

Towards Constraint-based Actuation Allocation for Hyper-redundant Manipulators*

Markus P.J. Fromherz, Maia Hoeberechts, Warren B. Jackson

Xerox PARC, 3333 Coyote Hill Road, Palo Alto, CA 94303
{fromherz,mhoebere,wjackson}@parc.xerox.com

October 16, 1999

Abstract

Hyper-redundant robotic manipulators are robots that have many more degrees of freedom than required for a task such as grasping an object in 3D space. The large number of joints, possibly ranging from dozens to thousands, offers both challenges and opportunities for control of such robots. A challenge is to develop algorithms that scale and adapt to different configurations while taking into account a variety of robot constraints. An opportunity is to use the extra degrees of freedom to optimally control multiple objectives. In this paper, we propose to use a control approach based on constrained optimization. We present robot and task models and discuss first results.

1. Introduction

Modular robots are robots consisting of many interchangeable robotic modules connected into a mechanical and functional assembly. Interest in modular robots has increased recently due to their potential for robustness, reduced costs, and wide range of applicability particularly in highly constrained environments [Yim 94]. Typically, there are only a small number of module types, and each module type only has limited motion capability, e.g., for the case considered in this paper only one rotational joint. The flexibility of modular robots is achieved through the large number of modules, expected to range from dozens to thousands, and their many possible configurations. Low cost can be achieved through the mass fabrication of the simpler mechanical components. Sensory information and computational resources as well as mechanical actuation are shared between modules. Because of the large number of degrees of freedom of the redundant modular limbs, such modular robots are also called hyper-redundant. This paper concerns the control of a subclass of such robots, namely hyper-redundant manipulators that consist of a chain of modules with rotational joints (Fig. 1).

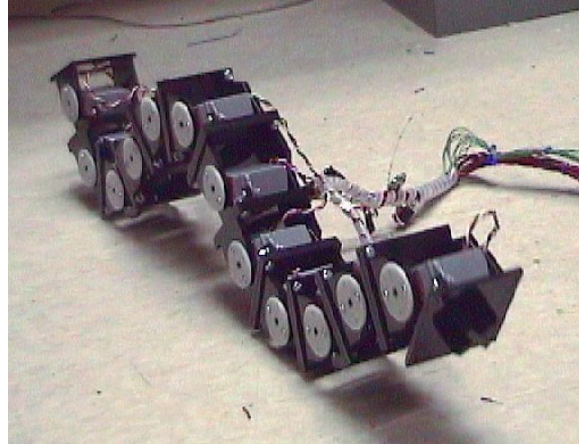


Fig. 1 Modular robot prototype PolyBot (12 joints)

In order to realize the potential of modular robots, robust, versatile and scaleable control algorithms must be developed while satisfying a number of changing joint and force constraints. A control approach should not only handle the many degrees of freedom with limited computational resources in real time, but should also attempt to utilize the redundancy in an optimal way. For example, the robot may not only be asked to achieve typical goals such as reaching a goal position and orientation with its end effector, but also to minimize velocities, distribute torques equally, and avoid complex obstacles. At the same time, each module has to obey its physical limits, such as limits on joint angles and motor torques. Finally, in order to simplify programming different configurations of modular robots, the software should also be generic with respect to constraints and goals.

In this work, we attempt to robustly handle a variety of constraints for the many degrees of freedom found in modular robots. We cast the control problem as a constrained optimization problem. The constraints represent the (mainly physical) limits of the robot, while the objective function expresses the various goals of interest. Furthermore, we assume that end-effector goals (e.g., position and orientation) are given by a higher-level controller such as a

* Presented at CP'99 Workshop on Constraints in Control (CC'99).

path planner, and we focus on the problem of *actuation allocation*, the problem to finding optimal solutions for the joint angles at each time step.

In the balance of this paper, we review related work, present our robot and task models, and discuss our initial implementation and first results.

2. Related Work

Previous work on the control of redundant manipulators has focused on their kinematics, involving the computation and inversion of the Jacobian mapping from joint space to the end-effector space [Khatib 87, Ghosal & Roth 88, Chiu 88]. These approaches typically include few or no constraints and have been applied to robots with a small number of degrees of freedom (less than 10). Attempts have been made to include more constraints by extending Jacobian methods with projected gradients of the constraints [Baillieul 86] or numerically computed gradients [Klein et al. 95]. These methods can work reasonably well for systems with a small number of joints and a few well-characterized constraints. However, modular robot control problems tend to have too many degrees of freedom and mixed task space, joint space, and dynamic inequality constraints to solve such systems in real time. Also, because these methods tend to require a detailed constraint analysis, it is difficult to robustly adjust the priorities of different constraints over time as robot environment and pose change. Moreover, since the solutions are computed in velocity space, position errors can build up over time.

