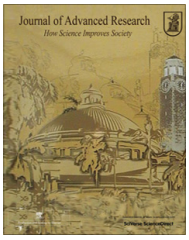
Journal of Advanced Research (2015) 6, 479–492

Cairo University

Journal of Advanced Research



ORIGINAL ARTICLE

Task based synthesis of serial manipulators

Sarosh Patel *, **Tarek Sobh**

Robotics, Intelligent Sensing and Control (RISC) Laboratory, School of Engineering, University of Bridgeport, 221 University Avenue, Bridgeport, CT 06604, USA

ARTICLE INFO

Article history:
Received 15 August 2014
Received in revised form 5 December 2014
Accepted 15 December 2014
Available online 3 January 2015

Keywords:
Global optimization
Manipulator synthesis
Simulated annealing
Task-based design

ABSTRACT

Computing the optimal geometric structure of manipulators is one of the most intricate problems in contemporary robot kinematics. Robotic manipulators are designed and built to perform certain predetermined tasks. There is a very close relationship between the structure of the manipulator and its kinematic performance. It is therefore important to incorporate such task requirements during the design and synthesis of the robotic manipulators. Such task requirements and performance constraints can be specified in terms of the required end-effector positions, orientations and velocities along the task trajectory. In this work, we present a comprehensive method to develop the optimal geometric structure (DH parameters) of a non-redundant six degree of freedom serial manipulator from task descriptions. In this work we define, develop and test a methodology to design optimal manipulator configurations based on task descriptions. This methodology is devised to investigate all possible manipulator configurations that can satisfy the task performance requirements under imposed joint constraints. Out of all the possible structures, the structures that can reach all the task points with the required orientations are selected. Next, these candidate structures are tested to see whether they can attain end-effector velocities in arbitrary directions within the user defined joint constraints, so that they can deliver the best kinematic performance. Additionally least power consuming configurations are also identified.

Introduction

The rapid growth in manufacturing technologies has increased the need for design and development of optimal machinery. No longer is the emphasis on machinery that works, but on machinery that works faster, consumes less power, and is more functional. The availability of cheap and easy computing power allows us to design and evaluate multiple structures based on user defined criteria and select the best design before physically constructing the manipulator. In this work we propose a method for designing optimal robotic manipulator structures.

What is the best manipulator configuration for soldering electronic components? What should be the ideal manipulator structure for a painting job? What is the optimal manipulator configuration for a material handling job? Robotic researchers over the years have tried to find answers to these questions.

* Corresponding author. Tel.: +1 (203) 583 9321.
E-mail addresses: saroshp@bridgeport.edu (S. Patel), sobh@bridgeport.edu (T. Sobh).

Peer review under responsibility of Cairo University.

ELSEVIER Production and hosting by Elsevier

http://dx.doi.org/10.1016/j.jare.2014.12.006

But in this case plenty is the problem; there is no unique solution or definite answer to these questions. Instead, in most cases there can be infinite answers to all of the above questions. Equations describing the kinematic behavior of serial manipulators are highly nonlinear with no closed solutions. The difficulty in most cases lies not in finding a solution, but finding the 'best' solution out of the numerous possible solutions, or in other words, an optimal solution.

There is a very close relationship between the structure of the manipulator and its kinematic performance. Robotic researchers have over the years tried to develop a framework to reverse engineer optimal manipulator geometries based on task requirements. Every robotic manipulator can only perform a certain set of tasks, some more efficiently than others. Deciding the best manipulator structure for a required job at the design stage is done mainly on the basis of experience and intuition. The rigorous analysis of a few widely used manipulator structures and a collection of a few ad-hoc analytical tools can be of some help. However, a comprehensive framework to design manipulator structures from task descriptions that can guarantee optimal task performance under a set of operating constraints is still lacking.

The aim of this work was to develop a task directed design methodology that can serve as a simple and easy tool for kinematic synthesis of robotic manipulators based on task descriptions. The proposed methodology allows a user to enter the task point descriptions and joint constraints, and generates the optimal manipulator structure for the specific task.

Existing approaches

The research area of robotic manipulator design can be broadly classified into general purpose designs and task specific designs. Even though general purpose manipulators are commonplace, they do not guarantee optimal task execution. Because industrial robotic manipulators perform a set of given tasks repeatedly, task-specific or task-optimized manipulator designs are preferred for industrial applications.

The existing approaches for design and synthesis of serial manipulators can be broadly classified into the following three types:

Geometric approach

Serial robotic manipulators are open-loop kinematic chains consisting of interconnected joints and links. There is a great body of research dealing with the mobility issues of closed loop kinematic chains. The principles of closed loop mechanical chains can be applied to design highly dexterous serial manipulators by assuming the distance between the base of the manipulator and the task point as a fixed and imaginary link in the closed mechanical chain.

Grashof [1] proposed a simple rule to judge the mobility of links in four-link closed kinematic chains. This rule was further extended and developed into Grashof's criterion by Paul [2]. Robotic researchers have applied Grashof's criterion to design manipulators with high dexterity at the given task points. Where dexterity refers to the ability of the manipulator to attain any orientation about a given point [3]. Li and Dai [4] and Patel and Sobh [5], proposed a method for the optimal design of three-link planar manipulators using Grashof's criterion. In their work Patel and Sobh [5] propose a simple algorithm for the optimal design of three link planar manipulators with full dexterity at the given task region or trajectory. The Grashof's criterion has also been extended by researchers to explain the behavior of longer kinematic chains. Ting introduced the five-link Grashof criterion [6] and later extended it to N-link chains [7,8]. The main advantage of this method is its independence from the necessity to calculate the inverse kinematic solutions to judge its performance.

Parametric optimization approach

Parametric optimization is a classical way of solving an optimization problem. One or more criterion that quantify the performance properties of the manipulator, and sometimes with associated weighing factors, are maximized or minimized to arrive at an optimal manipulator structure. Parametric optimization has been one of the widely adopted approaches for the synthesis of serial manipulators. Condition number was used by Angeles and Rojas [9] to obtain optimal dimensions for a three-DoF manipulator and three-DoF spherical wrist. Craig and Salisbury [10] used the condition number of the Jacobian as design criterion to optimize the dimensions of the fingers of the Stanford articulated hand.

Sobh and Toundykov [11] present a method for the optimal kinematic synthesis of the manipulator structure based on the Yoshikawa manipulability ellipsoid at a given set of task points. An objective cost function incorporating the Yoshikawa manipulability index was optimized using the *steepest-descent* algorithm over the manipulator's task trajectory to derive the optimal geometric structure. This work was implemented as a procedural package in Mathematica®[1] (version 4.1) and used the Robotica[2] version 3.60 (a robotics toolkit for Mathematica®). This work was further extended by Sobh et al. [12,13] to simulate the dynamic behavior of such an optimized manipulator.

Kucuk and Bingul [14,15], implement a multi-variable optimization. The manipulator workspace was optimized based on a combination of local and global performance indices: Structural length index, manipulability measure, condition number, and global conditioning index.

These parametric optimization methods are task independent and hence do not guarantee the non-existence of a better manipulator for a specific task [16]. Another limitation of this approach is that it has a very limited scope due to the inherent limitations and general shortcomings of the performance metrics. A comprehensive survey of manipulator performance parameters and their limitations can be found in this reference [17].

Task-based design approach

Task-based design of manipulators uses the prior knowledge of application of the manipulator to design the best possible structure that can guarantee task completion. Task specifications can either be kinematic or dynamic. The ultimate goal of task-based design model is to be able to generate both the manipulator kinematic and dynamic parameters, using task

[1] [© 2002] Wolfram Research Inc.
[2] [© 1993] Board of Trustees, University of Illinois.

descriptions and operating constraints [18]. Task-based design approach has seen considerable interest from researchers dealing with re-configurable modular manipulator systems (RMMS) that can be easily re-configured depending on the task at hand. A task-based approach for deciding the optimal configuration for metamorphic self-reconfigurable manipulators was proposed by Valsamos et al. [19].

Paredis and Kholsa [16], use the task requirements to find the optimal structure of a manipulator. They developed a numerical approach for determining the optimal structure of a six degree of freedom non-redundant manipulator. Their proposed method involves generating the DH parameters by minimizing an objective function using numerical optimization. This method does not check for non-singular positions at task points and the ability of the manipulator to generate effective velocities.

Al-Dios et al. [20], developed a method for optimizing the link lengths, masses and trajectory parameters of a serial manipulator with known DH table using direct non-gradient search optimization. This work was focused to optimize the task time and joint torques for a specific manipulator task.

Kholsa et al. [18,21], proposed the concept of Progressive Design as a frame work for the general design of manipulators and reconfigurable modulator manipulator systems, using task descriptions. The framework consists of three modules: kinematic design, planning and kinematic control. The kinematic design module encapsulates the task specifications, manipulator specifications and dexterity measure. Kholsa et al. [18], [21], applied the framework to develop an optimal manipulator for space shuttle tile changing operation, using dexterity as the optimizing criterion.

