OBSTACLE-FREE CONTROL OF THE HYPER-REDUNDANT NASA INSPECTION MANIPULATOR

Robert L. Williams II
James B. Mayhew IV
Ohio University
Athens, OH 45701

Proceedings of the
Fifth National Conference on Applied Mechanisms and Robotics
Cincinnati OH, October 12-15, 1997

Contact author information:
Robert L. Williams II
Assistant Professor
Department of Mechanical Engineering
257 Stocker Center
Ohio University
Athens, OH 45701-2979
Phone: (740) 593-1096
Fax: (740) 593-0476
E-mail: bobw@bobcat.ent.ohiou.edu

OBSTACLE-FREE CONTROL OF THE HYPER-REDUNDANT NASA INSPECTION MANIPULATOR

Robert L. Williams II
James B. Mayhew IV
Ohio University, Athens, OH
bobw@bobcat.ent.ohiou.edu

ABSTRACT

This paper presents a follow-the-leader algorithm for serpentine control of hyper-redundant manipulators. Given an obstacle-free trajectory for the manipulator tip generated by a path planner or teleoperation, the follow-the-leader algorithm ensures whole-arm collision avoidance by forcing ensuing links to follow the same trajectory. The algorithm requires two steps to place the tip on each trajectory point: first, fit the manipulator to the trajectory, tip to base; and second, solve inverse position kinematics, base to tip. The general method is presented and can be used off-line or on-line. It is then applied in a modular manner to a specific serpentine manipulator system, the Payload Inspection and Processing System (PIPS) at NASA Kennedy Space Center. PIPS is designed for inspection of Space Shuttle payloads after integration and prior to launch. The follow-the-leader algorithm completely constrains the kinematic redundancy and limits the manipulator workspace. This limitation is justified in sensitive, cluttered environments such as the Shuttle payload bay. The method has been successfully implemented on prototype hardware at NASA.

1. INTRODUCTION

Interest is growing in hyper-redundant serpentine manipulators for inspection, maintenance, assembly, and manipulation tasks in space, nuclear, undersea, and industrial environments. These devices possess highly kinematically-redundant degrees of freedom (much greater than the seven or eight joints of existing redundant manipulators) which enable snake-like motion through an obstacle field. Implementation of such systems requires development of efficient, modular schemes for controlling this serpentine motion.

Several authors have presented results in this area. Three groups of researchers have proposed use of a "backbone curve" to resolve the redundancy of these manipulators. Salerno (1989) uses parametric curves to place the intermediate links of variable geometry truss manipulators, and the solution is achieved by closed-form relationships. Chirikjian and Burdick (1991) use the backbone curve for the inverse kinematics of modular extensible hyper-redundant manipulators. They formulate the algorithm in a manner suitable for parallel computation. Naccarato and Hughes (1991) compare the backbone curve method to a more "traditional" approach to resolving inverse kinematics. They find reduced real-time computations using the backbone curve method.

Hooper and Tesar (1995) present an efficient inverse kinematics algorithm for serpentine manipulators which allows multiple criteria to be satisfied by the excess degrees-of-freedom. It is applied in simulation to a 21-dof serpentine robot model.

Other methods are based on the pseudoinverse of the manipulator Jacobian matrix. Salerno (1993) solved the inverse kinematics problem for hyper-redundant variable geometry truss manipulators using the pseudoinverse of the Jacobian matrix and projection of objective function gradients into the Jacobian null-space to achieve performance optimization. Byers (1994) used a similar approach, with a potential field for obstacle avoidance.

The current paper implements a general follow-the-leader (FTL) algorithm for serpentine motion control. A similar algorithm was proposed in Asano et.al. (1983). Large volumes are not swept by the manipulator. The method is stable and programmed off-line or executed on-line. Path planning is critical to avoid contact in cluttered environments. The robotics literature is rich with path planning and obstacle avoidance

techniques (e.g. see the 102 references in Sanderson, 1992, and several more in Williams and Sklar, 1996). In this paper the primary path planning mode is human-based teleoperation to define an obstacle-free trajectory for the manipulator tip step-by-step, either on- or off-line. The FTL algorithm ensures whole-arm collision avoidance by forcing each manipulator link to follow the tip trajectory. This algorithm was first presented, verified by graphical simulation, by Williams and Tamasy (1996). The general method is presented and then applied in a modular manner to the Payload Inspection and Processing System (PIPS) at NASA Kennedy Space Center. A discussion of implementation to prototype hardware at NASA is also given.

Inspection of Space Shuttle payloads after integration and prior to launch is essential for launch and mission safety. NASA Kennedy Space Center is developing PIPS for prelaunch inspection and light tasks in the Space Shuttle bay (Pasch, 1990 and Richardson et.al., 1993). This device features a hyper-redundant serpentine truss manipulator for carrying a camera along obstacle-free trajectories to required goal points for inspection. NASA is also interested in serpentine manipulators for in-space construction (Spanos and Berka, 1993) and in-space inspection tasks (Lee et.al., 1994).

2. TELEOPERATED PATH PLANNING

The follow-the-leader (FTL) algorithm requires an obstacle-free trajectory for the manipulator tip. This trajectory can be generated by any path planning algorithm (see references mentioned above). In the current paper, teleoperation is used as the primary path planning method, where a human operator determines the obstacle-free tip trajectory step-by-step via joystick input. Teleoperation may be performed either on-line with actual hardware or off-line with a graphical model of the hardware and workspace.

