Kinematically Optimal Hyper-Redundant Manipulator Configurations

Gregory S. Chirikjian, *Member, IEEE,* and Joel W. Burdick

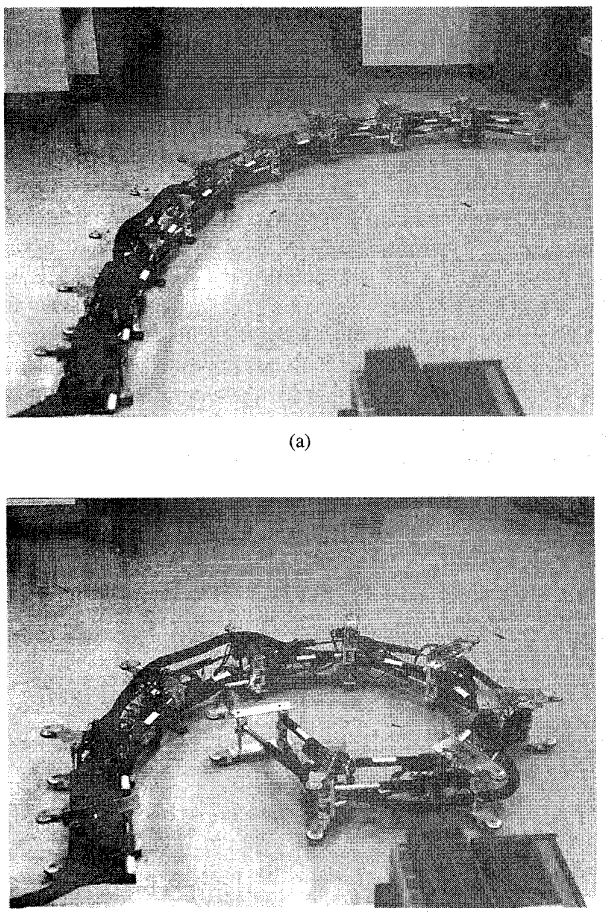
***Abstract*—"Hyper-redundant" robots have a very large or infinite degree of kinematic redundancy. This paper develops new methods for determining "optimal" hyper-redundant manipulator configurations based on a continuum formulation of kinematics. This formulation uses a backbone curve model to capture the robot's essential macroscopic geometric features. The calculus of variations is used to develop differential equations, whose solution is the optimal backbone curve shape. We show that this approach is computationally efficient on a single processor, and generates solutions in O(1) time for an N degree-of-freedom manipulator when implemented in parallel on O(N) processors. For this reason, it is better suited to hyper-redundant robots than other redundancy resolution methods. Furthermore, this approach is useful for many hyper-redundant mechanical morphologies which are not handled by known methods.**

I. Introduction

"HYPER-REDUNDANT" manipulators have a very large relative degree of kinematic redundancy. These robots are analogous in design and operation to "snakes" or "tentacles." For example, Fig. 1 shows a photograph of a 30 degree-of-freedom robot which has been constructed by the authors and to which the theory in this paper has been applied. This paper addresses the issue of computing the inverse kinematics of such highly redundant robots. We term this problem "hyper-redundancy resolution."

Traditionally, kinematic analysis and motion planning for redundant manipulators has relied upon a pseudo-inverse [17], generalized inverse [20], or extended inverse [1] of the manipulator Jacobian matrix. Some have considered schemes to find redundant manipulator trajectories which optimize joint rates or torques over a whole trajectory instead of point-to-point, e.g., [25] and [22]. Other inverse kinematics schemes have been developed based on the concept of dynamic isotropy [18].

The principal contribution of this paper is a new and computationally efficient method for determining the kinematically optimal configurations of hyper-redundant robots. This method has a number of advantages over other possible techniques. First, it is applicable to a wide variety of hyper-redundant robot mechanical morphologies which are not handled by known redundancy resolution techniques. For example, it is useful for devices driven by distributed actuators or constructed from pneumatic tube bundles [23], [26], where it is difficult or impossible to define a Jacobian matrix. Second, this method generates cyclic joint trajectories for a given cyclic end-effector trajectory. Third, it is computationally efficient for a large number of degrees of freedom. We show that for computation on a serial processor, the computational burden of the method grows as O(N), where N is the number of actuated joints. While competing method for configuration optimization based on a Jacobian can also have computational complexity that grows as O(N) (see Section II), our method can be implemented using a very simple parallel processing scheme, and so the computational burden of this new method can be reduced to O(1) in time if distributed over O(N)



(a)

(b)

Fig. 1. Photograph of 30 DOF hyper-redundant robot.

Manuscript received August 4, 1994; revised January 7, 1995. This work was supported by the National Science Foundation under Grants MSS-901779, MSS-9157843, and IRI-9357738, and by the Office of Naval Research Young Investigator Award N00014-92-J1920.

G. S. Chirikjian is with the Department of Mechanical Engineering, Johns Hopkins University, Baltimore, MD 21218 USA.

J. W. Burdick is with the Department of Mechanical Engineering, California Institute of Technology, Pasadena, CA 91125 USA.

IEEE Log Number 9414516.

processors. That is, the computational time can be *independent* of the number of degrees of freedom if $O(N)$ processors are used. This claim has not been made for any other redundancy resolution technique for general hyper-redundant manipulator morphologies.

The method presented here is based on a "backbone curve" that captures a hyper-redundant robot's macroscopic geometric features. The backbone concept was introduced in previous work [8], [11], and has been used as a basis for developing obstacle avoidance [7], locomotion, and grasping [9], [12] analysis and algorithms for hyper-redundant robots. This paper uses the calculus of variations to develop necessary conditions for determining backbone curve shapes which satisfy task constraints and a user-defined optimality criterion. To illustrate the idea, we focus on a cost criterion which is a weighted measure of mechanism bending and extension. Shapes generated from other optimality criteria can be formulated analogously.

A summary of the potential uses of hyper-redundant robots and previous work by other researchers in the kinematics and design of hyper-redundant robotic systems can be found in [5] and [11]. We list here those prior works which are most relevant to this paper. Reference [24] considers an optimal shape synthesis problem for a high degree of freedom Variable Geometry Truss (VGT). The solution in [24] is an approximate one, which can be considered a subset of the method in this paper. The work presented here is also useful for a broader class of mechanisms than VGT systems. Several authors [25], [22] have presented algorithms for the "global" (or pathwise) redundancy resolution problem. They typically use the calculus of variations or optimal control theory to find joint displacement *trajectories* which satisfy terminal end-effector constraints and minimize a user-defined cost function. This paper focuses on optimal manipulator *configurations*, though this method can also be used for trajectory generation. The optimal shape design of thin elastic rods which implement a desired robot wrist compliance was considered in [4]. This is analogous to finding the geometry of a "stiff" hyper-redundant robot which best implements a desired compliance behavior.

Section II reviews joint-based redundancy resolution techniques which are applicable to the problem of configuration optimization. Section III reviews the basic backbone curve modeling technique and the associated "fitting" procedures. Section IV develops an optimality criteria for hyper-redundant manipulator shape based on a weighted measure of backbone curve bending, twisting, and extension. Section V provides a detailed application of the technique to a VGT mechanism. Section VI compares the computational and qualitative advantages of the continuum approach versus joint-based methods. Section VII summarizes our conclusions.

II. Joint-Based Configuration Optimization

Based on our experience in building and controlling the robot seen in Fig. 1, we believe that configuration optimization is a more useful goal than trajectory optimization for hyper-redundant robots. The reasons for this are twofold. First, one is likely to build hyper-redundant robots in a modular fashion—i.e., as a cascade of many similar or identical mechanical modules (most practical hyper-redundant robots have NOT been constructed with serial chain topologies, as they are too weak). For example, the robot in Fig. 1 is comprised of a concatenation of 10 identical modules, where each module is a 3 degree of freedom planar parallel manipulator. In order to have reasonable strength-to-weight ratios, the actuators in these modules will often have nonnegligible restrictions on bending, extension, etc. Thus, it makes sense to insure that at each configuration, the local module kinematic and mechanical constraints are not violated. This can easily be done in an approximate way via configuration optimization. Configuration optimization algorithms can also be the basis for trajectory planning schemes, in which one insures that the robot's configuration is optimal at each point in the trajectory. The techniques of [25] extremize a criteria which is integrated over a trajectory, not at each configuration. Second, configuration optimization schemes are cyclic in subsets of the workspace void of singularities. Cyclicity can be quite important for hyper-redundant manipulators.

For the purpose of comparison to the continuum approach presented later in this paper, we now review how one might formulate configuration optimization procedures using a framework based on discrete joint displacements and a Jacobian matrix.

Recall that redundancy resolution techniques based on the pseudo-inverse (or weighted pseudo-inverse) of the manipulator Jacobian matrix are based on the idea of constrained optimization. Let $\overline{\theta}$ denote the vector of joint displacements. The weighted pseudo-inverse solution is the joint velocity vector, $\dot{\overline{\theta}}$, which minimizes the instantaneous cost function $\frac{1}{2}\dot{\overline{\theta}}^T \mathbf{W}\dot{\overline{\theta}}$ subject to the linearized end-effector kinematic constraints $\dot{\overline{x}}_D = \mathbf{J}\dot{\overline{\theta}}$. $\mathbf{W}$ is an $N \times N$ symmetric positive definite weighting matrix, N is the number of mechanism actuated joints, $\overline{x}_D \in \mathbb{R}^M$ is the desired end-effector or task coordinates, and $\mathbf{J}$ is the $M \times N$ manipulator Jacobian matrix. Recall that the solution to this optimization problem is $\dot{\overline{\theta}} = \mathbf{J}_W^{\dagger}\dot{\overline{x}}_D$, where $\mathbf{J}_W^{\dagger} = \mathbf{W}^{-1}\mathbf{J}^{\mathbf{T}}(\mathbf{J}\mathbf{W}^{-1}\mathbf{J}^{\mathbf{T}})^{-1}$ is the weighted pseudo-inverse. The computational complexity of the unweighted pseudo-inverse ($\mathbf{W} = 1$) is $O(N)$ when $\mathbf{J}$ is computed recursively (see Appendix B). The weighted pseudo-inverse requires $O(N^{\max(2,p)})$ computations for a general symmetric weighting matrix where $O(N^p)$ computations are required to compute the components of $\mathbf{W}^{-1}$. p depends on what matrix function is used, and whether or not $\mathbf{W}(\overline{\theta})$ is defined and inverted at each timestep, or $\mathbf{W}^{-1}$ is calculated symbolically off line. This, coupled with the fact that these computations cannot be performed completely in parallel, means that there is an undesirable rise in computation time when using the pseudo-inverse (weighted or not) as the number of joints increases even if $O(N)$ processors work in parallel.