In order to deal with constraints, certain kinds of dynamic constraints can be included in an unconstrained form using higher-order differential forms [Agrawal & Faiz 98] which can then be used to elegantly provide optimal motion planning and control [Gokce & Agrawal 99]. However, mixed task space and joint space inequality constraints do not map easily into these higher-order forms.

Control of hyper-redundant robots using continuous backbone curves has been well studied [Chirikjian & Burdick 94, Chirikjian & Burdick 95]. This approach scales very well to large numbers of modules and is of particular interest in hierarchical control approaches. Again, this approach has only been studied with kinematic constraints.

Taking into account constraints such as torque limits is particularly relevant for modular robots. Traditional robots are designed such that their joint torque limits are never exceeded (e.g., by making the base joints strong and the end joints light-

weight). In a modular robot, all modules are similar and often relatively weak.

In summary, in contrast to traditional approaches to manipulator control, we attempt to separate constraints, goals, and control algorithms from each other, and to include a larger class of constraints. So far, we have focused on finding effective constraints and objective functions. We assume a generic nonlinear constraint solver and leave the development of a dedicated solver for real-time control to future work.

3. Robot and Task Models

In this section, we first present a model for our manipulator and then show how constraints and task objectives can be expressed in terms of this model.

3.1 Manipulator Model

For the purpose of this paper, we focus on a robotic manipulator that consists of a chain of n links. This manipulator is attached to a base at one end (link 1), and the primary task is to reach a goal position, or follow a goal trajectory, with the other end (link n). Such a manipulator may, for example, be an arm (which is attached to a table or robot body and whose “hand” end grasps or pushes other objects) or a leg (which is attached to a robot body and whose “foot” end is moving over the ground during locomotion). Note that for now we consider only statically stable configurations, and, due to the relatively slow movement of the joints, we are not concerned with dynamics.

Each manipulator link i has its own local coordinate system, or *frame*, with unit vectors \hat{X}_i , \hat{Y}_i , and \hat{Z}_i defining the three axes. In addition, we have a base frame or frame 0, e.g., a table or body, to which the first link is attached, as well as the global frame, the world coordinate system. Following robotics conventions [Craig 89], the kinematics of each link is defined by its Denavit-Hartenberg parameters $\langle \alpha_i, a_i, \theta_i, d_i \rangle$, where (cf. Fig. 2)

- twist α_i is the angle between \hat{Z}_{i-1} and \hat{Z}_i measured about \hat{X}_{i-1} ,
- length a_i is the distance from \hat{Z}_i to \hat{Z}_{i+1} measured along \hat{X}_i ,
- rotation θ_i is the angle between \hat{X}_{i-1} and \hat{X}_i measured about \hat{Z}_i , and
- extension d_i is the distance from \hat{X}_{i-1} to \hat{X}_i measured along \hat{Z}_i .

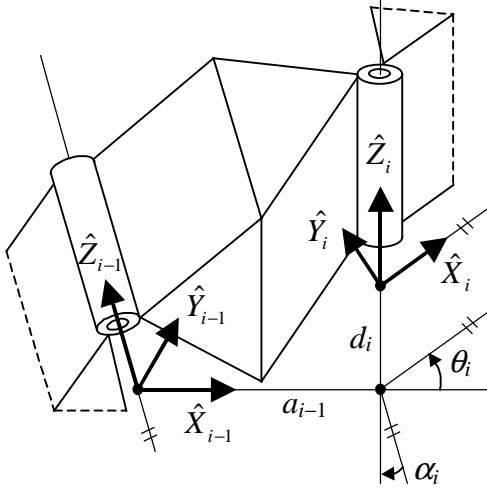


Fig. 2 Parameters for coordinate transformation from frame $i-1$ to frame i

θ_i is taken variable for revolute link joints, and d is taken variable for prismatic link joints. In this paper, we restrict ourselves to revolute joints, and d_i is always 0. Also, the origin o_i of frame i is located in the joint of link i .

These parameters define a link's frame with respect to its predecessor's frame in the chain of links. More precisely, these parameters define a *homogeneous transform matrix* ${}^{i-1}T_i$, a rotation plus a translation, that maps a point ${}^i p$ defined in frame i to point ${}^{i-1} p$ defined in frame $i-1$ by the multiplication ${}^{i-1} p = {}^{i-1}T_i {}^i p$ [Craig 89]. (Here, a point $(x \ y \ z)$ is represented by its homogeneous coordinates $p = [x \ y \ z \ 1]^T$, with the superscript "T" indicating the transpose. A point p or vector r is defined in the global frame, unless a prefix superscript identifying the local frame is given, as in ${}^i p$ and ${}^i r$.) The transform matrix is defined as