For spatial obstacle-free trajectory generation via teleoperation, three-dof input is sufficient, which controls relative *XYZ* positions. Manipulator tip orientation (camera pointing vector) is fixed by the relative locations of the last two trajectory points. The three-dof input could be chosen to be $\Delta X, \Delta Y, \Delta Z$. However, for follow-the-leader control, it is more convenient to use spherical coordinates to define the next trajectory point relative to the current trajectory point. Starting from the current trajectory point, the next point is defined using a hand controller to input a radius *P* and two spherical angles, ϕ, θ . As shown in Fig. 1, the hand controller can be aligned with the manipulator tip video monitor so teleoperation is natural. Teleoperation is enhanced by placing two or three cameras in the workspace to provide orthogonal views.

Figure 2 shows the i^{th} teleoperation step where the next trajectory point ${}^0\{{}^0P_{i+1}\}$ is determined based on the current trajectory point ${}^0\{{}^0P_i\}$ using the following vector-loop-closure equation. Input motion is relative to the manipulator tip coordinate frame $\{i\}$.

$$ {}^0\{{}^0P_{i+1}\} = {}^0\{{}^0P_i\} + {}^0\{{}^iP_{i+1}\} \tag{1} $$

(Note: ${}^A\{{}^BP_C\}$ is the vector to the origin of frame $\{C\}$ from the origin of frame $\{B\}$, expressed in the coordinates of frame $\{A\}$, (Craig, 1989). If $\{A\}$ is omitted, it is assumed to be $\{B\}$.) Coordinate frame $\{i+1\}$ is obtained by two rotations relative to $\{i\}$: 1) ϕ about X_i; and 2) θ about Y_{i+1} (the *Y*-axis resulting from the first rotation). This sequence is an *X-Y* (ϕ, θ) Euler rotation. The relative vector ${}^0\{{}^iP_{i+1}\}$ for Eq. 1 is found from:

$$ {}^0\{{}^iP_{i+1}\} = {}^0_iR\,{}^i\{{}^iP_{i+1}\} \tag{2} $$

The relative vector ${}^i\{{}^iP_{i+1}\}$ in frame $\{i\}$ is produced by rotating vector ${}^iP = \{0 \quad 0 \quad P\}^T$ through the *X-Y* (ϕ, θ) Euler rotation sequence described above:

$$ {}^i\{{}^iP_{i+1}\} = R_X(\phi)R_Y(\theta)\,{}^iP = \begin{Bmatrix} Ps\theta \\ -Pc\theta s\phi \\ Pc\theta c\phi \end{Bmatrix} \tag{3} $$

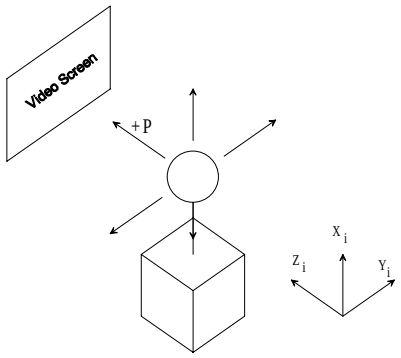

Figure 1. Teleoperation Hand Controller

In Eq. 2, the rotation matrix ${}^0_iR$ must be initialized to the starting orientation. The rotation matrix ${}^0_iR$ must be updated after each successful teleoperation input as follows, to prepare for the next input step:

$$ {}^0_iR \rightarrow {}^0_{i+1}R = {}^0_iR\,{}^i_{i+1}R \tag{4} $$

Where the rotation matrix ${}^i_{i+1}R$ comes from the current *X-Y* (ϕ, θ) Euler rotation sequence,

$$ {}^i_{i+1}R = R_X(\phi)R_Y(\theta) \tag{5} $$

3. FOLLOW-THE-LEADER ALGORITHM

This section presents the general, modular follow-the-leader (FTL) algorithm for serpentine motion control of hyper-redundant manipulators. Given an obstacle-free trajectory for the manipulator tip, the FTL algorithm ensures obstacle-free motion for the entire manipulator by forcing ensuing links to follow the tip link. The manipulator must be serial, or an in-parallel-

actuated variable geometry truss with a serial model (Williams, 1995a). The FTL algorithm moves the manipulator tip along the given trajectory from the start to the end. The FTL algorithm controls the locations of spine points along the manipulator.

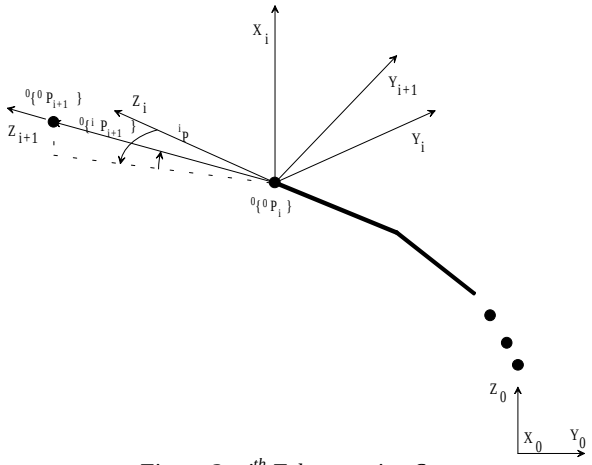

Figure 2. i^{th} Teleoperation Step

In this paper, a serpentine manipulator (SM) is assumed to have a motion base with a minimum of three-dof to position the base-most spine point at general locations. The motion base is used to feed the SM into the trajectory. The serpentine portion is assumed to be modular (allowing tapered modularity), with two-dof modules. The method can handle non-modular SMs, but the solution will be non-modular. Two-dof are required to position one point relative to another along the given trajectory. If each SM module is composed of intersecting revolute joints (such as universal joints or roll-pitch joints) every spine point can be placed on the trajectory. For general SMs consisting of offset revolute joints, only every other spine point can be placed on the trajectory. The FTL algorithm controls only the SM spine; therefore, the path planning algorithm must provide trajectories with sufficient clearance to allow collision-free motion considering the manipulator dimensions about the spine. Figure 3 gives the flow chart for the FTL algorithm.

