathbf{x}, \mathbf{h})$ forms the basic procedure to certify whether or not workspace box $\mathbf{x}$ and design parameter box $\mathbf{h}$ satisfy a set of given constraints. In this paper, this function is utilized to obtain 1) definition of solutions in end-effector workspace $(\mathbf{X})$ and 2) constraint satisfaction solutions in design parameter space $(\mathbf{H})$.

In the interval evaluation of a function, numerical values are substituted into a function, resulting in the loss of the relationships between various variables. Overestimation [22] occurs as multiple occurrences of the same variables within the function are regarded as independent variables. The evaluation of a function where all variables involved only appear once is *sharp* (within rounding errors), meaning it is bounded within the smallest possible "box." For example, let $X = [1, 2]$ and $Y = [3, 6]$. Evaluating $F(X, Y) = X + Y = [4, 8]$ would therefore be *sharp*. However, evaluating $G(X, Y) = X - X = [1, 2] - [1, 2]$ results in $[-1, 1]$ and is not equal to zero, although we know it should be. This is because the two variables X are taken as being independent and not as the same variable. Due to overestimation, it is difficult to obtain clear solution whether a box forms an inner or an outer box. This reduces the effectiveness and efficiency of the algorithm.

Branch-and-bound loops were utilized as the solution search algorithm. This allows the interval algorithm to consider only on smaller subsections of the boxes at each iteration. Solutions are searched throughout the two variables spaces, i.e., end-effector workspace $\mathbf{X}$ and design parameter $\mathbf{H}$. The following sections further explain the two modes of usage.

A. Definition of Constraint Satisfaction in the End-Effector Workspace

Evaluating constraints in the end-effector workspace for a specific set of manipulator design is carried out by setting a constant design parameter $\mathbf{H}$. This allows the algorithm to analyze only the workspace of a range of mechanisms of specific design $\mathbf{H}$. (Note that $\mathbf{H}$ is a set of interval design parameters.) In this usage, a single branch-and-bound loop is constructed to evaluate the end-effector workspace against the set of constraints $\mathbf{C}$. The aim is to "categorize" if a box $(\mathbf{X}, \mathbf{H})$ forms an inner or outer box to the constraints. If it yields a boundary box, then the workspace $\mathbf{X}_i$ is recursively divided into smaller subboxes to be evaluated further. Some of the smaller subboxes of the workspace will eventually evaluate to inner or outer boxes. The splitting of the workspace into smaller spaces is terminated once a threshold dimension of the workspace $\mathbf{X}_i$, which is denoted as ϵ_x, is reached. The remaining boxes at this point are assigned as boundary boxes of the system. It should be noted that the overestimation of interval analysis decreases with the size of the input intervals; hence, the process of splitting the workspace into smaller spaces also improves the accuracy of the interval extension.

Division of an interval pose $\mathbf{X}_i$ into smaller boxes is termed the *bisection* process. There are various methods of *bisection*, ranging from the simplistic approaches (e.g., round robin or largest first) to the more complicated but effective methods [23]. One such example is the *smear* function [24] that utilizes the derivative of the system. It aimed at recognizing the dominant variable that affects the system the most, hence potentially producing the most improvement through bisection process.

The result of this exercise is a map in the task space (workspace) showing the regions of inner and outer boxes, as well as boundary boxes, to the set of constraints, for a particular set of design parameters $\mathbf{H}$. The algorithm is summarized in Table I. In this summary, lists $\mathcal{L}_{\text{IN}}$, $\mathcal{L}_{\text{OUT}}$, $\mathcal{L}_B$ contain the subregions of $\mathbf{X}$ forming the inner, outer, and boundary boxes to the constraints, respectively.

B. Constraint Satisfaction Solutions in Design Parameter Space

To search for solutions in the design parameter space $\mathbf{H}$ such that all constraints $\mathbf{C}(\mathbf{X}, \mathbf{H})$ are satisfied for all points $\mathbf{x} \in \mathbf{X}$,

TABLE I
SUMMARY OF BRANCH-AND-BOUND LOOP TO OBTAIN THE DEFINITION OF SOLUTIONS IN END-EFFECTOR WORKSPACE

Input: $\mathbf{X}$ and $\mathbf{H}$
Output: 1, 0, -1, (representing inner, boundary, and outer solutions, respectively).
1 Initialise empty lists $\mathcal{L}_{IN}$, $\mathcal{L}_{OUT}$, and $\mathcal{L}_B$.
2 Initialise list $\mathcal{L}$ containing initial task space intervals $\mathbf{X}_0$ to be analysed.
3 While ($\mathcal{L}$ not empty)
(a) Extract interval pose $\mathbf{X}_i$ from $\mathcal{L}$
(b) Remove interval pose $\mathbf{X}_i$ from list $\mathcal{L}$
(c) Evaluate function $S(\mathbf{X}_i, \mathbf{H})$,
(d) If $S(\mathbf{X}_i, \mathbf{H}) == 1$
add $\mathbf{X}_i$ to list $\mathcal{L}_{IN}$
(e) Else If $S(\mathbf{X}_i, \mathbf{H}) == -1$
add $\mathbf{X}_i$ to list $\mathcal{L}_{OUT}$
(f) Else If $S(\mathbf{X}_i, \mathbf{H}) == 0$
(i) If Dimension ($\mathbf{X}_i > \epsilon_x$)
Bisect $\mathbf{X}_i$ into $\mathbf{X}_i(1)$ and $\mathbf{X}_i(2)$
Add $\mathbf{X}_i(1)$ and $\mathbf{X}_i(2)$ to $\mathcal{L}$.
(ii) Else If Dimension($\mathbf{X}_i \leq \epsilon_x$)
Add $\mathbf{X}_i$ to $\mathcal{L}_B$.
(iii) End If
(g) End If
4 End While

two (nested) branch-and-bound loops are required to search both the $\mathbf{X}$ and $\mathbf{H}$ spaces. While the algorithm in Section III-A produces the description of the regions in the given workspace that satisfy the set of constraints, the algorithm in this section produces the regions in the design parameter space that are certified to form serial-chain mechanism where all the given constraints are satisfied in the desired workspace. In other words, the aim of the algorithm is to obtain all design parameters $\mathbf{h} \in \mathbf{H}$ of the serial manipulator that would allow all constraints $\mathbf{C}$ to be satisfied within all pose in $\mathbf{x} \in \mathbf{X}$. In contrast to the algorithm in Section III-A, the design parameter $\mathbf{H}$ is now an interval parameter and is included in the bisection process and not set as constant. This implies that all the constraints have to be satisfied for all points $\mathbf{x} \in \mathbf{X}$ in order for a set of interval design parameter $\mathbf{H}_i$ to qualify as an inner box. A branch-and-bound loop, which is similar to that in Section III-A, is carried out to obtain the design parameters that produces inner and outer solutions. A function $S^*(\mathbf{X}, \mathbf{H}_i)$ is defined in a similar manner to $S(\mathbf{X}, \mathbf{H}_i)$ such that it returns 1, -1, 0 if $\mathbf{H}_i$ forms an inner, outer, and boundary boxes to the design constraints. The branch-and-bound loop that searches the design parameter space calls the function $S^*(\mathbf{X}, \mathbf{H}_i)$ to evaluate the design parameter space $\mathbf{H}$ against the given constraints. This is summarized in Table II.