$${}^{i-1}T_i = \begin{bmatrix} {}^{i-1}R & {}^{i-1}R {}^{i-1}o_i \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos \theta_i & -\sin \theta_i & 0 & a_{i-1} \\ \sin \theta_i \cos \alpha_i & \cos \theta_i \cos \alpha_i & -\sin \alpha_i & -\sin \alpha_i d_i \\ \sin \theta_i \sin \alpha_i & \cos \theta_i \sin \alpha_i & \cos \alpha_i & \cos \alpha_i d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1)$$

where ${}^{i-1}R$ is the rotation matrix from frame $i-1$ to i , and ${}^{i-1}o_i$ is the origin of frame i given in (the coordinate system of) frame $i-1$. The three columns of the rotation matrix define the *unit vectors* \hat{X}_i , \hat{Y}_i and \hat{Z}_i of frame i in frame $i-1$. We will also use the functional denotation ${}^{i-1}T_i = T(\alpha_i, a_{i-1}, \theta_i, d_i)$ to describe the matrix of Eq. (1).

Transform matrices can be composed by multiplying them. In particular, the transformation from base frame 0 to frame i is given by

$${}^0T_i = {}^0T_1 {}^1T_2 \dots {}^{i-1}T_i = {}^0T_i {}^{i-1}T_i$$

For example, a point ${}^i p$ defined in frame i , i.e., in the local coordinate system of link i , has position ${}^0 p = {}^0T_i {}^i p$ in the base coordinate system. As another example, the origin of frame i is given in the base frame by ${}^0 o_i = {}^0T_i [0001]^T$. In this paper, the base frame is always identical to the global frame, i.e., ${}^0 p = p$.

In addition to the kinematics parameters, each link has the following parameters:

- link mass m_i ,
- center of mass position ${}^i c_i$ (in the link's frame),
- joint angle limits $\theta_{\min,i}$ and $\theta_{\max,i}$,
- maximum rotational velocity $\omega_{\max,i}$ achievable by the joint, and
- maximum motor torque $\tau_{\max,i}$.

Angles are expressed in radians, times in seconds, distances in meters, masses in kilograms, and torques in Newton-meters.

3.2 Manipulator Constraints

The manipulator's only actuator variables are the joint angles $\theta_1, \dots, \theta_n$, and thus all constraints ultimately have to be expressed as constraints on these angles. Given the links' parameters, we take into account up to three types of constraint: joint angle limits, torque limits, and velocity limits.

Angle limits. The first type of constraint states that a joint's movement is restricted by the mechanics of the link in either direction of the rotation and is defined trivially as

$$\theta_{\min,i} \leq \theta_i \leq \theta_{\max,i} \quad \text{for all } i = 1, \dots, n$$

Torque limits. Each link has a mass, and for each joint i , the links from i through n together exert a torque onto joint i . This torque is limited by the strength of the joint, and the link will break or at least not be able to hold the subsequent links up if that limit is exceeded. As noted before, this problem has traditionally not been addressed in manipulator control, and related work has been restricted to 2D manipulators [Agrawal & Garimella 94, Gokce & Agrawal 99].

Since a joint with its driving motor are typically the weakest component of a link, we are interested in the motor torque. Its magnitude is determined pri-

marily by the gravitational force (applied at the center of gravity of links i through n) and the moment arm (the vector from the joint axis to where the force is applied). As a vector, this torque is both perpendicular to the moment arm and tangential to the rotation (Fig. 3). The torque τ_i on joint i can be computed as

$$\tau_i = \hat{Z}_i \bullet (\mathbf{R}_i \times \mathbf{F}_i)$$

where \hat{Z}_i is the axis of rotation (cf. Fig. 2), \mathbf{R}_i is the vector from joint i to the center of mass C_i of links i through n , and \mathbf{F}_i is the gravitational force on links i through n . (Also, \bullet is the vector dot-product and \times is the vector cross-product.) The magnitude of \mathbf{F}_i is $F_i = \sum_{j=i}^n m_j g$ (where g is the gravitational constant), and \mathbf{F}_i is directed vertically down, i.e., $\mathbf{F}_i = [0 \ 0 \ -F_i]^T$.

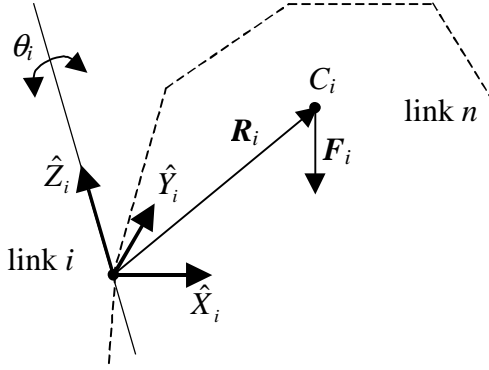


Fig. 3 Center of gravity C_i , gravity force \mathbf{F}_i , and moment arm \mathbf{R}_i for links i through n

The center of mass C_i of links i through n is the weighted average of the centers of mass of each link relative to the origin of link i , and thus \mathbf{R}_i can be computed as follows.