An obstacle-free trajectory for the manipulator tip must be generated and discretized into piecewise linear segments off-line (teleoperation automatically provides this discretization). The variable SM home position is defined as the fully retracted motion base position, with the SM portion straight out along the variable feed line. The manipulator tip in the home position is the trajectory start point. All trajectory points must be reachable subject to joint limits.

To place the manipulator tip at a given trajectory point, FTL performs two steps: 1) The manipulator is shaped to the trajectory from tip to base. The tip is placed on the current trajectory point and in-board spine point locations are calculated by intersecting the manipulator segment link sphere with the appropriate trajectory straight-line segment. Each solution becomes the sphere center for the next link. The process, pictured in Fig. 4, continues until the base-most spine point is placed on the trajectory. For this purpose, the trajectory is appended with the feed line from the base to the first trajectory point. In Fig. 4, the module lengths are Q_i and the spine points are numbered 1,2,...,*n*. The trajectory is shown in solid lines and the SM links in dotted lines. 2) Inverse position kinematics then calculates the required joint values, from the base to the tip. This solution is decoupled into the motion base solution, followed by *n* applications of the modular inverse position solution for the *n* SM modules. The resulting joint values place the spine points on the trajectory.

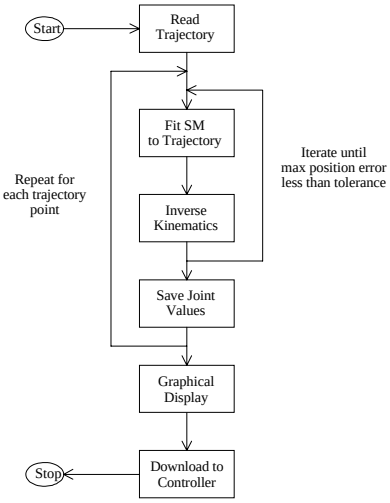

Figure 3. Follow-the-Leader Algorithm Flowchart

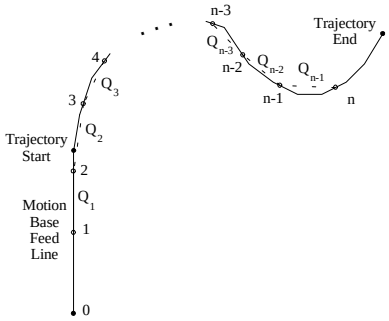

Figure 4. Fit SM to Given Trajectory

For SM modules with intersecting revolute joints the above steps are sufficient. However, for SMs with joint offsets, the link sphere radii are not fixed but functions of the intermediate joint angles. Therefore, steps 1) and 2) must be performed iteratively (the process starts with straight-out values for link radii) until the position errors between the desired spine points on the trajectory and the actual spine points achieved by inverse kinematics are sufficiently small. This is the inner loop in Fig. 3.

This process is repeated for each point on the trajectory, indicated by the outer loop of Fig. 3. At each step, the joint values are saved. Smooth serpentine motion may be obtained by providing a fine trajectory discretization. Retraction of the SM along the same obstacle-free trajectory is accomplished by reversing the joint values array.

4. IMPLEMENTATION TO PIPS

This section presents implementation of the general follow-the-leader algorithm to the Payload Inspection and Processing System (PIPS) at NASA Kennedy Space Center. For more

detailed reports, see (Williams, 1995b and 1996). PIPS features a hyper-redundant serpentine truss manipulator (STM) for prelaunch inspection of Shuttle bay payloads.

Eighteen-dof prototype PIPS hardware has been built, with a two-dof motion base and sixteen-dof STM. The motion base has a prismatic joint d_1 and a pitch joint θ_2, shown in Fig. 5. Figure 5 gives the zero position for θ_2; nominal FTL trajectories have $\theta_2 = -90^\circ$ so the spine is aligned with d_1. Frame {0} is the base coordinate frame for the system. Frame {3} is aligned 45° from horizontal. Fig. 5 also shows the first two STM spine segments.

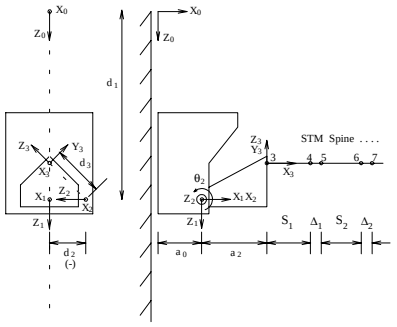

Figure 5. Motion Base for Prototype PIPS

The STM is a sixteen-dof tetrahedral variable geometry truss (VGT) as shown in Fig. 6. The rotation axes are orthogonal, rather than spaced by 120° as in the standard tetrahedron VGT (Subramaniam and Kramer, 1992). Also, the structure is tapered and there are joint offsets. The corners of the solid tetrahedra along the manipulator from base to tip, opposite the linear actuators, are the spine of the STM. Figure 6 is mirror-image of the as-built hardware, with one segment missing.