The optimal *configuration redundancy resolution* problem can be analogously defined. The goal is to minimize a cost which is a function of configuration while also satisfying end-effector position constraints

$$\min_{\overline{\theta}} g(\overline{\theta}) \quad \text{subject to} \quad \overline{c}(\overline{\theta}) = f(\overline{\theta}) - \overline{x}_D = \overline{0} \tag{1}$$

where $g(\overline{\theta})$ is the cost function, $f(\overline{\theta})$ is the forward kinematic function, and $\overline{x}_D$ is the desired end-effector location. This method is cyclic [2], unlike the pseudo-inverse which generally is not. For the purpose of discussion, let us assume that the configuration cost is

$$g(\overline{\theta}) = \frac{1}{2}\overline{\theta}^T \mathbf{W} \overline{\theta}. \tag{2}$$

For example, if the joint displacements are defined so that $\overline{\theta} = \overline{0}$ is the center of the joint range, then (3) is a measure of mechanism's deformation from the center of its joint range.

There are a number of methods by which one can computationally solve (2). One of the most popular methods is the *Lagrange–Newton* approach [19]. This method is based on the definition of a Lagrangian

$$L(\overline{\theta}, \overline{\lambda}) = g(\overline{\theta}) + \overline{\lambda}^T \overline{c}(\overline{\theta}) \tag{3}$$

where $\overline{\lambda}$ is a vector of undetermined Lagrange multipliers. An extrema of $L(\overline{\theta}, \overline{\lambda})$ extremizes $g(\overline{\theta})$ subject to the constraint $\overline{c}(\overline{\theta}) = \overline{0}$. Numerically, local extrema are found by starting with estimates $\overline{\theta}_0$ and $\overline{\lambda}_0$, and then iteratively solving the matrix equation

$$\begin{bmatrix} \mathbf{P}(\overline{\theta}_k, \overline{\lambda}_k) & \mathbf{J}(\overline{\theta}_k) \\ \mathbf{J^T}(\overline{\theta}_k) & \mathbf{0} \end{bmatrix} \begin{bmatrix} \delta\overline{\theta}_k \\ \delta\overline{\lambda}_k \end{bmatrix} = - \begin{bmatrix} \nabla g(\overline{\theta}_k) + \mathbf{J}^T(\overline{\theta}_k)\overline{\lambda}_k \\ \overline{c}(\overline{\theta}_k) \end{bmatrix} \tag{4}$$

for $\delta\overline{\theta}_k$ and $\delta\overline{\lambda}_k$. The matrix $\mathbf{P}(\overline{\theta}_k, \overline{\lambda}_k)$ in (4) has the form $\mathbf{P}(\overline{\theta}_k, \overline{\lambda}_k) = \nabla^2 g(\overline{\theta}_k) + \sum_{i=1}^{M} \lambda_i \nabla^2 c_i(\overline{\theta})$. New best estimates of the extrema are obtained by the update rule $\overline{\theta}_{k+1} = \overline{\theta}_k + \delta\overline{\theta}_k$ and $\overline{\lambda}_{k+1} = \overline{\lambda}_k + \delta\overline{\lambda}_k$. This method has the advantage that if the initial estimates are good, then convergence to a local minima is quadratic. While general configuration optimization implemented with the Lagrange–Newton method requires $\mathrm{O}(N^2)$ computations if the above quantities are defined recursively and solved iteratively (or $\mathrm{O}(N^3)$ computations if the $(N+M)\times(N+M)$ matrix is explicitly inverted), there are some manipulator morphologies and some choices of cost functions which permit as few as $\mathrm{O}(N)$ computations with this technique (when the joint angles are defined from an appropriate datum). In these cases $\mathbf{P}$ and $\mathbf{J}$ will have special structure.

Further, one must be able to compute second derivatives of $g(\overline{\theta})$ and the manipulator Hessian matrix (i.e., second derivatives of $f(\overline{\theta})$). These computations are often extremely difficult for the nonserial topologies used in practical hyper-redundant designs. A variant on this method has been derived in a different way and was implemented as a weighted pseudo-inverse for serial chain manipulators in [15], and the method was shown to be cyclic for a particular example.

Another class of numerical methods rely on a *projected gradient* [19]. In this approach, from an initial estimate, $\overline{\theta}_0$, one would like to minimize $g(\overline{\theta})$ by moving in the direction of the negative gradient of $g(\overline{\theta})$. However, only moves which satisfy the constraints are allowed. This procedure, which is also iterative, can be performed as follows. Let $\overline{\theta}_0$ be an estimate of the local minima, which is assumed to satisfy the constraints. Let $\mathbf{Z_k}$ be an $N \times N$ matrix whose columns are a basis for the null space of $\mathbf{J}(\overline{\theta}_k)$ (for example, the matrix $\mathbf{1} - \mathbf{J}^\dagger(\overline{\theta}_k)\mathbf{J}(\overline{\theta}_k)$). In the first step (termed the estimator step), a better estimate of the minima is obtained by taking a step in the direction of the projection of the negative gradient onto $\mathbf{Z_k}$:

$$\overline{\theta}_{k+1,0} = \overline{\theta}_k - \alpha \mathbf{Z_k} \nabla g(\overline{\theta}_k). \tag{5}$$

$\alpha > 0$ is either a small constant (typically $\alpha \leq 1$), or its value can be chosen using a line search.

Since the tangent space is a poor approximation to the curved constraint surface, the estimator step will result in a configuration $\overline{\theta}_{k+1,0}$ which is not on the constraint surface. This constraint violation is fixed in the subsequent "corrector" step, whose goal is to find a $\delta\overline{\theta}_{k+1,0}$ such that $\overline{c}(\overline{\theta}_{k+1,0} + \delta\overline{\theta}_{k+1,0}) = \overline{0}$. Assume that $\delta\overline{\theta}_{k+1,0} = \mathbf{Y}_k \overline{v}_k$, where $\mathbf{Y}_k = [\overline{Y}_{k_1}, \overline{Y}_{k_2}, \cdots, \overline{Y}_{k_N}]$ is a basis for the range space of $\mathbf{J}^T(\overline{\theta}_{k+1,0})$. In fact, one can choose $\mathbf{Y}_k = \mathbf{J}^T(\overline{\theta}_{k+1,0})$. The undetermined coefficients $\overline{v}_k$ which minimize the estimator error to 1^{st} order are obtained as the solution to

$$-\overline{c}(\overline{\theta}_{k+1,0}) = \mathbf{J}(\overline{\theta}_{k+1,0})\mathbf{Y}_k \overline{v}_k. \tag{6}$$

If we do in fact choose $\mathbf{Y}_k = \mathbf{J}^T(\overline{\theta}_k)$ and combine the estimator and corrector steps together, then we arrive at the adaptation of the well known "pseudo-inverse with null space projection" redundancy resolution scheme [2] to configuration optimization:

$$\overline{\theta}_{k+1} = \overline{\theta}_k + \mathbf{J}^\dagger(\overline{x}_D - f(\overline{\theta}_k)) - \alpha \mathbf{Z_k} \nabla g(\overline{\theta}_k). \tag{7}$$

While combining the estimator and predictor steps into one step is advantageous from a computational point of view, it does lead to poorer convergence than the proper estimator-corrector scheme.

This scheme has computational complexity $\mathrm{O}(N)$ when the Jacobian is computed recursively, and the null space projection term is computed by first computing the pseudo-inverse, performing the multiplication $\mathbf{J}\nabla g$, and then multiplying the pseudo-inverse. Otherwise, treating the Jacobian null space basis as an $N \times N$ matrix, and projecting ∇g onto this basis has computational complexity $\mathrm{O}(N^2)$. Either way, it is clear that these computations cannot be distributed among $\mathrm{O}(N)$ processors to achieve $\mathrm{O}(1)$ time performance because many of the computations are serial or recursive in nature.

Furthermore, neither the Lagrange–Newton or pseudo-inverse with gradient projection methods for configuration optimization can be easily applied to continuous morphology hyper-redundant robots (such as those made from pneumatic tube bundles (e.g., [23]) where it is not possible to define a Jacobian in the standard sense. The remainder of this paper is dedicated to *macroscopically* solving the configuration optimization problem in an entirely new way: using a continuum approach.

III. Continuum Kinematics of Backbone Curves

The continuum approach to hyper-redundancy resolution is based on a two-step modeling and computation procedure:

In the first step, we assume that, regardless of the mechanical implementation (e.g., the morphologies shown in Fig. 2), the hyper-redundant robot can be modeled using a continuous

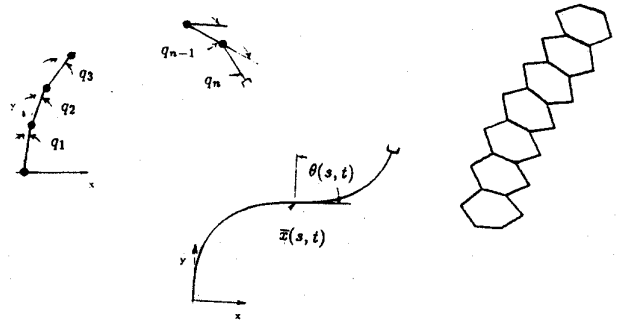


Fig. 2. Examples of hyper-redundant robot morphologies.