$$\mathbf{R}_i = \frac{\sum_{j=i}^n m_j (c_j - o_i)}{\sum_{j=i}^n m_j}$$

Recall that $c_i = {}^0T^i c_i$ is the center of mass of link i and o_i is the link's origin (both in the base frame). Note that τ_i and \mathbf{R}_i depend on all the joint angles, which we indicate by writing $\tau_i(\theta)$. The nonlinear torque constraint is then defined as

$$\tau_i(\theta) \leq \tau_{\max,i} \text{ for all } i = 1, \dots, n$$

Velocity limits. If used in a control scenario where the movement from a previous position is taken into account, we may at a minimum take into ac-

count the maximum velocity achievable by each joint, i.e., $|\omega_i| \leq \omega_{\max,i}$ for all i . We use a linear approximation for joint velocity, i.e., $\omega_i = (\theta'_i - \theta_i)/dt$, where dt is the time step for control and θ'_i is the previous angle value for joint i . Thus, this constraint is defined as

$$\theta'_i - \omega_{\max,i} dt \leq \theta_i \leq \theta'_i + \omega_{\max,i} dt$$

for all $i = 1, \dots, n$

3.3 Task Model

For the purpose of this paper, the main goal of the robot manipulator is to reach a target position with its end link. There may be secondary goals, however, such as also achieving a target orientation with the end link, minimizing the difference from a previous position, and distributing actuation equally. At the same time, the manipulator has to satisfy its constraints, and it may not always be able to achieve its goals exactly. Thus, we define the actuation allocation task as the constrained optimization problem

$$\begin{aligned} & \text{minimize} && h(\theta) \\ & \text{subject to} && \mathbf{c}(\theta) \end{aligned}$$

where $\theta = \{\theta_1, \dots, \theta_n\}$ are the free variables. The constraints $\mathbf{c}(\theta)$ are those presented above. The objective function is presented in the balance of this section.

Goals we have implemented include a position goal, an orientation goal, minimal actuation energy, and actuation distribution. The total objective function is a weighted, normalized sum of the individual goal functions $h_i(\theta)$:

$$h(\theta) = \frac{\sum_i w_i \frac{h_i(\theta)}{q_i}}{\sum_i w_i}$$

The individual goal functions are normalized by the factors q_i to be in the range $[0,1]$. The weights w_i determine the relative priority of the goals. By setting weights to 0, individual goals can be turned off. Finally, the entire sum is divided by the sum of the weights to be in the range $[0,1]$.

We now present the individual goals.

Position goal. Given a goal position $p_g = [x_g \ y_g \ z_g \ 1]^T$ for the end effector (the end of link n), the goal function is

$$h_1(\theta) = \sqrt{(x_g - x_e)^2 + (y_g - y_e)^2 + (z_g - z_e)^2}$$

where $p_e = [x_e \ y_e \ z_e \ 1]^T = {}^0T_n {}^nT_e [0\ 0\ 0\ 1]^T$ is the position of the end effector, with ${}^nT = T(0, a_n, 0, 0)$ the transform matrix from the origin of link n to its end (a translation by a_n along \hat{X}_n ; cf. Eq. (1)).

Our normalization factor for this goal is the sum of the distance from base origin to goal position and the maximum reach of the manipulator:

$$q_1 = \sqrt{x_g^2 + y_g^2 + z_g^2} + \sum_{i=1}^n a_i$$

Orientation goal. We currently allow one to define a goal twist α_g and rotation θ_g for the end effector. These angles define a frame ${}^gT = T(\alpha_g, 0, \theta_g, 0)$

(cf. Eq. (1)). Recall that ${}^0T = {}^0T_n {}^nT$ is the frame of the end effector. Also recall that each transform matrix 0T and eT contains a rotation matrix gR and eR . Thus, ${}^eR = {}^eR^T {}^gR$ is the rotation from end-effector frame to goal frame. Based on Euler's theorem on rotation [Craig 89], this rotation can be represented with a single axis and a single rotation angle ${}^e\Theta$, and the angle can be computed from the rotation matrix as

$${}^e\Theta = \arccos\left(\frac{\text{Tr}({}^eR) - 1}{2}\right) \quad (2)$$

where $\text{Tr}({}^eR) = \sum_{i=1}^3 r_{ii}$ is the trace of eR (r_{ij} being the elements of eR). We assume that the arccosine is defined in the interval $[-1, 1]$, such that Eq. (2) always returns an angle in the interval $[0, \pi]$. (If the argument of the arccosine is 1 then ${}^e\Theta = 0$, and if the argument is -1 then ${}^e\Theta = \pi$.)

We define the goal function for an orientation goal by this rotation angle:

$$h_2(\theta) = {}^e\Theta$$

Note that since the goal function is defined in terms of the rotation matrices of the two frames, this definition can be used for any definition of orientation.