Figure 6. Prototype STM

Figure 7 shows the as-built kinematic diagram for the STM. Views A and B are "flattened" about the spine. There are sixteen linear actuators $L_3, L_4, \cdots, L_{18}$, controlling the angles $\theta_3, \theta_4, \cdots, \theta_{18}$ about axes $Z_3, Z_4, \cdots, Z_{18}$. All *X* axes are along the spine. Figure 7 presents the zero position for all STM angles. The spine points *i* are the origins of coordinate frames {*i*}. Spine point 3 is attached to the motion base, and spine point 18 is associated with the last moving joint. The link lengths are given for each segment *i*, where S_i is the major length and Δ_i is the joint offset. Each joint pair is an offset universal joint. Table 1 gives the Denavit-Hartenberg (DH) parameters (Craig, 1989) for the prototype PIPS hardware. The modular pattern established by rows 4,5 and 6,7 and concluded by rows 16,17 are repeated for rows 8-15.

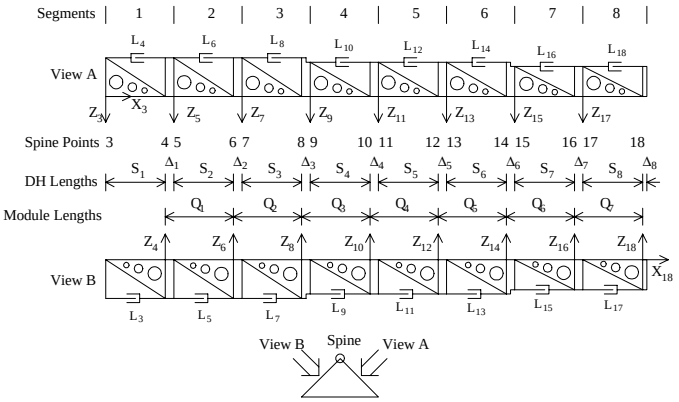

Figure 7. Kinematic Diagram for Prototype STM

Table 1. Prototype PIPS DH Parameters

i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	a_0	d_1	0
2	-90	0	d_2	θ_2
3	-45	a_2	d_3	θ_3
4	-90	S_1	0	θ_4
5	90	Δ_1	0	θ_5
6	-90	S_2	0	θ_6
7	90	Δ_2	0	θ_7
8 - 15	...	...	...	...
16	-90	S_7	0	θ_{16}
17	90	Δ_7	0	θ_{17}
18	-90	S_8	0	θ_{18}

4.1 Spine Points and STM Modules

Given a serpentine truss manipulator (STM) to control in follow-the-leader (FTL) mode, the first step is to select the spine points and identify the repeating modules. Since three-dof are required to position a point, the first serpentine joint θ_3 is combined with the motion base variables d_1 and θ_2 in order to position the first spine point 4. Two-dof are required to position a point a fixed distance from another point along a given trajectory; therefore, every second spine point following 4 can be fit to the trajectory. There are a total of eight spine points for the prototype PIPS hardware: 4, 6, 8, 10, 12, 14, 16, and 18. In this paradigm, STM joint values from neighboring segments are treated as modules to place each ensuing spine point. Figure 8 shows the spine of general STM module *i* controlling spine point *j*+2 with respect to spine point *j* using STM joint angles θ_{Ei} and θ_{Oi} (*E*, *O* stand for even and odd).

The intermediate spine point *j*+1 cannot be placed in general on the given trajectory. The fixed lengths are Δ_i and S_{i+1}, from segments *i* and *i*+1, respectively. The DH parameters for Module *i* are given in Table 2.

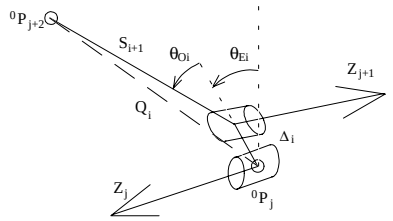

Figure 8. General Prototype STM Module

Table 2. DH Parameters for Module *I*

i	α_{i-1}	a_{i-1}	d_i	θ_i
j	-90	S_i	0	θ_{Ei}
$j+1$	90	Δ_i	0	θ_{Oi}
$j+2$	-90	S_{i+1}	0	θ_{Ei+1}

The variable distance Q_i from spine points j to j+2 is the magnitude of the vector $^jP_{j+2}$; it is a function of the intermediate joint angle θ_{Oi}:

$$\begin{Bmatrix} \begin{Bmatrix} ^jP_{j+2} \end{Bmatrix} \\ 1 \end{Bmatrix} = {}_{j+1}^{j}T \begin{Bmatrix} \begin{Bmatrix} ^{j+1}P_{j+2} \end{Bmatrix} \\ 1 \end{Bmatrix} = \begin{bmatrix} c\theta_{Oi} & -s\theta_{Oi} & 0 & \Delta_i \\ 0 & 0 & -1 & 0 \\ s\theta_{Oi} & c\theta_{Oi} & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{Bmatrix} S_{i+1} \\ 0 \\ 0 \\ 1 \end{Bmatrix}$$

$$^jP_{j+2} = \begin{Bmatrix} \Delta_i + S_{i+1}c\theta_{Oi} \\ 0 \\ S_{i+1}s\theta_{Oi} \end{Bmatrix}$$

$$Q_i(\theta_{Oi}) = \left\| {}^jP_{j+2} \right\| = \sqrt{\Delta_i^2 + 2\Delta_i S_{i+1} c\theta_{Oi} + S_{i+1}^2} \qquad (6,7,8)$$

Table 3 summarizes the spine point and module paradigm for the prototype hardware. The first row of Table 3 gives the motion base information; the seven STM modules follow.