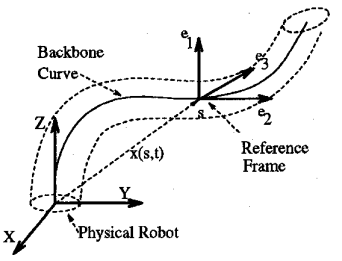


Fig. 3. Backbone curve abstraction.

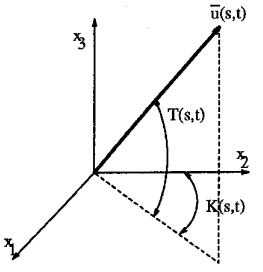


Fig. 4. Definition of angles $K(s,t)$ and $T(s,t)$.

backbone curve that captures the robot's macroscopic geometric features (Fig. 3). A backbone curve parametrization and an associated set of reference frames that evolve along the curve are collectively referred to as the *backbone reference set*. In this paradigm, hyper-redundancy resolution is reduced to the determination of the proper time varying behavior of the backbone reference set. Depending upon the robot's actual mechanical implementation, the associated backbone curve may be *inextensible* (or fixed length) or *extensible* (variable length). Techniques for physically meaningful parametrization of backbone curves are reviewed in Section III-A. For more details about the backbone curve approach and its suitability for modeling hyper-redundant robots, see [11].

In the second step, the continuous backbone curve redundancy resolution solution is used to specify the actual mechanism's joint displacements. The continuous solution can be used to directly specify actuator displacements in a continuous morphology robot. For discrete morphology robots we employ a "fitting procedure" to adapt the continuous solution to the discrete robot. The goal of the fitting procedure is to determine the actuator displacements which cause the discretely segmented robot to adhere exactly or as closely as possible to the continuous backbone curve solution. One fitting procedure which is specialized to a modular hyper-redundant robot mechanical architecture is reviewed in Section III-B because of its use in subsequent examples. Fitting procedures for other mechanical morphologies can be found in [5] and [11].

A. An Integral Backbone Curve Representation

Points, $\overline{x}(s,t)$, on an extensible spatial backbone curve can be parametrized as follows:

$$\overline{x}(s,t) = \int_0^s l(\sigma,t)\overline{u}(\sigma,t)d\sigma \tag{8}$$

where s is the backbone curve parameter and t is time. $\overline{u}(s,t)$ is the unit vector tangent to the curve at s. Unless otherwise specified, in this paper we use the following convention: $\overline{u}(0,t) = [0,1,0]^T$. $l(s,t)$ is a scaling factor that controls the length of the curve tangent and assumes the general form

$$l(s,t) = 1 + \epsilon(s,t) > 0. \tag{9}$$

$\epsilon(s,t)$ is the *local extensibility* of the manipulator at point s and time t. $\epsilon < 0$ indicates a local contraction, while $\epsilon > 0$ corresponds to local expansion. The backbone curve arc-length between the backbone curve base ($s = 0$) and any point along its length is

$$L(s,t) = \int_0^s l(\sigma,t)d\sigma. \tag{10}$$

Any parametrization of the unit sphere can be used to define $\overline{u}(s,t)$. In this work we choose

$$\overline{u} = [\sin K \cos T, \cos K \cos T, \sin T]^T. \tag{11}$$

The definitions of K and T are shown in Fig. 4. By convention, $K(0,t) = T(0,t) = 0$ is assumed. Henceforth, we will not express the dependence of functions on the variables s and t unless absolutely necessary.

The kinematics of planar curves is a degenerate case of (11) with $T = 0\ \forall s$. To distinguish the planar case, we use the symbol θ instead of K, where θ is the clockwise measured angle which the tangent to the planar curve makes with the x_2-axis at time t. Note that in the planar case $\overline{u} = [\sin\theta,\ \cos\theta]^T$.

A *backbone reference frame* at s has right-handed orthonormal basis vectors, $\{\overline{e}_1, \overline{e}_2, \overline{e}_3\}$, and its origin coincides with point $\overline{x}$. The set of backbone frame orientations can be written as

$$\mathbf{Q} = (\overline{e}_1\ \overline{e}_2\ \overline{e}_3) \in SO(3) \tag{12}$$

where $\overline{e}_i = \overline{e}_i(s,t)$, $\overline{e}_2 = \overline{u}$, and $\mathbf{Q}(0,t) = \mathbf{1}$. There is freedom in the assignment of backbone reference frames. A backbone curve parametrization will typically have a set of frames naturally associated with it. We call this frame the *parametrization induced reference frame*, or *induced frame*,

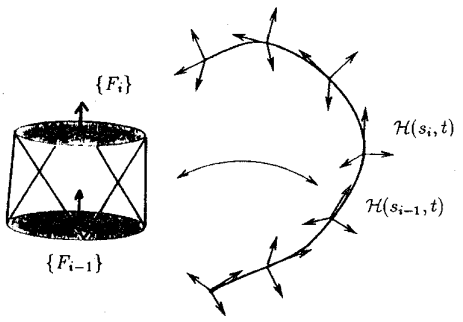


Fig. 5. Fitting a modular manipulator to a backbone reference set.

and denote it by $\mathbf{Q}_{\mathrm{IR}}$. For the parametrization of Fig. 4, we assign to every s the frame whose orientation is described by

$$\mathbf{Q}_{\mathrm{IR}} = \begin{pmatrix} \cos K & \sin K \cos T & -\sin K \sin \\ -\sin K & \cos K \cos T & -\cos K \sin T \\ 0 & \sin T & \cos T \end{pmatrix} \quad (13)$$

where $\mathbf{Q}_{\mathrm{IR}}(0,t) = \mathbf{1}$. The induced frame should not be confused with the backbone reference frame. It can differ from the backbone reference frame by an s-dependent twist about backbone curve tangent, which we term the *roll distribution*, $R(s,t)$. R measures how $\mathbf{Q}$ twists about the backbone curve with respect to $\mathbf{Q}_{\mathrm{IR}}$, and it is defined as $\mathbf{Q} = \mathrm{Rot}(\overline{e}_2, R)\mathbf{Q}_{\mathrm{IR}}$, where $\mathrm{Rot}(\overline{v}, \phi)$ is rotation about axis $\overline{v}$ by angle ϕ. $R(0,t) = 0$ follows from the fact that $\mathbf{Q}(0,t) = \mathbf{Q}_{\mathrm{IR}}(0,t) = \mathbf{1}$.

In summary, the backbone curve reference set, which consists of the backbone curve and associated set of orthonormal frames, is described with a small set of *shape functions*, which we denote by $\overline{y} = [K, T, R, L]^T$. The choice of shape function basis is not unique, and other possibilities are described in [11]. Note that the backbone reference set can also be expressed as a parametrized set of homogeneous matrices

$$\mathcal{H} = \begin{pmatrix} \mathbf{Q}(\overline{y}) & \overline{x}(\overline{y}) \\ \overline{0}^T & 1 \end{pmatrix} \quad (14)$$

where $\overline{x}(\cdot)$ and $\mathbf{Q}(\cdot)$ are defined in (8) and (12).

B. "Fitting" Procedures

Here we consider a fitting procedure for a hyper-redundant robot structure built from a concatenation of n identical modules. For example, the VGT structure of the robot in Fig. 1 fits this paradigm.

Consider the ith module (Fig. 5). Attach a frame, $\{F_{i-1}\}$, to the "input," or base, of the module, and a frame, $\{F_i\}$, to the "output," or top, of the module. For the discretely segmented modular manipulator configuration to conform to the continuous curve geometry, the frames $\{F_{i-1}\}$ and $\{F_i\}$ are chosen to coincide with the backbone reference frames at a set of $n+1$ "fitting" points: $\{s_i\}$. We typically choose $s_i = i/n$ for $i = 0, \cdots, n$. Recall that equal partitioning of the curve parameter need not imply equal physical spacing along the curve, because $L(\cdot)$ can be chosen from a broad class of functions.

The 4×4 homogeneous transform relating $\{F_i\}$ to $\{F_{i-1}\}$ is denoted by $\mathbf{H}_{i-1}^i$. This consists of the relative translation, $\overline{r}_{i-1}^i$, and rotation, $\mathbf{R}_{i-1}^i$, of $\{F_i\}$ with respect to $\{F_{i-1}\}$, i.e.,

$$\mathbf{H}_{i-1}^i(\overline{q}^{M_i}) = \begin{pmatrix} \mathbf{R}_{i-1}^i(\overline{q}^{M_i}) & \overline{r}_{i-1}^i(\overline{q}^{M_i}) \\ \overline{0}^T & 1 \end{pmatrix}. \quad (15)$$

$\overline{q}^{M_i} \in \mathbf{R}^{\mathbf{M}}$ is the vector of joint displacements which determine the geometry of the ith module. It is assumed that the inverse kinematics of the module, which relates $\{F_i\}$ to $\{F_{i-1}\}$, can be solved in a closed or numerically efficient form.

The manipulator configuration will exactly conform to the backbone reference set at points $\{s_i\}$ if

$$\mathbf{H}_{i-1}^i(\overline{q}^{M_i}(t)) = \mathcal{H}^{-1}(s_{i-1}, t)\mathcal{H}(s_i, t) \quad (16)$$

where $\mathcal{H}(s,t)$ is defined in (14). That is, the right hand side of (16) expresses the relative displacement of the backbone curve reference frame at s_i with respect to the backbone curve reference frame at s_{i-1}, while the left hand side describes the relative displacement of the ith module output frame with respect to its input frame. When the two are equated, the discrete mechanism aligns exactly with the continuous backbone curve at the n fitting points. We typically choose one of the fitting points to be the end-effector frame, so that distal position and orientation of both the continuous backbone curve and the discrete mechanism are in exact alignment. An example of this method is given in Section V.