Function $S^*(\mathbf{X}, \mathbf{H}_i)$ is in turn constructed as another branch-and-bound loop that assigns $\mathbf{H}_i$ to an inner, outer, or boundary box—given the required workspace $\mathbf{X}$—by calling function $S(\mathbf{X}, \mathbf{H}_i)$. Within iteration i of $S^*(\mathbf{X}, \mathbf{H}_i)$, where a box $\mathbf{H}_i$ out of the design parameter is evaluated, a search in the workspace $\mathbf{X}$ is carried out using the previously defined function $S(\mathbf{X}, \mathbf{H}_i)$. Bisection is performed to refine the search on a smaller region of the workspace. If $S(\mathbf{x}, \mathbf{H}_i) == -1$ for any point $\mathbf{x}$, then the set of design parameters $\mathbf{H}_i$ is an outer box. If no outer box is found while executing $S(\mathbf{x}, \mathbf{H}_i)$, but $S(\mathbf{x}, \mathbf{H}_i) == 0$ for any point within $\mathbf{X}$, then $\mathbf{H}_i$ is a boundary box. If no outer and no boundary box is found, then $\mathbf{H}_i$ is an inner box. Therefore,

TABLE II
SUMMARY OF BRANCH-AND-BOUND LOOP TO OBTAIN THE DESIGN PARAMETER RANGE OF VALUES WHERE ALL CONSTRAINTS ARE SATISFIED WITHIN THE GIVEN WORKSPACE $\mathbf{X}$

Input: $\mathbf{X}$ and $\mathbf{H}_i$
Output: 1, 0, -1, (representing inner, boundary, and outer solutions, respectively).
1 Initialise empty lists $\mathcal{L}_{IN}$, $\mathcal{L}_{OUT}$, and $\mathcal{L}_B$.
2 Initialise list $\mathcal{L}$ containing initial design parameter intervals $\mathbf{H}_0$ to be analysed.
3 While ($\mathcal{L}$ not empty)
(a) Extract design parameter box $\mathbf{H}_i$ from $\mathcal{L}$
(b) Remove design parameter box $\mathbf{H}_i$ from $\mathcal{L}$
(c) Evaluate function $S^*(\mathbf{X}, \mathbf{H}_i)$,
(d) If $S^*(\mathbf{X}, \mathbf{H}_i) == 1$
add $\mathbf{H}_i$ to list $\mathcal{L}_{IN}$
(e) Else If $S^*(\mathbf{X}, \mathbf{H}_i) == -1$
add $\mathbf{H}_i$ to list $\mathcal{L}_{OUT}$
(f) Else If $S^*(\mathbf{X}, \mathbf{H}_i) == 0$
(i) If Dimension ($\mathbf{H}_i > \epsilon_d$)
Bisect $\mathbf{H}_i$ into $\mathbf{H}_i(1)$ and $\mathbf{H}_i(2)$
Add $\mathbf{H}_i(1)$ and $\mathbf{H}_i(2)$ to list $\mathcal{L}$.
(ii) Else If Dimension($\mathbf{H}_i \leq \epsilon_d$)
Add $\mathbf{H}_i$ to $\mathcal{L}_B$.
(iii) End If
(f) End If
4 End While

TABLE III
SUMMARY OF ALGORITHM FOR FUNCTION $S^*(\mathbf{X}, \mathbf{H}_i)$ USED IN THE ALGORITHM IN TABLE II

Input: $\mathbf{X}_i$ and $\mathbf{H}_i$
Output: 1, 0, -1, (classifying $(\mathbf{X}, \mathbf{H})$ as inner, boundary, and outer boxes, respectively).
1 Initialise empty lists $\mathcal{L}_B$.
2 Initialise list $\mathcal{L}$ containing initial pose $\mathbf{X}$.
3 While ($\mathcal{L}$ not empty)
(a) Extract interval pose $\mathbf{X}_i$ from $\mathcal{L}$
(b) Evaluate function $S(\mathbf{X}_i, \mathbf{H}_i)$,
(c) If $S(\mathbf{X}_i, \mathbf{H}_i) == -1$
Return (-1)
Exit function $S^*(\mathbf{X}, \mathbf{H}_i)$
(e) Else If $S(\mathbf{X}_i, \mathbf{H}_i) == 0$
(i) If Dimension ($\mathbf{X}_i > \epsilon_x$)
Bisect $\mathbf{X}_i$ into $\mathbf{X}_i(1)$ and $\mathbf{X}_i(2)$
Add $\mathbf{X}_i(1)$ and $\mathbf{X}_i(2)$ to list $\mathcal{L}$.
(ii) Else If Dimension($\mathbf{X}_i \leq \epsilon_x$)
Add $\mathbf{X}_i$ to $\mathcal{L}_B$.
(iii) End If
(f) End If
4 End While
5 If ($\mathcal{L}_B$ is empty)
Return 1
6 Else
Return 0
7 End If

$S^*(\mathbf{X}, \mathbf{H}_i)$ can be defined as a function that describes a design parameter box $\mathbf{H}_i$ by returning

1) -1 for if for any pose $\mathbf{x} \in \mathbf{X}$, $S(\mathbf{x}, \mathbf{H}_i) == -1$;
2) 1 if for all pose $\mathbf{x} \in \mathbf{X}$, $S(\mathbf{x}, \mathbf{H}_i) == 1$;
3) 0 if threshold dimension ϵ_d is reached and there is at least one boundary box $\mathbf{X}_i$ where $S(\mathbf{x}, \mathbf{H}_i) == 0$ and no outer solution was found.

The algorithm of function $S^*(\mathbf{X}, \mathbf{H}_i)$ is summarized in Table III.

Simplification can be performed on $S^*(\mathbf{X}, \mathbf{H}_i)$ by substituting real numbers (can be thought of as intervals with zero width or *degenerate intervals*) for $\mathbf{X}_i$ at the beginning of the procedure in order to find any possibility of rejecting $\mathbf{H}_i$ as an outer box early in the loop. Calculation with real variables offer a much faster evaluation. An interval box $\mathbf{X}_i$ can be sampled quickly but taking its vertices (combination of all the extremal values) as well as its center point. If any of these yields an outer solution, the algorithm as summarized in Table III returns (-1) and immediately exits the function $S^*(\mathbf{X}, \mathbf{H}_i)$. It should be noted that even if a point interval is used, the calculation is still interval-based; therefore, roundoff errors will not affect the process, and the results are still certified.

Any point contained in the resulting boxes of design parameters is certified to constitute a manipulator where all points in its workspace satisfy the given constraints. Having interval design parameters means that any bounded uncertainty known to affect the design parameters is also included in the consideration when certifying the boxes. The design solution resulting from this process is therefore robust with respect to such uncertainties.

IV. Solving Inverse Kinematics of Serial Manipulators In Interval Analysis

The limit of displacements for the manipulator joints is selected as the design constraints in this paper as it is one of the most common in design problems. As joint displacement is involved given the end-effector pose requirements, it is necessary to be able to evaluate or solve the inverse kinematics of the manipulator. In this section, the strategies for solving interval kinematic relationships of serial manipulators are presented. These calculations are carried out within $S(\mathbf{X}, \mathbf{H})$.

Conventionally, the inverse kinematics of nonredundant serial manipulators are solved symbolically to obtain the closed-form solutions. However, explicit solution to the inverse kinematics is specific to the manipulator design/topology. When the topology of the mechanism is unknown, as is the case in this paper (e.g., during design process or in the case of reconfigurable robots), explicit inverse kinematics does not exist. Hence, it is necessary to construct an inverse kinematics algorithm that is general enough to be able to evaluate the performance of all possible topologies within a specified window of design parameters. In our algorithm, the forward kinematic relationship is utilized as kinematic constraints in establishing the inverse kinematic solutions. It is well known that the forward kinematics of a serial manipulator is simpler to obtain than its inverse kinematics and that the opposite is true for parallel manipulators.