Since the maximum rotation angle is by definition π , the normalization factor for this goal is

$$q_2 = \pi$$

Minimal actuation energy. This objective, usually secondary to achieving a goal position and/or ori-

entation, is to minimize the difference between a previous solution for the manipulator and the new solution. In other words, we prefer solutions that move the joints as little as possible. This can be expressed by the goal function

$$h_3(\theta) = \sum_{i=1}^n (\theta'_i - \theta_i)^2$$

where θ'_i is the previous angle value for joint i .

Given the limits on the joint angles, the maximum difference between two values of joint angle i is $\theta_{\max,i} - \theta_{\min,i}$. Thus, the normalization factor for this goal is

$$q_3 = \sum_{i=1}^n (\theta_{\max,i} - \theta_{\min,i})^2$$

Actuation distribution. This objective, usually of lowest priority, expresses a preference for solutions that result in an equal distribution of actuation (smooth curves) over solutions that have "kinks" (i.e., where some joints are at their limits while others are hardly actuated). For example, instead of one joint-to-joint rotation by 0 degrees and the next by 90 degrees, we prefer if both have rotations of 45 degrees.

Recall that the rotation from frame $i-1$ to frame i is given by matrix ${}^{i-1}R$. Again, we can determine the Euler rotation angle as in Eq. (2), and by directly substituting the trace of ${}^{i-1}R$ we get

$${}^{i-1}\Theta = \arccos\left(\frac{\cos \theta_i + \cos \theta_i \cos \alpha_i + \cos \alpha_i - 1}{2}\right)$$

As expected, if α_i is 0 or π then ${}^{i-1}\Theta = \theta_i|_{\pi}$, where $\theta_i|_{\pi}$ is θ_i normalized to the interval $[0, \pi]$ (to conform with the assumptions for Eq. (2)). Since α_i can't be changed, one may argue that using ${}^{i-1}\Theta = \theta_i|_{\pi}$ is a sufficient measure, and we'll be using this definition. The goal function is defined as

$$h_4(\theta) = \sum_{i=1}^n ({}^{i-1}\Theta)^2 = \sum_{i=1}^n \theta_i|_{\pi}^2$$

Using the square will prefer distributed allocations. Given a maximum value of π for all $\theta_i|_{\pi}$, the normalization factor for this goal is

$$q_4 = n \pi^2$$

4. Implementation and Evaluation

In order to explore the possibility of using constrained optimization in a control approach for modular robots, a testbed was implemented in Matlab in which a manipulator can be represented according to the model described in the previous section. The testbed allows manipulators with an arbitrary number of links to be entered, although, at present, calculations for more than approximately 15 joints are prohibitively slow. This section describes the testbed implementation and some of the tests and results so far.

4.1 Testbed Implementation

The testbed consists of a set of functions for defining and testing a manipulator, and a graphical user interface for accessing those functions (Fig. 4). The manipulator is represented as a structure whose principal element is an array of links. Each link is itself a structure with the set of parameters described in Section 3.1: link twist α_i , link length a_i , rotation θ_i , extension d_i , link mass m_i , center of mass position ${}^i c_i$, joint angle limits $\theta_{\min,i}$ and $\theta_{\max,i}$, maximum rotational velocity $\omega_{\max,i}$ achievable by the joint, and maximum motor torque $\tau_{\max,i}$. In addition, the transform matrix ${}^{i-1}T_i$ is stored for each link.

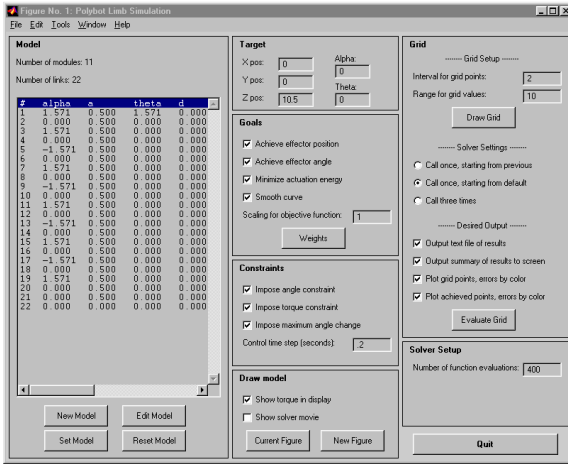


Fig. 4 Testbed interface

We define a default configuration for a manipulator with n links to have $n-1$ joints and a rigid link, the *end-effector*, as the last link in the chain. The first joint is fixed to the base such that its rotation, θ_1 , is in the x-z plane. The subsequent joints are attached at offsets of $\pi/2$ radians (90 degrees) to one another so that their rotations are in orthogonal planes. In terms of link twist, the latter condition can be ex-