Dash et al. [22], proposed a two stage methodology for structure and parameter optimization of reconfigurable parallel manipulator systems. They proposed a *'TaskToRobot Map'* database that maps task description to a suitable manipulator configuration depending on the degrees of freedom required for a given task.

The manipulator configuration search space is prohibitively large, even if unacceptable solutions are eliminated early in the evaluation process. Two of the most applied approaches to search the Configuration Space are Random Line search and Genetic algorithms. The use of Genetic Algorithms (GA) for designing the structure of self-organizing and modular robotic systems was recommended by Izumi et al. [23], Chung et al. [24] and Kim et al. [25]. Shiakolas et al. [26] use evolutionary optimization approaches to optimize the design of a SCARA manipulator.

Need for a comprehensive task based design methodology

Geometric optimization is limited to special cases, where certain criteria such as the Grashof's criterion can be applied to design manipulators. This methodology cannot be generalized or extended to design manipulators with prismatic links as an example. Also, this method does not allow the user to input multiple task requirements hence task satisfaction cannot be guaranteed using this method.

The major drawback of parametric optimization of manipulators lies in the limited scope of the parameters themselves. While it might be useful to fine tune existing manipulators configurations to improve their specific parametric performance, for example generating isometric manipulators by adjusting their link lengths, this approach has not evolved into a design methodology for manipulators because it only improves upon an existing structure and does not generate new configurations.

Another limitation of the above two methods is that existing methods do not consider practical design issues such as limited joint freedoms or constrained joint limits. Also, most methods avoid dealing with prismatic joints, especially the parametric optimization methods.

Since manipulators are expected to do certain tasks repeatedly it is essential that the task requirements are incorporated with in the design process, so that satisfactory task performance is guaranteed. Task based design is a promising avenue for developing a comprehensive manipulator design methodology. Firstly, because it is by definition based on task requirements, and secondly because, multiple criteria can be specified by the users.

The major limitation of existing task-based methodology is that they are limited to the design of manipulators composed of only revolute links. In this work, we define the necessary and sufficient conditions for a holistic task-based methodology. Next, we define a function to judge the reachability of a manipulator configuration to the task point(s). This methodology can generated manipulators composed of both revolute as well as prismatic joints.

Another contribution of this work is that operating constraints can be specified with the task descriptions. The configurations generated by this methodology guarantee task satisfaction under the constraints. We test this methodology for real-life robotic applications under joint constraints. Such a framework should also have the capability to optimize existing general configurations based on a specific task and constraints.

Problem statement

The task descriptions can be given in terms of the task points p that the manipulator is supposed to reach with a specified orientation. Let P be the set of m task points that define the manipulator's performance requirements.

$$P = \{p_1, p_2, \ldots, p_m\} \in TS \tag{1}$$

All these points belong to the six-dimensional Task Space (TS) that defines both the position and orientation of the manipulator's end-effector. Each point in the Task Space (TS) can be given as:

$$p_i = \{x, y, z, \varphi, \theta, \psi\} \quad \forall i = 1, 2, \ldots, m \in TS \tag{2}$$

Fig. 1 shows an example of a manipulator doing multiple tasks that require specific positioning and orientation of the manipulator at different points in the workspace.

In this work, we use the standard DH (Denavit–Hartenberg) notation to represent the manipulator structures [27]. The standard DH notation uses four parameters to define each link in the serial manipulator. In the case of a revolute link the design parameters are $\{a, \alpha, d\}$, and in the case of a prismatic link the design parameters are $\{a, \alpha, \theta\}$. A n degree serial manipulator configuration set (DH) can be given as:

$$\mathrm{DH} = \{a_0, \alpha_0, \theta_0 \text{ or } d_0, a_1, \alpha_1, \theta_1 \text{ or } d_1, \ldots, a_{n-1}, \alpha_{n-1}, \theta_{n-1} \text{ or } d_{n-1}\} \tag{3}$$

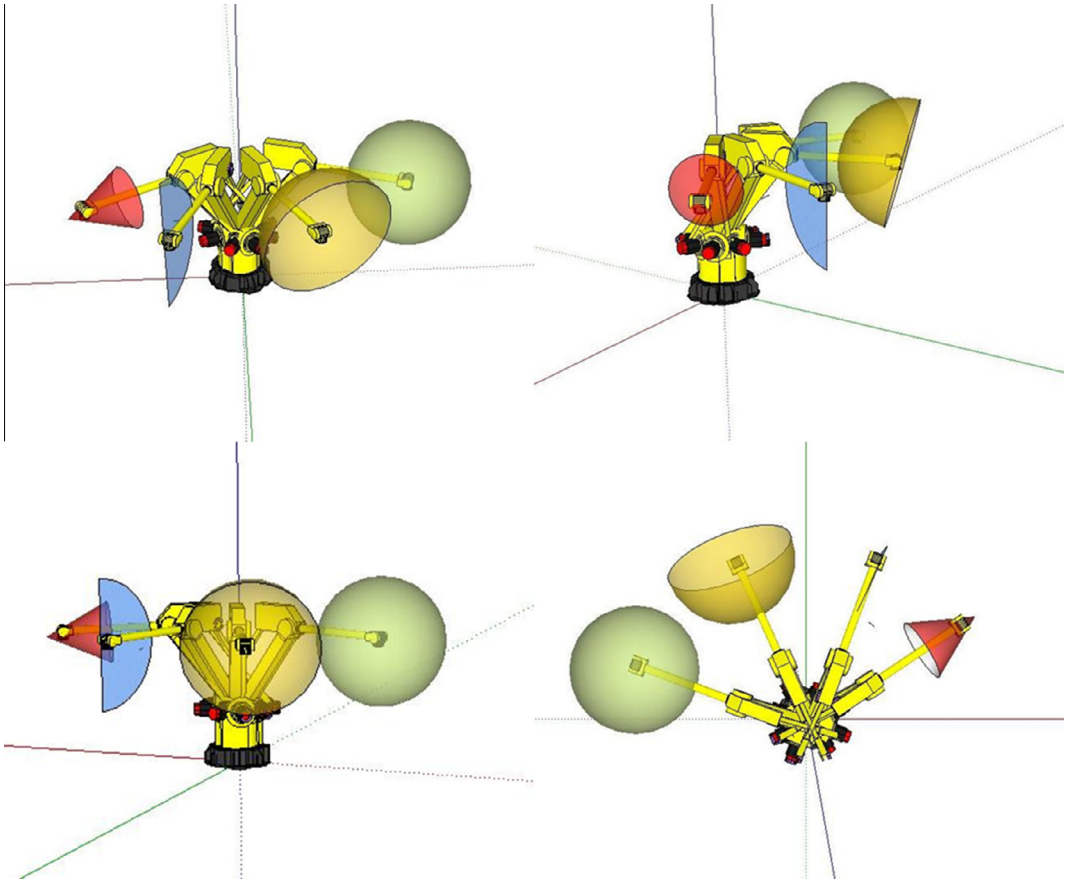


Fig. 1 Manipulator with different orientations at a set of task points.

Therefore, a n – link serial manipulator will have $3n$ design parameters. Every set of manipulator configuration parameters can be said to be a point in the Configuration Space (C). Each set of values of the DH vector represents a unique manipulator configuration and a distinct point in the $3n$ dimensional Configuration Space (C).

$$\mathrm{DH} = \{a_0, \alpha_0, \theta_0 \text{ or } d_0, a_1, \alpha_1, \theta_1 \text{ or } d_1, \ldots, a_{n-1}, \alpha_{n-1}, \theta_{n-1} \text{ or } d_{n-1}\} \in C \tag{4}$$

Similarly, for a n degree of freedom manipulator, the joint vector q can be said to be a point in the n dimensional Joint Space (Q), such that:

$$q = [q_1, q_2, \ldots, q_n] \in Q \tag{5}$$

Each joint vector q represents unique manipulator posture and a distinct point in the n dimensional Joint Space (Q). The natural Joint Space assumes there are no joint limitations (fully revolute ideal joints). But in practice the joints are not fully revolute and are bounded by lower and upper bounds. The values of the joint angles are range bound by user defined joint limits (upper and lower bounds). Hence, we define Q_c as the Constrained Joint Space, such that the joint displacements always satisfy the constraints:

$$q_{i,\min} \leqslant q_i \leqslant q_{i,\max} (q_i \in Q_c) \text{ and } Q_c \subset Q \tag{6}$$

Similarly, the manipulator's Reachable Workspace (WS) is defined as the set of points in the world coordinate system that the manipulator's end-effector can reach when no joint constraints are imposed. The manipulator's forward kinematic equations form a mapping $f(C)$:$Q \rightarrow$ WS between these three spaces: the Configuration Space (C), the Joint Space (Q) and the Workspace (WS).