Table 3. STM Modules

Module i	Spine Point	DH Lengths	Joint Variables	Module Length
Base	4	-	d_1, θ_2, θ_3	-
1	6	Δ_1, S_2	θ_4, θ_5	Q_1
2	8	Δ_2, S_3	θ_6, θ_7	Q_2
3	10	Δ_3, S_4	θ_8, θ_9	Q_3
4	12	Δ_4, S_5	θ_{10}, θ_{11}	Q_4
5	14	Δ_5, S_6	θ_{12}, θ_{13}	Q_5
6	16	Δ_6, S_7	θ_{14}, θ_{15}	Q_6
7	18	Δ_7, S_8	θ_{16}, θ_{17}	Q_7

In this paradigm, the last STM joint angle θ_{18} is not required to place the last spine point, 18. It can be used in conjunction with a wrist mechanism for fine camera pointing. For the prototype STM hardware, module index i and spine point index j are related by $j = 2(i+2)$.

4.2 General Feed-Line

The prototype hardware is designed primarily to push the serpentine manipulator onto trajectories straight-out along the prismatic joint d_1. A greater motion range is possible if general feed lines are enabled. See Williams (1996) for more detail.

Figure 9 shows the general feed line trajectory geometry. Length L_1 is the X_0 distance from the origin to point {4} in the reset position and length L_2 is the straight STM distance from spine points {4} to {18}. As discussed in Section 4.1, joints d_1, θ_2, and θ_3 are used to push spine point {4} onto trajectories. Since spine point {4} is the first to be pushed onto the trajectory, the feed line starts at the nominal reset position for this spine point. The initial position for the STM tip, spine point {18}, is determined by a sequence of two rotations of length L_2: 1) α about Y_0; and 2) β about X_0. This sequence is a *Y-X* (α, β) fixed rotation, described by the rotation matrix:

$${}_F^0R = R_X(\beta)R_Y(\alpha) = \begin{bmatrix} c\alpha & 0 & s\alpha \\ s\alpha s\beta & c\beta & -c\alpha s\beta \\ -s\alpha c\beta & s\beta & c\alpha c\beta \end{bmatrix} \qquad (9)$$

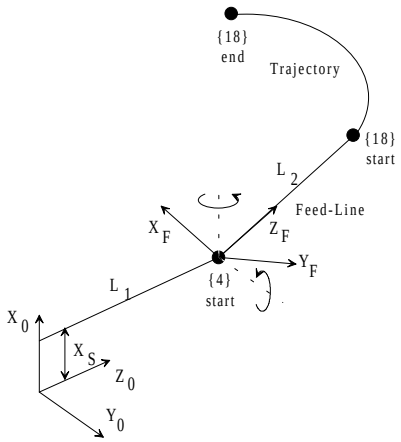

Figure 9. General Feed Line Geometry

For all trajectories, the first two points are: $^FP_1 = \{0 \quad 0 \quad 0\}^T$ and $^FP_2 = \{0 \quad 0 \quad L_2\}^T$. The first point cannot be reached by the STM tip but must be defined in order to intersect the STM back onto the feed line. If remaining path is determined in the $\{F\}$ frame, we must first transform all trajectory points to $\{0\}$: $^0P_i = {}_F^0T\,^FP_i$, where:

$${}_F^0T = \begin{bmatrix} c\alpha & 0 & s\alpha & X_S \\ s\alpha s\beta & c\beta & -c\alpha s\beta & 0 \\ -s\alpha c\beta & s\beta & c\alpha c\beta & L_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (10)$$

4.3 Fit STM to Trajectory

To fit the STM to a given trajectory, the following process is used. The manipulator tip (the last spine point) is placed on the current trajectory point. The next in-board spine point is fit to the trajectory by intersecting a sphere (centered at the last spine point and radius equal to the module length) with the piecewise linear trajectory. The module lengths initialize at their maximum values and are updated by Eq. 8 with the current joint angles after each inverse kinematics solution. The intersection attempt starts at the trajectory line segment containing the current trajectory point. If no intersection is found, preceding trajectory segments are tried until the intersection is found. For ensuing modules, the newly calculated spine point becomes the sphere center and the next in-board module length the radius. This process repeats until the first spine point is placed on the trajectory (which includes the feed line). The intersection of a sphere with a 3D line segment must be solved repeatedly and is presented below.

The parametric form for a line from points P_1 to P_2 is:

$$\begin{array}{lll} x = at + P_{1x} & a = P_{2x} - P_{1x} & t = 0 @ P_1 \\ y = bt + P_{1y} & b = P_{2y} - P_{1y} & t = 1 @ P_2 \\ z = ct + P_{1z} & c = P_{2z} - P_{1z} & \end{array} \tag{11}$$

The equation of a sphere with radius *r* and center x_c, y_c, z_c is:

$$(x - x_c)^2 + (y - y_c)^2 + (z - z_c)^2 = r^2 \tag{12}$$

The intersection of the sphere surface and line segment is found by substituting *x,y,z* from Eqs. 11 into Eq. 12, yielding:

$$At^2 + Bt + C = 0 \tag{13}$$

$$\text{where:} \quad \begin{array}{ll} A = a^2 + b^2 + c^2 & d = P_{1x} - x_c \\ B = 2(ad + be + cf) & e = P_{1y} - y_c \\ C = d^2 + e^2 + f^2 - r^2 & f = P_{1z} - z_c \end{array} \tag{14}$$

Equation 13 is solved for *t* using the quadratic formula, and *x, y, z* is evaluated from Eqs. 11. There are two solutions, easily seen by imagining a pencil passed through a softball. The following intersection conditions exist for each of the two solutions.

