1995 NASA/ASEE SUMMER FACULTY FELLOWSHIP PROGRAM
JOHN F. KENNEDY SPACE CENTER
UNIVERSITY OF CENTRAL FLORIDA

532-61
7762
P. 29

FOLLOW-THE-LEADER ALGORITHM FOR THE PAYLOAD INSPECTION AND PROCESSING SYSTEM

Dr. Robert L. Williams II
Assistant Professor
Mechanical Engineering Department
Ohio University
Athens, Ohio

KSC Colleagues - Eduardo Lopez del Castillo and Gabor Tamasi
Robotics

Contract Number NASA-NGT-60002
Supplement 19

August 18, 1995

FOLLOW-THE-LEADER ALGORITHM FOR THE PAYLOAD INSPECTION AND PROCESSING SYSTEM

Robert L. Williams II, Ph.D.

ABSTRACT

This report summarizes the author's summer 1995 work at NASA Kennedy Space Center in the Advanced Systems Division. The assignment was path planning for the Payload Inspection and Processing System (PIPS). PIPS is an automated system, programmed off-line for inspection of Space Shuttle payloads after integration and prior to launch. PIPS features a hyper-redundant 18-dof serpentine truss manipulator capable of snake-like motions to avoid obstacles. The path planning problem was divided into two segments: 1) Determining an obstacle-free trajectory for the inspection camera at the manipulator tip to follow; and 2) Development of a follow-the-leader (FTL) algorithm which ensures whole-arm collision avoidance by forcing ensuing links to follow the same tip trajectory.

The summer 1995 work focused on the FTL algorithm. This report summarizes development, implementation, testing, and graphical demonstration of the FTL algorithm for prototype PIPS hardware. The method and code was developed in a modular manner so the final PIPS hardware may use them with minimal changes. The FTL algorithm was implemented using MATLAB software and demonstrated with a high-fidelity IGRIP model. The author also supported implementation of the algorithm in C++ for hardware control.

The FTL algorithm proved to be successful and robust in graphical simulation. The author intends to return to the project in summer 1996 to implement path planning for PIPS.

ACKNOWLEDGMENTS

The author acknowledges Gabor Tamasi of the Advanced Systems Division at KSC for an interesting theoretical and hardware problem, and for guidance in its solution. Many thanks also to Carey Cooper of Intergraph Corporation who performed the IGRIP modeling and provided enlightening insights.

TABLE OF CONTENTS

FIGURE LIST

TABLE LIST

I. INTRODUCTION

1.1 BACKGROUND

Inspection of Space Shuttle payloads after integration and prior to launch is essential for launch and mission safety. Currently, this inspection is completed by humans, which is dangerous, costly, labor-intensive, and not versatile in the cluttered and sensitive Shuttle bay environment. With shrinking budgets, development of efficient, labor saving methods are warranted. Therefore, the Advanced Systems Division at NASA Kennedy Space Center (KSC) is developing an automated tool, the Payload Inspection & Processing System (PIPS), for prelaunch inspection and light tasks in the Space Shuttle bay [1],[2]. Figure 1 shows the design concept for PIPS. This unique device features a hyper-redundant serpentine truss manipulator (STM) for carrying a camera along obstacle-free trajectories to required goal points for inspection. The prototype PIPS hardware (built by Foster-Miller, Inc. [3] and modified by NASA) has eighteen degrees-of-freedom. The development of PIPS hardware for use at either lauchpad Payload Changeout Room (PCR) is underway. NASA is also interested in serpentine manipulators for in-space construction [4] and in-space inspection tasks [5].