pressed as $\alpha_2 = \pi/2$, $\alpha_3 = -\pi/2$, $\alpha_4 = \pi/2$, \dots , $\alpha_i = (-1)^i \pi/2$. The center of mass for each joint, ${}^i c_i$, is set at the origin of frame i , or in other words at the point where the motor would be located for a physical manipulator. We have made the assumption that the mass of the link is negligible when compared with the mass of the motor. Link length a_i , link mass m_i , and maximum motor torque $\tau_{\max,i}$ are set such that a joint can lift approximately 5 other joints without exceeding the maximum motor torque. Joint angle limits $\theta_{\min,i}$ and $\theta_{\max,i}$ are set to -80 degrees and 80 degrees respectively. The results discussed in this paper were achieved using a manipulator composed of 10 joints and an end-effector, and for which the link parameters were defined according to the described defaults (cf. Fig. 5).

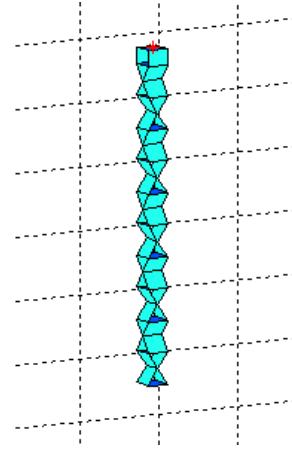


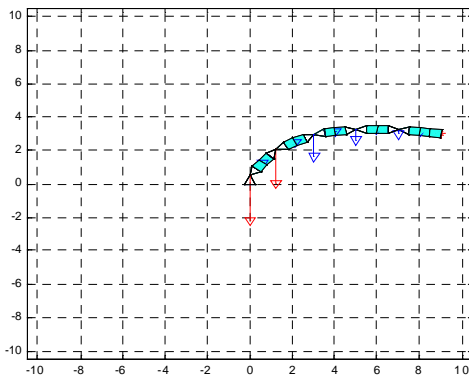
Fig. 5 Rendering of the default manipulator

4.2 Allocation of Joint Angles

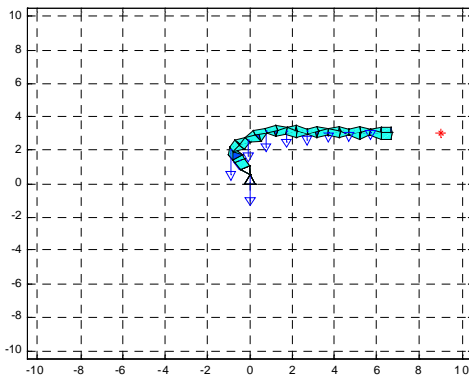
The principal function of the testbed is to allow a user to select a set of goals and constraints for a manipulator, and then run the solver using those settings to find a new set of joint angles. The joint angles which are found should satisfy the constraints and optimize the performance on the selected goals (i.e., minimize the value of the objective function). The solver options which can be set through the graphical user interface are: the target position and orientation, the active goals and their respective weights, active constraints, number of objective function evaluations, and the desired output.

For example, as explained earlier, torque limits on robot joints become an important factor in determining possible positions for the manipulator. By using the testbed to turn torque constraints on and off for a given manipulator and target point, the effect of torque constraints for the type of manipu-

lator we are considering can be seen clearly. For example, in Fig. 6(a) the solver was given the goal position $[9 \ 0 \ 3 \ 1]^T$ with only the angle limit constraint imposed. The target point is shown by an asterisk, and the torque for each joint is indicated by a vertical arrow centered at the joint. The torque arrows are scaled such that the maximum torque is represented by an arrow one unit long. As the diagram shows, in order to achieve the goal position, the maximum motor torques for joints 1 and 3 are exceeded. (In a physical realization, the manipulator arm would drop.) Fig. 6(b) shows the resulting manipulator position when the torque constraints are imposed. In this case, the first few joints are bent in such a way that the manipulator's center of mass has been shifted toward the base in order to counter-balance some of the mass of the outer links, and therefore the torque constraints can now be satisfied. However, as the figure shows, the goal position can no longer be reached.



(a)



(b)

Fig. 6 Solutions for reaching a goal position (a) without and (b) with torque constraints

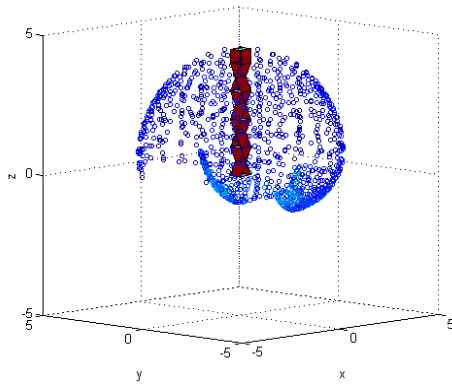
At present we are using the Matlab toolbox function *fmincon* as the constraint solver and optimizer.

