Advanced Robotics

ISSN: 0169-1864 (Print) 1568-5535 (Online) Journal homepage: www.tandfonline.com/journals/tadr20

Task-based optimization of reconfigurable robot manipulators

Jason A. Kereluk & M. Reza Emami

To cite this article: Jason A. Kereluk & M. Reza Emami (2017) Task-based optimization of reconfigurable robot manipulators, Advanced Robotics, 31:16, 836-850, DOI: 10.1080/01691864.2017.1362995

To link to this article: https://doi.org/10.1080/01691864.2017.1362995

Published online: 21 Aug 2017.

Submit your article to this journal

Article views: 355

View related articles

View Crossmark data

Citing articles: 9 View citing articles

Full Terms & Conditions of access and use can be found at
https://www.tandfonline.com/action/journalInformation?journalCode=tadr20

ADVANCED ROBOTICS, 2017
VOL. 31, NO. 16, 836–850
https://doi.org/10.1080/01691864.2017.1362995

FULL PAPER

Task-based optimization of reconfigurable robot manipulators

Jason A. Kereluk[a] and M. Reza Emami[a,b]

[a]University of Toronto Institute for Aerospace Studies, Toronto, Canada; [b]Onboard Space Systems, Space Technology Division, Luleå University of Technology, Kiruna, Sweden

ABSTRACT

Reconfigurable Manipulators are structurally redundant robots that utilize a subset of their joints to perform a specific task optimally. This paper presents a method of finding a task-based optimal configuration for a new type of reconfigurable robot manipulator, called the modular autonomously reconfigurable serial (MARS) manipulator. The reconfiguration optimization treats the joint space of the MARS manipulator as a 12-dimensional smooth configuration manifold. The manifold is discretized and ranked based on a variety of criteria, and then clustered into attractive and repellent regions. The user then specifies which regions are desired in the target configuration, and the manifold is reduced in dimension in order to maximize the number of attractive regions and minimize the number of repellent regions. Six manipulator configurations are synthesized using this approach, and their effectiveness is compared.

ARTICLE HISTORY
Received 3 October 2016
Revised 25 May 2017
Accepted 24 July 2017

KEYWORDS
Robot manipulator; reconfigurable system; task-based optimization; reconfigurable manipulator

1. Introduction

Serial-link robot manipulators (or serial manipulators) are popular tools in a wide range of industrial fields including manufacturing, assembly, and quality control. A primary incentive for serial manipulators is to be reprogrammable for a variety of tasks. Therefore, they are more versatile than their fixed-automation counterparts in that they are able to switch from one task to another with minimal adjustments or modifications. Conventional serial manipulators, however, have a limited range of effective usage for set of tasks by their mechanical structure: the number and types of joints, the joint organization, and the link lengths between joints. Ideally, a manipulator would be selected with the number and configuration of joints suited for their expected range of tasks. This is often not the case, as custom configuration manipulators are expensive and time-consuming to specify. It is usually the case that the best manipulator configuration from a range of available industrial manipulators is chosen and implemented for a range of tasks. However, even in the ideal case of custom implementation, the manipulator is not optimized for each individual task, but rather for a collection of different tasks. For each individual job, it could have too many or too few joints, and likely have the joints in a less than optimal configuration. A reconfigurable serial manipulator can provide a range of kinematic configurations by including or removing joints and changing their relative positioning, which enables the robot to optimally perform a much larger variety of tasks, thus expanding the versatility of the robot manipulator.

The field of reconfigurable robotics refers to a wider range of systems beyond reconfigurable serial manipulators, and it is densely populated with a large variety of applications. The vast majority of reconfigurable robots are a small collection of modules that dynamically link and separate between themselves, often in serial or parallel configurations [1,2]. These modules often have an onboard power supply and computer, and are not tethered to any base station [3–6]. There has been comparatively very little work in the field of reconfigurable serial manipulators. Of the few that have been developed, there has not been any work on autonomous reconfiguration; rather, these manipulators rely on a human operator to manually reconfigure the robot. One of these systems is the reconfigurable modular manipulator system (RMMS) [7,8]. The RMMS is a collection of pneumatically powered revolute 1-DOF (Degree of Freedom) modules that when jointed together make a multi-DOF serial manipulator. The system is reconfigured, as mentioned above, by a human operator, making reconfiguration a time-consuming and involved process. Another, more recent modular reconfigurable serial manipulator was developed by Chen et al.

CONTACT M. Reza Emami emami@utias.utoronto.ca, reza.emami@ltu.se

© 2017 Taylor & Francis and The Robotics Society of Japan

[9,10]. This system is electrically powered instead of pneumatically, and is much smaller, lighter and faster than the RMMS. Further, it is the first serial manipulator of its kind to employ prismatic motion along with revolute motion, and to have a multi-DOF module: a 2-DOF module is often used as the wrist of the robot. This serial manipulator system, like the RMMS, is also manually reconfigurable. As it is lighter and less complex than the RMMS, its reconfiguration time is reduced. It is still, however, a lengthy and involved process to generate new configurations. It is also limited to a finite number of configurations, as it is made up of discrete modules. A similar system with multi-DOF modules is presented in [11].

There are several existing methods of determining an optimal configuration for a given task: a heuristic search paired with simulated annealing as discussed in [12,13], genetic algorithms as discussed in [14], a discrete optimization as discussed in [15], and the method of Linguistic Mechatronics as discussed in [16,17]. While they all work to various levels of success, they all share the same shortcoming: they are designed to work with smaller DOF manipulators, typically 5-6 DOF. If these methods were applied on a manipulator of similar dimension to the MARS Manipulator, it is likely they would either take several years to complete, or else run out of memory.

This paper presents a method of reconfiguring a reconfigurable serial manipulator based on a user-defined task. Section 2 describes the reconfigurable manipulator that this method is applied to, Section 3 details the steps for the reconfiguration, and Section 4 presents the implementation and results of the reconfiguration. Some concluding remarks are made in Section 5.

2. The MARS manipulator

The modular autonomously reconfigurable serial (MARS) manipulator, presented in [18,19] and shown in Figure 1, is a serial-link manipulator with 12 revolute and 6 prismatic joints. By locking a subset of its joints in a specific set of positions, the MARS Manipulator can assume a continuous domain of configurations with different kinematic and dynamic properties.

The manipulator is comprised of six modules of three cascading sizes, namely module A1, A2, B1, B2, C1, and C2. Each module is homogenous in layout, consisting of a prismatic actuation and two revolute actuations, which are mutually perpendicular (P⊥R⊥R). Each module is connected to the previous one such that the kinematic layout for the complete manipulator is (P⊥R⊥R ‖ P⊥R⊥R ‖ … ‖ P⊥R⊥R). This layout was selected for three reasons. First, many serial manipulators have a wrist mechanism located at the end of the manipulator. The wrist configuration is usually either roll-pitch-roll (R⊥R⊥R) or only pitch-roll (R⊥R). Such a wrist configuration would be readily available with the selected layout which aids in creating useful new configurations. The second reason for the selected manipulator layout is that by having the first DOF in the module as a prismatic joint, various link lengths of the final configuration can be achieved. Without a prismatic motion capability, link lengths would be limited to combinations of unchanging module lengths, which would drastically reduce the variety of the possible configurations that the manipulator can assume. The third reason for the selected manipulator layout is related to the mutual orientation of the 3 DOF's of each module, as

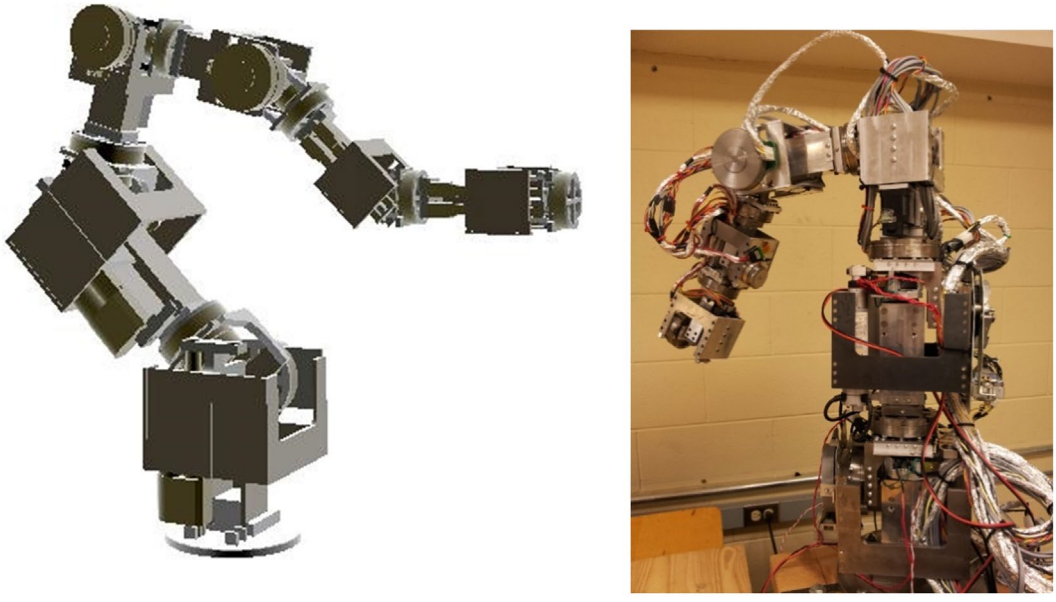

Figure 1. The MARS manipulator.

well as the orientation of the adjacent modules. With their selected layout orientation, it is easy to have two or more joints parallel to each other. This allows for the motion of the actuators to stack, increasing the speed, range and force of the active DOF. Further and more specifically, the stacking of actuators also allows for significantly increasing the speed of linear actuation, which is a design issue for serial manipulators employing a prismatic DOF.