When the manipulator's joint motion is restricted between joint limits the manipulator can only reach a part of the Reachable Workspace, known as the Constrained Reachable Workspace (CWS), such that CWS $\subset$ WS. Constrained Reachable Workspace is defined as the set of points in the real coordinate system that the manipulator's end-effector can reach when joint constraints are imposed. This is given by the forward mapping: $f(C)$:$Q_c \rightarrow$ CWS and $Q_c \subset Q$.

Fig. 2 shows the Reachable Workspace (WS) and Constrained Reachable Workspace (CWS) for a simple planar two-link manipulator as an illustrative example.

When a given manipulator with configuration set DH, with joint vector q can reach a specific task point p, the mapping can be represented as:

$$f(\mathrm{DH}, q) = p \tag{7}$$

Therefore, the problem can be stated as follows: Find a solution set DH in the $3n$ dimensional Configuration Space such that there exists at least one q in the Constrained Joint Space that can reach the required position and orientation of the end-effector. i.e.,

$$\text{Find all DH such that } \forall\, p \in TS; \exists q \in Q_c | f(DH, q) = p$$

Even though this might seem to be a necessary and sufficient condition required for designing a manipulator, simulations and experience will suggest that this solution set might

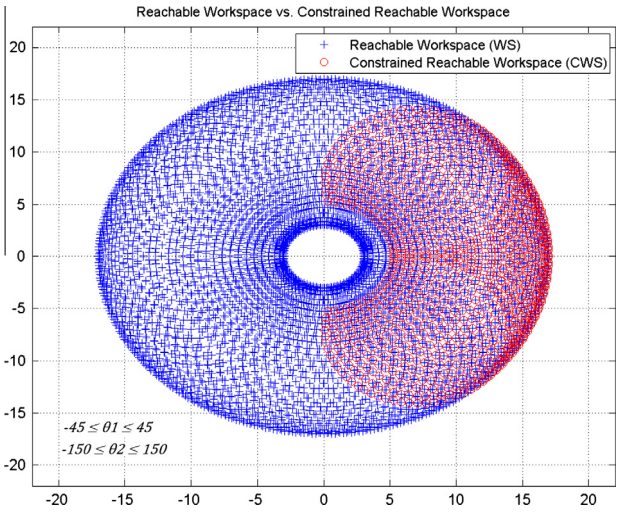


Fig. 2 Reachable Workspace (WS) compared with Constrained Reachable Workspace (CWS).

include a few manipulators that are able to reach the one or more of the task points only in singular positions. Such manipulators, if constructed, will not be able to attain good end-effector velocities in one or more directions due to their singular postures at the task point(s). Such manipulators will have very limited mobility at the required task point(s). Infinite forces have to be applied in order to generate motion along one or more directions at singularities. Therefore such manipulator configurations should be removed from the solution set. The test for singularity is the determinant of the Jacobian matrix, which for a square Jacobian also happens to be the Yoshikawa manipulability index [28].

The Jacobian mapping from joint velocities to end-effector velocities for a manipulator is given as:

$$\xi = J(\mathrm{DH},q)\dot{q} \tag{8}$$

where $\xi = [\dot{x}\,\dot{y}\,\dot{z}\,\dot{\phi}\,\dot{\theta}\,\dot{\psi}]$ is the end-effector velocity vector.

The Jacobian matrix is posture dependent matrix. It is also important to evaluate the Jacobian of the manipulator because the Jacobian matrix maps joint velocities to end-effector velocities, according to the mapping $J(\mathrm{DH},q) : \dot{q} \rightarrow \xi$. Hence, it is important to check whether the Jacobian of manipulator at a given task point is well conditioned, and not in a singular posture. A manipulator with well-conditioned Jacobian at the task point(s) will easily be able to transform joint velocities into end-effector velocities in any required direction, however the opposite cannot be said to be true on the basis of just the Jacobian determinant.

$$\xi = J_1\dot{q}_1 + J_2\dot{q}_2 + \cdots + J_n\dot{q}_n \tag{9}$$

Therefore, we modify the problem statement as follows:

$$\text{Find all DH such that } \forall p \in TS; \exists q \in Q_c | f(DH,q) = p \text{ and } \det(J(q)) \neq 0$$

Solution methodology

In this section we define two functions for evaluating the reachability and kinematic performance of the manipulator. To solve the problem we make the following assumptions:

1. The robot base is fixed and located at the origin O.
2. The task points are specified with respect to the manipulator's base frame.
3. The joint limitations are known to the designer.
4. If a joint is prismatic, the joint angle (θ) can assume values in the interval [−180, 180].
5. If a joint is revolute, the joint twist angle (α) can assume values [−180, 180].
6. The last three axes of the six degree of freedom manipulator intersect at a point to form a spherical wrist.
7. To limit the number of inverse kinematic solutions only non-redundant configurations are considered.

Let the task points be represented as $p = [x, y, z, \phi, \theta, \psi]$. The position of the operating point (OP) on the end-effector is given by $p_P = [x, y, z]$ and its orientation by $p_0 = [\phi, \theta, \psi]$.

$$p_i = [p_P p_0] \in TS \quad \forall i = 1, 2, 3 \ldots, m \tag{10}$$

In cases where multiple orientations are required at the same point the vector p_P remains same while the orientation vector p_O will assume different values.

The first criterion that needs to be satisfied is that all the points in the Task Space should be a part of the manipulator's Constrained Reachable Workspace. We define the Constrained Reachable Workspace as the set of points that the manipulator is able to reach under constrained joint limitations Q_c, while the normal Reachable Workspace (WS) is the set of points that the manipulator can reach with no joint limits, such that CWS ⊂ WS. Hence, given a set of task points P, the first objective is to find all possible manipulator configurations such that all task points in P are a part of the manipulator's Constrained Reachable Workspace (CWS).

$$\text{Find all DH such that } \forall p \in TS; p \in \mathrm{CWS}$$

The Constrained Reachable Workspace (CWS) of the manipulator is given by the forward kinematic mapping $f(C):Q_c \rightarrow \mathrm{CWS}$. With the help of the standard DH notation parameters, the forward kinematic relationship is given as:

$$f(\mathrm{DH},q) = p \tag{11}$$

Due to the highly non-linear nature of the kinematic equations describing this forward kinematic mapping from the Joint Space to the Task Space, multiple manipulator postures or points in the Joint Space can lead to the same point in the Task Space. In such cases, point(s) in the Task Space will have more than one inverse kinematic solution.

$$q = f^{-1}(\mathrm{DH},p) \tag{12}$$

The inverse kinematic equations often have no unique solutions. Depending on the manipulator's structure (DH) and location of the task points (p), the number of solutions might range from zero to infinite. And, even in the case where there are multiple known solutions to the above equations, it is still possible that none of them lie within the Constrained Joint Space (Q_c).

$$q = f^{-1}(\mathrm{DH},p) | q \in Q_c \tag{13}$$

In this work we use Particle Swarm Optimization based inverse kinematic approach for finding the inverse kinematic solutions within the constrained joint space. This numerical approach finds all possible inverse kinematic solutions within the specified joint constraints.

To determine whether the structure manipulator is able to reach a given task point with required orientation we construct a reachability function. The reachability function determines whether the manipulator can reach and orient the end-effector at the task point within the set joint limitations.

$$\text{reachability (DH)} = \max\left[\min\left(\frac{(q_{i,\max}-q_i)(q_i-q_{i,\min})}{(0.5(q_{i,\max}-q_{i,\min}))^2}\right)_{i=1}^{n}\right]_{j=1}^{g} \tag{14}$$

where g is the number of inverse kinematic solutions.

When the joint angle displacements required to reach a task point are within the joint constraints the reachability function is bounded between zero and unity. And, if the manipulator reaches the task point with at least one joint angle at it maximum displacement, the reachability function will have a value of zero. The reachability function will have a maximum value of unity if the manipulator reaches the task point with all joint displacement being mid-range of their joint limits. A reachability value of unity is the ideal case and is only possible with a single task point. If one of the bounds is violated by any given joint out of the n manipulator joints the function will have a negative value. The reachability function value for different locations of the task point is shown in Table 1.

Since we take a minimum of all the n joints, the reachability indicates the worst joint performance. This reachability function can help in the design of optimal manipulator structures by checking if they can reach the task point with proper joint displacements. To find the best reachable configurations the reachability function needs to be maximized.

Next, to select the best manipulator out of this set of manipulator configurations based their kinematic performance and manipulability we write an objective function that can be maximized or minimized to obtain the optimal manipulator configuration.

$$f(\text{DH})_{\text{velocity}} = \max\left[\det(J(q^1)), \det(J(q^2)), \ldots, \det(J(q^g))\right] \tag{15}$$

where g is the number of inverse kinematic solutions.