On a serial processor, the computational burden of the fitting procedure for modular morphologies is $\mathrm{O}(N)$. More importantly, the algorithm can easily be parallelized to great advantage. Assume that each module contains one computer processor and is connected to a central computer by a communications network. Once the backbone curve shape functions which solve a hyper-redundancy resolution problem have been computed by the central processor, each $\mathbf{R}_{i-1}^i$, $\overline{r}_{i-1}^i$, and module inverse kinematics can be computed in parallel on the relevant processors. Thus, for modular geometries, such as the one in Fig. 1, the computational complexity of the fitting procedure is $\mathrm{O}(1)$, or *constant*, in time for N processors. That is, it is independent of the number of modules if one chooses this simple parallel processing model and broadcast communications architecture.

IV. Optimal Backbone Curve Configurations

In this work we focus on "optimal" configurations that minimize a weighted combination of bending, twisting, and local extension/contraction of the backbone curve while also satisfying task constraints. Using the continuous backbone curve model, this section uses the calculus of variations to compute the optimal backbone curve shapes. The optimal configurations that satisfy other criteria can be formulated in an analogous fashion.

A. Quantifying Backbone Curve Optimality

Deformation of the backbone curve (and the resulting change in hyper-redundant manipulator configuration) results

from mechanism bending, twisting, rolling, and extension/contraction at each s. A dimensionally consistent cost function which includes these effects is

$$I = \frac{1}{2}\int_0^1 \left[\mathrm{tr}\left(\dot{\mathbf{Q}}\mathbf{W}\dot{\mathbf{Q}}^T\right) + \beta(\dot{L}-1)^2\right]ds \qquad (17)$$

where $\mathrm{tr}(\mathbf{A})$ denotes the trace of matrix $\mathbf{A}$, and "$\cdot$" represents differentiation with respect to s. In the problem at hand, the cost function and constraints are functions of time, but since we are extremizing from point to point in time, the calculus of variations formulated for a single dependent variable (see Appendix A) is directly applicable. $\mathrm{tr}\left(\dot{\mathbf{Q}}\mathbf{W}\dot{\mathbf{Q}}^T\right)$ is a measure of mechanism bending and twisting. $\mathbf{W}$ is a 3×3 symmetric positive semi-definite weighting matrix. We make the reasonable assumption that there is no preferred direction of bending, and hereafter $\mathbf{W}$ is restricted to the isotropic form $\mathbf{W} = \alpha\mathbf{1}$, where $\mathbf{1}$ is the 3×3 identity matrix. Similarly, $(\dot{L}-1)^2$ is a measure of a mechanism's extension and contraction from its nominal length. Thus, α weights the relative cost of bending, twisting, and rolling, while β weights extension/contraction. In this section, the calculus of variations is used to generate backbone curve shapes which extremize (17). Other criteria can be similarly handled.

At $s = 0$, the backbone reference frame must coincide with the base frame. At $s = 1$, the backbone reference frame must correspond to the desired end-effector orientation, $\mathbf{Q}_D$. Thus, the boundary conditions

$$\mathbf{Q}(0,t) = \mathbf{1}; \qquad \mathbf{Q}(1,t) = \mathbf{Q}_D(t) \qquad (18)$$

are imposed on the Euler–Lagrange equations. The minimum bending problem can be stated as the minimization of (17) subject to the isoperimetric constraints $\overline{x}(1,t) = \overline{x}_D(t)$ (the desired end-effector position, where $\overline{x}(s,t)$ takes the form of (9) with boundary conditions (18). (See Appendix A for a review of variational calculus and explanation of the above terminology).

The corresponding Lagrangian is

$$\mathcal{L} = \frac{1}{2}\alpha\mathrm{tr}\left(\dot{\mathbf{Q}}\dot{\mathbf{Q}}^T\right) + \frac{1}{2}\beta(\dot{L}-1)^2 + \dot{L}\overline{\mu}_c\cdot\overline{u}. \qquad (19)$$

$\overline{\mu}_c$ is a vector of undetermined Lagrange multipliers arising from the isoperimetric end-effector constraint $\overline{x}(1,t) = \overline{x}_D(t)$.

The only issue which needs to be resolved is how to choose the functions $\alpha(s)$ and $\beta(s)$. The choice naturally depends on the particular hyper-redundant robot morphology and the intuition of the hyper-redundant robot user. In instances where the robot is constructed from a concatenation of uniform modules (such as in Fig. 1) or pneumatic tubes, these functions are independent of s, and the following analysis leads to a useful choice of the weighting functions.

First consider a planar robot. Imagine a deformable envelope or "tube" which closely fits the structure. We will assume that the undeformed position of the manipulator is straight with a nominal uniform extension. This is equivalent to a straight and unstretched state of the tube. For each manipulator configuration that is different from the nominal state, there is an associated configuration which is a deformation of the surrounding tube. Therefore, tube configurations that correspond to the least variation from an undeformed state will correspond approximately to manipulator configurations with the least variation from the initial configuration.

Let $\overline{x}$ denote the backbone curve (or centerline) of a planar tube in its distorted shape at any fixed time. Let $\overline{x}_+$ and $\overline{x}_-$ denote the respective sides of the planar tube in the distorted state

$$\overline{x}_+ = \overline{x} + r\overline{n}, \qquad \overline{x}_- = \overline{x} - r\overline{n} \qquad (20)$$

where $\overline{n}$ is the unit vector normal to the backbone curve at s, and r is the constant tube radius. A reasonable measure of the local deviation of the tube at a point s from its nominal configuration (which is assumed here to be a straight tube with no extension/contraction) is the sum of the squared difference in length between the tube tangents and the length of the nominal reference tangent

$$\begin{aligned} f &= \tfrac{1}{2}\left((||\dot{x}_+|| - 1)^2 + (||\dot{x}_-|| - 1)^2\right) \\ &= \tfrac{1}{2}\left((||\dot{x} + r\dot{\overline{n}}|| - 1)^2 + (||\dot{x} - r\dot{\overline{n}}|| - 1)^2\right) \\ &= \tfrac{1}{2}\left((||\overline{u}\dot{L} - r\dot{L}\kappa\overline{u}|| - 1)^2 + (||\overline{u}\dot{L} + r\dot{L}\kappa\overline{u}|| - 1)^2\right) \\ &= (\dot{L}-1)^2 + (r\dot{L}\kappa)^2 = (\dot{L}-1)^2 + (r\dot{\theta})^2. \end{aligned} \qquad (21)$$

Equation (21) corresponds to the integrand of (17) in the planar case, with $\beta = 1$ and $\alpha = \frac{1}{2}r^2$. We take the tube radius, r, to be half the width of the physical manipulator. This provides for a cost function which is dimensionally homogeneous.

In the spatial case, it can be shown that similar functions result by taking the magnitudes of the tangents of the deformed and underformed fibers which lay longitudinally along a spatial tube, and integrating the square of this magnitude around the tube (which is the spatial analog of the sum in (21)).

Other physically meaningful choices for $\alpha(s)$ and $\beta(s)$ are based on inertial properties of the manipulator. For instance, one can define a mass density per unit curve parameter, $\rho(s)$, to approximate manipulator inertial properties. This can be used to weight bending and extension so that the base of the manipulator (which has the largest inertia to move) bends less than the end. One choice to achieve this is $\beta(s) = \int_s^1 \rho(\sigma)d\sigma$, $\alpha = \frac{1}{2}r^2\int_s^1 \rho(\sigma)d\sigma$. Because the density per unit curve parameter does not change even if the manipulator stretches or contracts, ρ is not a function of time. Dynamics algorithms based on the continuum model can be computed in $\mathrm{O}(N)$ computations, and can be completely distributed over N processors to yield $\mathrm{O}(1)$ time performance [14].

B. Solving the Inverse Problem

The previous subsection developed a cost function for a configuration. Substituting this cost function into the Euler–Lagrange equations results in a set of differential equations whose solution is a backbone curve shape which extremizes this cost for a given set of initial conditions (i.e., conditions of the curve at $s = 0$) and a given set of Lagrange multipliers. Let the set of undetermined initial conditions and the Lagrange multipliers be termed the *reduced configuration variables*, which we denote by $\overline{\gamma}$. For a given cost function and its associated Euler–Lagrange equations, the end-effector location is strictly a function of the small set of these reduced configuration variables. That is, for a given value of $\overline{\gamma}$, integration of the

Euler–Lagrange equations will produce a unique end-effector location. This section considers the "inverse" problem of determining the reduced configuration variables for a desired end-effector location. Many other techniques are known for solving this problem, as reviewed in [6]. We develop a particular method here because of its generality, because of its close resemblance to standard methods in manipulator kinematics, and for the purposes of computational complexity analysis.

Equation (8), when evaluated at $s = 1$, can be expressed in the general form

$$\overline{x}(1,t) = \int_0^1 \overline{v}(\overline{y})ds \tag{22}$$

where $\overline{y}$ is the set of shape functions. If $\overline{y}(s,t)$ is a solution which extremizes (17) at t, subject to the constraints that $\overline{x}(1,t) = \overline{x}_D(t)$, then $\overline{y}$ is a function of t via the reduced configuration variables $\overline{\gamma}$: $\overline{y}(s,t) = \hat{\overline{y}}(s,\overline{\gamma}(t))$.

The corresponding rate linearized (or "resolved rate") kinematics commonly used in robotics can then be written symbolically as

$$\frac{d}{dt}(\overline{x}(1,t)) = \left[\int_0^1 \frac{\partial \overline{v}}{\partial \overline{y}}\frac{\partial \hat{\overline{y}}}{\partial \overline{\gamma}}ds\right]\frac{d\overline{\gamma}}{dt} = \mathrm{J}\frac{d\overline{\gamma}}{dt} \tag{23}$$

where J is the Jacobian associated with the reduced configuration variables $\overline{\gamma}$ and is termed the *reduced Jacobian*. Note that J is typically a 3×3 (4×4) or 6×6 (7×7) matrix for inextensible (extensible) planar or spatial backbone curves. Note that the size of J does *not* depend upon the number of mechanism degrees of freedom, but only on the dimension of the task coordinates (where the length of the manipulator is considered a task variable in the extensible case).