With regard to the dynamic parameters, the largest modules reside at the base of the manipulator, followed by modules of a size smaller, and finally the smallest modules at the end of the manipulator. This cascading module size arrangement allows the mass distribution in the robot from heavy to light as we move from the base to the end-effector, such that more powerful modules reside at the base of the manipulator. The centre of mass of each module is designed to be close to the intersection of the three axes of motion in the module to reduce the moment of inertial components. In addition, the assembly of each module is compact by design to further reduce the system moments of inertia. Considerations were made in the overall system design to minimize cost and to reduce undesirable effects such as backlash.

3. Reconfiguration

The reconfiguration process consists of seven phases:

(1) Construct the configuration manifold, $\mathcal{C}$.
(2) Discretize $\mathcal{C}$ into a set of points $\mathcal{C}_d$.
(3) Rank α based on manipulator criteria.
(4) Construct the feature space $\mathcal{C}_f$.
(5) Specify the task-group weights.
(6) Construct the target configuration manifold $\mathcal{C}_T$.

These phases are separated into an offline and an online group. The offline group (phases 1–4) is the reconfiguration preparation. The online stage (phases 5, 6) is the reconfiguration calculation. The reconfiguration preparation is performed only once, as part of the offline setup of the manipulator. This preparation is then used to speed up runtime of the online optimization. Between the two stages, a user defines a task which specifies optimization criteria used by the optimization.

3.1. Phase 1

The kinematic layout of an n-DOF serial manipulator can be represented with $4n$ Denavit and Hartenberg (DH) parameters: a_i, d_i, α_i, and θ_i [20]. From these parameters, n homogeneous transform matrices, shown in Equation (1), can be derived, one for each joint. Note that there are two conventions for DH parameters, standard and modified. This paper uses standard DH notation.

$$ {}^{i-1}T_i = \begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\sin\alpha_i & a_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\cos\theta_i\sin\alpha_i & a_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1} $$

The set of all homogeneous transformation matrices from the base to end-effector, Equation (2), is a smooth manifold which is the configuration space [21].

$$ \mathcal{C} = \left\{ {}^0T_1,\ {}^1T_2,\ \ldots,{}^{n-1}T_n \right\} \tag{2} $$

The configuration space for the MARS Manipulator is a 12-manifold which is topologically expressed as the Cartesian product of its revolute joints. At this stage, the prismatic joints are all considered locked at a position midway along their stroke. The reason for such assumption is that varying the position of a locked prismatic joint at this stage will only change the magnitude of the twist or wrench vector at the end-effector, not its direction, whereas the revolute joints can influence the manipulator's attributes more drastically. Reducing the manifold dimension to 12 will significantly lessen the computational efforts for the optimization. After this stage, a fine-tuning can be performed through which the position values of the prismatic actuators will be adjusted, as discussed in detail in [18].

The pitch joints are also homeomorphic to $\mathbb{R}^1$ as their range of motion is from $[-\pi, \pi]$. The Roll joints, with their range of $[0,\ 2\pi)$ would be thought of as an isomorphism to a circle $\mathbb{S}^1$. However, due to the real-world application of these joints, the position of 0 is very different from the position of 2π, largely due to wiring constraints. Therefore, the topological range is $(0, 2\pi)$ which is also homeomorphic to $\mathbb{R}^1$. Practically speaking, the fact that the joint can achieve a position exactly equal to 0 is of no consequence. Therefore, the 12-dimensional configuration manifold is homeomorphic to $\mathbb{R}^{12}$.

Alongside the configuration manifold is a second 6-manifold known as $SE(3)$ or the Special Euclidean space [22]. $SE(3)$ is a space which describes rigid body rotation and translation through 3-dimensional space. The rotation through 3-dimensional space is described by the rotation group $SO(3)$ whose elements are 3×3 rotation matrices R. The topology of $SO(3)$ is homeomorphic to the 3-dimensional real projective space $\mathbb{RP}^3$. Translation through 3-dimensional space can be described as a single vector v in $\mathbb{R}^3$. As translation and rotation are independent, the space $SE(3)$ is homeomorphic to $\mathbb{R}^3 \times \mathbb{RP}^3$, the Cartesian product of the isomorphic spaces of translation and rotation. The elements of $SE(3)$ can be arranged as a 4×4 translation matrix T of the form:

$$\left(T = \begin{bmatrix} \mathbf{R} & \mathbf{P} \\ 0 & 1 \end{bmatrix} \right) \in SE(3) \tag{3}$$

For a serial manipulator, a point in the configuration manifold represents a pose of all joints of the manipulator. A point in the Special Euclidian Space represents a pose of the manipulator's end-effector. It is by exploring the links between these two spaces that the end-effector's performance, as governed by the manipulator's behavior is reviled.

3.2. Phase 2

In preparation for creating the feature space (phase 3), the configuration manifold C needs to be discritized. The Van Der Corput sequence is a method of densely discretizing the unit interval (0, 1), as discussed in [23,24].

To cover N dimensions with m Van Der Corput numbers per dimension, D^N vectors need to be generated that express every permutation with repetition of numbers 1 through D. These numbers are then replaced with their Van Der Corput counterpart using Equation (4), which calculates the ith number of the Van Der Corput sequence.

$$n_i = \mathbf{b}_i \begin{bmatrix} 2^{-j} & 2^{-(j-1)} & \dots & 2^{-1} \end{bmatrix}^{\mathbf{T}} \tag{4}$$

where $\mathbf{b}_i$ is a row vector of the binary bits representing the number i, $\left(\text{i.e.,} \mathbf{b}_{10} = \begin{bmatrix} 1 & 0 & 1 & 0 \end{bmatrix} \right)$, and j is the number of columns in $\mathbf{b}_i$.

3.3. Phase 3

Once the manifold has been discretized, the next phase is to rank each point against performance features of the manipulator. The two types of features that are implemented in Section 4 are End-Effector Twist and Manipulator Compliance. Both of these features are measures of the effectiveness of the manipulator's configuration.

3.3.1. End-effector twist

The end-effector twist refers to the linear and angular velocities of the manipulator's end-effector. Calculating the end-effector twist from the joint position and velocities is known as the differential forward kinematics problem. The methods of obtaining the twist given a point in the configuration space are well established. For each of the 3 principle axes: x, y, and z of the end-effector space, the maximum linear and rotational speeds are calculated. For N potential joints, $6N$ scalar values must be calculated for each point: $\boldsymbol{q} = (q_1, \dots, q_N) \in C$. These values are termed $v_{x,i}$, $v_{y,i}$, $v_{z,i}$, $\omega_{x,i}$, $\omega_{y,i}$, $\omega_{z,i}$ for joint i. These values are calculated by inputting the maximum capable speed for joint i into the forward kinematics formulation of the robot, with all other joint speeds at zero. For the MARS Manipulator, the joint speeds are obtained from the technical specifications of the actuators, and they are listed in the following table:

Based on the manipulator's geometry, coordinate frames are attached to the location of every actuator. These actuator frames are aligned such that their z-axis is along the axis of motion for the joint, i.e. the rotational axis for revolute joints and the axis of motion for prismatic joints. For the purposes of evaluating twist, the directions of the x and y axes for these actuator coordinate frames are irrelevant. The origins of these frames are at the intersection of the two links surrounding the joint. The actuator frame of the actuator currently being tested is denoted as frame $\{a\}$. An additional frame is attached to the end-effector, denoted as frame $\{e\}$ aligned such that its z-axis is along the approach vector of the end-effector.

There are two cases which must be considered to obtain the end-effector twist values: if the joint in question is (a) prismatic or (b) revolute in nature. If the joint is prismatic, the values for ω_x, ω_y, and ω_z are all zero, as a prismatic actuator cannot impose an end-effector angular velocity, regardless of the manipulator's configuration. The values for v_x, v_y, and v_z are given by Equation (5):

$$v_x\hat{\boldsymbol{i}} + v_y\hat{\boldsymbol{j}} + v_z\hat{\boldsymbol{k}} = v_{\max}\left({}^{a}u_x\hat{\boldsymbol{i}} + {}^{a}u_y\hat{\boldsymbol{j}} + {}^{a}u_z\hat{\boldsymbol{k}} \right) \tag{5}$$

where $v_{\max}$ is the corresponding value from Table 1 and ${}^{i}u_x$, ${}^{i}u_y$, and ${}^{i}u_z$ are the components of the unit vectors $\hat{\boldsymbol{i}}$, $\hat{\boldsymbol{j}}$, and $\hat{\boldsymbol{k}}$, which describes $\hat{z}_i$ (the axis of motion for frame $\{i\}$).

For the case of a revolute joint, the cross product of $\hat{z}_a$ and the moment-arm from $\{a\}$ to $\{e\}$ must be computed. This is shown in Equation (6).

$$v_x\hat{\boldsymbol{i}} + v_y\hat{\boldsymbol{j}} + v_z\hat{\boldsymbol{k}} = v_{max}\begin{vmatrix} \hat{\boldsymbol{i}} & \hat{\boldsymbol{j}} & \hat{\boldsymbol{k}} \\ {}^{a}u_x & {}^{a}u_y & {}^{a}u_z \\ {}^{e}p_x - {}^{a}p_x & {}^{e}p_y - {}^{a}p_y & {}^{e}p_z - {}^{a}p_z \end{vmatrix} \tag{6}$$

where ${}^{i}p$ is the point describing the location of frame $\{i\}$. As before, $v_{\max}$ is the corresponding value from Table 1. For calculating ω_x, ω_y, and ω_z with a revolute joint, Equation (6) is used with ω_x, ω_y, and ω_z in place of v_x, v_y, and v_z.

Table 1. Joint speed information for the MARS manipulator.

Module	Pitch speed (rpm)	Roll speed (rpm)	Linear speed (mm/s)
A1 & A2	32.1	37.1	23.0
B1 & B2	26.5	52.5	50.0
C1 & C2	16.4	26.5	50.0

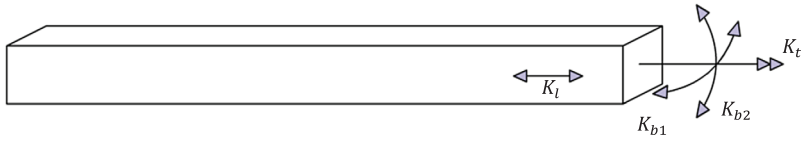

Figure 2. Beam stiffness naming convention.