As previously mentioned, the evaluation of interval functions result in overestimation due to multiple occurrences of a single variable. Therefore, evaluating the forward kinematics would result in an overestimation of the resulting workspace. Hence, to certify that a specific workspace is achievable by a given joint displacement range, it is necessary to obtain the inverse kinematics—which yields an overestimated joint space displacement required for the workspace. If the overestimated joint displacement required is within the joint limits, then it is guaranteed that the required end-effector workspace is achievable—although it is an underestimation of the true achievable workspace. A major challenge is to obtain the minimum overestimation possible for an accurate result.

Two methods were proposed for solving the inverse kinematics of serial manipulator: 1) solving method using interval Newton and 2) consistency filtering with solution existence check.

A. Solving Using Interval Newton Method

Interval Newton method is a solving method based on evaluation of a function about the center point of the intervals. It is based on a first-order Taylor series expansion, where

$$F(Q) = f(q_m) + J_q(Q) \cdot (Q - q_m) \tag{6}$$

where Q is the interval extension of variable q (i.e., $q \in Q$), $F(Q)$ is the interval extension of function $f(q)$, $J_q(Q)$ is the derivative of function $f(q)$ with respect to variable q, evaluated at interval Q, and q_m is the center of interval Q. For a multivariate function, for example, $f(q_1, q_2)$

$$\begin{aligned} F(Q_1, Q_2) = f(q_{1m}, q_{2m}) &+ J_{q_1}(Q_1, Q_2) \cdot (Q_1 - q_{1m}) \\ &+ J_{q_2}(Q_1, Q_2) \cdot (Q_2 - q_{2m}). \end{aligned} \tag{7}$$

Solving for $F(Q_1, Q_2) = 0$, the equation can be rearranged to be

$$-f(q_{1m}, q_{2m}) = [J_{q_1}(Q_1, Q_2) J_{q_2}(Q_1, Q_2)] \begin{bmatrix} Q_1 - q_{1m} \\ Q_2 - q_{2m} \end{bmatrix} \tag{8}$$

It is well established that evaluating and solving for equations with real variables requires considerably less computational resources than with interval variables. Equation (8) can therefore be rearranged to develop the Taylor series expansion in subsequent order of the variables involved. For the example of $F(Q_1, Q_2)$, Taylor expansion is first carried out with respect to q_1

$$F(Q_1, Q_2) = f(q_{1m}, Q_2) + J_{q_1}(Q_1, Q_2) \cdot (Q_1 - q_{1m}) \tag{9}$$

then with respect to q_2

$$f(q_{1m}, Q_2) = f(q_{1m}, q_{2m}) + J_{q_2}(q_{1m}, Q_2) \cdot (Q_2 - q_{2m}). \tag{10}$$

From (9) and (10), and solving $F(Q_1, Q_2) = 0$

$$-f(q_{1m}, q_{2m}) = [J_{q_1}(Q_1, Q_2) J_{q_2}(q_{1m}, Q_2)] \begin{bmatrix} Q_1 - q_{1m} \\ Q_2 - q_{2m} \end{bmatrix}. \tag{11}$$

The difference between (8) and (11) is that it is now possible to replace some of the interval variables in matrix $\mathbf{J}$ by real variables q_{im} for some joint variables q_i, providing a large improvement in computational efficiency. Equation (11) involves only two joint variables for the simplicity of presentation; however, it is obvious that this simplification in interval Taylor expansion becomes more significant with larger number of joint variables involved.

In solving for the inverse kinematics problem, let the given end-effector interval pose be described as $\mathbf{X}$, the interval joint displacement vector as $\mathbf{Q}$, and the forward kinematics of the serial manipulator as $f_{FK}(\mathbf{q})$. Solving the inverse kinematics

solution becomes a problem of solving for joint displacement interval Q, given the task space displacement X, with the following constraints:

$$F_{FK}(\mathbf{Q}) - X = 0 \tag{12}$$

$$f_{FK}(\mathbf{q}_m) - X + \mathbf{J}(\mathbf{Q}, \mathbf{q}_m) \cdot (\mathbf{Q} - \mathbf{q}_m) = 0 \tag{13}$$

where

$$\mathbf{J} = \begin{pmatrix} J_{q_1}^T(Q_1, Q_2, \ldots, Q_N) \\ J_{q_2}^T(q_{1m}, Q_2, \ldots, Q_N) \\ J_{q_3}^T(q_{1m}, q_{2m}, \ldots, Q_N) \\ \vdots \\ J_{q_N}^T(q_{1m}, q_{2m}, \ldots, q_{(N-1)m}, Q_N) \end{pmatrix}^T \tag{14}$$

and $\mathbf{q}_m$ is the vector containing the middle values of the intervals of the joint displacement (i.e., the mid of $\mathbf{Q}$). The joint displacement intervals were initialized at the allowable joint displacement limits, and the middle values $\mathbf{q}_m$ are calculated accordingly. Solving for $(\mathbf{Q} - \mathbf{q}_m)$ through the linear equations results in

$$X - f_{FK}(\mathbf{q}_m) = \mathbf{J}(\mathbf{Q}, \mathbf{q}_m) \cdot (\mathbf{Q} - \mathbf{q}_m). \tag{15}$$

In iterative manner, the interval inverse kinematic solutions of the serial manipulator is obtained.

B. Consistency Filtering Method

Instead of directly solving for the solutions of $\mathbf{q}$, it is also possible to perform consistency filtering on the forward kinematic equations. Consistency filtering techniques originated from the constraint programming field of study. These techniques serve to contract interval variables by removing part(s) of each interval variable that is not consistent within the given constraints. To perform this, each joint displacement variable is expressed initially as an interval bounded by the allowable joint limits as dictated by the physical constraints. For a given value of the end-effector interval pose, consistency filtering removes the extremal part(s) of the intervals that do not conform to the forward kinematic equations of the manipulator. One of the following cases could happen.

1) If consistency filtering fails, then the allowable joint space displacements inside the joint limits are inconsistent with the forward kinematic equations. This means that there is no inverse kinematic solutions to the given manipulator end-effector pose, in which case end-effector pose $\mathbf{X}$ is an outer box.
2) If consistency filtering causes the N-dimensional interval joint displacement variables $\mathbf{Q}$ to contract on all sides, it means that there is no inverse kinematic solution at the boundary of the joint limits. If inverse kinematic solution exists, it would lie completely within the joint limits. To verify the existence of the inverse kinematic solution, a Newton method could be performed on the contracted joint displacement intervals $\mathbf{Q}$. If Newton method shows that solution exists within the box, then $\mathbf{X}$ is an inner box. Otherwise, $\mathbf{X}$ is an outer box. Another approach can be as simple as evaluating the forward kinematics of manipulator at the center point of the resulting joint displacement interval. If the forward kinematics evaluated at $\mathrm{mid}(\mathbf{Q})$ results in a point within the given end-effector pose $\mathbf{X}$, then solution to inverse kinematics within the joint limits is shown to exist. However, the opposite does not necessarily mean that no solution exists within box $\mathbf{Q}$; hence, box $\mathbf{X}$ is assigned to be bisected further. As $\mathrm{mid}(\mathbf{Q})$ is a real vector, evaluating the forward kinematics of this (noninterval) point requires much less computational effort than solving for interval Newton method to establish the existence of the solution.
3) The N-dimensional interval joint displacement variable $\mathbf{Q}$ does not contract on all sides after the consistency filtering process. In this case, the box Q is assigned to be bisected further.

In this paper, interval 2B and 3B consistency techniques [25]–[27] were utilized to solve the system of equation. Other filtering techniques are available in interval methods such as Newton-based techniques [21], [22], [28], [29].