This objective function should be maximized to find the optimal manipulator structure that has the best conditioned Jacobian at the task points. Such a manipulator will be able to easily transform joint velocities into needed end-effector velocities.

We extend the above formulation for reachability and kinematic performance to include all m points that define the Task Space, as a summation of the function values at the individual task points.

$$\text{reachability (DH)} = \sum_{\forall p \in TS}\left(\max\left[\min\left(\frac{(q_{i,\max}-q_i)(q_i-q_{i,\min})}{(0.5(q_{i,\max}-q_{i,\min}))^2}\right)_{i=1}^{n}\right]_{j=1}^{g}\right) \tag{16}$$

$$f(\text{DH})_{\text{velocity}} = \sum_{\forall p \in TS}\left(\max\left[\det(J(q^1)), \det(J(q^2)), \ldots, \det(J(q^g))\right]\right) \tag{17}$$

Manipulator power or torque requirements depend on not only the structure of the manipulator but also on the task trajectory. Industrial manipulators repeatedly traverse a trajectory of set task points. Hence the operating cost of the manipulator over a long period of time greatly depends on torque requirements and proper design of the trajectory. Although the path/trajectory planning is outside the focus of the this work, we address the problem by identifying manipulator postures such that the torque required is minimum while moving from one task point to another.

Lower joints typically have more powerful motors and high torque requirements as any displacement about the lower joint moves the entire structure or the set of manipulator links that follow. For example any motion on the first joint of the manipulator will certainly move the complete arm as opposed to a wrist joint that will only turn the end-effector or the gripper. Any displacement of joint displaces not only that link but also all successive links are moved irrespective of their joints being displaced or not. We use the structural length index for each of the joints to estimate the power requirement of each of the joints of the manipulator.

The structural length index for the first three joints is given as the sum of the link lengths and link offsets.

$$S_1 = \sum_{i=1}^{n}(a_i + d_i) \tag{18}$$

$$S_2 = \sum_{i=2}^{n}(a_i + d_i) \tag{19}$$

$$S_3 = \sum_{i=2}^{n}(a_i + d_i) \tag{20}$$

In the case of a spherical wrist where all three axes coincide the structural index for the last three joints are equal, because any displacement about any one of the axes affects only the end-effector.

$$S_4 = S_5 = S_6 = \sum_{i=4}^{n}(a_i + d_i) \tag{21}$$

Assuming uniform mass distribution of the links of the manipulator, we can estimate the amount of power required by each of the joints for a displacement $\Delta\theta_i$ about the ith joint as $S_i\Delta\theta_i$. The good estimate of the total power required by the manipulator to displace itself by $\Delta\theta = [\Delta\theta_1\,\Delta\theta_2\,\Delta\theta_3\,\Delta\theta_4\,\Delta\theta_5\,\Delta\theta_6]$ is given by:

$$\begin{aligned}\text{Torque Factor} &= \sum_{i=1}^{n} S_i\Delta\theta_i \\ &= S_1\Delta\theta_1 + S_2\Delta\theta_2 + S_3\Delta\theta_3 + S_4\Delta\theta_4 \\ &\quad + S_5\Delta\theta_5 + S_6\Delta\theta_6\end{aligned} \tag{22}$$

Table 1 Reachability function values.

Location of the task point 'p'	Reachability function value
When p is inside the workspace and at least one solution is within joint constraints	[0, 1]
When p is inside the workspace and the best solution has at least one of the joint angles at its extreme position	0
When p is inside the workspace and the best solution is one with all joint displacements mid-range	1

If for a given set of task points, one manipulator structure requires major displacement of the lower joints to reach the task point, while the another structure requires the same amount of displacement about higher order joints for the same task points. Then the power required by the first manipulator structure will be more, as the work done by the lower joints in moving the entire structure is more.

This metric can help identify structures that will require minimum power from the set of all reachable structures in executing a trajectory, assuming all other parameters are the same for the manipulators being compared. By minimizing the torque factor we can find manipulator structures that require less power and hence the operating cost will also be lower for such manipulators when compared to others.

To convert these functions into general optimization problems, such that minimizing them will yield optimal solutions we add a negative sign. The functions then become:

$$\text{reachability (DH)} = -\sum_{\forall p \in TS}\left(\max\left[\min\left(\frac{(q_{i,\max}-q_i)(q_i-q_{i,\min})}{(0.5(q_{i,\max}-q_{i,\min}))^2}\right)_{i=1}^{n}\right]_{j=1}^{g}\right) \tag{23}$$

$$f(\text{DH})_{\text{velocity}} = -\sum_{\forall p \in TS}\left(\max[\det(J(q^1)),\det(J(q^2))\ldots,\det(J(q^g))]\right) \tag{24}$$

While the dynamic criterion for minimum torque/power is by definition a function to be minimized

$$\begin{aligned}\text{Torque Factor} &= \sum_{i=1}^{n} S_i \Delta\theta_i \\ &= S_1\Delta\theta_1 + S_2\Delta\theta_2 + S_3\Delta\theta_3 + S_4\Delta\theta_4 \\ &\quad + S_5\Delta\theta_5 + S_6\Delta\theta_6\end{aligned} \tag{25}$$

When multiple task points constitute a task goal these functions will have many local minima. This should be kept in mind while selecting a proper optimization algorithm. Using local minimization routines to find optimal solutions will yield acceptable solutions but not global solutions. Only global minimization routines will be able to deliver an optimal solution for the problem. The choice of the global minimization algorithm to be used depends on the number of iterations required, number for function evaluations and the speed of convergence.

Methodology flowchart

The presented mathematical formulation and methodology can be represented in the form of a flow chart shown in Fig. 3. Random configurations are generated and tested for the existence of the inverse solutions within the joint limits range. In case a solution exists within the joint constraints,

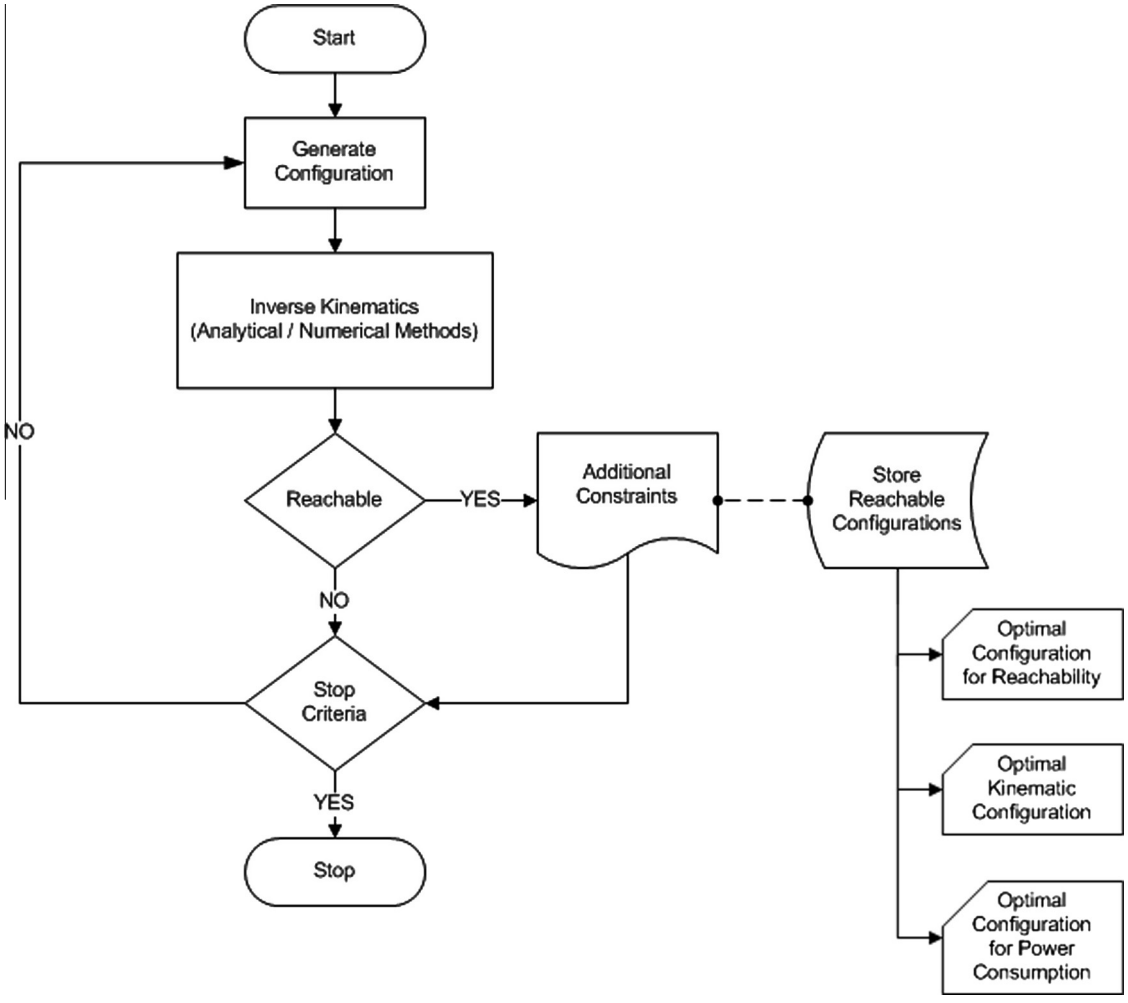


Fig. 3 Proposed methodology flowchart.

we further test the configurations for good manipulability and other additional performance criteria. Every reachable configuration is saved so that it can be used for further analysis and testing. Some of these configurations can also be used as initial starting points or seed values for optimization search algorithms. The final stop criteria can be set in terms of either the number of iterations, number of functional evaluations, or desired objective function value limit or a time limit.