This nonlinear optimizer allows us to solve the constraints directly in joint space with a set of current joint angles as starting point. Such solvers numerically compute gradients of the object function, which include the Jacobian augmented by various goal weight function and constraint gradients pointing towards feasible solutions, and even accumulates information on the curvature as the optimization proceeds. A different optimizer, perhaps more specifically suited to the problem we are addressing, could be implemented and integrated with the rest of the software.

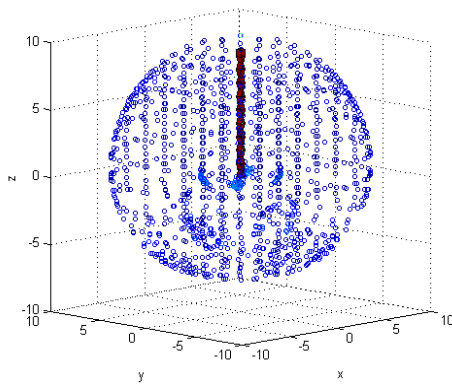
4.3 Workspace Evaluation

The second function of the testbed is to permit a given manipulator and set of options to be evaluated over a large set of points. A direct application is to determine the workspace, the reachable space, for a manipulator under the specified conditions. Traditionally, the workspace of a manipulator would be determined either analytically or by enumerating all possible values for each of the actuation variables (the joint angles in our case). The former is beyond today's methods when considering the number of modules and our range of constraints, while the latter only captures the theoretical workspace as opposed to the workspace achievable with a particular control approach. As an alternative, we propose an experimental approach to determining the workspace, in which, for a given manipulator and solver settings, we select a set (e.g., grid) of goal points in 3D space and run the solver on each of the goal points, recording objective function value and several other measures of the solver's success. As an example, Fig. 7. shows the workspace of the default configuration (for both 5 and 10 joints) in terms of the points reached when evaluating a grid with axis ranges $[-5,5]$ respectively $[-10,10]$. As another example, Fig. 8 shows the workspace of the default configuration without and with torque constraints. (The points indicate actually reached end-effector positions. The darker the color, the closer the reached position is to the goal position.)

This approach can be extended naturally to other workspace evaluation. For example, the dexterous workspace, the workspace reachable with any desired orientation, can be computed by evaluating the 6-dimensional space of goal positions and orientations. Finally, we can also evaluate alternative solvers by comparing how well they achieve the desired goals in the workspace.



(a)



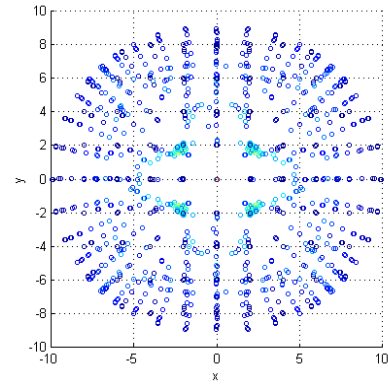
(b)

Fig. 7 Workspaces of a manipulator with (a) 5 joints and (b) 10 joints

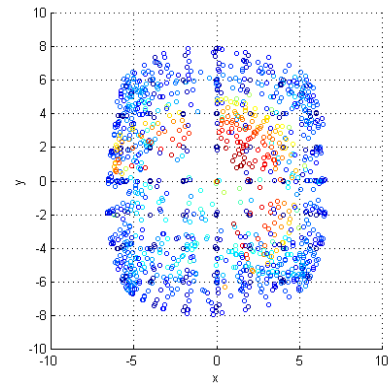
4.4 Scaling

One goal of our investigation was to develop techniques for allocating joint angles which could scale to large numbers of modules (e.g., several hundred joints). As it is currently implemented, the software could handle large sets of joints, although calculation becomes quite slow as the number of joints is increased.

Our current solver is an anytime solver, i.e., it can be stopped after a specified number of evaluations of the objective function and returns the then-best solution. One possibility for improving the speed of the solver is to select an error tolerance for the solution, and to eliminate function calls which do not make a significant change in the error value. For example, for the default configuration, Fig. 9 shows an exponential decay in how the position error improves with increasing number of function evaluations. We expect that results like these will allow us to make trade-offs of efficiency vs. quality in our control approach.