C. Bisection on the Joint (Solution) Space

It is well known that inverse kinematics problem can possess multiple solutions, even for well-constrained problems (nonredundant serial manipulators). For example, a planar two-link RR manipulator can have an *elbow up* and *elbow down* joint displacement solutions for one end-effector position. When such case exists, the consistency filtering technique will converge only to the outer bounds of the multiple solutions, instead of to the individual sets of solution. A quick *bisection* procedure on the interval joint variables $\mathbf{Q}$ is therefore necessary. The summary of the procedure is described in Table IV. In the procedure, each set of possible solution goes through a bisection process, which splits the interval along joint displacement interval variable Q_i, for $i = 1, \ldots, N$, where N is the number of joints (steps 6(b) in Table IV). Each time a set of possible solution is bisected, the algorithm tries to solve for the inverse kinematics [steps 10(b)–(d)]. If the bisected branch produces a solution, then an inverse kinematic solution within the joint limit is found. If it is confirmed to have no solution, then the branch of possible solution is eliminated. If the result is not sufficient to conclude either way, then the branch is kept for bisection on subsequent joint variable [steps 10(e)–(i)].

The essential point about the bisection process in the solution space ($\mathbf{Q}$) is to be able to isolate the individual solutions such that the solving or consistency filtering algorithm is able to converge. When there are two sets of solutions, for instance, splitting the solution space in between the solutions would tremendously increase the efficiency of the solving or filtering techniques. However, not knowing where the solutions are, it is possible that the bisection is carried out at the point of solution/in the range that contains the solution. Hence, the proposed procedure also checks whether or not the bisection process produces two separate solutions. If no separate solutions are found after bisection, the set of joint displacement $\mathbf{Q}_i$ is returned as the *Hull* (i.e., the union box) of the two bisected halves.

TABLE IV
SUMMARY OF ALGORITHM $IK(\mathbf{X}_i, \mathbf{H}, \mathcal{F})$: INTERVAL INVERSE KINEMATICS FOR A SERIAL MANIPULATOR WITH JOINT LIMIT CONSTRAINTS

Input: $\mathbf{X}_i$, $\mathbf{H}$, and $\mathcal{F}$
Output: 1, 0, -1, (representing inner, boundary, and outer solutions, respectively).
1 Initialise empty list for solutions $\mathcal{L}_{\mathcal{S}}$.
2 Initialise initial estimates for the joint displacements $\mathbf{Q}^0 = (Q_1^0, Q_2^0, ..Q_N^0)$ at the joint limits.
3 $\mathbf{Q}' = SolveQ(\mathbf{X}_i, \mathbf{H}, \mathbf{Q}, \mathcal{F})$
4 If (solution $\mathbf{Q}'$ does not exist)
(a) Return (-1)
(b) Exit function $IK(\mathbf{X}_i, \mathbf{H}, \mathcal{F})$
5 Else If (solution $\mathbf{Q}'$ exists)
(a) Load $\mathbf{Q}'$ onto $\mathcal{L}_{\mathcal{S}}$
(b) Return (1)
(c) Exit function $IK(\mathbf{X}_i, \mathbf{H}, \mathcal{F})$
6 Else
(a) SolutionFound = 0; i = 1;
(b) While i <= N AND SolutionFound $\neq$ 1
SolutionFound = Bisect-and-Solve($\mathcal{L}_{IN}$, $\mathcal{L}_{OUT}$, i);
$\mathcal{L}_{IN}$ = $\mathcal{L}_{OUT}$;
(c) End While
(d) Return (SolutionFound)
7 End If

8 Sub-Procedure:
9 SolFound = **Bisect-and-Solve**($\mathcal{L}_{IN}$, $\mathcal{L}_{OUT}$, i)
Input: $\mathcal{L}_{IN}$: list of joint displacement $\mathbf{Q}$ which possibly contain solutions.
i: index of joint displacement Q_i to be bisection this round.
Output: SolFound (1 solution is found, -1 no solution, and 0 when it is not clear).
$\mathcal{L}_{OUT}$ list of joint displacement $\mathbf{Q}$ which possibly contain solutions for next round of bisections.
10 While ($\mathcal{L}_{IN}$ not empty AND SolFound==0)
(a) Extract $\mathbf{Q}$ from $\mathcal{L}_{IN}$.
(b) Bisect Q_i to $Q_i'(1)$ and $Q_i'(2)$ to form $\mathbf{Q}'(1)$ and $\mathbf{Q}'(2)$
(c) $\mathbf{Q}"(1) = SolveQ(\mathbf{X}_i, \mathbf{H}, \mathbf{Q}'(1), \mathcal{F})$
(d) $\mathbf{Q}"(2) = SolveQ(\mathbf{X}_i, \mathbf{H}, \mathbf{Q}'(2), \mathcal{F})$
(e) If $\mathbf{Q}"(1)$ or $\mathbf{Q}"(2)$ are inverse kinematic solution
Return SolFound = 1;
exit $Bisect - and - Solve$.
(f) Else If $\mathbf{Q}"(1)$ and $\mathbf{Q}"(2)$ do not form separate solutions
Push Hull($\mathbf{Q}"(1)$, $\mathbf{Q}"(2)$) onto $\mathcal{L}_{OUT}$.
(g) Else If both $\mathbf{Q}"(1)$ and $\mathbf{Q}"(2)$ do not exist
This entry does not contain solution.
(h) Else If $\mathbf{Q}"(1)$ exists
Load $\mathbf{Q}"(1)$ onto $\mathcal{L}_{OUT}$.
(i) Else If $\mathbf{Q}"(2)$ exists
Load $\mathbf{Q}"(2)$ onto $\mathcal{L}_{OUT}$.
11 End While
12 If no entries contain inverse kinematic solution:
(a) Return SolFound = -1;
13 Else Return SolFound = 0;

Variable *SolFound* in Table IV denotes the state of whether or not inverse kinematics solution exists for a box $\mathbf{X}_i, \mathbf{H}$. SolFound $= 1$ when an inverse kinematic solution is found, SolFound $= -1$ when all possible bisections of the joint space $\mathbf{Q}_i$ are not solutions to the inverse kinematics, or SolFound $= 0$ when it is not clear.

D. Implementation on an Example

To decide whether a desired end-effector task space $\mathbf{X}$ is reachable given the joint limit constraints, an inverse kinematic function $\mathrm{IK}(\mathbf{X}_i, \mathbf{H})$ is defined. It returns 1, 0, -1, if $\mathbf{X}_i$ yields an inner, boundary, and outer box, respectively, to the inverse kinematics, given the allowable joint displacement limits. Note that function $\mathrm{IK}(\mathbf{X}_i, \mathbf{H})$ can be performed inside $S(\mathbf{X}_i, \mathbf{H})$, or it can even replace $S(\mathbf{X}_i, \mathbf{H})$ if joint limit by inverse kinematics is the only constraints to be enforced in the analysis and design process.

Within $\mathrm{IK}(\mathbf{X}_i, \mathbf{H})$, the interval inverse kinematics is solved. The relationship, as shown in (5), can be expressed as

$$
\begin{aligned}
{}^0\mathbf{x}_E &= F_{FK}(\mathbf{q}) = {}^0\mathbf{T}_1 \cdot {}^1\mathbf{T}_2 \cdot \ldots \cdot {}^{N-1}\mathbf{T}_N \\
\begin{pmatrix} \mathbf{R}_E & \mathbf{p}_E \\ \mathbf{0}_{1\times 3} & 1 \end{pmatrix} &= \begin{pmatrix} \mathbf{R}_{FK}(\mathbf{Q}) & \mathbf{p}_{FK}(\mathbf{Q}) \\ 0_{1\times 3} & 1 \end{pmatrix}
\end{aligned} \tag{16}
$$

where $\mathbf{R}_E$ and $\mathbf{p}_E$ are the given intervals of rotational matrix and position vector for the end-effector describing the required workspace and $\mathbf{R}_{FK}$ and $\mathbf{p}_{FK}$ contain the expressions of the interval rotational matrix and position vector, as functions of the joint displacement variables, obtained through forward kinematics (multiplication of transformation matrices).