Simulated annealing

There are many approaches to solve a given global optimization problem. The choice of the algorithms greatly depends on factors such as the dimensionality of the problem, the nature of the variables (discrete or continuous), availability of a function derivative. A good global optimization method for a given problem can only be found by matching the features of the problem to the algorithm characteristics and its problem handling capabilities.

In this case, the objective or cost function – which is the reachability function – does not have a direct analytical expression, and is computationally expensive to calculate as it depends on the inverse kinematic solutions. It is also important to note here that this problem does not have a formulation for a function derivative or any function gradient data. The objective function will have multiple local and global minima points where the function value attains the desirable value. The search space is also very exhaustive. Keeping in mind the above factors we chose to implement the problem using Simulated annealing algorithm. The simulated annealing method is a heuristic algorithm.

Simulated annealing was developed in the 1980s by Scott Kirkpatrick [29] based on a statistical algorithm developed much earlier by Metropolis [30], to improve designs of Integrated Circuit (IC) chips by emulating the actual process of annealing.

Simulated Annealing (SA) is a generic probabilistic metaheuristic algorithm for finding the global minimum of a cost function that has many local minima. The SA algorithm uses random generated inputs based on a probabilistic model. Only under certain conditions there is a change in the objective function due to a new random input accepted. The acceptance condition for a new input is given as follows:

$$\Delta f_{obj} \leqslant 0 \tag{26}$$

$$\exp\left(-\frac{\Delta f_{obj}}{T}\right) > random[0,1) \tag{27}$$

where Δf_{obj} is the change in the objective function and T is the temperature of the algorithm.

Starting with a high temperature, the algorithm, with every iterative step, gradually lowers the temperature simulating the annealing process. And, after every fixed number of iterations known as the annealing period, the temperature is raised back again. Higher temperatures mean greater randomization of the input variables. Therefore, a slow annealing method that lowers the temperature gradually will explore the search space to a greater extent that a fast annealing method that lowers the temperature quickly. At lower temperatures the search space is exploited while at high temperature the algorithm explores the search space.

The algorithm stops when there is no change in the objective function for a certain number of consecutive inputs. SA algorithm remembers the best inputs throughout its run. SA works well with high dimensionality problems even when the search space is extensive.

The Simulated Annealing Method first generates random manipulator configurations that are then tested for reachability using the inverse solutions found by the Particle Swarm Optimization. The PSO based inverse kinematic module only searches for solutions within the user specified joint constraints. All the configurations that are found to be reachable are then further tested based on additional criteria. We keep re-annealing, by raising the temperature of the simulated annealing algorithm when the temperature of the algorithm reaches a minimum. The best reachability table is updated every time a better configuration is found.

Inverse kinematics

This methodology works well with both analytical and numerical methods of calculating the inverse kinematic solutions. Analytical methods such as the GIK [31–34] are fast and offer closed form solutions for select class of manipulators. Sometimes analytical methods that convert the inverse kinematic problem into a polynomial with 'n' solutions may yield complex solutions for points that lie outside the reachable or constrained Reachable Workspace. In such cases the following formulation of the reachability can be used to eliminate complex solutions.

$$\begin{aligned}&\text{reachability (DH)}\\&= -\sum_{\forall p\in TS}\left(\max\left[\sum_{i=1}^{n} -imag(q_i)^2\right.\right.\\&\left.\left.+\min\left[\left(\frac{(real(q_i)-q_{i,\min})(q_{i,\max}-real(q_i))}{(0.5(q_{i,\max}-q_{i,\min}))^2}\right)_{i=1}^{n}\right]\right]_{j=1}^{g}\end{aligned} \tag{28}$$

The extra term in the reachability function is for the imaginary solutions that may result from the analytical approach. Even though analytical methods are fast and desirable some configurations do not have closed solutions and we have to resort to numerical approaches to find the solutions. In this work we have used a novel inverse kinematic approach based on Particle Swarm Optimization (PSO) algorithm. This method is not described here as it is out of the scope of this paper.

Experimental results

In this section we test the proposed methodology to design manipulators based on task point descriptions. The task goals differ in the number of task points and also in the orientation required at these task points. For a prismatic link the joint limit is constrained between zero and unity. The joint limit constraints for the revolute joints are set as follows:

Lower Bound = [−160, −45, −225, −110, −100, −266]
Upper Bound = [160, 225, 45, 170, 100, 266]

We test the proposed methodology by simulating the three tasks by specifying their task requirements in terms of the positioning and orientation required by the end-effector. These three task specifications are adaptations of real life applications of industrial serial manipulators. The three tasks are – 1. Horizontal Goal task, 2. Circular Ring Goal task, and 3. Spherical Goal task.

The spherical goal task is an example of an inspection task where the manipulator needs to achieve different orientations about a given task point in three dimensional space. At the same time it also demonstrates that dexterous manipulators (manipulators that can achieve all possible orientations about a given task point) can be designed using this approach.

Next, we present the circular ring task where the manipulator needs to have the same orientation about eight points on the circumference of a circle. This is an example of a real life manipulator job where the robot is required to drill holes or drive screws at these task points to fasten a circular disk or plate.

Another task is the horizontal task. Here the manipulator is required to have a constant orientation about nine points on a horizontal plane. This task mimics a task industrial manipulators commonly have in the packaging industry, where the manipulator could be either applying labels to the individual products or inspecting individual products in a box.

Finally, in the last example we optimize the structure of a Puma560 manipulator to demonstrate the frameworks ability to optimize existing general purpose manipulators for specific tasks. In this example we choose a simple cone task to optimize the Puma560's structure.

For each of the task scenarios the methodology was used to generate three manipulator configurations, the best configuration based reachability, second the best kinematic structure (a configuration that can easily achieve high end-effector velocities), and finally, a configuration with least power consumption for the given task (operating cost). This methodology investigates all possible configurations – RRR, RRP, RPR, PRR, RPP, PRP, PPR, PPP – within the search space, that can meet the task specifications.

Spherical goal

In this task the manipulator is required to have the ability to reach a task point from all possible approaches or angles. This task involves approaching a point from six different angles separated by 90 degrees, such that they represent the three diagonals of a sphere perpendicular to each other. The task points for a sphere goal are given below and the task visualization is shown in Fig. 4.

$$\text{Sphere goal} = \begin{bmatrix} 0 & 0.75 & 0 & 0 & 0 & 0; \\ 0 & 0.75 & 0 & -3.142 & 0 & -3.142; \\ 0 & 0.75 & 0 & 0 & 1.565 & 0; \\ 0 & 0.75 & 0 & 0 & -1.565 & 0; \\ 0 & 0.75 & 0 & -1.372 & 1.541 & -3.142; \\ 0 & 0.75 & 0 & 1.784 & -1.571 & -0.213 \end{bmatrix};$$

Based on the evaluations of all possible configurations, the best configuration that has the maximum overall reachability value for this set of points of the sphere is an RRR–RRR manipulator. This configuration has a reachability value of −0.5441

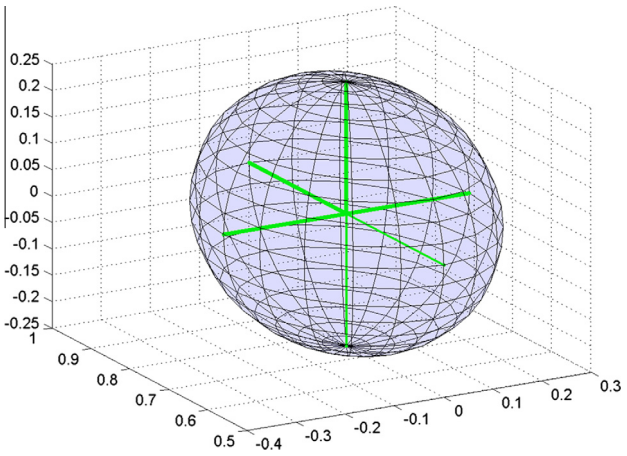


Fig. 4 Task description for the spherical goal.