- If $0 < t_i < 1$, the intersection lies on the line segment.
- If $t_i < 0$ or $t_i > 1$ the intersection lies on the line but off the line segment.
- If $t_1 = t_2$ the one intersection occurs where the line is tangent to the sphere; the above two rules apply.
- If $t_{1,2}$ is imaginary, no intersection exists.

4.4 Inverse Kinematics

Given an STM fit to a given trajectory, this section presents the second basic FTL component, inverse position kinematics. This problem solves the joint values given the spine points. First, the motion base solution is presented, followed by the general STM module *i* solution.

4.4.1 Motion Base Solution. The motion base inverse position problem is: Given ${}^0P_4$, calculate d_1, θ_2, θ_3. The DH parameters are the first three lines of Table 1. The kinematic diagram is shown in Fig. 5. Equation 15 gives the transformation relating the variables to the given information.

$${}^0P_4 = {}^0_3T\,{}^3P_4 = {}^0_1T(d_1)\,{}^1_2T(\theta_2)\,{}^2_3T(\theta_3)\begin{Bmatrix} S_1 \\ 0 \\ 0 \end{Bmatrix} \tag{15}$$

This expands to Eq. 16, where ${}^0P_4$ is given and $K = \frac{\sqrt{2}}{2}$.

$${}^0P_4 = \begin{Bmatrix} X \\ Y \\ Z \end{Bmatrix} = \begin{Bmatrix} a_0 + c\theta_2(S_1 c\theta_3 + a_2) - Ks\theta_2(S_1 s\theta_3 + d_3) \\ d_2 - K(S_1 s\theta_3 - d_3) \\ d_1 - s\theta_2(S_1 c\theta_3 + a_2) - Kc\theta_2(S_1 s\theta_3 + d_3) \end{Bmatrix} \tag{16}$$

The solution of Eq. 16 follows the order: 1) θ_3; 2) θ_2; 3) d_1. From the *Y* component of Eq. 16:

$$\theta_3 = \sin^{-1}\left(\frac{d_2 + Kd_3 - Y}{KS_1}\right) \tag{17}$$

There is a unique solution: considering joint limits, the inverse sine ambiguity presents no problem. Given the θ_3 solution, the *X* component of Eq. 16 yields:

$$E\cos\theta_2 + F\sin\theta_2 + G = 0 \qquad \begin{array}{l} E = S_1 c\theta_3 + a_2 \\ F = -K(S_1 s\theta_3 + d_3) \\ G = a_0 - X \end{array} \tag{18}$$

Equation 18 is solved using the tangent half-angle substitution (Craig, 1989), which yields two valid solutions for θ_2. The solution chosen must lie within joint limits, closest to the nominal STM home position, $\theta_2 = -90^\circ$.

$$\theta_{2_{1,2}} = 2\tan^{-1}\left(\frac{-F \pm \sqrt{E^2 + F^2 - G^2}}{G - E}\right) \tag{19}$$

Given the θ_3 and θ_2 solutions, the *Z* component of Eq. 16 solves the prismatic joint variable d_1. There is one solution for each θ_2, θ_3.

$$d_1 = Z + s\theta_2(S_1 c\theta_3 + a_2) + Kc\theta_2(S_1 s\theta_3 + d_3) \tag{20}$$

4.4.2 Module Solution. When the solutions for the motion base variables are known, the remaining STM joint angles can be found, proceeding from the base to the manipulator tip. The prototype STM hardware consists of repeating modules, as discussed previously. Though the STM is tapered so the DH lengths are not repeating, the kinematic structure of the solution is identical for each module. The general solution derived

below is applied seven times, from module one through seven. Since θ_{18} is not required to position the STM tip, spine point 18, it is set to zero, and can be used in conjunction with camera pointing after the serpentine motion is complete.

The general module inverse kinematics problem is: Given two consecutive spine point locations, plus the previous joint values, calculate θ_{Ei},θ_{Oi}. The vector difference between neighboring spine points can be expressed in two ways which are equated (see Fig. 8):

$$ {}^{0}P_{j+2} - {}^{0}P_{j} = {}^{0}_{j}R\,{}^{j}P_{j+2} \tag{21} $$

Equation 21 is rearranged to show the dependence on the unknown joint angles.

$$ {}^{0}P_{j+2} - {}^{0}P_{j} = {}^{0}_{j-1}R^{j-1}_{j}R(\theta_{Ei})\,{}^{j}P_{j+2}(\theta_{Oi}) \tag{22} $$

The left-hand-side vectors are given from the known spine points, referred to the {0} frame. The rotation matrix ${}^{0}_{j-1}R = {}^{0}_{1}R\,{}^{1}_{2}R\cdots{}^{j-2}_{j-1}R$ is a known function of previously-determined joint angles.

$$ {}^{0}_{j-1}R^{T}\left({}^{0}P_{j+2} - {}^{0}P_{j}\right) = \begin{Bmatrix} x \\ y \\ z \end{Bmatrix} = {}^{j-1}_{j}R(\theta_{Ei})\,{}^{j}P_{j+2}(\theta_{Oi}) \tag{23} $$

The vector ${}^{j}P_{j+2}(\theta_{Oi})$ was given in Eq. 7. The rotation matrix expressing the unknown θ_{Ei} is:

$$ {}^{j-1}_{j}R(\theta_{Ei}) = \begin{bmatrix} c\theta_{Ei} & -s\theta_{Ei} & 0 \\ 0 & 0 & 1 \\ -s\theta_{Ei} & -c\theta_{Ei} & 0 \end{bmatrix} \tag{24} $$