The constraints for the kinematic relationship can therefore be elaborated as

$$
\begin{aligned}
\mathcal{F}_1 &= p_{FK}(i) - p_E(i) = 0 \\
\mathcal{F}_2 &= R_{FK}(i,j) - R_E(i,j) = 0
\end{aligned} \tag{17}
$$

where $i, j = 1, 2, 3$, $p_{FK}(i), p_E(i)$ are the (x, y, z) components of the position vectors, and $R_{FK}(i,j)$, $R_E(i,j)$ are the element (i,j) of the 3×3 rotational matrices. Additionally, due to the redundant representation of orientation through rotational matrix, the following constraints are added:

$$
\begin{aligned}
\mathcal{F}_3 &= R_{FK}(i,1)^2 + R_{FK}(i,2)^2 + R_{FK}(i,3)^2 - 1 = 0 \\
\mathcal{F}_4 &= R_{FK}(:,1) \cdot R_{FK}(:,2) = 0 \\
\mathcal{F}_5 &= R_{FK}(:,2) \cdot R_{FK}(:,3) = 0 \\
\mathcal{F}_6 &= R_{FK}(:,1) \cdot R_{FK}(:,3) = 0
\end{aligned} \tag{18}
$$

where $R_{FK}(:,i)$ is the column number i of R_{FK}, to enforce the orthonormality of the rotational matrix.

In the summary of inverse kinematic algorithm in Table IV, the solving technique is labelled as *SolveQ*, which could be either the interval Newton or the consistency filtering plus solution existence check method, as described earlier in Section IV.

V. RESULTS AND DISCUSSION

An anthropomorphic arm with spherical wrist is used as an example. The diagram and frames assignment of the 6-DOF manipulator is shown in Fig. 1, and its Denavit–Hartenberg parameters are given in Table V. Two cases are presented, and in both cases, it is desired to verify the workspace and the topology of the manipulator, given a physical limit in its joint displacement. In all the examples given shortly, the joint displacements of all revolute joints were set at $\pm 30^\circ$. The first example (Section V-A) considers only a 3-DOF manipulator, using the first three joints of the manipulator shown in Fig. 1, addressing only the position of the end-effector, while the second (Section V-B) is a 6-DOF problem addressing the position and orientation of the end-effector. The third example (Section V-C) further explores the variations in the topology by testing various values of offset angle α_1 and α_2. In this example, a mixture

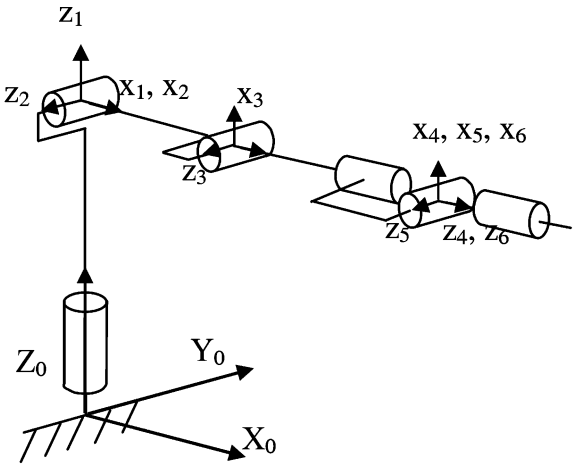


Fig. 1. Frame assignment of the robot used in the discussion.

TABLE V
DENAVIT–HARTENBERG PARAMETERS OF THE ROBOT IN FIG. 1 USED IN THE EXAMPLES

i	α_{i-1}	a_{i-1}	q_i	d_i
1	0	0	q_1	d_1
2	90^o	0	q_2	0
3	0	a_2	$q_3 + 90^o$	0
4	90^o	0	q_4	d_4
5	-90^o	0	q_5	0
6	90^o	0	q_6	0

TABLE VI
VALUES OF THE DENAVIT–HARTENBERG PARAMETERS IN EXAMPLE 2

d_1	[0.48, 0.52] m
a_2	[0.48, 0.52] m
d_4	[0.28, 0.32] m

between real and interval variables are utilized to demonstrate the cases where some design parameters are expressed in discrete real variables. The algorithm was used to validate possible topologies of the manipulators where the design constraints (i.e., the joint limit constraint) are satisfied within the desired workspace. The values for the Denavit–Hartenberg parameters used in the example are given in Table VI.

A. Example 1: Position Only

1) Workspace Definition: The algorithm summarized in Table I was implemented, with function $IK(\mathbf{X}_i, \mathbf{H})$ (Table IV) used as constraint evaluating function $S(\mathbf{X}_i, \mathbf{H})$ to obtain the workspace definition of the serial manipulator. The Denavit–Hartenberg parameters used were $d_1 = 0.5$, $a_2 = 0.5$ m, and $d_4 = 0.3$ m. Fig. 2 shows the workspace of the serial manipulator, evaluated within a boundary box of $([0.76, 0.80], [-0.01, 0.01], [0.59, 0.65])^T$. The inner solution of this workspace is certified as the region of workspace that is reachable by the manipulator (position only), given a joint displacement limits of $\pm 30^\circ$. Conversely, it is certified that the end-effector workspace represented by the outer boxes would not satisfy the given joint limit constraints. The gap between the inner and outer boxes are the boundary boxes. The loop termination thresholds were set at $\epsilon_x = 0.001$.

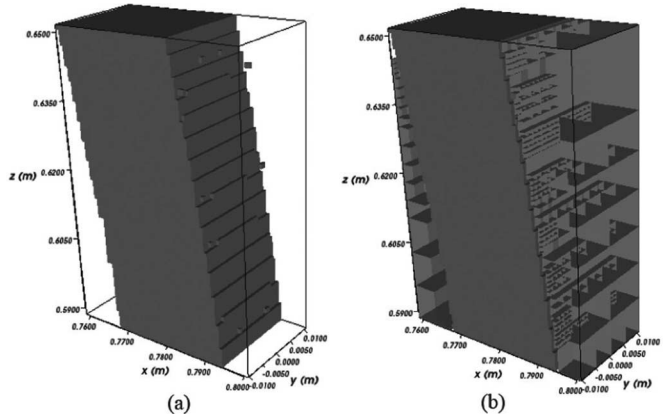


Fig. 2. Workspace definition of a 3-DOF serial anthropomorphic arm, based on joint limits constraints of $\pm 30^\circ$, evaluated within workspace $\mathbf{X} = ([0.76, 0.80], [-0.01, 0.01], [0.59, 0.65])^T$. The inner solution of the workspace is shown in (a), while the inner and outer solution (in slightly transparent display) is shown in (b).

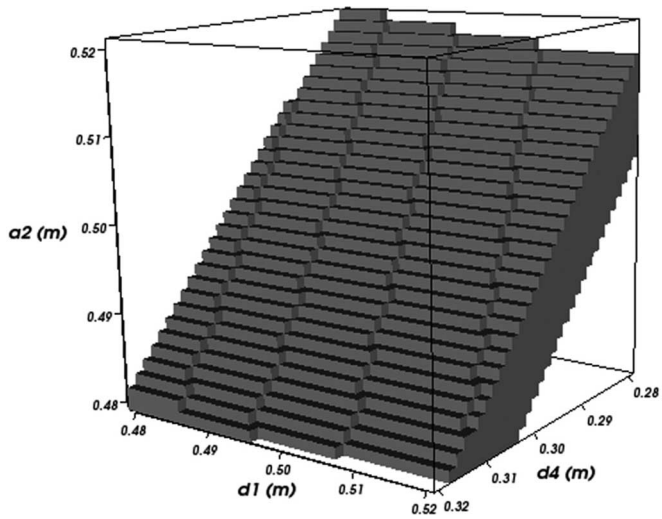


Fig. 3. Design parameters that guarantee that end-effector position $\mathbf{X} = ([0.77, 0.78], [-0.01, 0.01], [0.59, 0.60])^T$ (*note*: position only) is reachable by the manipulator within the joint limit of 30°.