The DH parameters of the manipulator are:

robot (6 axis, RRRRRR, stdDH)

j	theta	d	a	alpha
1	q1	0.9979	0.8345	-2.375
2	q2	0.7467	0.9979	3.101
3	q3	0.0025	0.9978	2.269
4	q4	0	0	-1.571
5	q5	0	0	1.571
6	q6	0.25	0	0

Fig. 5 shows superimposed manipulator positions at the required task points.

For this task the best kinematic performance structure was found to be:

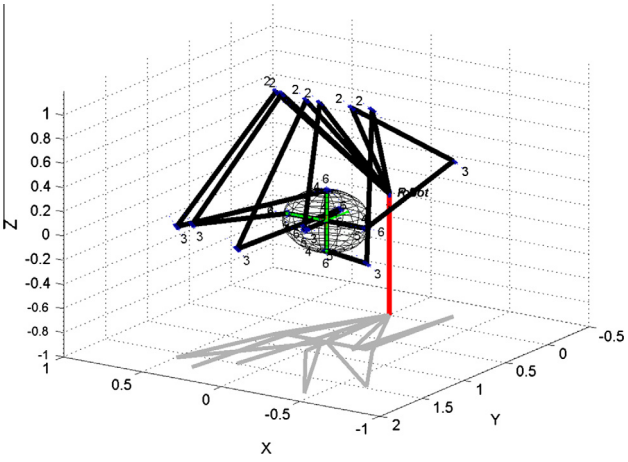


Fig. 5 Designed manipulator reaching all the task points of the spherical goal.

```
robot (6 axis, RRRRRR, stdDH)

+---+-----------+-----------+-----------+-----------+
| j |     theta |         d |         a |     alpha |
+---+-----------+-----------+-----------+-----------+
|  1|         q1|     0.9956|     0.6492|     -2.342|
|  2|         q2|     0.9973|     0.9956|      3.057|
|  3|         q3|   0.005252|     0.9954|      2.167|
|  4|         q4|          0|          0|     -1.571|
|  5|         q5|          0|          0|      1.571|
|  6|         q6|       0.25|          0|          0|
+---+-----------+-----------+-----------+-----------+
```

The manipulator configuration with the least power consumption is:

```
robot (6 axis, RRRRRR, stdDH)

+---+-----------+-----------+-----------+-----------+
| j |     theta |         d |         a |     alpha |
+---+-----------+-----------+-----------+-----------+
|  1|         q1|     0.9518|     0.7191|     -2.016|
|  2|         q2|     0.9491|     0.9643|     -3.036|
|  3|         q3|       0.28|     0.7201|      2.287|
|  4|         q4|          0|          0|     -1.571|
|  5|         q5|          0|          0|      1.571|
|  6|         q6|       0.25|          0|          0|
+---+-----------+-----------+-----------+-----------+
```

As expected the best configuration is a RRR–RRR structure, however the joint twist angles are non-intuitive and not conventional. This unique arrangement of the links with the joint twist respect to each other gives the structure the high reachability and kinematic performance. As seen from the generated configurations the least power consuming configuration has a longer offset for the third joint and the link lengths are slightly shorter when compared to the other two configurations, this helps the manipulator navigate the task points with the amount least joint movement.

While configurations were able to meet the task goal with varying performance metrics, no RPP–RRR configuration could complete the task with the set constraints.

Circular Ring Goal

In this task the manipulator is required to reach eight points on the circumference of a circle with the same orientation at all the task points. The task points for the Ring Goal are given below and the task visualization is shown in Fig. 6.

$$\text{Ring Goal} = \begin{bmatrix} 0.7000 & 0.5000 & 0 & -3.142 & 0-3.142 \\ 0.6414 & 0.6414 & 0 & -3.142 & 0-3.142 \\ 0.5000 & 0.7000 & 0 & -3.142 & 0-3.142 \\ 0.3586 & 0.6414 & 0 & -3.142 & 0-3.142 \\ 0.3000 & 0.5000 & 0 & -3.142 & 0-3.142 \\ 0.3586 & 0.3586 & 0 & -3.142 & 0-3.142 \\ 0.5000 & 0.3000 & 0 & -3.142 & 0-3.142 \\ 0.6414 & 0.3586 & 0 & -3.142 & 0-3.142 \end{bmatrix};$$

Based on the evaluations of all possible configurations, the best configuration that has the maximum overall reachability

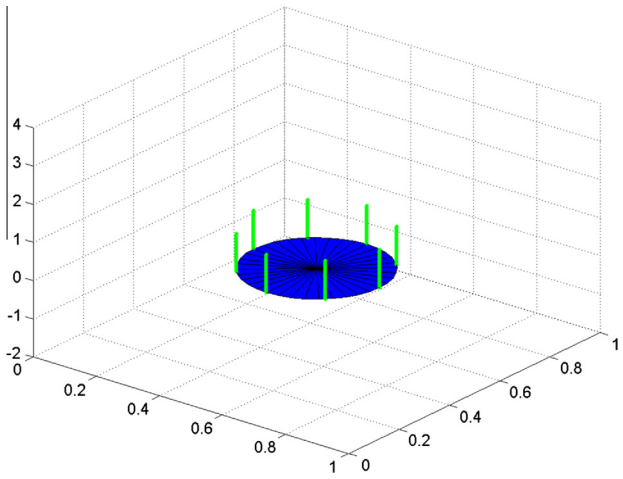


Fig. 6 Task description for the Ring Goal.

value for this set of points of the ring task is an RRR–RRR manipulator. This configuration has a reachability value of −0.833

The DH parameters of the manipulator are:

```
robot (6 axis, RRRRRR, stdDH)

+---+-----------+-----------+-----------+-----------+
| j |     theta |         d |         a |     alpha |
+---+-----------+-----------+-----------+-----------+
|  1|         q1|      0.049|     0.6576|     0.7544|
|  2|         q2|      0.817|      0.908|       3.02|
|  3|         q3|     0.9482|     0.6897|      1.264|
|  4|         q4|          0|          0|     -1.571|
|  5|         q5|          0|          0|      1.571|
|  6|         q6|       0.25|          0|          0|
+---+-----------+-----------+-----------+-----------+
```

Fig. 7. Shows superimposed manipulator positions at the required task points.

For this goal the best kinematic performance structure was found to be:

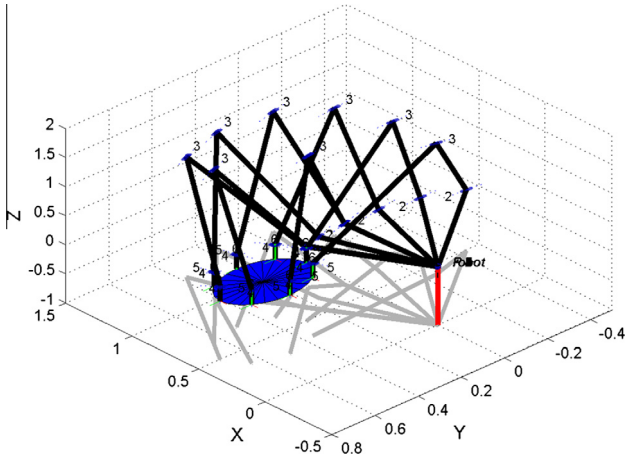


Fig. 7 Designed manipulator reaching all the task points of the Ring Goal.

robot (6 axis, RRRRRR, stdDH)

j	theta	d	a	alpha
1	q1	0.04902	0.6576	0.7545
2	q2	0.8169	0.908	3.02
3	q3	0.9482	0.6897	1.264
4	q4	0	0	-1.571
5	q5	0	0	1.571
6	q6	0.25	0	0

The manipulator configuration with the least power consumption is:

robot (6 axis, RRRRRR, stdDH)

j	theta	d	a	alpha
1	q1	0.04909	0.6577	0.7545
2	q2	0.8169	0.9081	3.02
3	q3	0.948	0.6898	1.264
4	q4	0	0	-1.571
5	q5	0	0	1.571
6	q6	0.25	0	0

As seen from the above three DH tables that the most reachable configuration, the best kinematic performance configuration and the least power consumption configuration are fairly the same. The link length and twist are the same. Therefore, in this case one configuration can achieve all the three criteria. This can be considered to be an ideal case from a designer's point of view as one configuration achieves all three goal. But, may not be the case for most task objectives.