Substituting these values into Eq. 23 yields:

$$ \begin{Bmatrix} x \\ y \\ z \end{Bmatrix} = \begin{Bmatrix} c\theta_{Ei}\left(\Delta_i + S_{i+1}c\theta_{Oi}\right) \\ S_{i+1}s\theta_{Oi} \\ -s\theta_{Ei}\left(\Delta_i + S_{i+1}c\theta_{Oi}\right) \end{Bmatrix} \tag{25} $$

In Eq. 25, the left-hand-side is known, while the right-hand-side contains the unknowns θ_{Ei},θ_{Oi}. Solution of these unknown joint angles is obtained by equating like components. θ_{Ei} is solved from a ratio of the *z* to the *x* component. The four-quadrant inverse tangent function, *atan2*, must be used in Eq. 26.

$$ \frac{z}{x} = \frac{-s\theta_{Ei}\left(\Delta_i + S_{i+1}c\theta_{Oi}\right)}{c\theta_{Ei}\left(\Delta_i + S_{i+1}c\theta_{Oi}\right)}; \qquad \theta_{Ei} = a\tan 2(-z, x) \tag{26} $$

θ_{Oi} is solved from a ratio of the *y* to the *x* component using *atan2*.

$$ \frac{y}{\dfrac{x}{c\theta_{Ei}} - \Delta_i} = \frac{S_{i+1}s\theta_{Oi}}{S_{i+1}c\theta_{Oi}}; \qquad \theta_{Oi} = a\tan 2\left(y, \frac{x}{c\theta_{Ei}} - \Delta_i\right) \tag{27} $$

This completes the general solution for θ_{Ei},θ_{Oi}. Joint angle pairs θ_4,θ_5 through θ_{16},θ_{17} are solved from Eqs. 26 and 27.

4.5 Convergence

As shown in Fig. 3, iteration is required due to the joint offsets. However, if the initial segment radii are chosen to be $Q_i = \Delta_i + S_i$ and updated continuously with Eq. 8, only one iteration is required for the entire FTL trajectory to achieve tip errors less than 0.1 inch. Therefore, to achieve this adequate error tolerance, the solution is essentially closed-form for all tested trajectories.

5. IMPLEMENTATION TO PROTOTYPE HARDWARE

This section discusses prototype hardware implementation of the FTL algorithm at NASA KSC.

5.1 Prototype Hardware

The kinematic structure of the prototype hardware is presented in Section 4. Each joint is driven by a ball-screw mechanism. Each ball screw is actuated by a rotary stepper motor through a gear box. The feedback element for each joint is a LVDT across the ball-screw which gives a linear relationship between the length and the output voltage. Eighteen commands are sent to the eighteen joint controllers via serial communication from the host control PC. Currently teleoperation is implemented through keyboard inputs; a 3-dof joystick would greatly improve teleoperation.

5.2 Calibration and Mapping

The FTL algorithm output is d_1, followed by seventeen joint angles. Low-level control of the serpentine truss manipulator is through voltage commands to all eighteen joints. Therefore, two transformations are required for each joint: 1) Angle-to-length calibration for each joint excluding the first (prismatic) joint; and 2) Length-to-voltage mapping for all joints. The length-to-voltage mapping is a linear relationship. Both transformations required extensive physical measurements of the prototype hardware. For more details, see Williams (1996).

5.3 Programming Options

There are several off- and on-line programming options for the prototype hardware system. The FTL algorithm was originally developed in the MATLAB simulation environment with basic graphical animation. An IGRIP (then upgraded to TELEGRIP) model was developed to display the MATLAB results with greater graphical fidelity. The low-level manipulator control program was written in *C* code and the operational FTL algorithm in *C++* code interfaces with it. To ensure safety in sensitive environments, off-line trajectory development and simulation is performed before the hardware is operated.

The prototype hardware can be run under FTL via two main options: 1) Teleoperation; and 2) Run Trajectories.

With teleoperation, the user specifies an obstacle-free trajectory on-line, step-by-step. Under trajectories there are three sub-options: a) Load Existing *XYZ* Trajectory (from off-line teleoperation with the graphical model or from a model-based path planning algorithm); b) Calculate *XYZ* Trajectory (by combining basic curve primitives); and c) Load Voltage Commands (created by the MATLAB code under any of the possible modes). In all cases, retraction of the manipulator after FTL motion is accomplished by reversing the joint commands.

5.4 STS-82 Payload Hardware Simulation

The prototype hardware FTL capability was demonstrated for inspection of the STS-82 payload, which is scheduled to fly in February, 1997 on the second Hubble Telescope repair mission. The hardware setup includes the eighteen-dof STM, Shuttle pallet, and mock-up STS-82 payload.

Four follow-the-leader trajectories were developed and implemented in hardware control to demonstrate representative inspection locations and tasks for the STS-82 payload. All trajectories were developed free of hardware joint limits and proved to be free of collisions in hardware. Figures 10 are photographs of the initial and final STM configurations for one of these trajectories. A videotape (Williams et.al. 1997) was produced to summarize the FTL simulation and hardware results.