3.3.2. Manipulator compliance

The compliance of the manipulator is how much the end-effector moves when a force is applied. The measures of compliance are δ_x, δ_y, δ_z, ϕ_n, ϕ_o, ϕ_a which indicate distance (δ) and rotation (ϕ) quantities. The distances is measured in the three principal directions (x, y, z), and the rotation is measured as the angle of the three vectors, normal, orientation, and approach. These variables are measured from an initial pose, which is the position and orientation of the end-effector if the manipulator was a perfectly rigid body, to a final pose, which is taking into account manipulator flexibility. These variables are calculated by determining the change in position and orientation of a set of coordinate frames. The first step is to divide the MARS Manipulator into 12 segments, each segment constituting half of a module. These segments are treated as cantilever-beams with a certain torsional stiffness (K_t), linear stiffness for tension and compression (K_l) and bending stiffness (K_{b1} and K_{b2}) shown in Figure 2.

These stiffnesses are only an approximation of how the manipulator experiences compliance. However, as only a rough ranking is desired as opposed to a specific calculation, this approximation is sufficient. The values of these stiffnesses are shown in Table 2. These values were obtained by performing an FEA (Finite Element Analysis) on the sub-module in question using SolidWorks and COSMOS solid modeling tools. Note that these values are for structural stiffness only. There is also stiffness due to the harmonic drive gearboxes, which will be discussed later in this section.

Equations (7)–(12), give the displacement of the end of a beam given an applied wrench. Equation (7) is the rotation angle given a torque. Equation (8) is the linear compression displacement. Equations (9) and (10) are the displacement and angle of bending given an applied force. Equations (11) and (12) are the displacement and bending angle given an applied moment.

$$\theta = \frac{\mathcal{T} L}{K_t} \tag{7}$$

$$x = \frac{F}{K_l} \tag{8}$$

$$\delta = \frac{FL^3}{3K_b} \tag{9}$$

Table 2. Joint force information.

Module	K_t (Nm²/rad)	K_l (N/m)	K_{b1} (Nm²)	K_{b2} (Nm²)
A1 & A2 – lower	2.26×10^1	4.17×10^5	3.24×10^1	1.05×10^2
A1 & A2 – upper	2.43×10^2	4.15×10^6	2.77×10^2	4.44×10^1
B1 & B2 – lower	3.68×10^1	4.29×10^5	6.44×10^1	2.61×10^2
B1 & B2 – upper	6.21×10^1	1.31×10^6	3.53×10^1	1.58×10^1
C1 & C2 – lower	1.66×10^2	4.76×10^5	1.30×10^2	6.99×10^2
C1 & C2 – upper	7.66×10^2	1.67×10^6	8.10×10^2	3.67×10^2

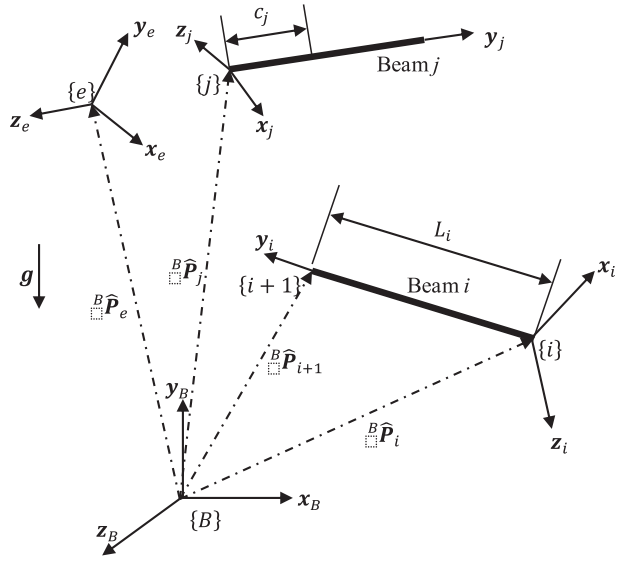

Figure 3. Compliance beam diagram.

$$\phi = \frac{FL^2}{2K_b} \tag{10}$$

$$\delta = \frac{ML^2}{2K_b} \tag{11}$$

$$\phi = \frac{ML}{K_b} \tag{12}$$

The second step is to find the resulting wrench on the end of the segments.

Figure 3 shows two beams, i and j with respect to a base frame. Note that beam j is located between beam i and the end-effector. A unit wrench is applied at the end-effector (frame $\{e\}$).

The length, mass, and location of centre of mass are shown in Table 3.

Table 3. Beam information.

Module	Mass (kg)	Length (m)	Location of CG (m)
A1 & A2 – lower	0.646	$9.0\times10^{-2}+d$	$5.25\times10^{-2}+0.55d$
A1 & A2 – upper	1.092	5.71×10^{-2}	1.98×10^{-2}
B1 & B2 – lower	1.385	$1.48\times10^{-1}+d$	$9.07\times10^{-2}+0.44d$
B1 & B2 – upper	2.105	8.44×10^{-2}	2.37×10^{-2}
C1 & C2 – lower	5.087	$1.78\times10^{-1}+d$	$1.10\times10^{-1}+0.33d$
C1 & C2 – upper	4.341	1.03×10^{-1}	3.74×10^{-2}

Note that for lower modules, length and location of the centre of mass are a function of d. This is because their length changes based on the position of the linear actuator, which is governed by the point $\boldsymbol{q} \in \boldsymbol{C}$. The position (${}^{B}\hat{\boldsymbol{P}}$) and origination ($\boldsymbol{x}$, $\boldsymbol{y}$, and $\boldsymbol{z}$) of all coordinate frames are known with respect to the base frame $\{\boldsymbol{B}\}$. The frame positions and orientations are calculated at the beginning of the algorithm in a similar manner to the frames for calculating end-effector wrench and speed. Based on these beams, the equations for the resulting force and moment applied to beam i are given in Equations (13) and (14).

$$ {}^{B}\hat{\boldsymbol{F}}_i = {}^{B}\hat{\boldsymbol{F}}_e - \left(\sum_{j=i+1}^{12} m_j g\right)\hat{\boldsymbol{j}} \tag{13} $$

$$ \begin{aligned} {}^{B}\hat{\boldsymbol{M}}_i =& {}^{B}\hat{\boldsymbol{M}}_e + \left({}^{B}\hat{\boldsymbol{P}}_e - {}^{B}\hat{\boldsymbol{P}}_{i+1}\right)\times {}^{B}\hat{\boldsymbol{F}}_e \\ &+ \sum_{j=i+1}^{12}\left[\left({}^{B}\hat{\boldsymbol{P}}_j - {}^{B}\hat{\boldsymbol{P}}_{i+1} + {}^{B}R_j\begin{bmatrix}0\\ c_j\\ 0\end{bmatrix}\right)\times\left(-m_j g\hat{\boldsymbol{j}}\right)\right] \end{aligned} \tag{14} $$

where ${}^{B}R_j$ is a rotation matrix from the base to the jth frame, shown in Equation (15).

$$ {}^{B}\boldsymbol{R}_j = \begin{bmatrix} {}^{B}\hat{\boldsymbol{x}}_j & {}^{B}\hat{\boldsymbol{y}}_j & {}^{B}\hat{\boldsymbol{z}}_j \end{bmatrix} \tag{15} $$

The third step is to find the deflections, both angular and linear, of beam i due to the Equations (7)–(12). Equation (16) gives the change in length of beam i, in the base frame, due to the applied force.

$$ {}^{B}\widehat{\Delta \boldsymbol{L}}_i = \frac{\left({}^{B}\hat{\boldsymbol{F}}_i \cdot {}^{B}\hat{\boldsymbol{y}}_i\right){}^{B}\hat{\boldsymbol{y}}_i}{K_{li}} \tag{16} $$

Equation (17) gives the deflection of beam i, in the base frame, due to the applied force and moment.

$$ \begin{aligned} {}^{B}\hat{\boldsymbol{\delta}}_i =& \left(\frac{L_i^3\left({}^{B}\hat{\boldsymbol{F}}_i \cdot {}^{B}\hat{\boldsymbol{z}}_i\right)}{3} + \frac{L_i^2\left({}^{B}\hat{\boldsymbol{M}}_i \cdot {}^{B}\hat{\boldsymbol{x}}_i\right)}{2}\right)\frac{{}^{B}\hat{\boldsymbol{z}}_i}{K_{b1_i}} \\ &+ \left(\frac{L_i^3\left({}^{B}\hat{\boldsymbol{F}}_i \cdot {}^{B}\hat{\boldsymbol{x}}_i\right)}{3} - \frac{L_i^2\left({}^{B}\hat{\boldsymbol{M}}_i \cdot {}^{B}\hat{\boldsymbol{z}}_i\right)}{2}\right)\frac{{}^{B}\hat{\boldsymbol{x}}_i}{K_{b2_i}} \end{aligned} \tag{17} $$

Table 4. Harmonic drive stiffness value.