2) Design Space Solutions: The design algorithm as summarized in Table II was implemented, utilizing the function $S^*(\mathbf{X}_i, \mathbf{H})$ (Table III). Function $IK(\mathbf{X}_i, \mathbf{H})$ (Table IV) was used as the constraint evaluating function $S(\mathbf{X}_i, \mathbf{H})$. The range of design parameters in this example is given in Table VI. The desired workspace where all constraints have to be satisfied is $\mathbf{X} = ([0.77, 0.78], [-0.01, 0.01], [0.59, 0.60])^T$. The loop termination thresholds are $\epsilon_x = 0.002$ and $\epsilon_d = 0.002$.

The results in the form of inner boxes to the constraints are shown in Fig. 3 The design parameters contained within the inner boxes guarantee that all the points within the desired workspace is reachable given a joint limit of 30°.

B. Example 2: Position and Orientation

The algorithm proposed allows the evaluation of both position and orientation workspace of a serial manipulator. For instance, if the constraint is defined as joint displacements that are no

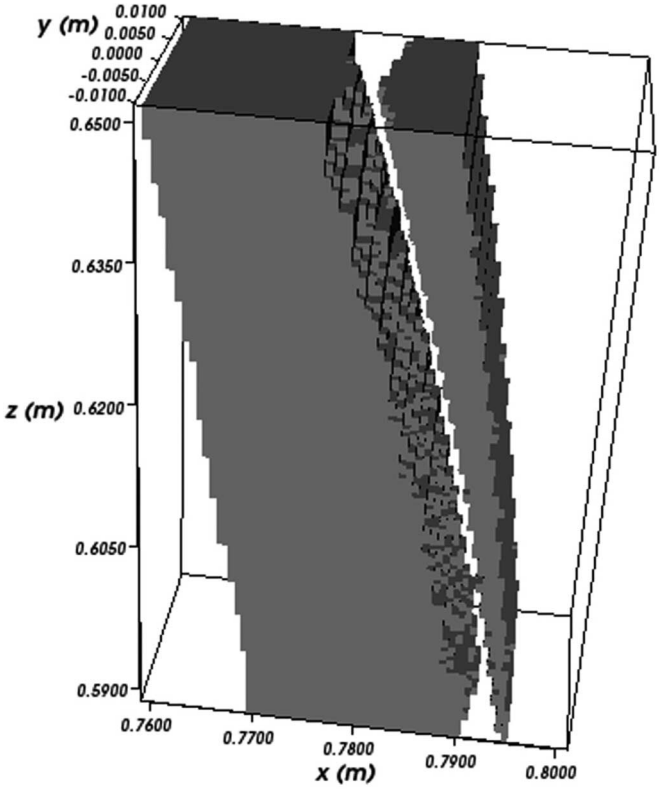


Fig. 4. Constant orientation workspace [as defined in (19)] of the serial manipulator, as defined in Table V, is evaluated as $\mathbf{X} = ([0.76, 0.80], [-0.01, 0.01], [0.59, 0.65])^T$.

larger than $\pm 30^\circ$, then it is possible to find the solutions in 6-D space of end-effector position and orientation that satisfy the constraint. However, this result will be difficult to present on paper.

In this paper, an example is presented for the case of constant orientation in a serial manipulator, while maintaining the joint limit constraint $\pm 30^\circ$. For the following results, the constant orientation required is selected to be

$$\mathbf{R}_E^0 = \begin{pmatrix} 0 & 0 & 1 \\ 0 & -1 & 0 \\ 1 & 0 & 0 \end{pmatrix} \tag{19}$$

where $\mathbf{R}_E^0$ represents the rotational matrix of the end-effector with respect to the ground frame. The Denavit–Hartenberg parameters are still as defined in Table V and the end-effector frame is attached to the sixth frame.

1) Workspace Definition: The result of constant orientation workspace can be represented easily in 3-D plots of the position of the end-effector, as the inner boxes are solutions where the inverse kinematics of end-effector position and orientation lie within the joint limits. For a constant orientation workspace, the end-effector orientation is kept constant throughout the *SolveQ* procedure.

The inner solution of the achievable constant orientation workspace is presented in Fig. 4, evaluating the constraint within the same workspace bounds as that shown in Fig. 2, i.e., $\mathbf{X} = ([0.76, 0.80], [-0.01, 0.01], [0.59, 0.65],)^T$. As can be seen from the results, the additional orientation constraint has created two clusters of inner solutions. This is because there are two sets of admissible solutions to the inverse kinematics of the end-effector position, corresponding to "elbow up" and "elbow down" configurations. The gap between the two clusters of solutions is caused by the joint displacement having to go beyond the $\pm 30^\circ$ limit while maintaining the constant orientation. Compare the results with Fig. 2(a) where no constant orientation requirement was imposed as a constraint. The loop termination threshold for the results in Fig. 4 was set at $\epsilon_x = 0.001$, the same as that in Section V-A1.

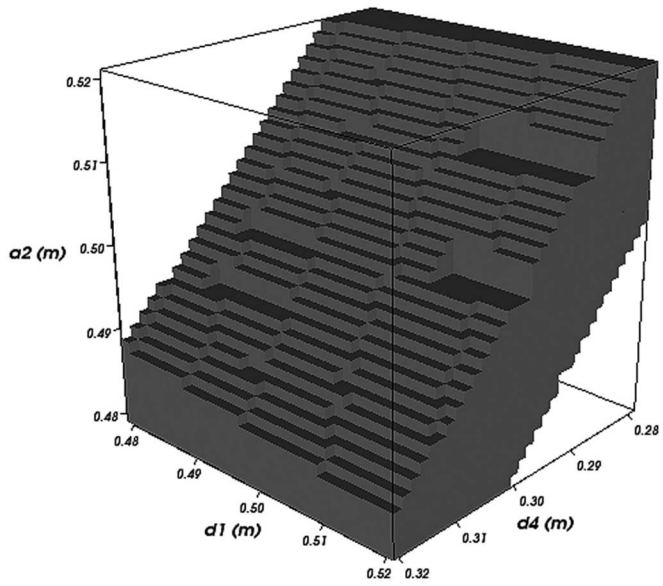


Fig. 5. Design parameters that guarantee constant orientation workspace $\mathbf{X} = ([0.773, 0.777], [-0.005, 0.005], [0.598, 0.602])^T$ is reachable within joint limit constraints of 30°. The constant orientation is defined in (19).

2) Design Space Solutions: For the design parameter space solution, the example given here was defined such that the inner boxes in the design space would guarantee that the resulting manipulator would be able to achieve the desired constant orientation workspace within desired workspace, given the constraint of joint limit ($\pm 30^\circ$).

Evaluating the problem with the desired workspace of $\mathbf{X} = ([0.77, 0.78], [-0.01, 0.01], [0.59, 0.60])^T$ as used by the example in Section V-A2 yields no inner boxes. It means that the algorithm cannot guarantee any set of design parameters to a manipulator that can achieve a constant orientation workspace of $\mathbf{X} = ([0.77, 0.78], [-0.01, 0.01], [0.59, 0.60])^T$. A smaller desired workspace was selected for this example, set at $\mathbf{X} = ([0.775 \pm 0.002], [0 \pm 0.005], [0.600 \pm 0.002])^T$. The design parameter parameters, certified to produce a manipulator capable of achieving all points within the desired constant orientation workspace, is given by Fig. 5. The loop termination thresholds are $\epsilon_x = 0.002$ and $\epsilon_d = 0.002$, the same as those in the position-only example in Section V-A2.