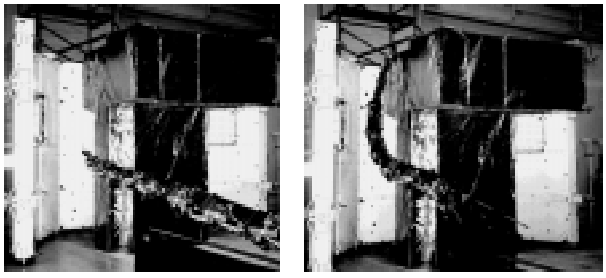

Figure 10. Initial and Final STM Configurations

6. CONCLUSION

This paper presents development of a follow-the-leader (FTL) algorithm for whole-arm obstacle-free control of hyper-redundant serpentine manipulators. It was successfully applied to the Payload Inspection and Processing System (PIPS) at NASA Kennedy Space Center. The algorithm was developed, implemented, and tested in a modular manner for the prototype hardware, and validated using computer graphics and hardware control. Given an obstacle-free trajectory for the manipulator tip (from teleoperation or path planner), the FTL algorithm ensures whole-arm collision avoidance for the manipulator by forcing ensuing links to follow the trajectory.

Implementation of the FTL algorithm to prototype hardware yielded several design lessons which should be improved in the final hardware: 1) Joint offsets should be zero; 2) Motion base translational travel should be equal to the STM length; 3) Motion base must have more range in three dimensions; 4) Joint limits should be increased; 5) Control system must allow all motors to reach each command set in the same time interval; 6) The hardware must be lighter yet stiffer; 7) Actuation redundancy should be provided for joint failures; and 8) LVDT voltage noise must be reduced.

REFERENCES

K. Asano, M. Obama, Y. Arimura, M. Kondo, and Y. Hitomi, 1983, "Multi-Joint Inspection Robot", *IEEE Transactions on Industrial Electronics*, Vol. IE-30, pp. 277-281.

R.M. Byers, 1994, "Automated Path Planning of the Payload Inspection and Processing System", Final Report, NASA/ ASEE Summer Faculty Fellowship Program, NASA KSC.

G.S. Chirikjian and J.W. Burdick, 1991, "Parallel Formulation of the Inverse Kinematics of Modular Hyper-Redundant Manipulators", *Proceedings of the 1991 IEEE International Conference on Robotics and Automation*, Sacramento, CA, pp. 708-713.

J.J. Craig, 1989, *Introduction to Robotics: Mechanics and Control*, Addison-Wesley Publishing Co., Inc., Reading, MA.

R. Hooper and D. Tesar, 1995, "Motion Coordination Based on Multiple Performance Criteria with a Hyper-Redundant Serial Robot Example", *Proceedings of the 10th IEEE International Symposium on Intelligent Control*, pp. 133-138.

T.S. Lee, T. Ohms, and S. Hayati, 1994, "A Highly Redundant Robot System for Inspection", AIAA Paper AIAA-94-1194-CP, *Proceedings of the Conference on Intelligent Robotics in Field, Factory, Service, and Space (CIRFFSS '94)*, Houston, TX.

F. Naccarato and P.C. Hughes, 1991, "Inverse Kinematics of Variable-Geometry Truss Manipulators", *Journal of Robotic Systems*, Vol. 8, No. 2, pp. 249-266.

K. Pasch, 1990, "Self-Contained Deployable Serpentine Truss for Prelaunch Access of the Space Shuttle Orbiter Payloads", NAS-1659-FM-9106-387, Final Report, Contract NAS 10-11659, NASA KSC.

B. Richardson, M. Sklar, and M. Fresa, 1993, "PCR Inspection and Processing Robot Study, Final Report", McDonnell Douglas Space Systems - Kennedy Space Division.

R.J. Salerno, 1989, "Shape Control of High Degree-of-Freedom Variable Geometry Truss Manipulator", *M.S. Thesis*, VPI&SU, Blacksburg, VA.

R.J. Salerno, 1993, "Positional Control Strategies for a Modular, Long-Reach, Truss-Type Manipulator", *Ph.D. Dissertation*, VPI&SU, Blacksburg, VA.

A.C. Sanderson, 1992, "Path Planning for Robotic Truss Assembly", Final Report, Contract NAG-1-1413, NASA Langley Research Center.

P.D. Spanos and R.B. Berka, 1993, "Development of a Large Space Robot: A Multi-Segment Approach", AIAA Paper AIAA-93-1463-CP, *Proceedings of the 34th AIAA/ASME/ ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, LaJolla, CA.

M. Subramaniam and S.N. Kramer, 1992, "The Inverse Kinematic Solution of the Tetrahedron Based Variable-Geometry Truss Manipulator", *ASME Journal of Mechanical Design*, Vol. 114, pp. 433-437.

R.L. Williams II and M. Sklar, 1996, "Manipulator Path Planning for Redundant Arms", *AIAA Forum on Advanced Developments in Space Robotics*, Madison, WI.

R.L. Williams II and G. Tamasy, 1996, "Follow-the-Leader Control for the Payload Inspection and Processing System", *Proceedings of the 1996 ASME 24th Biennial Mechanisms Conference*, Irvine, CA.

R.L. Williams II, 1995a, "Survey of Active Truss Modules", *Proceedings of the 21st Design Automation Conference*, Boston, MA.

R.L. Williams II, 1995b, "Follow-the-Leader Algorithm for the Payload Inspection and Processing System", Final Report, NASA/ ASEE Summer Faculty Fellowship Program, NASA KSC.

R.L. Williams II, 1996, "Follow-the-Leader Control for the Payload Inspection and Processing System Prototype Hardware", Final Report, NASA/ ASEE Summer Faculty Fellowship Program, NASA KSC.

R.L. Williams II, J.B. Mayhew, T. Lippitt, and C. Cooper, 1997, "Follow-the-Leader Control of Hyper-Redundant Serpentine Truss Manipulator Prototype Hardware", submitted to the *1997 IEEE Conference on Automation and Robotics Video Proceedings*, Albuquerque.