Equation (23) can be used to develop numerical procedures for solving the reduced configuration variable inverse problem. In the most simplistic approach, let $\overline{\gamma}_0$ be an initial guess of the reduced variables. One can iterate the following approximation to (23) to find the values of $\overline{\gamma}$ which solve the inverse problem

$$\Delta\overline{\gamma}^{k+1} = \overline{\gamma}_k + \mathrm{J}^{-1}(\overline{\gamma}_k)\Delta\overline{x}^k, \tag{24}$$

where k is the iteration index and $\Delta\overline{x}^k$ is the error between the actual end-effector location (computed with estimated reduced variable vector $\overline{\gamma}^k$) and the desired end-effector location. However, to compute $\mathrm{J}(\overline{\gamma}_k)$, one requires an expression for $\partial\overline{v}(s)/\partial\overline{y}$ (which is easily obtained) and an expression for the function $\partial\hat{\overline{y}}(s,\overline{\gamma})/\partial\overline{\gamma}$, where $\hat{\overline{y}}(s,\overline{\gamma})$ extremizes (17). Unfortunately, the extremal value of $\partial\hat{\overline{y}}(s,\overline{\gamma})/\partial\overline{\gamma}$ can generally be realized only in numerical, and not symbolic form. We must therefore develop indirect methods for computing the extremal value of $\partial\hat{\overline{y}}/\partial\overline{\gamma}$, and hence J.

The Euler–Lagrange equations will generally be of the form

$$\ddot{\overline{y}} + \overline{f}(\dot{\overline{y}},\overline{y},\overline{\gamma},s) = \overline{0} \tag{25}$$

with initial conditions

$$\overline{I}(\dot{\overline{y}}(0,\overline{\gamma}),\overline{y}(0,\overline{\gamma}),\overline{\gamma}) = \overline{0}, \tag{26}$$

where $\overline{y}, \overline{f}(\cdot) \in \mathbb{R}^P$, $\overline{\gamma} \in \mathbb{R}^M$, and $\overline{I}(\cdot) \in \mathbb{R}^{2P}$. We have dropped the "hat" from $\hat{\overline{y}}$, but there is no ambiguity because of the context in which $\overline{y}$ is being used. P is the number of shape functions needed to fully specify the hyper-redundant-manipulator configuration, and recall that M is the number of end-effector or task coordinates. In the plane $P = 2$ and M is typically 3 for nonextensible robots, or $M = 4$ for extensible robots. For spatial manipulators $P = 4$ and $M = 6$ or 7.

Given (25) and (26), one can determine $\partial\overline{y}/\partial\overline{\gamma}$ by a system of *auxiliary* differential equations. These M sets of auxiliary equations are derived by taking the derivatives of (25) and (26) with respect to the M components of $\overline{\gamma}$. Since derivatives of smooth functions commute, the auxiliary equations can be expressed as the $P \times M$ matrix equation:

$$\frac{d^2}{ds^2}\left(\frac{\partial\overline{y}}{\partial\overline{\gamma}}\right) + \frac{\partial\overline{f}}{\partial\dot{\overline{y}}}\frac{d}{ds}\left(\frac{\partial\overline{y}}{\partial\overline{\gamma}}\right) + \frac{\partial\overline{f}}{\partial\overline{y}}\frac{\partial\overline{y}}{\partial\overline{\gamma}} = -\frac{\partial\overline{f}}{\partial\overline{\gamma}}. \tag{27}$$

Note the linearity of the above equations in the auxiliary variables $\partial y_i/\partial\gamma_j$ for $(i,j) \in (1,\cdots,P) \times (1,\cdots,M)$. This linearity simplifies the numerical solution. The initial conditions of the 2nd order auxiliary equations are written symbolically as the $2P \times M$ matrix equation

$$\left[\frac{\partial\overline{I}}{\partial\dot{\overline{y}}}\frac{d}{ds}\left(\frac{\partial\overline{y}}{\partial\overline{\gamma}}\right) + \frac{\partial\overline{I}}{\partial\overline{y}}\frac{\partial\overline{y}}{\partial\overline{\gamma}} + \frac{\partial\overline{I}}{\partial\overline{\gamma}}\right]_{s=0} = \mathbf{0} \tag{28}$$

which can generally be separated. In addition to the commutation of derivatives, we have also made use of the fact that differentiation and function evaluation commute in the following cases:

$$\frac{\partial\overline{y}(0,\overline{\gamma})}{\partial\gamma_i} = \left[\frac{\partial\overline{y}}{\partial\gamma_i}\right]_{s=0} \qquad \frac{\partial\dot{\overline{y}}(0,\overline{\gamma})}{\partial\gamma_i} = \left[\frac{d}{ds}\left(\frac{\partial\overline{y}}{\partial\gamma_i}\right)\right]_{s=0}. \tag{29}$$

Thus the extremal value of $\partial\overline{y}/\partial\overline{\gamma}$ can be computed numerically from the auxiliary equations, and subsequently used to compute the reduced Jacobian in (23). The simultaneous (possibly parallel) solution of the original system of equations and the auxiliary equations provide the means by which the instantaneous end-effector kinematics of the hyper-redundant manipulator backbone curve is computed at each time step. If the algorithm is parallelized over $M + 1$ processors, the required computation time will be no greater than that of the original set of Euler–Lagrange equations plus the time required to invert the reduced Jacobian matrix.

V. A Detailed Example: The Optimal Backbone Curve for a Planar VGT

A detailed application of this approach to a variable geometry truss (VGT) manipulator having the same geometry as the robot in Fig. 1 is developed in this section. The next section compares the computational complexity of this method to the noncontinuum approaches of Section II for the same VGT robot geometry.

In the planar case, $\mathbf{Q}$ consists of a rotation, by angle θ, about the axis normal to the plane. Thus,

$$I = \frac{1}{2}\int_0^1 \left[\alpha\ \mathrm{tr}\left(\dot{\mathbf{Q}}\dot{\mathbf{Q}}^T\right) + \beta\left(\dot{L} - 1\right)^2\right]ds$$
$$= \frac{1}{2}\int_0^1 \left[2\alpha\ \dot{\theta}^2 + \beta\left(\dot{L} - 1\right)^2\right]ds. \tag{30}$$

In the case of $\alpha = \frac{1}{2}$ and $\beta = 0$ (i.e., an inextensible backbone curve), this cost function becomes the integral of squared curvature. This is a problem which has been considered in detail in the mathematics, mechanics, and computer science literature [21]. Solutions for this case in terms of Elliptic functions are known.

In the extensible case, the following choice is made for hyper-redundant manipulators with homogeneous structure: $\alpha(s) = \frac{1}{2}r^2$ and $\beta(s) = 1$ (consistent with the arguments of Section IV-A). The Euler–Lagrange equations corresponding to the functions θ and L are, respectively,

$$r^2\ddot{\theta} - \mu_1\dot{L}\cos\theta + \mu_2\dot{L}\sin\theta = 0 \tag{31}$$

$$\frac{d}{ds}(\dot{L} - 1 + \mu_1\sin\theta + \mu_2\cos\theta) = 0, \tag{32}$$

where μ_1 and μ_2 are the undetermined Lagrange multipliers associated with the planar end-effector position constraint.

We must now determine the boundary conditions and the reduced configuration variables. There are generally four task coordinates for a planar robot: x_{ee}, y_{ee}, θ_{ee}, and L_{ee}. That is, in addition to the end-effector position and orientation, the total manipulator length from base to end-effector is also treated as a task variable. Let us consider the problem of extremizing the integral in (30) while letting $\theta(1,t)$ and $L(1,t)$ be free. That is, we do not care about the end-effector orientation or the total length of the robot in the optimal configuration. Thus the free boundary conditions ((51) in Appendix A) are used:

$$\dot{\theta}(1) = 0 \tag{33}$$

$$\dot{L}(1) - 1 + \mu_1\sin\theta(1) + \mu_2\cos\theta(1) = 0. \tag{34}$$

The initial conditions $\theta(0) = 0$ and $L(0) = 0$ are imposed by the fixed base conditions. The undetermined initial conditions are $\dot{\theta}(0)$ and $\dot{L}(0)$. Thus, the reduced configuration variables consist of the two Lagrange multipliers and the two unspecified base boundary conditions:

$$\overline{\gamma} = [\gamma_1,\ \gamma_2,\ \gamma_3,\ \gamma_4]^T = [\mu_1,\ \mu_2,\ \dot{\theta}(0),\ \dot{L}(0)]^T. \tag{35}$$

These reduced configuration variables map to the four task variables through the Euler–Lagrange equations.

The solution proceeds as follows. Equation (32) has the exact first integral:

$$\dot{L} + \gamma_1\sin\theta + \gamma_2\cos\theta = \gamma_4 + \gamma_2. \tag{36}$$

$\dot{L}$ from this equation can then be substituted into (31), effectively decoupling the θ-variable Euler–Lagrange equation from L-variable Euler–Lagrange equation. Integrating Equation (31) with respect to s, while observing the boundary conditions at $s = 0$ and (33), yields

$$-r^2\gamma_3 = \gamma_1 y_{ee} - \gamma_2 x_{ee}. \tag{37}$$

Observing that (34) and (36) must hold simultaneously, we obtain the relation

$$\gamma_4 = 1 - \gamma_2. \tag{38}$$

Thus, γ_4 can be eliminated, and γ_3 is represented in a way that directly describes its influence on end-effector position (as opposed to slope at the base of the manipulator).