Module	K_{HD} (Nm/rad)
A1 & A2 – lower	0.4×10^4
A1 & A2 – upper	0.4×10^4
B1 & B2 – lower	0.84×10^4
B1 & B2 – upper	0.84×10^4
C1 & C2 – lower	2.7×10^4
C1 & C2 – upper	2.7×10^4

Table 5. Harmonic drive Boolean variables.

i	ς_{ix}	ς_{iy}
Odd	1	0
Even	0	1

For the rotations, first the 3 principle rotations for the ith coordinate frame are calculated using Equations (18)–(20).

$$ {}^{i}\theta_x = \frac{L_i\left({}^{B}\hat{\boldsymbol{M}}_i \cdot {}^{B}\hat{\boldsymbol{x}}_i\right)}{K_{b1_i}} + \frac{L_i^2\left({}^{B}\hat{\boldsymbol{F}}_i \cdot {}^{B}\hat{\boldsymbol{z}}_i\right)}{2K_{b1_i}} + \zeta_{ix}\frac{\left({}^{B}\hat{\boldsymbol{M}}_i \cdot {}^{B}\hat{\boldsymbol{x}}_i\right)}{K_{HD_i}} \tag{18} $$

$$ {}^{i}\theta_y = \frac{L_i\left({}^{B}\hat{\boldsymbol{M}}_i \cdot {}^{B}\hat{\boldsymbol{y}}_i\right)}{K_{t_i}} + \zeta_{iy}\frac{\left({}^{B}\hat{\boldsymbol{M}}_i \cdot {}^{B}\hat{\boldsymbol{y}}_i\right)}{K_{HD_i}} \tag{19} $$

$$ {}^{i}\theta_z = \frac{L_i\left({}^{B}\hat{\boldsymbol{M}}_i \cdot {}^{B}\hat{\boldsymbol{z}}_i\right)}{K_{b2_i}} - \frac{L_i^2\left({}^{B}\hat{\boldsymbol{F}}_i \cdot {}^{B}\hat{\boldsymbol{x}}_i\right)}{2K_{b2_i}} \tag{20} $$

Note the presence of two additional constants, $\boldsymbol{\zeta_{iy}}$ and $\boldsymbol{K_{HD_i}}$. These take into account the compliance of the harmonic drives of the robot. The MARS Manipulator employs harmonic drive gearboxes which have inherent compliance. The value of $\boldsymbol{K_{HD_i}}$ is provided by the manufacturer of these gearboxes, and is given in Table 4. For a given sub-module, there is one harmonic drive at the end. For lower modules, the harmonic drive rotates about the local x-axis, and for upper modules, it rotates about the local y-axis. Therefore, the variable $\boldsymbol{\zeta_{ix}}$ and $\boldsymbol{\zeta_{iy}}$ are Boolean variables which follow the values in Table 5.

At this stage it is assumed that the values of θ produced by Equations (12)–(14) are small. Because of this assumption, it does not matter in what order these rotations are performed. From this assumption, a rotation matrix can be formed my multiplying principle rotation matrices together. This rotation matrix is shown in Equation (21), and is sometimes referred to as the x–y–z fixed rotation matrix [20]. Note that the pre-super script of i was dropped from this formulation for simplicity.

$$R_{\text{bending}} = \begin{bmatrix} \cos\theta_z \cos\theta_y & \cos\theta_z \sin\theta_y \sin\theta_x - \sin\theta_z \cos\theta_x & \cos\theta_z \sin\theta_y \cos\theta_x - \sin\theta_z \sin\theta_x \\ \sin\theta_z \cos\theta_y & \sin\theta_z \sin\theta_y \sin\theta_x + \cos\theta_z \cos\theta_x & \sin\theta_z \sin\theta_y \cos\theta_x - \cos\theta_z \sin\theta_x \\ -\sin\theta_y & \cos\theta_y \sin\theta_x & \cos\theta_y \cos\theta_x \end{bmatrix} \tag{21}$$

This rotation matrix is the amount frame $\{i + 1\}$ rotates $i + 1$ due to bending of beam i. At this stage, all beams after i are considered as rigid bodies. Therefore, all frames after frame $\{i\}$ can be rotated with Equation (15). However, this rotation matrix is a rotation about frame $\{i\}$. Therefore it must be pre and post multiplied by the rotation matrix in Equation (15) to translate it to and from the base frame. This is shown in Equation (22).

$$\begin{bmatrix} {}^B\hat{\boldsymbol{x}}_j & {}^B\hat{\boldsymbol{y}}_j & {}^B\hat{\boldsymbol{z}}_j \end{bmatrix}^{\text{new}} = {}^B R_i R_{\text{bending}} {}^B R_i{}^T \begin{bmatrix} {}^B\hat{\boldsymbol{x}}_j & {}^B\hat{\boldsymbol{y}}_j & {}^B\hat{\boldsymbol{z}}_j \end{bmatrix}^{\text{old}} \tag{22}$$

Due to the rotation of frame $\{i + 1\}$, frames further down the link will translate as well as rotate. This translation is described in Equation (23), which uses a small angle approximation. Clearly the value for ${}^B\hat{\boldsymbol{d}}_j$ is zero for ${}^B\hat{\boldsymbol{P}}_j = {}^B\hat{\boldsymbol{P}}_{i+1}$.

$$\begin{aligned} {}^B\hat{\boldsymbol{d}}_j = {}^i\theta_x \left[{}^B\hat{\boldsymbol{x}}_i \times \left({}^B\hat{\boldsymbol{P}}_j - {}^B\hat{\boldsymbol{P}}_{i+1} \right) \right] \\ + {}^i\theta_y \left[{}^B\hat{\boldsymbol{y}}_i \times \left({}^B\hat{\boldsymbol{P}}_j - {}^B\hat{\boldsymbol{P}}_{i+1} \right) \right] \\ + {}^i\theta_z \left[{}^B\hat{\boldsymbol{z}}_i \times \left({}^B\hat{\boldsymbol{P}}_j - {}^B\hat{\boldsymbol{P}}_{i+1} \right) \right] \end{aligned} \tag{23}$$

Using Equations (10), (11), and (16), a formula can be developed for the translation of frame $\{j\}$ based on an applied wrench and the masses of all links. This is shown as Equation (24).

$${}^B\hat{\boldsymbol{P}}_j^{\text{new}} = {}^B\hat{\boldsymbol{P}}_j^{\text{old}} + {}^B\hat{\boldsymbol{d}}_i + {}^B\widehat{\Delta \boldsymbol{L}}_i + {}^B\hat{\boldsymbol{\delta}}_i \tag{24}$$

The fourth and final step is to evaluate Equations (22) and (24) for all $j > i$ in sequence for $i = [1, 12]$. This is explained by the flow chart in Figure 4.

Once this process terminates, the values of $\begin{bmatrix} {}^B\hat{\boldsymbol{x}}_{13} & {}^B\hat{\boldsymbol{y}}_{13} & {}^B\hat{\boldsymbol{z}}_{13} \end{bmatrix}^{\text{new}}$ and ${}^B\hat{\boldsymbol{P}}_{13}^{\text{new}}$ describe the position of the end-effector based on the bending due to compliance. The difference of these values with the end-effectors original positions represents the compliance of the manipulator. The difference of position can be easily divided into three values δ_x, δ_y, δ_z. The difference of the unit vectors describing orientation of the end-effector can be represented by angles using the dot-product, shown in Equation (25).

$$\theta_x = \cos^{-1}\left[\left({}^B\hat{\boldsymbol{x}}_{13}^{\text{final}} \right) \cdot \left({}^B\hat{\boldsymbol{x}}_{13}^{\text{origional}} \right) \right] \tag{25}$$

From this equation, values of θ_x, θ_y, θ_z can be generated. It is these 6 values, δ_x, δ_y, δ_z, θ_x, θ_y, and θ_z, which represent the compliance rank for the point $\boldsymbol{q} \in \mathcal{C}$.

3.4. Phase 4

Once all the points in the discretized configuration manifold $\mathcal{C}_d$ have values for the feature types, phase 4 is to cluster the points by distance within each category. Each point can be evaluated against a certain criteria. Based on this evaluation, it can be determined whether the point represents a positive attribute, a negative attribute, or a

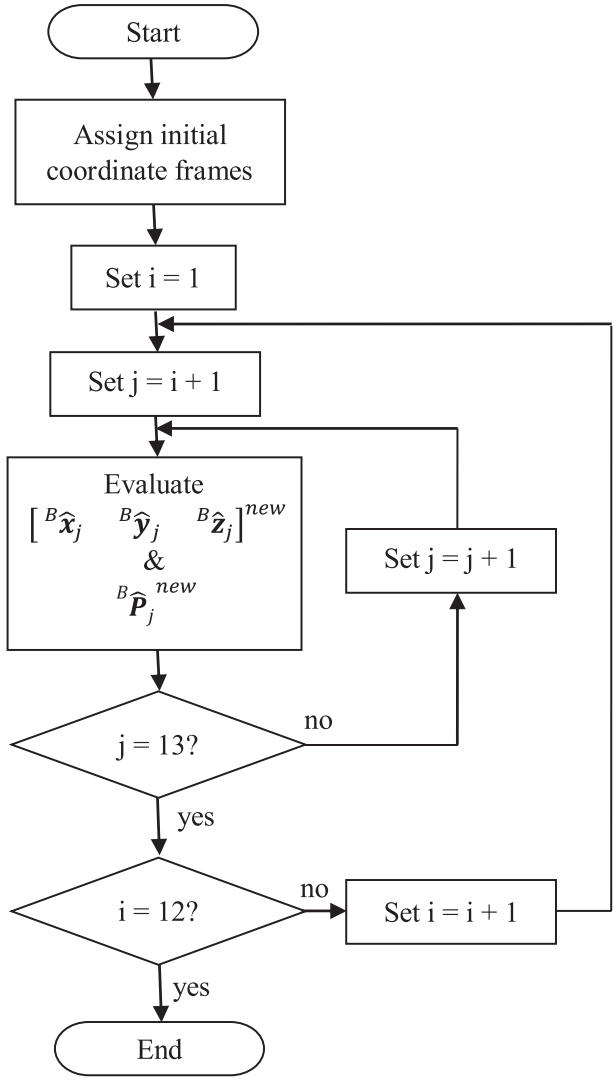

Figure 4. Compliance evaluation flowchart.

Table 6. Threshold values.

	T_{min}	T_{max}
Deflection (linear)	0.0084 m	0.0530 m
Deflection (angular)	0.3488 rad	0.3587 rad
Twist (linear)	0.0596 m/s	0.8657 m/s
Twist (angular)	0.33 rad/s	2.53 rad/s

neutral attribute. This is done by comparing the computed value to a pair of threshold values, T_{max} and T_{min}. If the computed value is between the two threshold values, it is deemed a neutral attribute. For compliance, if the computed value is less than T_{min} it is deemed a positive attribute whereas for twist if the computed value is greater than T_{max} it is deemed as a positive attribute. The values of the thresholds are described in Table 6.