Horizontal plane goal

This task comprises of nine points that lie in a horizontal plane, the manipulator is supposed to reach all of the task points with the same orientation. This task is similar to the task manipulators execute in the packaging/soldering application. The task points for the horizontal plane goal are given below and the task visualization is shown in Fig. 8.

$$\text{Horizontal Plane Goal} = \begin{bmatrix} 0.9 & -0.5 & 0-3.142 & 0-3.142; \\ 0.9 & 0 & 0-3.142 & 0-3.142; \\ 0.9 & 0.5 & 0-3.142 & 0-3.142; \\ 0.7 & -0.5 & 0-3.142 & 0-3.142; \\ 0.7 & 0 & 0-3.142 & 0-3.142; \\ 0.7 & 0.5 & 0-3.142 & 0-3.142; \\ 0.5 & -0.5 & 0-3.142 & 0-3.142; \\ 0.5 & 0 & 0-3.142 & 0-3.142; \\ 0.5 & 0.5 & 0-3.142 & 0-3.142; \end{bmatrix};$$

Based on the evaluations of all possible configurations, the best configuration that has the maximum overall reachability value for this set of points of the horizontal goal is an RRR–RRR manipulator. This configuration has a reachability value of −0.68127

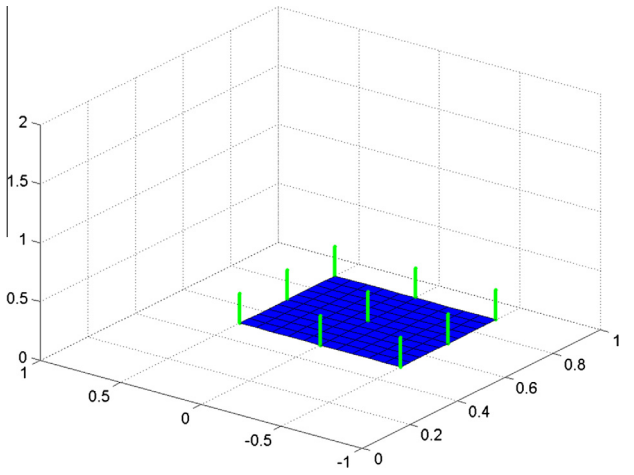


Fig. 8 Task requirements for the horizontal plane goal.

The DH parameters of the manipulator are:

robot (6 axis, RRRRRR, stdDH)

j	theta	d	a	alpha
1	q1	0.2472	0.6404	0.104
2	q2	0.0019	0.6147	1.404
3	q3	0.3707	0.3709	-1.135
4	q4	0	0	-1.571
5	q5	0	0	1.571
6	q6	0.25	0	0

Fig. 9 shows superimposed manipulator positions at the required task points.

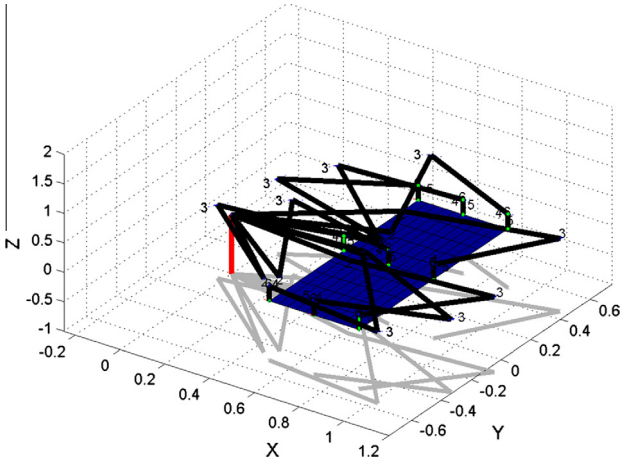


Fig. 9 Designed manipulator reaching all the task points of the horizontal goal.

For this goal the best kinematic performance structure was found to be:

robot (6 axis, RRRRRR, stdDH)

j	theta	d	a	alpha
1	q1	0.1855	0.9	2.619
2	q2	0.07636	0.3638	-1.332
3	q3	0.5625	0.3354	-2.649
4	q4	0	0	-1.571
5	q5	0	0	1.571
6	q6	0.25	0	0

The manipulator configuration with the least power consumption:

robot (6 axis, RPPRRR, stdDH)

j	theta	d	a	alpha
1	q1	0.1406	0.6185	-3.118
2	2.686	q2	0.6054	2.229
3	-2.932	q3	0.04216	3.007
4	q4	0	0	-1.571
5	q5	0	0	1.571
6	q6	0.25	0	0

This example demonstrates the ability of this methodology to evaluate configurations that include prismatic joints as well. In this case the best reachable and kinematic configurations are RRR. This is expected as revolute joints can achieve better reachability and kinematic performance. But the least power consuming configuration is a RPP–RRR manipulator which is very non-intuitional. It is important to note here that the in case of a prismatic link the joint limits are set in terms of the allowable linear displacement of the joint. The methodology therefore generates the optimal values for the joint offset, joint twist and link offset.

Optimized Puma560 compared to the original Puma560

The Puma560 manipulator is one of the widely used and common generalized manipulators. In this example we optimize the Puma's structure using this methodology and compare it with the original manipulator for reachability and kinematic performance. This also demonstrates the ability of the methodology to optimize existing manipulator configurations based on the task descriptions. For this purpose we define a small cone task involving four task points, such that the manipulator orientations at these task points form a conic section. The task points for a cone section goal are as follows:

$$\text{Cone Goal} = \begin{bmatrix} 0 & 0.7 & 0 & -3.142 & 1.162 & -3.142; \\ 0 & 0.7 & 0 & 0 & 1.131 & 0; \\ 0 & 0.7 & 0 & -1.028 & 0.383 & -1.393; \\ 0 & 0.7 & 0 & -1.873 & 0.845 & -1.557; \end{bmatrix};$$

The DH parameter table of the original Puma560 manipulator is given below.

Puma 560 (6 axis, RRRRRR, stdDH)

j	theta	d	a	alpha
1	q1	0	0	1.571
2	q2	0	0.4318	0
3	q3	0.15	0.0203	-1.571
4	q4	0.4318	0	1.571
5	q5	0	0	-1.571
6	q6	0	0	0

When this original Puma was applied to the cone task a reachability function value of -0.248 was achieved. This is clearly a very low value for the reachability indicating that the manipulator joints are close to their limits when reaching the task points. Also the kinematic performance measure was is -0.7232.

Next, the proposed methodology is applied to optimize the link lengths of a Puma560 manipulator for better reachability and kinematic performance for the task. Only the specific link lengths and link offsets are optimized in this process, and the rest a kept the same as the original Puma manipulator. The joint constraints of the original Puma560 manipulator apply for both during task performance.

The DH parameter table of the optimized Puma is shown below.

Puma Mod (6 axis, RRRRRR, stdDH)

j	theta	d	a	alpha
1	q1	0	0	1.571
2	q2	0	0.7299	0
3	q3	0.0207	0.4572	-1.571
4	q4	0.0007	0	1.571
5	q5	0	0	-1.571
6	q6	0.25	0	0

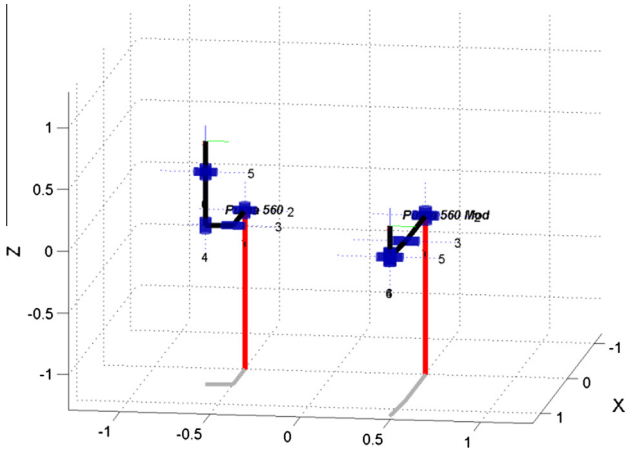


Fig. 10 Puma and optimized Puma in their home position.

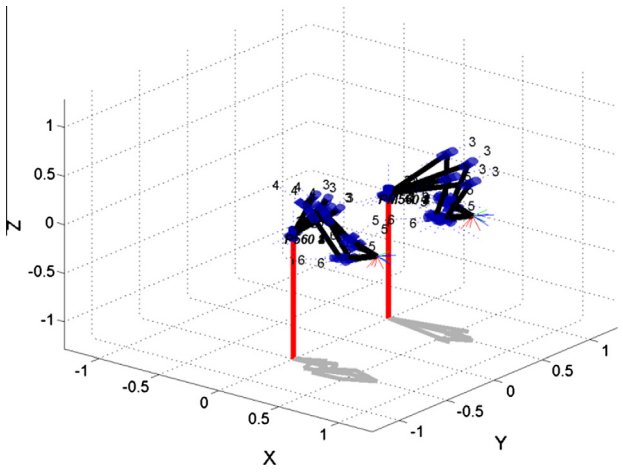


Fig. 11 Superimposed positions of the Puma and Optimized Puma performing the cone task.