In this example, there are three sets of auxiliary equations corresponding to the remaining three reduced configuration variables (since γ_4 was eliminated). Each of these sets of equations consists of two separate equations of the form

$$r^2\frac{d^2}{ds^2}\left(\frac{\partial\theta}{\partial\gamma_i}\right) - \delta_{i1}\dot{L}\cos\theta - \gamma_1\frac{\partial}{\partial\gamma_i}\left(\dot{L}\cos\theta\right) + \delta_{i2}\dot{L}\sin\theta + \gamma_2\frac{\partial}{\partial\gamma_i}\left(\dot{L}\sin\theta\right) = 0, \tag{39}$$

$$\frac{d}{ds}\left(\frac{\partial L}{\partial\gamma_i}\right) + \delta_{i1}\sin\theta + \gamma_1\frac{\partial\theta}{\partial\gamma_i}\cos\theta + \delta_{i2}\cos\theta - \gamma_2\frac{\partial\theta}{\partial\gamma_i}\sin\theta = 0, \tag{40}$$

with initial conditions

$$\frac{\partial\theta}{\partial\gamma_i}(0) = \frac{\partial L}{\partial\gamma_i}(0) = 0 \qquad \frac{d}{ds}\left(\frac{\partial\theta}{\partial\gamma_i}\right)(0) = \delta_{i3} \tag{41}$$

for $i = 1,2,3$. $\delta_{ij} = 1$ when $i = j$ and zero otherwise. Equations (39) and (40) are solved to find $\partial\theta/\partial\gamma_i$ and $\partial L/\partial\gamma_i$.

The Jacobian matrix for this example is a 3×3 matrix, since one of the reduced configuration variables was eliminated. The first two rows have components

$$\frac{\partial x_1}{\partial\gamma_i}(1) = \int_0^1\left[\frac{\partial\epsilon}{\partial\gamma_i}\sin\theta + (1+\epsilon)\frac{\partial\theta}{\partial\gamma_i}\cos\theta\right]ds \tag{42}$$

$$\frac{\partial x_2}{\partial\gamma_i} = \int_0^1\left[\frac{\partial\epsilon}{\partial\gamma_i}\cos\theta - (1+\epsilon)\frac{\partial\theta}{\partial\gamma_i}\sin\theta\right]ds \tag{43}$$

for $i = 1,2,3$. Recall that ϵ is the extensibility: $\dot{L} = l = 1+\epsilon$.

The last row in the Jacobian matrix comes from differentiating (37) with respect to time, yielding

$$\gamma_2\frac{dx_{ee}}{dt} - \gamma_1\frac{dy_{ee}}{dt} = y_{ee}\frac{d\gamma_1}{dt} - x_{ee}\frac{d\gamma_2}{dt} - r^2\frac{d\gamma_3}{dt} \tag{44}$$

and so the last row in the Jacobian matrix is $[y_{ee}, -x_{ee}, r^2]$. Having calculated the reduced Jacobian, (24) is used to find the inverse solution. The initial values of the reduced variables are $\overline{\gamma} = \overline{0}$, which corresponds to the underformed reference state $\theta(s,0) = 0$, $L(s,0) = s$.

Until this point, not a single joint-based computation has been performed. In order to "algorithmically link" the backbone curve model with an actual physical device, we apply a fitting procedure to determine the discrete joint angles which cause the mechanism to exactly or closely adhere to the continuous backbone curve shape. The VGT mechanism is a modular structure, and therefore the fitting paradigm of Section III-B can be applied. All that is required is the inverse kinematics of a VGT module, which is easily solved.

For example, Fig. 6 shows one module of the planar variable geometry truss manipulator (which is the same geometry as the robot in Fig. 1 and is described in detail in [10]). The three

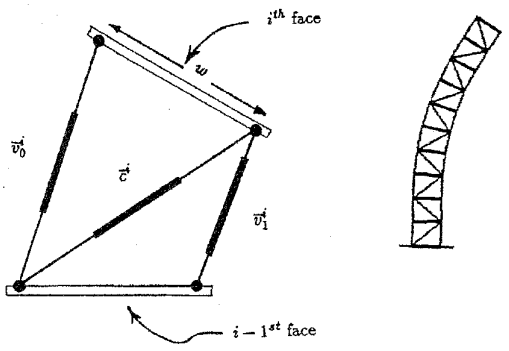


Fig. 6. Planar variable geometry truss fitting geometry.

vectors which are collinear with the prismatic actuators can be determined as follows:

$$\begin{aligned} \vec{v}_j^i &= \overline{p}_{i-1}^i - \overline{n}_j^{i-1} + \mathrm{ROT}(-\overline{e}_3, \theta_M^i)\overline{n}_j^i \qquad j = 1,2 \\ \vec{c}^i &= \overline{p}_{i-1}^i - \overline{n}_1^{i-1} + \mathrm{ROT}(-\overline{e}_3, \theta_M^i)\overline{n}_2^i \qquad i = 1,3,5,\cdots \\ \vec{c}^i &= \overline{p}_{i-1}^i - \overline{n}_2^{i-1} + \mathrm{ROT}(-\overline{e}_3, \theta_M^i)\overline{n}_1^i \qquad i = 2,4,6,\cdots \end{aligned} \tag{45}$$

where $\theta_M^i = \theta(s_i) - \theta(s_{i-1})$ and $\overline{n}_j^i$ are the vectors to the jth vertex of the ith platform in the frame affixed to that platform. For this example, $\overline{n}_1^i = [-w_i/2, 0]^T$ and $\overline{n}_2^i = [w_i/2, 0]^T$, where w_i is the width of each horizontal truss module face (Fig. 6). The controlled degrees of freedom are the lengths

$$q_j^i = ||\vec{v}_j^i||; \qquad q_3^i = ||\vec{c}^i|| \tag{46}$$

for $i = 1, \cdots, n$, and $j = 1, 2$. Equations (45) and (46) provide the inverse kinematics solution for this module geometry. This procedure is used in Section VI to generate optimal configurations for a given end-effector trajectory.

VI. A Comparison of the Continuum and Joint-Based Approaches

In this section we consider the qualitative and quantitative features of the continuum approach for configuration optimization versus joint-based configuration optimization methods. We also compare the two approaches in terms of numerical efficiency for a VGT manipulator. Section VI-A discusses the applicability and computational burden of both methods. Section VI-B provides the results of numerical trials.

A. Generality and Order of Computation

Practical hyper-redundant robots *have not and will not* be constructed of a serial chain of rigid links and motors, since such designs are too weak. Consequently, most of the hyper-redundant structures built to date use nonserial structural designs and actuation schemes, such as tendons, pneumatic hoses, or serial/parallel mechanisms like the VGT. It is often difficult or impossible to define a Jacobian matrix for many of these structures. One particular example of this is the case of hyper-redundant manipulators composed of a cascade of modules where each module does not have closed form forward kinematics—as is often the case for parallel manipulators with revolute joints. In this case, the joint-based approaches of Section II cannot even be applied directly. However, it is always possible to relate actuator displacements to a backbone curve (since the inverse kinematics of all kinematically sufficient serial and parallel manipulators can be performed efficiently), thereby establishing a fitting procedure. With a fitting procedure, the continuum approach is then applicable.

Moreover, the continuum approach has computational advantages over the discrete methods of Section II for general manipulator structures as the number of joints becomes large if sufficiently parallel computer architectures are used. Recall that the computation of the optimal shape using the continuum approach is a two phase process. First, the optimal continuous backbone curve shape is computed. The computational cost of this step is independent of the number of mechanical degrees of freedom of the actual discrete mechanism, and is therefore $\mathrm{O}(1)$. Next, the mechanism is "fitted" to the resulting curve. For modular designs, such as the one in Fig. 1, this process has a computational burden of $\mathrm{O}(N)$ when implemented on a serial processor, or $\mathrm{O}(1)$ when implemented with a simple parallel computing architecture with one processor per module (or even one processor for every m modules for some number $m > 1$ which is independent of N). Thus, the total computational burden of this approach scales as $\mathrm{O}(N)$ on a serial processor, or is $\mathrm{O}(1)$ in time on a simple parallel processing architecture with $\mathrm{O}(N)$ processors.

Alternatively, all of the discrete joint based procedures for performing configuration optimization are at best $\mathrm{O}(N)$ in complexity when implemented on a serial processor, and some are much worse for arbitrary macroscopically serial manipulator morphologies, e.g., the Lagrange–Newton method. However, as the number of degrees of freedom increases, the continuum based approach becomes more computationally attractive *for all morphologies* if it and competing methods are implemented in parallel on $\mathrm{O}(N)$ processors. This is because the computations required for the continuum approach completely decouple, whereas the $\mathrm{O}(N)$ computations of competing approaches do not.

One can also reduce the scaling of the computational burden of discrete approaches by going to parallel architectures. However, the architecture required for parallel computation of the Jacobian pseudo-inverse and other matrix/vector manipulations of conventional redundancy resolution cannot achieve $\mathrm{O}(1)$ time performance with $\mathrm{O}(N)$ processors. This is true in part because to achieve $\mathrm{O}(N)$ performance for the gradient projection method applied to a general manipulator morphology, the Jacobian must be computed by recursion. This does not parallelize completely, as is also the case for the matrix vector multiplications required to compute the pseudo-inverse once the Jacobian has been computed. In addition, such parallelization will require very complicated interprocessor communication and synchronization. Conversely, the parallel computing scheme for the continuum approach is trivial.

Thus, the continuum approach is favored for large numbers of degrees of freedom. To make this comparison more concrete, we now apply the projected gradient method of Section II to configuration optimization of a planar VGT truss manipulator, and compare the computation time with that of the continuum approach—both running on a single processor.

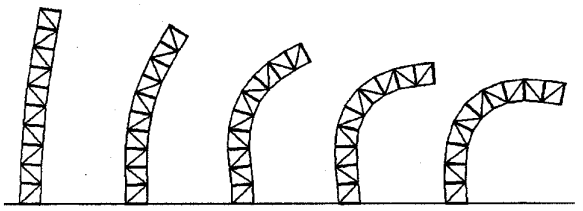


Fig. 7. Continuum method (calculus of variations) solution.

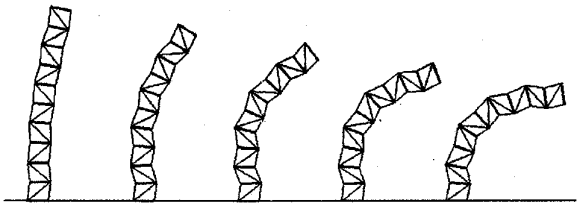


Fig. 8. Joint-based configuration optimization.