This clustering is done via the BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) clustering algorithm, first proposed in [25,26]. BIRCH is used because it is designed to work for data-sets that are very large and memory intensive, as well as for dimension much higher than 2. One of the main advantages of BIRCH comes from the 3 variables that describe each cluster: Number of patterns, Linear Sum and Sum Squared (N, $\widehat{LS}$, SS). Each of these variables can be updated as a new pattern is added to a cluster, shown in Equations (26)–(29).

$$\hat{X} = [X_1, X_2, \ldots, X_n] \tag{26}$$

$$N_{new} = N + 1 \tag{27}$$

$$\widehat{LS}_{new} = \widehat{LS} + \hat{X} \tag{28}$$

$$\widehat{SS}_{new} = \widehat{SS} + [X_1^2, X_2^2, \ldots, X_n^2] \tag{29}$$

These 3 variables can in turn be translated into the cluster's center of mass and variance:

$$\hat{C} = \frac{\widehat{LS}}{\boldsymbol{N}} \tag{30}$$

$$\sigma_i^2 = \frac{\boldsymbol{SS_i}}{\boldsymbol{N}} - \left(\frac{\boldsymbol{LS_i}}{\boldsymbol{N}}\right)^2 \tag{31}$$

Once the clustering is complete, each cluster is represented by a 12-dimensional ellipse. The radii of the resulting hyperellipse are a function of the standard deviation, found in Equation (32). A multiplier is applied to each index of the standard deviation to obtain the associated radius:

$$r_i = r_m \sigma_i \quad \text{for } i = 1, 2, 3 \ldots 12 \tag{32}$$

This multiplier was found experimentally to have an optimal value of $r_m = 2$ by testing sample points, and whether or not they were contained within hyperellipses of various sizes. More details are discussed in [27].

3.5. Phase 5

Phase 5 involves the user specifying design weights (termed as the task-group) to govern the optimization. Each of the criteria has 12 weights, divided in two groups of 6 maximum and 6 minimum, for which the user assigns a value on the scale of [−1, 1]. Each group is further divided into 3 translation and 3 rotation weights. An example is shown in Table 8, in which the 12 twist weights are assigned, as well as a single compliance weight (leaving the other 11 compliance weights as zero). A negative weight indicates an undesirable trait, and a positive weight indicates a desirable trait. Such weights would trend the optimization to eliminate performance in a certain area. For example, for designing a planar configuration that has no linear motion in the x-dimension, (large) negative values can be assigned to both weights of the x-component of the translational twist. This method is used to guide the optimization toward certain areas of interest, as desired by the user.

3.6. Phase 6

Phase 6 involves reducing the 12-dimentional configuration manifold into a lesser dimensional target configuration manifold suited for the task defined in phase 5. This manifold reduction takes the form of an optimization. This optimization is performed by systematically eliminating dimensions of the manifold as long as it is still desirable to do so.

The first step in eliminating a subspace is to check all potential subspaces and find the optimal point for elimination for each. A single subspace can be thought of as a series of hyperellipses along a one-dimensional axis; note that a hyperellipse may be above or below this axis, the only relevant coordinate is the one associated with that axis.

First a search threshold (ξ) is defined. This value is the resolution in which each dimension is searched to find its optimal elimination point. At each point searched, the value for elimination ($\mathcal{V}_e$) is calculated as a function of the dimension to eliminate (M) and the point $p \in [0, 1]$ representing the point associated with the dimension, shown below:

$$\mathcal{V}_e(p, M) = \sum_{k=1}^{K} \left(Vw_k\right) \left| Vw_k \right| \tag{33}$$

The value for elimination is essentially a signed sum of the square of the weighted volume of all K hyperellipses that have an intersection of the point p on dimension M, where w_k is the weight associated with hyperellipse k.

The formula describing the region of a non-rotated N-ellipsoid is given by:

$$\left(\frac{x_1}{r_1}\right)^2 + \left(\frac{x_2}{r_2}\right)^2 + \cdots + \left(\frac{x_N}{r_N}\right)^2 \leq 1 \tag{34}$$

where r_i is the radius of the hyperellipse along the ith coordinate axis. The volume of this hyperellipse is given by:

$$V(R,N) = \left[\frac{\pi^{0.5N}}{\Gamma(0.5N+1)}\right] \prod_{i=1}^{N} r_i \tag{35}$$

where

$$\Gamma(a+1) = a! \quad \text{for } a = 2t, t \in \mathbb{N}$$
$$\Gamma(a+0.5) = \frac{(1)(3)(5)\ldots(2a-1)}{(2^a)}\sqrt{\pi} \quad \text{for } a = 2t+1, t \in (\mathbb{N} \cup \{0\})$$

When a dimension M is eliminated, a resulting hyperellipse of dimension $N-1$ is created, described by:

$$\left(\frac{x_1}{r'_1}\right)^2 + \left(\frac{x_2}{r'_2}\right)^2 + \cdots + \left(\frac{x_{M-1}}{r'_{M-1}}\right)^2 + \left(\frac{x_{M+1}}{r'_{M+1}}\right)^2 \cdots + \left(\frac{x_N}{r'_N}\right)^2 \leq 1 \tag{36}$$

The $N-1$ new radii are given as:

$$r'_i = r_i \sqrt{1 - \left(\frac{c_M - p}{r_M}\right)^2} \tag{37}$$

where c_M is the location of the centroid of the hyperellipse in dimension M.

Then, each dimension is searched for the optimal point. Once this point has been found for each dimension, the dimension with the highest value is chosen for elimination, and the process repeats. This is shown in Figure 5.

4. Implementation in simulation environment

Because of the dimensional size of the configuration manifold, the phase 4 clustering was implemented on a bank of 18 PCs. The features that were implemented in

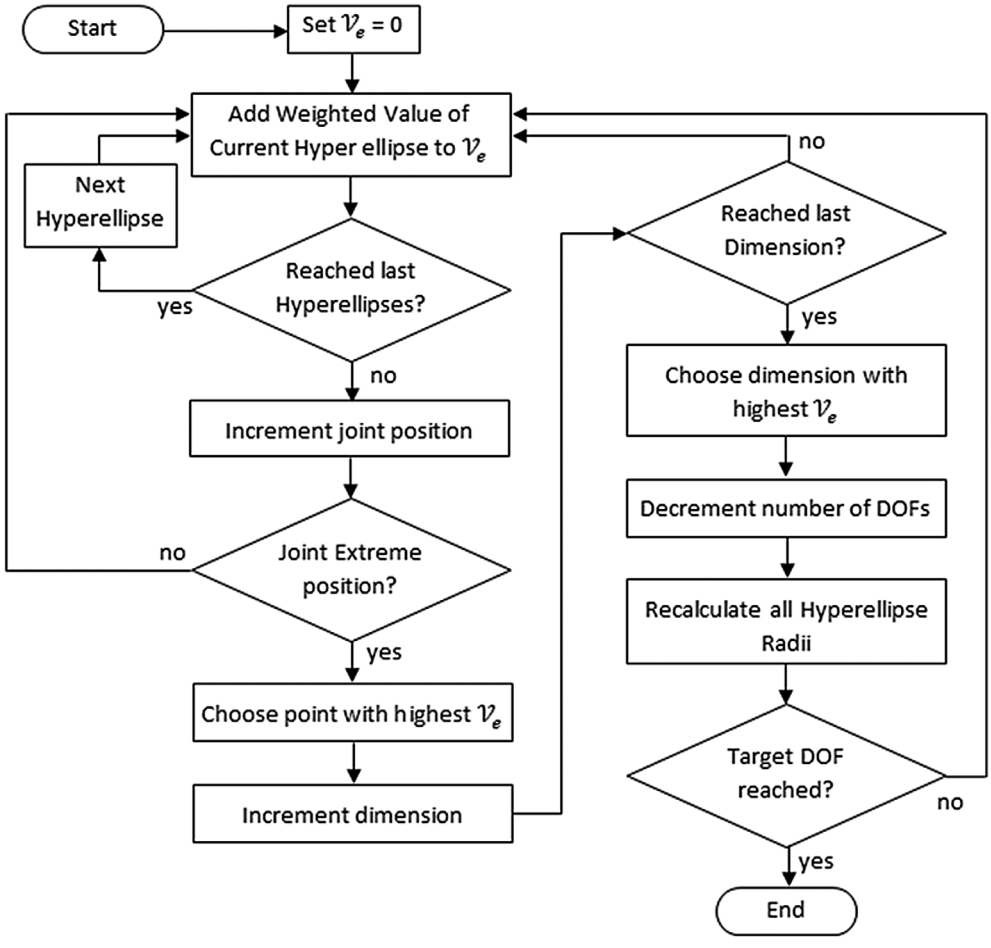

Figure 5. Constructing target configuration manifold flowchart.

the clustering process were Twist and Compliance, each requiring approximately 16,000 computational hours to cluster. Each PC is of identical hardware specification, listed as follows:

- E2180 Dual Core CPU with 1MB Cache and 64 bit Registers
- 2GB RAM
- 320 GB HHD
- Windows Server 2003 (R2) OS

The PCs were networked together in order to share data, and equipped with a scheduling system, as their availability was limited, and they would have to regularly hand off jobs and free up resources for other operations.

The clustering process was run using a combination of the Java implementation of the BIRCH clustering method, from [25] and a complied MATLAB® space evaluation function, running the Matlab Compiler Runtime (MCR). A summary of the clustering results in terms of compression ratio is listed in Table 7. The initial size assumes an 8-byte (double) variable for each of the criteria for the space.

4.1. Reducing the dimension of the configuration manifold

The first sought manipulators are two 6-DOF manipulators that have similarly high twist in all directions and varying compliance requirements. For this reduction, and all future reductions, the search threshold is $\xi = 22.5°$, which is chosen as a trade-off between accuracy of results and computational efficiency. The two manipulator's task-group weights are defined in Table 8 (note that all compliance weights are zero unless otherwise specified).