(a)



(b)

Fig. 8 Workspaces (seen from above) of a manipulator with 10 joints (a) without and (b) with torque constraints

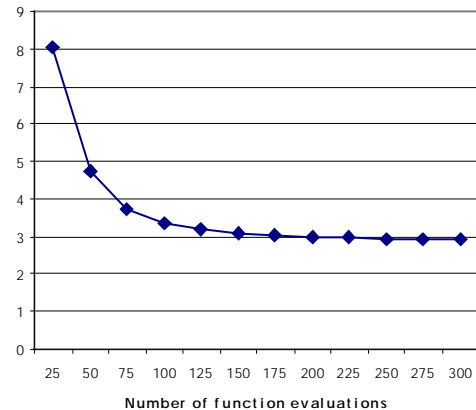


Fig. 9 Mean position error with limits on number of function evaluations

5. Conclusions and Future Work

In this paper, we have presented a parametric model for hyper-redundant, reconfigurable manipulators that can be extended readily with additional constraints on joints and sets of joints without detailed analysis. We have also presented a task model for reaching various objectives such as goal position and actuation distribution, which can be extended to new objectives as well. Together, these models allow one to formulate new constraints and objectives that are of particular interest to hyper-redundant manipulators, such as torque constraints and the equal distribution of actuation. Finally, we discussed a first implementation and evaluation with a standard, nonlinear constrained optimizer.

There are a number of interesting properties for a generic constraint-based representation and control approach. In addition to being extensible, this approach can deal with dynamically changing constraints and goals as well as dynamically changing configurations. We found that it may be possible to meet the real-time requirements by limiting the number of solver iterations, and our first results indicate that this may be possible without trading off solution quality.

A primary problem of our current implementation is that performance is not acceptable for large numbers of links. Currently, all calculations for constraints and objective function are linear in the number of links, but this may not be true for different constraints (e.g., constraints for obstacle avoidance). Even linear scaling may not be acceptable if the number of modules ranges into the thousands. In order to address this issue, we plan to investigate hierarchical and distributed representations and solver implementations. For example, a higher-level controller can determine allocations at a more abstract level, e.g., using splines, and then provide its solutions as goals for low-level optimization.

We also plan to investigate the use of actuation allocation in the context of path control, where a path planner determines a sequence of goal via points. In both hierarchical allocation and path control, high-level optimization solutions will serve as goals for low-level optimization problems, and errors from low-level solutions will adjust goals of higher-level optimization problems. Thus, although the approach presented in this paper may occasionally find the inherent singularities and local minima, we expect that this will be compensated for by the ability to adjust the goal and constraint weights at higher levels of control.

6. References

- [Agrawal & Faiz 98]
S. K. Agrawal and N. Faiz, "Optimization of a class of nonlinear dynamic systems: new efficient method without Lagrange multipliers." In: *Journal of Optimization Theory and Application*, vol. 97, no. 1, April 1998, pp. 11-28.
- [Agrawal & Garimella 94]
S. K. Agrawal and R. Garimella, "Workspace boundaries of free-floating open and closed chain planar manipulators." In: *Journal of Mechanical Design*, vol. 116, March 1994, pp. 105-110.
- [Baillieul 86]
J. Baillieul, "Avoiding obstacles and resolving kinematic redundancy." In: *Proc. 1986 IEEE Int. Conf. on Robotics and Automation*, 1986, pp. 1698-1704.
- [Chirikjian & Burdick 94]
G. S. Chirikjian and J. W. Burdick, "A modal approach to hyper-redundant manipulator kinematics." In: *IEEE Trans. On Robotics and Automation*, 10, 1994, pp. 343-354.
- [Chirikjian & Burdick 95]
G. S. Chirikjian and J. W. Burdick, "Kinematically optimal hyper-redundant manipulator configurations." In: *IEEE Trans. On Robotics and Automation*, vol. 11, 1995, pp. 794-806.
- [Chiu 88]
S. L. Chiu, "Task compatibility of manipulator postures." In: *Int. Journal of Robotics Research*, vol. 7, no. 5, 1988, pp. 13-21.
- [Craig 89]
J. J. Craig, *Introduction to Robotics Mechanics and Control*. Addison Wesley, 1989.
- [Ghosal & Roth 88]
A. Ghosal and B. Roth, "A new approach for kinematic resolution of redundancy." In: *Int. Journal of Robotics Research*, vol. 7, no. 2, March/April 1988, pp. 22-35.
- [Gokce & Agrawal 99]
A. Gokce and S. K. Agrawal, "Mass center of planar mechanism using auxiliary parallelograms." In: *Trans. of the ASME*, vol. 121, March 1999, pp. 166-168.
- [Khatib 87]
O. Khatib, "A unified approach to motion and force control in robotic manipulators: The operational space formulation." In: *IEEE Journal on Robotics and Automation*, vol. 3, no. 1, 1987, pp. 43-53.
- [Klein et al. 95]
C. A. Klein, C. Chu-Jenq, and S. Ahmed, "A new formulation of the extended Jacobian method and its use in mapping algorithmic singularities for kinematically redundant manipulators. In: *IEEE Trans. On Robotics and Automation*, vol. 11, 1995, pp. 50-55.
- [Yim 94]
M. Yim, *Locomotion with a Unit-Modular Reconfigurable Robot*. Ph.D. Thesis, Dept. of Mechanical Engineering, Stanford University, 1994.