B. Numerical Comparison of Two Approaches

In this section, we compare the computation required to determine the optimal configurations for the VGT manipulator described previously. In this numerical comparison, the end-effector of the VGT mechanism follows the straight line trajectory

$$\overline{x}_{\text{ee}}(t) = [0,1]^T + t[1,-1]^T \quad \text{for} \quad 0 \le t \le \frac{1}{2}$$

with no prescribed orientation. This trajectory is approximated by 101 discrete points, i.e., $\overline{x}_{\text{ee}}(t)$ is evaluated at intervals of $\Delta t = 0.005$. For configuration optimization, the goal is to determine the optimal configuration at each point along the trajectory.

We define the optimal configuration as the one which requires the least deformation, or joint displacement, from a reference configuration. In this case, the reference configuration at $t = 0$ corresponds to a VGT configuration in which the outside actuators have length $q_{3i}(0) = q_{3i+2}(0) = \frac{1}{n}$ and the diagonal actuators have length $q_{3i+1}(0) = \frac{\sqrt{2}}{n}$. That is, the reference configuration is straight, with no extension or contraction. The vector $\overline{\theta} = \overline{q} - \overline{q}(0)$ measures the amount of joint displacement from the reference configuration. Note that the desired trajectory was chosen so that the manipulator is initially in its reference configuration at $t = 0$.

For the sake of comparison, we use two redundancy resolution methods on the trial trajectory: 1) the continuum approach with the minimum deformation criteria discussed earlier and 2) joint-based configuration optimization with the square of the norm of joint displacements as the optimization criteria implemented via gradient projection onto the Jacobian null space. The latter is implemented in two ways: Jacobian calculated numerically column by column (using the standard centered difference approximation for derivatives applied to the forward kinematic function), and the Jacobian calculated recursively. The forward kinematics for the VGT described earlier and used here can be found in [13]. The Lagrange–Newton method is not used in this comparison, because it is generally far slower than the Jacobian-based approaches.

Our implementation of the continuum approach follows directly the developments in Section V. The weighting factors on mechanism bending in the cost function of (17) are is chosen to be $B(s) = 1$ and $\alpha(s) = r^2/2$, where $r = \frac{1}{2n}$ because the width of each fixed truss element is taken to be $w = \frac{1}{n}$, and $r = w/2$. The backbone curve configuration that corresponds to the undeformed reference configuration can be found by taking $\gamma_i(0) = 0$ in (35) for $i = 1, 2, 3$ and integrating (31) and (32). The resulting backbone curve is a uniformly parametrized straight line described by the shape functions $\theta(s, 0) = 0$ and $L(s, 0) = s$. The reduced Jacobian elements are calculated at each time step using Liebnitz's rule and Euler integration. For purposes of numerical integration with respect to s, the backbone curve interval $s \in [0, 1]$ is subdivided into $5n$ intervals so that the backbone curve interval $s \in [s_{i-1}, s_i]$, which defines the motion of the ith module, is approximated by five segments. The configurations resulting from this method are shown in Fig. 7 for a 10 module VGT mechanism (with the same topology as the robot in Fig. 1). The configurations are shown at intervals of $\Delta t = 0.1$ starting at $t = 0.1$.

The joint-based configuration optimization simulation uses the projected gradient method reviewed in Section II. The objective function is

$$g(\overline{\theta}) = \frac{1}{2}\overline{\theta} \cdot \overline{\theta},$$

where $\overline{\theta}$ is the displacement of the joints from their reference configuration position. We have taken $\alpha = 3\sqrt{g(\theta)} < 1$, so the influence of the null space term becomes more pronounced the further the configuration is from the global minimum of the cost function. If one were to do a line search for the optimal value of α (which would be the most rigorous approach) this would add to the computational requirements of this method.

Unlike the continuum approach, the joint-based approach requires us to compute the derivatives of the VGT forward kinematics equations. Since we are comparing this method for VGT structures consisting of 2 to 20 modules, it would be too tedious to derive closed form algebraic expressions for all the derivatives. In fact, the need to derive expressions for these derivatives for the parallel/serial structures often used in real hyper-redundant systems is a major drawback of the Jacobian based methods. Instead, we numerically evaluate derivatives for each module and recursively compute the Jacobian as outlined in Appendix B.

The configurations resulting from the joint-based configuration optimization method implemented with pseudo-inverse with gradient projection onto the Jacobian null space are shown in Fig. 8 for the case of a 10 module VGT mechanism.

Fig. 9 shows the computation time (in seconds) required on a SUN SPARCstation ELC to compute the redundancy resolution solution for each method along the whole trajectory, and to display the results at all timesteps. CONT indicates time required for the continuum approach, PSEUDO is the time required for the pseudo-inverse with nullspace projection where each column of the Jacobian is generated separately, and REC is the same as PSEUDO except that the Jacobian is recursively computed. Since both methods (CONT or PSEUDO/REC)

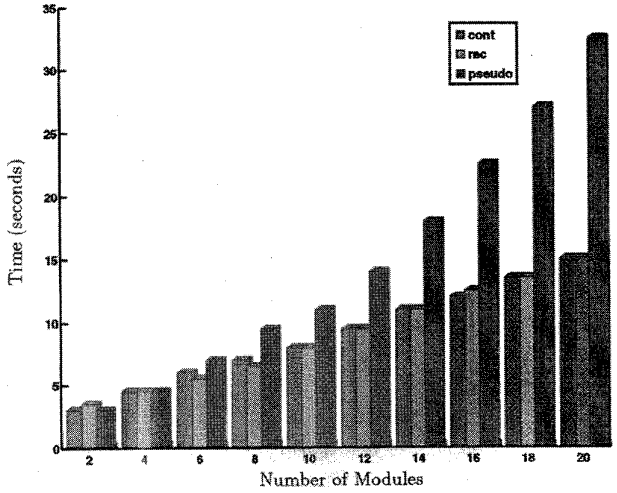


Fig. 9. Computation time for calculus of variations, and configuration optimization via gradient projection.

are aimed at the same objective, the configurations of the manipulators are qualitatively similar using both methods, though the continuum approach appears to give "smoother" results.

The computation time for PSEUDO grows quadratically in N, whereas the computation time for CONT and REC are both linear in N. Furthermore, the slope and intercept for CONT and REC are remarkably close. We believe this is due to the fact that the $O(N)$ computations required to display the manipulator during the simulations contributes a great deal to the computation time for both methods on the workstation that was used. However, removing the computation time required for the simulations in this comparison favors our method because it is the fitting procedure and display that makes our method $O(N)$ instead of $O(1)$ on a serial processor.

This is important because for point-to-point motions in the workspace, the fitting procedure does not have to be performed at each timestep when using the continuum approach, and so the only $O(N)$ computations that have to be performed can be done infrequently. This is not true for the joint based methods, which require $O(N)$ computations at each timestep. To emphasize this point, Fig. 10 plots the time required to compute both methods as before, but only display both methods at intervals of $\Delta t = 0.1$, instead of $\Delta t = 0.005$. Here we see that the continuum approach is much better than even the recursive Jacobian-based approach.

Furthermore, the continuum method can be implemented in ways which accommodate real-time control. For example, a look-up table scheme or neural network can store the relationship between end-effector coordinates and reduced configuration parameters. This relationship can be computed off-line and stored using $O(1)$ amount of memory. Table lookup and interpolation (or neural network generalization) can approximate the mapping between reduced parameters and end-effector coordinates in $O(1)$ time. Manipulator configurations can then be reconstructed (joint values calculated) via a fitting procedure in $O(N)$ time. If one were to form a look-up table of discrete joint values, this would require $O(N)$

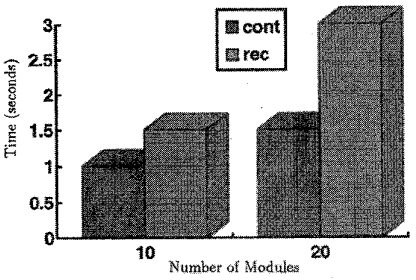


Fig. 10. Computation time for calculus of variations, and joint-based configuration optimization with intermittent display.

memory, and $O(N)$ time to interpolate. In this mode, the continuum method can be viewed as a "data compression" technique.

VII. Conclusion

This paper developed a method for determining "optimal" hyper-redundant manipulator configurations based on the calculus of variations and a continuous backbone curve model. This method also serves as the basis for trajectory planning schemes in which each configuration along the trajectory is optimal. Using the backbone curve approach and an associated optimality criteria, the entire backbone configuration becomes a function of a set of reduced configuration variables. It was shown that this method is computationally competitive with the most efficient joint-based approaches when implemented on a serial processor, and possesses the property that it can be computed in $O(1)$ time if computations are distributed over $O(N)$ processors—a statement which is not true for other methods.

Appendix A
Review of Variational Calculus

Recall [16] that vector functions $\overline{y}(s) \in \mathbb{R}^P$ (where in our case, $\overline{y}(s)$ will be interpreted as the set of backbone curve shape functions and s is the backbone curve length parameter) will extremize the integral

$$I = \int_0^1 f(s, \overline{y}(s), \dot{\overline{y}}(s), \ddot{\overline{y}}(s), \cdots, \overline{y}^{(n)}) ds \tag{47}$$

which is subject to the *isoperimetric* or integral constraints

$$\int_0^1 \overline{h}(s, \overline{y}(s), \dot{\overline{y}}(s), \cdots, \overline{y}^{(n)}(s)) ds = \overline{x}_D \tag{48}$$

if the *Lagrangian*

$$\mathcal{L}(s) = f(s, \overline{y}, \cdots) + \overline{\mu}_c \cdot \overline{h}(s, \overline{y}, \cdots) \tag{49}$$

is a solution to the Euler–Lagrange equations

$$\sum_{i=0}^{n} (-1)^i \frac{d^i}{ds^i} \left(\frac{\partial \mathcal{L}}{\partial y_j^i} \right) = 0 \qquad j = 1, \cdots, P. \tag{50}$$

$\overline{y}^i = \frac{d^i \overline{y}}{ds^i}$ and $\overline{\mu}_c$ is the vector of Lagrange multipliers that is associated with constraint (48), and is independent of s. In our problem, the isoperimetric constraints arise from end-effector

position constraints of the form (8). Thus, $\overline{h}(\cdot)$ will take the form $l(s)\overline{u}(s)$; and $\overline{x}_D$ is the desired end-effector location.