Based on these weights, the two manipulators are expected to have the following characteristics relative to each other:

(1) Both manipulators should have full motion in SE(3) for the majority of their workspaces.
(2) Manipulator 2 should have more maximum linear twist in all directions compared to Manipulator 1.
(3) Manipulator 2 should have more maximum rotation twist, especially in the y and z directions.
(4) Manipulator 1 should have lower compliance in the y direction compared to Manipulator 2.

The manipulators obtained from the optimization have configurations as shown in Table 9.

At first glance, the two manipulators are very similar, with only one locked joint different between them. To investigate the performance differences between the two manipulator configurations, a joint trajectory was plotted that spans the complete workspace for each manipulator. Then, each position in the workspace was evaluated in the relevant areas for performance. The trajectory that the manipulators follow simply visits each hypersphere's center in the order in which they were created. More efficient trajectories undoubtedly exist, however, provided that they span the space, they will give a accurate picture of the manipulator's capabilities. The resulting evaluations were sorted in ascending order (otherwise, they would appear as little more than white noise) and plotted in Figures 6–8.

Table 7. Clustering compression ratio.

	Initial memory size (GB)	Clustered memory size (MB)	Compression ratio
A2 Roll Twist	10.91	0.62	18026:1
A2 Pitch Twist	16.4	16.5	1016:1
A1 Roll Twist	16.4	3.36	4989:1
A1 Pitch Twist	16.4	20.8	818:1
B2 Roll Twist	21.83	13.5	1655:1
B2 Pitch Twist	21.83	17.5	1277:1
B1 Roll Twist	21.83	7.83	2855:1
B1 Pitch Twist	21.83	21.0	1064:1
C2 Roll Twist	21.83	15.6	1432::1
C2 Pitch Twist	16.4	22.5	745:1
C1 Roll Twist	18.2	18.7	996:1
C1 Pitch Twist	7.28	21.6	345:1
Compliance	21.83	42.2	530:1

Table 8. Task-group weights for first reduction.

Weight	Manipulator 1	Manipulator 2
v_x min	−0.9	−0.9
v_x max	0	0.6
v_y min	−0.9	−0.9
v_y max	0	0.6
v_z min	−0.9	−0.9
v_z max	0	0.6
ω_x min	−1	−1
ω_x max	0.25	0.25
ω_y min	−1	−1
ω_y max	0	0.25
ω_z min	−1	−1
ω_z max	0	0.25
d_y min	0.5	0

Table 9. Configurations for first reduction.

Joint	Manipulator 1	Manipulator 2
A2 Roll	−180°	−180°
A2 Pitch		
A1 Roll		
A1 Pitch		45°
B2 Roll		
B2 Pitch	−90°	
B1 Roll		
B1 Pitch		
C2 Roll	−180°	−180°
C2 Pitch	−90°	−90°
C1 Roll	180°	180°
C1 Pitch	90°	90°

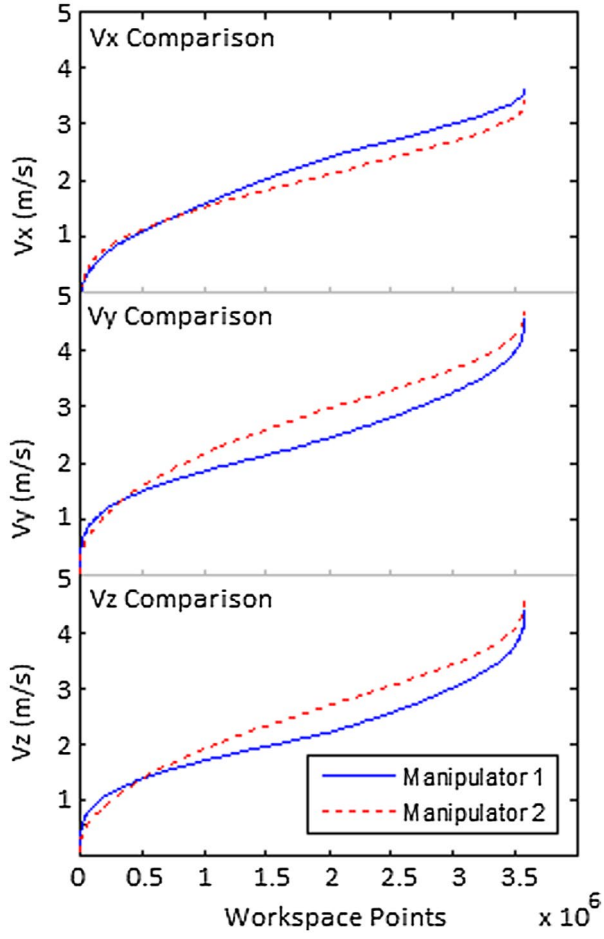

Figure 6. Linear twist comparison for manipulators 1 and 2.

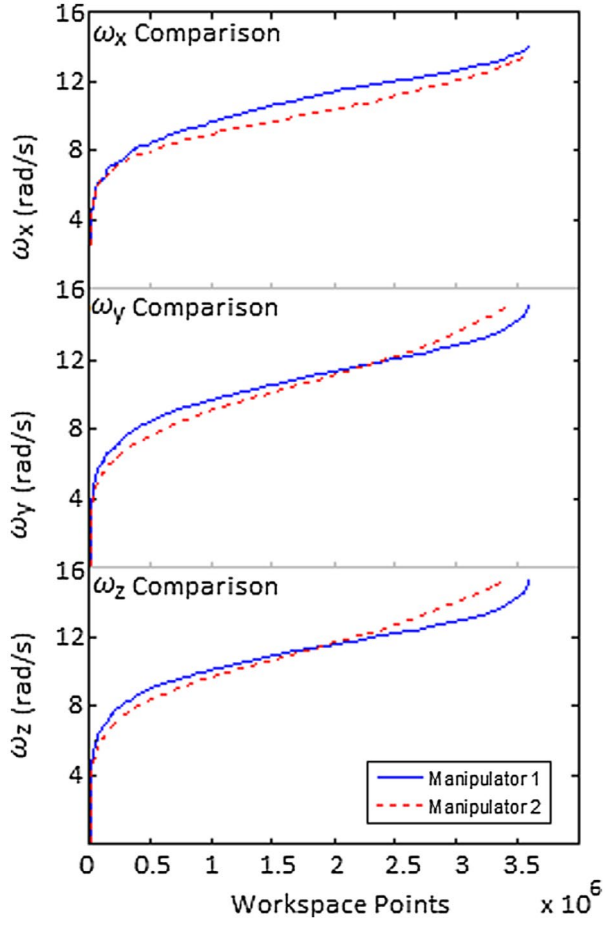

Figure 7. Rotational twist comparison for manipulators 1 and 2.

Based on the above results, the expected characteristics mirror what is observed, but only slightly. Both manipulators have full motion in SE(3). Overall, Manipulator 2 has more maximum linear twist compared to Manipulator 1, except in the x-direction, where it is reversed. Taking an average over all three linear directions, however, the linear twist behavior follows the task-group weights specified, however, this difference is minor and the two manipulators have very similar performance. Manipulator 2 has increased maximum rotation twist in the y and z directions. It is interesting to note that, because there are zero weights for Manipulator 1 in the y and z maximum rotational twist, this inherently puts more of an emphasis on the x direction maximum rotational twist, which is shown in the results in Figure 7. Finally, Manipulator 1 has a lower compliance in the y for the majority of its workspace compared to Manipulator 2.

4.2. Creating 3-DOF manipulator arm with spherical wrist

In serial manipulator design, it is often advantageous to design only the arm of the manipulator, and install a spherical wrist to take care of end-effector orientation. The arm often consists of a collection of 3-DOFs near the base, responsible for positioning the end-effector. To this end, two more manipulators were designed without weights for rotational twist, and constraining the optimization to always leave the final 3 joints unlocked (constituting a spherical wrist). The task-group weights for these two manipulators are defined in Table 10.

Based on these weights, the three manipulators are expected to have the following characteristics relative to each other:

(1) All three manipulators should have full motion in SE(3) for the majority of their workspaces.
(2) Manipulator 3 should have slightly decreased linear twist relative to Manipulators 4 and 5 which should have roughly equal linear twist relative to each other.
(3) Manipulator 3 should have the lowest compliance in the y direction, and Manipulator 4 should have the highest.

The final manipulator configurations are shown in Table 11.

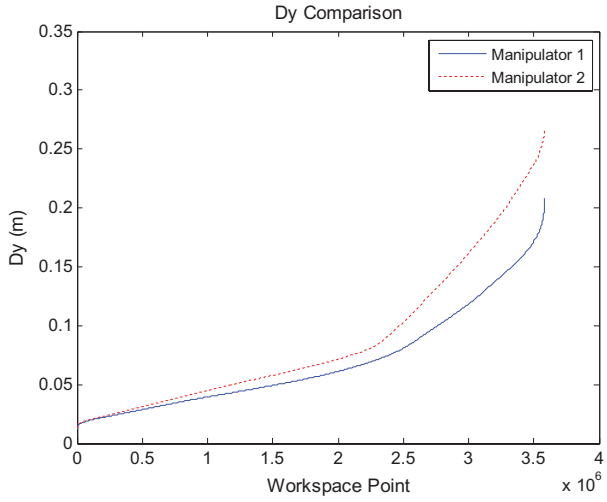

Figure 8. d_y Comparison for Manipulators 1 and 2.

Table 10. Task-group weights for second reduction.

Weight	Manipulator 3	Manipulator 4	Manipulator 5
v_x min	−1	−1	−1
v_x max	0	0.2	0.2
v_y min	−1	−1	−1
v_y max	0	0.2	0.2
v_z min	−1	−1	−1
v_z max	0	0.2	0.2
ω_x min	0	0	0
ω_x max	0	0	0
ω_y min	0	0	0
ω_y max	0	0	0
ω_z min	0	0	0
ω_z max	0	0	0
d_y min	1	0.4	0.8

Table 11. Configurations for second reduction.