C. Example 3: Design Exercise With Combined Real Parameters

It should be noted, however, that in most design cases, some of the design parameters are often of discrete, rather than of

TABLE VII
TABULATED VARIATIONS OF OFFSET ANGLES α_1 AND α_2

		α_1				
		-90^o	-45^o	0	45^o	90^o
α_2	-90^o	X	X	X	X	X
	-45^o	X	X	X	X	X
	0	X	X	X	X	1
	45^o	X	X	X	1	X
	90^o	X	X	X	X	X

Configurations marked by "1" guarantee that the desired constant orientation workspace is an Inner box (reachable workspace). An "X" marks an outer box.

continuous values. For example, offset angles α_i in Denavit–Hartenberg parameters, are often in the values of 0, $\pm 90^\circ$, or sometimes $\pm 45^\circ$. In reconfigurable modular robots, offset lengths are in discrete multiples of the module lengths. Combining real parameters, instead of using the interval continuous range of values, produces tremendous improvements in the computational time of the algorithm.

If the desired workspace has been determined, and a set of values for design parameters have been given, then it is only necessary for the algorithm to evaluate if the entire desired workspace is an inner box or comprise only inner boxes. That often takes very few iterations. An algorithm can be constructed quickly to produce the combinations of all possible design parameters to be tested, either as sets of real variables, or as sets of very narrow intervals, reflecting discrete possible values of the design parameter, plus associated uncertainties.

In this example, we will investigate the possible configurations of the serial manipulator by running *for loops* on discrete values of α_1 and α_2 (Table V). The values of d_1, a_2, and d_4 are kept at $(0.5, 0.5, 0.3)$, respectively. In this exercise, it is desired to know what different configurations, obtained by varying the offset angles α_1 and α_2, would be able to yield the desired workspace, as defined in Section V-B2, namely, position $\mathbf{X} = ([0.775 \pm 0.002], [0 \pm 0.005], [0.600 \pm 0.002])^T$, with constant orientation defined in (19). The results are tabulated in Table VII. It is shown that aside from the original configuration where $(\alpha_1, \alpha_2) = (90^\circ, 0^\circ)$ that was evaluated in example Section V-B2, the desired constant orientation workspace is also achievable through $(\alpha_1, \alpha_2) = (45^\circ, 45^\circ)$.

D. Note on Computational Efficiency

The choice of implementation technique used in interval-based method plays a large role in determining the efficiency of the algorithm. Most simplistic approaches can yield the same results with much longer computational time, or even lower quality results. The quality of the result is determined by the sharpness of the definition of the inner and outer regions. A large region of boundary boxes does not provide any certification information; hence, it is desired to obtain a sharp result by minimizing the boundary region.

An efficient implementation would allow early determination on whether a box is an inner or an outer box, reducing the need for bisection. The termination threshold, at which the branch-and-bound loop is terminated, also determines the computational expense, as decreasing the threshold would increase computational time exponentially.

TABLE VIII
COMPUTATIONAL TIME AND NUMBER OF BOXES OBTAINED FOR EXAMPLES IN SECTION V

Example no.	Computation time (s)	Number of boxes		
		inner	outer	boundary
V-A1	264	2518	2218	17178
V-A2	2374	1507	1174	9615
V-B1	2442	5245	2508	12812
V-B2	1315	1796	383	3393
V-C	9	2	23	0

As the aim of this paper is to propose a concept to generate a certified serial chain topology and certified workspace against a set of constraints, the efficiency of implementation techniques is not emphasized in the content of this paper. Some examples of more advanced implementation of interval techniques can be found in [30] and [31]. In this paper, the algorithms were implemented in C++ with ALIAS library developed upon BIAS/Profil platform within the COPRIN project. As an indication of the computational resources required, the examples in Section V-A1–C were implemented on an Intel Dual-Core 2.4-GHz processor with 1-GB RAM, and the computational time is given in Table VIII. The termination thresholds were as given in the description of each example. Note that the last set of experiment was conducted by combining discrete values of possible design parameters (offset angles) and was required only to certify which topologies were able to satisfy the constraints, thus cutting the computational efforts.

VI. CONCLUSION

In this paper, a certified workspace evaluation algorithm and a serial-chain design algorithm were proposed and presented. In the process, an effective interval inverse kinematics algorithm for serial chain, without explicit inverse kinematic expressions, was also proposed and presented. These algorithms are shown to be effective in certifying that the various design constraints, given in the form of equality or inequality constraints, are satisfied. The proposed algorithm is also demonstrated to be effective in obtaining all variations of the kinematic topology of a serial manipulator such that the given constraints are satisfied in all points within the desired workspace. Further work is required to improve the efficiency of the algorithm in admitting or rejecting the various interval boxes, such as by exploring an efficient interval representation of serial kinematic transformations, hence reducing the necessity for further bisections in the process. As this study was carried out under European Union project ARES on endoluminal surgery through reconfigurable modular robots, future work also includes improving and adapting the proposed strategy to the challenging biomedical environment and requirements to compute the most suitable topology for the task.

REFERENCES

[1] R. Vijaykumar, K. Waldron, and M. Tsai, "Geometric optimization of serial chain manipulator structures for working volume and dexterity," *Int. J. Robot. Res.*, vol. 5, no. 2, pp. 91–103, 1986.

[2] E. Ottaviano, M. Husty, and M. Ceccarelli, "Identification of the workspace boundary of a general 3-r manipulator," *ASME J. Mech. Design*, vol. 128, no. 1, pp. 236–242, 2006.