With constraint (48) and boundary conditions $\overline{y}^i(0) = \overline{y}_0^i$ and $\overline{y}^i(1) = \overline{y}_1^i$ for $i \in [0, 1, \cdots, n]$, (50) can be solved to find $\overline{y}(s)$, and $\overline{\mu}_c$ that extremize (47). In some cases we may not choose to impose boundary conditions at $s = 1$, in which case the "free" end conditions at $s = 1$ will be [3]:

$$\frac{\partial f}{\partial y_j^i}(1, \overline{y}(1), \cdots) = 0. \tag{51}$$

Existence of solutions to the Euler–Lagrange equations is discussed in [16]. We develop one solution method in Section IV-B for the purposes of computational complexity analysis and for comparison to the joint based approaches of Section II.

Appendix B
Recursive Computation of Macroscopically Serial Manipulator Jacobians Is $O(N)$

Let $\overline{\theta}_i$ denote the set of joint variables of the ith module of a hyper-redundant manipulator comprised of a cascade of *identical* modules. Assume that there are p such joint variables, and n modules, i.e., $N = pn$. Let $g_i(\overline{\theta}_i)$ denote the displacement (e.g., homogeneous 4×4 matrix) between a reference frame attached to module $i-1$ and module i. The location of the end-effector or tool frame with respect to the base frame is thus given by

$$g_{0n} = g_1(\overline{\theta}_1)g_2(\overline{\theta}_2)\cdots g_n(\overline{\theta}_n).$$

In general, for an object whose location in space is given by a displacement $g(t)$, its *body velocity* is computed as

$$g^{-1}\dot{g}$$

where the "$\cdot$" denotes differentiation with respect to time. Note that matrix $g^{-1}\dot{g}$ takes the form

$$\begin{bmatrix} \hat{\omega} & \overline{v} \\ \overline{0}^T & 0 \end{bmatrix}$$

where $\hat{\omega}$ is a 3×3 skew symmetric matrix. We can convert this to 6×1 "twist" coordinates via the "$\vee$" operator:

$$\begin{bmatrix} \overline{v} \\ \overline{\omega} \end{bmatrix} = \begin{bmatrix} \hat{\omega} & \overline{v} \\ \overline{0}^T & 0 \end{bmatrix}^{\vee}.$$

The body Jacobian is thus

$$J_{0n}^b = \left[(g_{0n}^{-1}\frac{\partial g_{0n}}{\partial \theta_{11}})^{\vee} \cdots (g_{0n}^{-1}\frac{\partial g_{0n}}{\partial \theta_{1p}})^{\vee} \cdots (g_{0n}^{-1}\frac{\partial g_{0n}}{\partial \theta_{np}})^{\vee} \right].$$

The first p columns take the form

$$\left(g_n^{-1} g_{n-1}^{-1} \cdots g_1^{-1} \frac{\partial g_1}{\partial \theta_{1i}} g_2 \cdots g_n \right)^{\vee}$$

while columns $jp+1$ to $(j+1)p$ take the form

$$\left(g_n^{-1} g_{n-1}^{-1} \cdots g_j^{-1} \frac{\partial g_j}{\partial \theta_{ji}} g_{j+1} \cdots g_n \right)^{\vee}$$

for $i = 1, \cdots, p$ and $j = 1, \cdots, n-1$.

Hence, to compute the body Jacobian, one needs to compute all sequences of the form $g_j \cdots g_n$ for $j = 1, \cdots (n-1)$, and their inverses. This can be done recursively in $O(n)$ calculations (though it needs $O(n)$ memory storage). Next, the individual matrix columns are computed using a constant time algorithm.

Note that the Jacobian which is used in standard robotics practice is neither the rigorously correct body coordinates Jacobian or spatial coordinates Jacobian. Instead, it is a "hybrid" Jacobian. The hybrid Jacobian, J^H, can be computed as

$$J^H = \begin{bmatrix} R_{0n} & 0 \\ 0 & R_{0n} \end{bmatrix} J_{0n}^b$$

where R_{0n} is the rotation matrix part of g_{0n}, which presumably has already been computed. However, this too is a linear time operation.

Hence, the Jacobian of modular structures can in general be computed in $O(n) = O(N)$ time.

Acknowledgment

The authors would like to thank the anonymous reviewers for many helpful suggestions.

References

[1] J. Baillieul, "Kinematic programming alternatives for redundant manipulators," in *Proc. IEEE Int. Conf. Robotics and Automation*, St. Louis, MO, Mar. 1985, pp. 722–725.
[2] D. R. Baker and C. W. Wampler, "On the inverse kinematics of redundant manipulators," *Int. J. Robot. Res.*, vol. 7 no. 2, pp. 3–21, 1988.
[3] U. Brechtken-Manderscheid, *Introduction to the Calculus of Variations*. New York: Chapman and Hall, 1991.
[4] R. W. Brockett and A. Stokes, "On the synthesis of compliant mechanisms," in *Proc. IEEE Int. Conf. Robotics and Automation*, Sacramento, CA, 1991, pp. 2168–2173.
[5] G. S. Chirikjian, "Theory and applications of hyper-redundant robotic manipulators," Ph.D. dissertation, Dep. Eng. Appl. Sci., California Inst. Technol., Pasadena, May 1992.
[6] ——, "A general numerical method for hyper-redundant manipulator inverse kinematics," in *Proc. IEEE Int. Conf. Robotics and Automation*, Atlanta GA, May 1993.
[7] G. S. Chirikjian and J. W. Burdick, "An obstacle avoidance algorithm for hyper-redundant manipulators," in *Proc. IEEE Int. Conf. Robotics and Automation*, Cincinnatti, OH, May 13–18, 1990.
[8] ——, "Parallel formulation of the inverse kinematics of modular hyper-redundant manipulators," in *Proc. IEEE Int. Conf. Robotics and Automation*, Sacramento, CA, Apr. 1991.
[9] ——, "Kinematics of hyper-redundant locomotion with applications to grasping," in *Proc. IEEE Int. Conf. Robotics and Automation*, Sacramento, CA, Apr. 1991.
[10] ——, "Design and experiments with a 30 degree-of-freedom robot," in *Proc. IEEE Int. Conf. Robotics and Automation*, Atlanta, GA, May 1993.
[11] ——, "A modal approach to the kinematics of hyper-redundant robots," *IEEE Trans. Robot. Automat.*, vol. 10, no. 3, pp. 343–354, June 1994.
[12] ——, "The kinematics of hyper-redundant robotic locomotion," *IEEE Trans. Robot. Automat.*, vol. 11, no. 6, pp. 781–793, Dec. 1995.
[13] G. S. Chirikjian, "A binary paradigm for robotic manipulators," in *Proc. 1994 IEEE Int. Conf. Robotics and Automation*, San Diego, CA, May 1994.
[14] ——, "Hyper-redundant manipulator dynamics: A continuum approximation," *Advanced Robot.*, Special Issue on Highly Redundant Manipulators, vol. 9, no. 3, pp. 217–243.
[15] Y. S. Chung, M. W. Griffis, and J. Duffy, "Repeatable joint displacement generation for redundant robotic systems," *ASME J. Mech. Design*, vol. 116, pp. 11–16, Mar. 1994.
[16] G. M. Ewing, *Calculus of Variations with Applications*. New York: Norton, 1969.

[17] C. A. Klein and C. H. Huang, "Review of the pseudoinverse for control of kinematically redundant manipulators," *IEEE Trans. Syst., Man, Cyber.*, vol. SMC-13, no. 2, pp. 245–250, Mar. 1983.
[18] O. Ma and J. Angeles, "The concept of dynamic isotropy and its applications to inverse kinematics and trajectory planning," in *Proc. IEEE Int. Conf. Robotics and Automation*, Cincinnatti, OH, May 13–18, 1990.
[19] L. E. Scales, *Introduction to Non-Linear Optimization.* New York: Springer-Verlag, 1985.
[20] T. Shamir and Y. Yomdin, "Repeatability of redundant manipulators: Mathematical solution of the problem," *IEEE Trans. Automat. Contr.*, vol. 33, no. 11, pp. 1004–1009, 1988.
[21] B.-Q. Su and D.-Y. Liu, *Computational Geometry: Curve and Surface Modeling.* Boston: Academic, 1989.
[22] K. Suh and J. Hollerbach, "Local versus global torque optimization of redundant manipulators, in *Proc. IEEE Int. Conf. Robotics and Automation*, Raleigh, NC, 1987.
[23] K. Suzumori, S. Iikura, and H. Tanaka, "Development of flexible microactuator and its applications to robotic mechanisms," in *1991 IEEE Conf. Robotics and Automation*, Sacramento, CA, Apr. 1991.
[24] S. Tavakkoli and S. G. Dhande, "Shape synthesis and optimization using intrinsic geometry," in *Proc. ASME Design Conf.*, Chicago, Il, Sept. 16–19, 1990.
[25] Z. Wang and K. Kazerounian, "An efficient algorithm for global optimization in redundant manipulations," *ASME J. Mechanisms, Transmissions, Automation Design*, vol. 111, pp. 488–493, Dec. 1989.
[26] J . F. Wilson and U. Mahajan, "The mechanics and positioning of highly flexible manipulator limbs," *ASME J. Mechanisms, Transmissions, Automation Design.* vol, 111, no. 2, pp. 232–237, June 1989.

Gregory S. Chirikjian (M'93), for a photograph and biography, see this issue, pp. 793.

Joel W. Burdick, for a photograph and biography, see this issue, pp. 793.