Joint	Manipulator 3	Manipulator 4	Manipulator 5
A2 Roll			
A2 Pitch			
A1 Roll			
A1 Pitch	−90°	0°	
B2 Roll			45°
B2 Pitch			
B1 Roll		90°	180°
B1 Pitch	−90°	−90°	−90°
C2 Roll	45°	22.5°	45°
C2 Pitch	−90°		
C1 Roll	180°	180°	180°
C1 Pitch	90°	90°	90°

Similar to the previous test, the three manipulators each had a joint trajectory that spans their entire workspace plotted and evaluated, and their points sorted based on performance. These results are graphed in Figures 9–11.

The results mirror the expected characteristics, with one minor exception: Manipulator 3 ended up having the best linear twist performance. This is likely due to two reasons. First, the individual performance outcomes

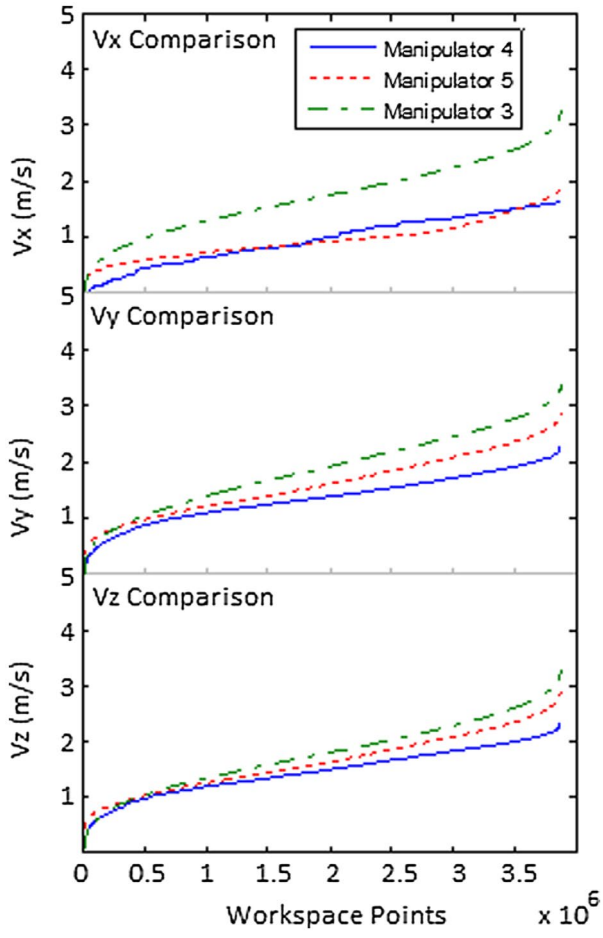

Figure 9. Linaer velocity comparison for manipulators 3–5.

are not completely independent. Therefore, it is possible that, while the optimization was guided to be looking for a better performance in compliance, due to the nonlinearity of the search space, it stumbled upon a local maximum for twist in the meantime. Secondly, because it does not have maximum twist performance weights, the minimum weights are comparatively more emphasized. Its improved performance could be partly due to better minimum avoidance.

With the exception of this one anomaly, the other expected characteristics are all upheld. All three manipulators have full range of motion in SE(3), they all have roughly comparable linear twist, and Manipulator 3 has significantly lower compliance in the y direction, with Manipulator 4 having slightly worse compliance compared to Manipulator 5. What is interesting to note is, the manipulator with the largest increase in twist performance is Manipulator 3, which had looser constraints compared to Manipulator's 4 and 5, which have almost identical twist capabilities compared to each other. It is reasonable to conclude that one can over-constrain a manipulator by specifying too many design weights.

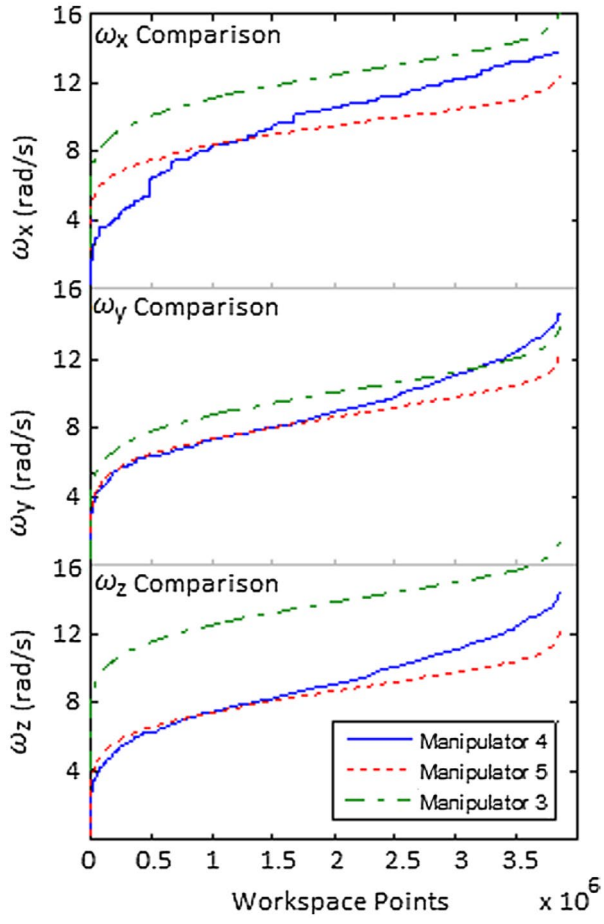

Figure 10. Rotational velocity comparison for manipulators 3–5.

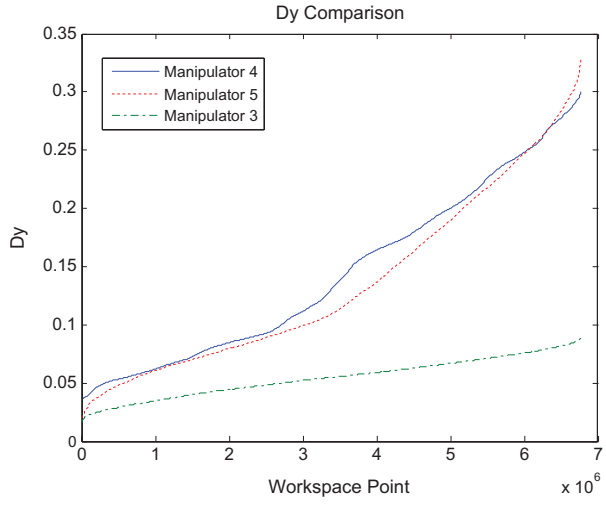

Figure 11. d_y comparison for manipulators 3–5.

4.3. Comparing 6-DOF to 3-DOF plus wrist

The next manipulator generated is another 3-DOF manipulator with spherical wrist which can be compared with Manipulator 2, a 6-DOF generated manipulator. Table 12 outlines the task-group weights for the new manipulator

Table 12. Task-group weights for third reduction.

Weight	Manipulator 6	Manipulator 2
v_x min	−0.9	−0.9
v_x max	0.5	0.6
v_y min	−0.9	−0.9
v_y max	0.5	0.6
v_z min	−0.9	−0.9
v_z max	0.5	0.6
ω_x min	0	−1
ω_x max	0	0.25
ω_y min	0	−1
ω_y max	0	0.25
ω_z min	0	−1
ω_z max	0	0.25

Table 13. Configurations for third reduction.

Joint	Manipulator 6	Manipulator 2
A2 Roll		−180°
A2 Pitch		
A1 Roll		
A1 Pitch		45°
B2 Roll		
B2 Pitch	0°	
B1 Roll	22.5°	
B1 Pitch		
C2 Roll	45°	−180°
C2 Pitch	0°	−90°
C1 Roll	180°	180°
C1 Pitch	90°	90°

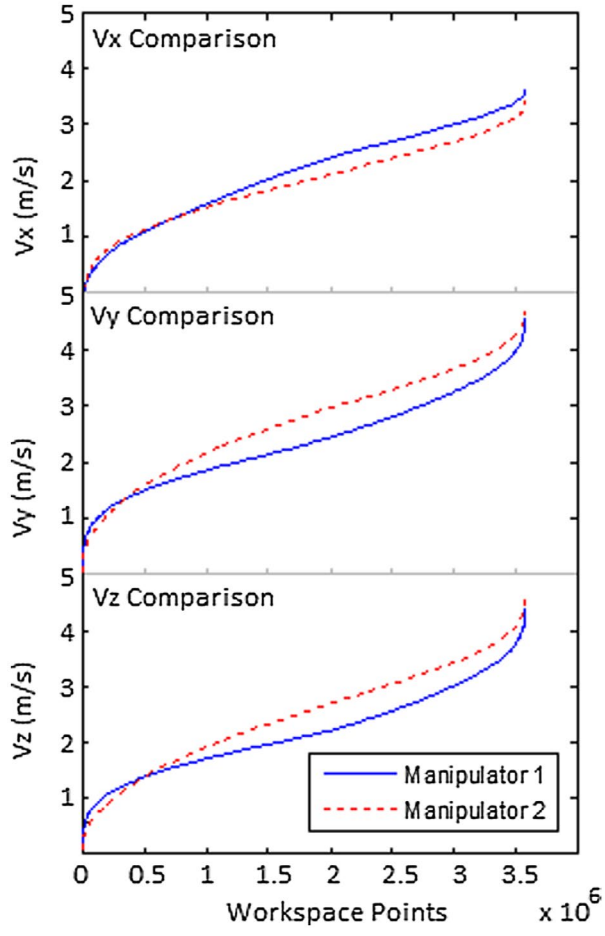

Figure 12. Linear twist comparison for manipulators 6 and 2.