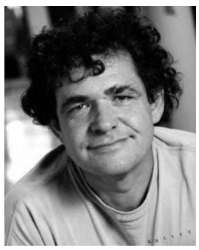
[3] J.-P. Merlet, C. Gosselin, and N. Mouly, "Workspaces of planar parallel manipulators," *Mechanism Mach. Theory*, vol. 33, no. 1/2, pp. 7–20, 1998.
[4] G. Pennock and D. Kassner, "The workspace of a general geometry planar three degree of freedom platform manipulator," *ASME J. Mech. Design*, vol. 115, no. 2, pp. 269–276, 1993.
[5] M. Gouttefarde and C. Gosselin, "Wrench-closure workspace of six-dof parallel mechanisms driven by 7 cables," *Trans. CSME*, vol. 29, no. 4, pp. 541–552, 2005.
[6] V. Kumar, "Characterization of workspaces of parallel manipulators," *ASME J. Mech. Design*, vol. 114, no. 3, pp. 368–375, 1992.
[7] F. Hao and J.-P. Merlet, "Multi-criteria optimal design of parallel manipulators based on interval analysis," *Mechanism Mach. Theory*, vol. 20, no. 2, pp. 151–171, 2005.
[8] Y. Papegay, J.-P. Merlet, and D. Daney, "Exact kinematic analysis of car's suspension mechanisms using symbolic computation and interval analysis," *Mechanism Mach. Theory*, vol. 40, pp. 395–413, 2005.
[9] J.-P. Merlet, "Solving the forward kinematics of a Gough-type parallel manipulator with interval analysis," *Int. J. Robot. Res.*, vol. 23, pp. 221–235, 2004.
[10] J. Porta, L. Ros, T. Creemers, and F. Thomas, "Box approximations of planar linkage configuration spaces," *ASME J. Mech. Design*, vol. 129, pp. 397–405, 2007.
[11] I. Bonev and C. Gosselin, "Analytical determination of the workspace of symmetrical spherical parallel mechanisms," *IEEE Trans. Robot.*, vol. 22, no. 5, pp. 1011–1017, Oct. 2006.
[12] O. Ma and J. Angeles, "Optimum design of manipulators under dynamic isotropy conditions," in *Proc. IEEE Conf. Robot. Autom.*, 1993, vol. 1, pp. 470–475.
[13] R. Mayorga, J. Carrera, and M. Oritz, "A kinematics performance index based on the rate of change of a standard isotropy condition for robot design optimization," *Robot. Auton. Syst.*, vol. 53, no. 3–4, pp. 153–163, Dec. 1992.
[14] A. Bowling and O. Khatib, "Design of non-redundant manipulators for optimal dynamic performance," in *Proc. Int. Conf. Adv. Robot.*, Monterey, CA, Jul. 7–9, 1997, pp. 865–872.
[15] M. Yim, W. Shen, B. Salemi, D. Rus, M. Moll, H. Lipson, E. Klavins, and G. Chirikjian, "Modular self-reconfigurable robot systems [Grand Challenges of Robotics]," *IEEE Robot. Autom. Mag.*, vol. 14, no. 1, pp. 865–872, Mar. 2007.
[16] F. Masahiro, H. Kitano, and K. Kageyama, "Reconfigurable robot platform," *Robot. Auton. Syst.*, vol. 29, no. 2, pp. 119–132, Nov. 1999.
[17] M. Yim, D. Duff, and K. Roufas, "Polybot: A modular reconfigurable robot," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2000, vol. 1, pp. 514–520.
[18] Y. Fei and X. Zhao, "Design and dock analysis for the interactive module of a lattice-based self-reconfigurable robot," *Robot. Auton. Syst.*, vol. 55, no. 2, pp. 87–95, Feb. 2007.
[19] W. Shen and P. Will, "Docking in self-reconfigurable robots," in *Proc. IEEE Int. Conf. Intell. Robots Syst.*, 2001, vol. 2, pp. 1049–1054.
[20] K. Stoy, W. Shen, and P. Will, "A simple approach to the control of locomotion in self-reconfigurable robots," *Robot. Auton. Syst.*, vol. 44, no. 3–4, pp. 191–199, Mar. 2003.
[21] R. Moore, *Interval Analysis*. Englewood Cliffs, NJ: Prentice–Hall, 1966.
[22] E. Hansen and G. Walster, *Global Optimization Using Interval Analysis*, 2nd ed. New York: Marcel Dekker, 2004.
[23] R. B. Kearfott, "Corrigenda: Some tests of generalized bisection," *ACM Trans. Math. Softw.*, vol. 14, no. 4, pp. 399–399, 1988.
[24] R. B. Kearfott and M. Novoa III, "Algorithm 681: INTBIS, a portable interval Newton/bisection package," *ACM Trans. Math. Softw.*, vol. 16, no. 2, pp. 152–157, Jun. 1990.
[25] F. Benhamou, F. Goualard, and L. Granvilliers, "Revising hull and box consistency," in *Proc. Int. Conf. Logic Program.*, Las Cruces, NM, 1999, pp. 230–244.
[26] M. Collavizza, F. Delobe, and M. Rueher, "Comparing partial consistencies," *Reliable Comput.*, vol. 5, pp. 1–16, 1999.
[27] O. Lhomme, "Consistency techniques for numeric CSPs," in *Proc. Int. Joint Conf. Artif. Intell. (IJCAI)*, Chambery, France, Aug. 1993, pp. 232–238.
[28] A. Neumaier, *Interval Methods for Systems of Equations.*. Cambridge, U.K.: Cambridge Univ. Press, 1990.
[29] G. Alefeld, "On the convergence of some interval-arithmetic modifications of Newton's method," *SIAM J. Numer. Anal.*, vol. 21, no. 2, pp. 363–372, 1984.
[30] J.-P. Merlet and P. Donelan, "On the regularity of the inverse jacobian of parallel robot," in *Proc. Int. Symp. Adv. Robot Kinematics (ARK)*, Ljubljana, Slovenia, Jun. 2006, pp. 41–48.
[31] J. Rohn, "Cheap and tight bounds: The recent result by E.Hansen can be made more efficient," *Interval Comput.*, vol. 4, pp. 13–21, 1993.

Denny Oetomo (M'04) received the B.Eng. degree (with first-class honors) from the Australian National University, Acton, A.C.T., Australia, in 1997 and the Ph.D. degree in mechanical engineering from the National University of Singapore, Singapore, in 2004.

From 1998 to 1999, he was a Photolithography Engineer with Hewlett-Packard Singapore. From 2002 to 2004, he was a Research Engineer at the Singapore Institute of Manufacturing Technology. From 2004 to 2007, he was a Postdoctoral Fellow at Monash University, Melbourne, Vic., Australia, and also with the Institut National de Recherche en Informatique et en Automatique (INRIA) Sophia-Antipolis, France. He is currently a Lecturer with the University of Melbourne. His current research interests include robotics, specifically the kinematic and dynamic analysis of manipulators, mobile manipulation, unified force/motion control, and computational kinematics. He is also involved in the application of robotics in biomechanics.

David Daney received the Ph.D. degree in robotics from the University of Nice Sophia-Antipolis, Nice, France, in 2000.

During 2000, he was a Postdoctoral Associate with Constructions Mecaniques Des Vosges (CMW) France, where he studied a parallel kinematic machining (PKM) milling machine. During 2001, he was a Postdoctoral Associate with the LORIA Laboratory, Nancy, France, where he worked on computer arithmetic. In 2002, he was invited to attend Mc Gill University, Rutgers University, and Laval University. Since 2003, he has been an Research Scientist with the Constraints Solving, Optimization, Robust Interval Analysis (COPRIN) Project, Institut National de Recherche en Informatique et en Automatique (INRIA) Sophia-Antipolis. His current research interests include the area of robotics, analysis, design, and calibration of parallel robots. He is also involved in interval analysis and its applications in robotics, in particular for wire-driven robots performance analysis.



Jean-Pierre Merlet (M'01) received the M.S. degree in mathematics from the University of Nantes, Nantes, France, in 1978, the Engineer Title from the National Superior School of Mechanics, Nantes, in 1980, the Ph.D. degree from Paris VI University, Paris, France, in 1986, and the Research Habilitation degree from Nice University, Nice, France, in 1993.

He was an Engineer in the food industry (Lu) and in civil engineering. He was also a Research Engineer with the Commissariat à l' Energie Atomique (CEA) (French Nuclear Agency) and a Research Associate with Kyoto University, Kyoto, Japan, Mechanical Engineering Laboratory (MEL), Tsukuba, Japan, and McGill University, Montreal, QC, Canada. He is currently a Scientific Leader with the Constraints Solving, Optimization, Robust Interval Analysis (COPRIN) Project, Institut National de Recherche en Informatique et en Automatique (INRIA) Sophia-Antipolis, Nice (French National Research Institute in control theory and computer science). He has authored or coauthored over 200 conference and journal papers in the field of force control of robots, algebraic geometry, constraint solving, and parallel robots, and a book on the latter subject with two editions in French and two editions in English (Springer-Verlag, 2001 and 2005). He holds a patent on a parallel robot that is currently licensed to a U.S. company (Gerber) and a Japanese company (Hephaist Seiko). His current research interest includes interval analysis, optimal design for mechanisms, nanotechnology, and medical robotics. He is an Associate Editor of *Mechanism and Machine Theory.*

Dr. Merlet is the Chairman of the French Section of the International Federation for the Theory of Machines and Mechanisms (IFToMM).