Fig. 10 shows the Puma and optimized Puma manipulator in their home positions. When this manipulator is applied to the task of a cone section presented above we get an improved reachability function value of −0.76187. Also the kinematic performance measure for the optimized Puma560 was −0.9017. Fig. 11 shows superimposed positions of the Puma and optimized Puma performing the cone task. For this task the structural length of the optimized Puma is higher than the original Puma manipulator. And therefore, the torque requirement will be higher. This is a common compromise that designer have to make where better performance dictates higher torque requirements. The torque factor for the original puma manipulator is 1.2839 and the torque for the modified Puma560 manipulator is 1.4585.

Discussion

In all the task experiments, the initial seed to the algorithm was a set of random values such that the resultant configuration did not constitute an existing manipulator structure and did not reach even a single task point. The methodology then iteratively found a set of reachable configurations from which task suitable configurations are selected such that they meet the performance requirements under the set constraints.

The optimal manipulator structures for the best reachability, best kinematic performance and least power consumption are not always the same. They can be three different manipulators. A manipulator structure having a very good reachability value for a set task may not actually be the most efficient manipulator. Therefore, selecting the right manipulator will involve a certain intelligent trade off with respect to these parameters.

As expected for most of the tasks, the best manipulator structure found happened to be a RRR/RRR manipulator. This supports the fact that most industrial manipulators are RRR robots with spherical wrists as they provide better reachability at the task points and also the ability to orient the end-effector arbitrarily in the workspace.

The manipulator structures that were generated by the methodology for each of the tasks are not ones that would intuitively come to mind for those tasks. Using this task based tool to design manipulators can help the designer in evaluating and testing better configurations.

In some cases a few structures failed to reach all the task points with the necessary orientation required for task completion. This methodology also helps to determine which configurations can be eliminated from design considerations. For example no RPP/RRR configuration could be found that could successfully complete the sphere goal task within the set joint constraints.

Conclusions

In this work we have presented a general methodology for task-based prototyping of serial robotic manipulators. This framework can be used to generate specialized goal oriented manipulator structures based on the task descriptions. The framework allows for practical joint constraints to be imposed during the design stage of the manipulator. This methodology incorporates the necessary criteria for the design of a manipulator, such as reachability, orientation and non-singularity. However any additional sufficient condition(s) can be specified by the user. Using a set of practical task requirements and constraints we have generated the manipulator configurations such that the task performance is guaranteed even under the imposed joint constraints. Also, the configurations with the best kinematic performance and least power requirement have also been identified. This framework can also be used to optimize a given manipulator for a specific task. This work can be viewed as part of a broader program to develop a general framework for the reverse prototyping of robotic manipulators based on task descriptions and operating constraints.

Conflict of interest

The authors have declared no conflict of interest.

Compliance with Ethics Requirements

This article does not contain any studies with human or animal subjects.

References

[1] Grashof F. Thertische Mshinenlehre Leipzig 1883:113–83.
[2] Paul B. A reassessment of Grashof's criterion. J Mech Des Trans ASME 1979;101(3):515–8.
[3] Vijaykumar R, Waldron KJ, Tsai M. Geometric optimization of serial chain manipulator structures for working volume and dexterity. Int J Robotics Res 1986;5(2):91–103.
[4] Li R, Dai JS. Orientation angle workspaces of planar serial three-link manipulators. Sci China Ser E: Technol Sci 2009;52(4):975–85.
[5] Patel S, Sobh T. Optimal design of three-link planar manipulators using Grashof's criterion. In: Sobh T, Xiong X, editors. Prototyping of robotic systems: applications of design and implementation. USA: IGI Global; 2012. p. 321.
[6] Ting K-L. Five-bar Grashof criteria. J Mech Trans Automat Des 1986;108(4):533–7.
[7] Ting K-L. Mobility criteria of single-loop N-bar linkages. J Mech Trans Automat Des 1989;111(4):504–7.
[8] Ting K-L, Liu Y-W. Rotatability laws for N-bar kinematic chains and their proof. J Mech Des 1991;113(1):32–9.

[9] Angeles J, Rojas A. Manipulator inverse kinematics via condition-number minimization and continuation. Int J Robotics Autom 1987;2(2):61–9.
[10] Salisbury JK, Craig JJ. Articulated hands: force control and kinematic issues. Int J Robotics Res 1982;1(1):4–17.
[11] Sobh TM, Toundykov DY. Optimizing the tasks at hand [robotic manipulators]. IEEE Robot Autom Mag 2004;11(2):78–85.
[12] Sobh T, Wang B, Patel S. A mobile wireless and web based analysis tool for robot design and dynamic control simulation from task points description. J Internet Tech 2003;4(3):153–62.
[13] Sobh T, Wang B, Patel S. Web enabled robot design and dynamic control simulation software solutions from task points description. In: Proc 29th annual conf of the IEEE industrial electronics society; November 2–6, 2003.
[14] Kucuk S, Bingul Z. Robot workspace optimization based on a novel local and global performance indices. In: Proc IEEE int symp on industrial electronics; June 20–23, 2005.
[15] Kucuk S, Bingul Z. Comparative study of performance indices for fundamental robot manipulators. Robotics Autonom Syst 2006;54(7):567–73.
[16] Paredis CJJ, Khosla PK. Kinematic design of serial link manipulators from task specifications. Int J Robotics Res 1993;12(3):274–87.
[17] Patel S, Sobh T. Manipulator performance measures – a comprehensive literature survey. J Intell Robotic Syst; 2014. doi: 10.1007/s10846-014-0024-y.
[18] Kim J, Khosla PK. A formulation for task based design of robot manipulators. In: Proc IEEE/RSJ int conf on intell robots and syst; 26–30 July 1993.
[19] Valsamos C, Moulianitis V, Aspragathos N. Index based optimal anatomy of a metamorphic manipulator for a given task. Robotics Comput-integrated Manuf 2012;28(4):517–29.
[20] Al-Dois H, Jha AK, Mishra RB. Task-based design optimization of serial robot manipulators. Eng Optimiz 2012;45(6):1–12.
[21] Kim J, Khosla PK. Design of space shuttle tile servicing robot: an application of task based kinematic design. In: Proc IEEE int conf on robotics and autom; 2–6 May 1993.
[22] Dash AK, Chen IM, Yeo SH, Yang G. Task-oriented configuration design for reconfigurable parallel manipulator systems. Int J Comp Integrated Manuf 2005;18(7):615–34.
[23] Izumi K, Tamura H, Watanabe K. Task-oriented optimal configuration structure in a three-dimensional self-organizing robot by genetic algorithms. In: Proc IEEE Int conf on systems, man, and cybernetics; 1999.
[24] Chung WK, Jeongheon H, Youm Y, Kim SH. Task based design of modular robot manipulator using efficient genetic algorithm. In: Proc. IEEE int conf on robotics and autom; 20–25 April, 1997.
[25] Kim J, Khosla PK. A Multi-population genetic algorithm and its application to design of manipulators. In: Proc. lEEE/RSJ int conf on intell robots and syst; 7–10 July, 1992.
[26] Shiakolas PS, Koladiya D, Kebrle J. Optimum robot design based on task specifications using evolutionary techniques and kinematic, dynamic, and structural constraints. Inverse Problems Eng 2002;10(4):359–75.
[27] Denavit J, Hartenberg RS. A kinematic notation for lower-pair mechanisms based on matrices. Trans ASME J Appl Mech 1955;22:215–21.
[28] Yoshikawa T. Manipulability of robotic mechanisms. Int J Robotics Res 1985;4(2).
[29] Kirkpatrick S, Gelatt Jr CD, Vecchi MP. Optimization by simulated annealing. Science (New York, NY). 1983;220(4598):671–80 [Epub 1983/05/13] doi: 10.1126/science.220.4598.671.
[30] Metropolis N, Rosenbluth AW, Rosenbluth MN, Teller AH, Teller E. Equation of state calculations by fast computing machines. J Chem Phys 1953;21(6):1087–92.
[31] Pieper DE. The kinematics of manipulators under computer control. PhD Thesis. Stanford University; 1968.
[32] Lee H-Y, Liang C-G. Displacement analysis of the general spatial 7-link 7R mechanism. Mech Mach Theory 1988;23(3):219–26.
[33] Raghaven M, Roth B. Kinematic analysis of the 6R manipulator of general geometry. In: Proc fifth int symp on robotics research; 1990. p. 263–9.
[34] Tsai LW, Morgan AP. Solving the kinematics of the most general six- and five-degree-of-freedom manipulators by continuation methods. ASME J Mech Trans Automat Des 1985;107(2):189–200.