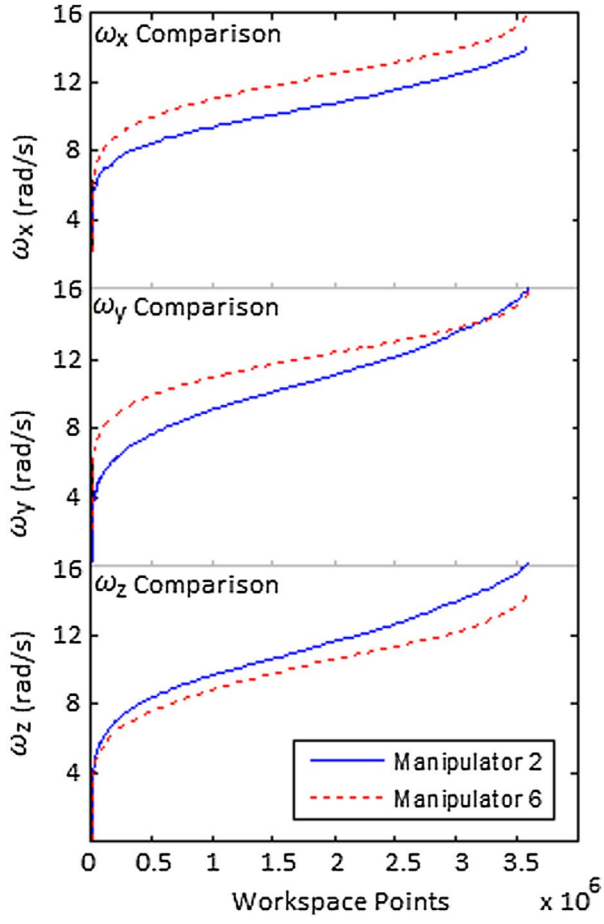

Figure 13. Rotational twist comparison for manipulators 6 and 2.

(Manipulator 6) as well as reiterating the weights for Manipulator 2 for easy comparison. For this comparison, both manipulators have no weights associated with compliance. Further, because Manipulator 6 has no rotational twist weights, it was given slightly reduced linear twist weights to be more comparable.

The final manipulator configurations are shown in Table 13.

The results of the configurations are graphed in Figures 12 and 13 below.

Based on these results, it is clear that Manipulator 2 has superior linear twist, while Manipulator 6 has superior rotational twist. This would suggest that by imposing a spherical wrist constraint on the optimization, it guarantees the resulting manipulator will have better-than-most rotational twist capabilities, as that is the primary feature of a spherical wrist, but at the cost of reduced capabilities in other areas. This is primarily due to the fact that the reduction has fewer choices in terms of which joints to leave active and which joints to lock.

From these results, it is clear that the reduction phase does produce manipulator configurations that correspond to the task-group weights defined by the user. The complete reduction phase took approximately 11–13 h, running on a single PC of the specifications listed in at the beginning of Section 4. Most of the time was consumed by choosing the first joint, taking on average 9 h. Choosing the 4th joint took on average 9 min, and both of the final two joints required less than 1 minute. If the user, therefore, provided the first two joints to be locked (and the position in which to lock them), the complete reduction procedure could be completed in under an hour. Comparing these times to a brute force method, it takes the same PC 1 second to evaluate 400 points in the configuration manifold. There are 8.3×109 possible configurations to evaluate, given the search threshold used. The manipulators designed above had between 3 and 12 task-group weights each, which translates into a brute force method taking between 720 and 2882 days to complete, although it would almost certainly produce solutions of slightly superior quality.

5. Conclusions

The reconfiguration approach utilizes a 12-dimensional configuration manifold representing the 12 revolute actuators of the MARS Manipulator. This manifold is populated with sets of points representing positive and negative performance attributes of the manipulator. These points are clustered into hyperellipses which are given weights by the user that guide the optimization in systematically reducing the dimension of the configuration manifold. As the dimension is reduced, the number of DOFs of the manipulator is reduced. The reduction is performed by checking each dimension for an optimal reduction point, and then comparing each optimal point to each other to determine the dimension with the highest value for reduction. This process is repeated until the manifold is of the dimension specified by the user.

The application of the configuration optimization produced 6 manipulator configurations. Each configuration was developed through different weights in the regions of end-effector twist and manipulator compliance. The method showed that the manipulators were optimized to have certain characteristic as specified by the user, and this optimization was done is a fraction of the time that traditional methods would require. The main limitation of this method is that there is some skill required by the designer in specifying the user-defined weights. It was shown that if a optimization was over-constrained, it could yield inferior results to a more sparsely constrained optimization.

Disclosure statement

No potential conflict of interest was reported by the authors.

Notes on contributors

Jason Kereluk received his BASc degree in Mechanical Engineering in 2008 from the University of Victoria, and his PhD degree in the Aerospace Mechatronics group at the University of Toronto Institute for Aerospace Studies in 2016. His research interest is in the field of reconfigurable manipulator systems and their optimal configuration selection.

Reza Emami is a professor and chair of Onboard Space Systems at the Luleå University of Technology. He has also been the director of the Aerospace Mechatronics Group and coordinator of the Aerospace and Design Laboratories at the University of Toronto Institute for Aerospace Studies (UTIAS) in Canada since 2001. He obtained his PhD in Robotics and Mechatronics from the University of Toronto, and worked in the industry as a Project Manager before joining the academia. His research focuses on concurrent engineering of multidisciplinary systems, such as spacecraft and robotic manipulators and rovers. His current research includes: concurrent base-arm control of free-flying space manipulators, satellite formation flying, social learning in heterogeneous rover/satellite teams, asteroid redirection for exploration and mining, reconfigurable manipulators and legged locomotion for exploratory rovers. Emami is the Associate Editor of the International Journal of Advanced Robotic Systems. He has been nominated for several teaching awards, including National Innovation for Technology Award and OPAS Award for Excellence in Teaching. He was also recognized by MathWorks as a Featured Professor in 2013 and 2003, and was in the list of University of Toronto's Miracle Workers in 2000.

References

[1] Yim M, Roufas K, Duff D, et al. Modular reconfigurable robots in space applications. Auton Rob. 2003;14(2–3):225–237.

[2] Shen YM, Salemi WM, Rus B, et al. Modular self-reconfigurable robot systems. Rob Autom IEEE. 2007 Mar;14(1):43–52.

[3] Murata S, Kurokawa H. Self-reconfigurable robots. Rob Autom IEEE. 2007 Mar;14(1):71–78.

[4] Suh J, Homans S, Yim M. Telecubes: mechanical design of a module for self-reconfigurable robotics. Rob Autom IEEE. 2002 May;4:4095–4101.

[5] Rus D, Vona M. Crystalline robots: self-reconfiguration with compressible unit modules. Auton Rob. 2001 Jan;10(1):107–124.

[6] Brandt D, Christensen D. A new meta-module for controlling large sheets of atron modules. International Conference on Intelligent Robots and Systems, IEEE/RSJ; 2007 Nov. p. 2375–2380.

[7] Paredis C, Brown H, Khosla P. A rapidly deployable manipulator system. IEEE International Conference on Robotics and Automation; 1996. p. 1434–1439.

[8] Kelmar L, Khosla K. Automatic generation of kinematics for a reconfigurable modular manipulator system. IEEE International Conference on Robotics and Automation; 1988. p. 663–668.

[9] Chen W, Lang G, Ho E, et al. Interactive-motion control of modular reconfigurable manipulators. IEEE International Conference on Robotics and Automation; 2003. p. 1620–1625.

[10] Chen I, Yang G. Inverse kinematics for modular reconfigurable robots. IEEE International Conference on Robotics and Automation; 1998. p. 1647–1652.

[11] Bi Z, Lin Y, Zhang W. The general architecture of adaptive robotic systems for manufacturing applications. Robot Cim-Int Manuf. 2010;26:461–470.

[12] Paredis C, Khosla P. Synthesis methodology for task based reconfiguration of modular manipulator systems. Proceedings of the International Symposium on Robotics Research; 1993 Oct 2–5; Hidden Valley (PA).

[13] Paredis CJ, Khosla PK. Kinematic design of serial link manipulators from task specifications. Int J Rob Res. 1993;12(3):274–287.

[14] Chen I, Burdick J. Determining task optimal modular robot assembly configurations. IEEE International Conference on Robotics and Automation; 1995. p. 132–137.

[15] Bi ZM, Zhang WJ. Concurrent optimal design of modular robotic configuration. J Rob Syst. 2001;18:77–87.

[16] Chhabra R. Concurrent design of reconfigurable robots using a robotic hardware-in-the-loop simulation [master's thesis]. Toronto: University of Toronto Institute for Aerospace Studies; 2008.

[17] Chhabra R, Emami MR. Linguistic mechatronics. IEEE/ASME International Conference on Advanced Intelligent Mechatronics, AIM; 2008 Jul 2–5. p. 1315–1320.

[18] Kereluk J, Emami MR. A new modular, autonomously reconfigurable manipulator platform. IJARS. 2015;12(71).

[19] Kereluk J, Emami MR. A remotely-accessible reconfigurable platform for robotics education. 120th ASEE Annual Conference & Exposition; 2013.

[20] Craig JJ. Introduction to robotics – mechanics and control. 3rd ed. Upper Saddle River (NJ): Pearson Prentice Hall; 2005.

[21] LaValle S. Planning algorithms. New York (NY): Cambridge University Press; 2006.

[22] Belta C, Kumar R. Euclidean metrics for motion generation on *SE*(3). Proc Inst Mech Eng. 2002;216(1):47–60.

[23] Lertchoosakul P, Nair R. Distribution functions for subsequences of the van der Corput sequence. Indagationes Math. 2013;24(3):593–601.

[24] van der Corput JG. Verteilungsfunktionen I. Akademie van Wetenschappen. 1935;38:813–821.

[25] Zhang T, et al. BIRCH: A new data clustering algorithm and its applications. Data Min Knowl Discov. 1997;1(2):141–182.

[26] Zhang T, Ramakrishnan R, Livny M. BIRCH: an efficient data clustering method for very large databases. Int Conf Manag Data. 1996;25(2):103–113.

[27] Kereluk JA. A new approach to robot manipulation [Ph.D. dissertation, Section 6.2]. Toronto: University of Toronto; 2016. p. 69–71.