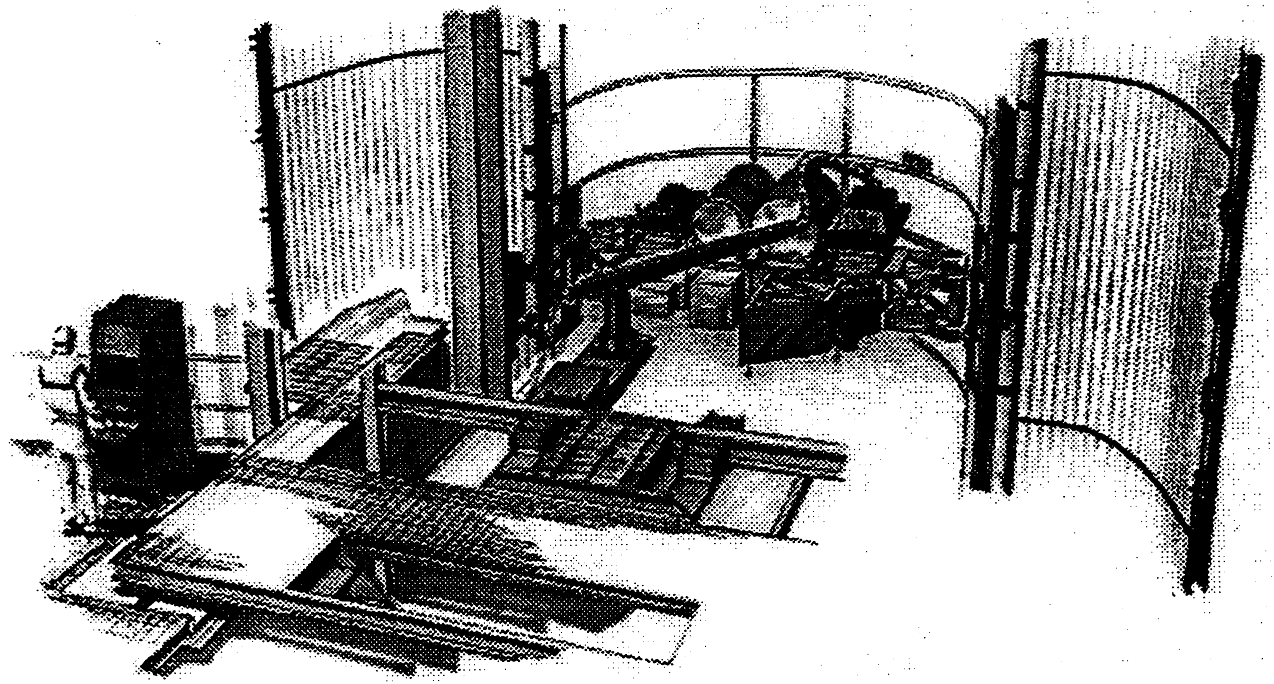


Figure 1.1 PIPS Conceptual Design

1.2 THE PATH PLANNING PROBLEM

PIPS is an automated system which will be programmed off-line. Path planning is critical for PIPS to avoid contact in the cluttered Shuttle payload environment. The robotics literature is rich with path planning and obstacle avoidance techniques (e.g. [6], [7], [8], [9]). Path planning for this project is divided into two phases: 1) Determination of collision-free manipulator tip trajectories to reach required goal points. 2) Development and implementation of a follow-the-leader (FTL) algorithm which forces ensuing links to follow the same tip trajectory. Given an obstacle-free manipulator tip trajectory, the FTL algorithm ensures whole-arm collision avoidance. The path planning algorithm must be constrained to yield trajectories which are achievable using the FTL algorithm with specific manipulator kinematics and constraints such as joint limits.

The author presents a general, modular FTL algorithm. This is an extension of the methods in [2] and [3]. Asano [6] suggests a similar scheme. The FTL algorithm was derived, implemented, and tested for the prototype hardware. It is adaptable with minimum change to new manipulator kinematics.

The author proposes to return to the PIPS project in summer 1996 to focus on the path planning algorithm. Information gathering and preliminary development for path planning was accomplished in summer 1995. Three methods are recommended, in order of preference: 1) Human-generated obstacle-free trajectories using IGRIP for visualization. 2) Same as 1), with the human producing the initial global trajectory. A computer algorithm then would fine-tune this result subject to potential field and curvature (joint limit) constraints. This mode is termed human-assisted path planning. 3) Automatic computer generation of trajectories, where the manipulator tip link is placed at the goal position and orientation. In-board links are placed optimally in turn, using potential fields .

1.3 REPORT ORGANIZATION

The focus of this report is the PIPS FTL algorithm development for use in path planning. The following are summer 1995 accomplishments:

- FTL algorithm development, MATLAB implementation, and testing
- High-fidelity prototype hardware modeling in IGRIP (with Carey Cooper) to demonstrate FTL
- Kinematics modeling and information compilation for prototype hardware
- Worked with programmer to implement FTL algorithm in C++
- Preliminary development for path planning algorithm

This report is organized as follows. Section II presents kinematic modeling and information compilation for the prototype PIPS hardware, plus a discussion of the developed IGRIP model. Section III presents the general FTL algorithm. The subsections give detailed derivations of the required mathematics. Section III concludes with simulated results. The conclusion follows, including a summary of design lessons learned from simulation. The last section is references. Four appendices are given to support the text.

1.4 REPORT ACRONYMS

DH	Denavit-Hartenberg
dof	degrees-of-freedom
FTL	Follow-the-Leader
IGRIP	Commercial Robot Simulation Software
KSC	Kennedy Space Center
MATLAB	Commercial Control Simulation Software
NASA	National Aeronautics and Space Administration
PCR	Payload Changeout Room
PIPS	Payload Inspection and Processing System
STM	Serpentine Truss Manipulator
VGT	Variable Geometry Truss

II. PROTOTYPE PIPS HARDWARE

2.1 MODIFIED FOSTER-MILLER HARDWARE

Prototype PIPS hardware was built for NASA by Foster-Miller, Inc. The original hardware consisted of a motion base and serpentine truss manipulator (STM), with eight degrees-of-freedom (dof) [3]. Two "dummy boxes" and several static members were included in place of planned actuators. NASA has modified the original hardware significantly to give the prototype good serpentine capability. The "dummy boxes" and static members were replaced with active elements to give the system eighteen-dof.

The motion base has two-dof, a ball-screw track (prismatic joint) plus a pitch joint (crane joint). The kinematic diagram for the motion base is shown in Fig. 2.1. The primatic joint variable is d_1 and the pitch joint angle is θ_2. Frame {0} is the base coordinate frame for the system. Frame {3} is aligned 45° with respect to the ball-screw track. Fig. 2.1 also shows the attachment and first two segments of the STM. Figure 2.1 shows the zero position for θ_2; nominal follow-the-leader trajectories have $\theta_2 = -90^\circ$.

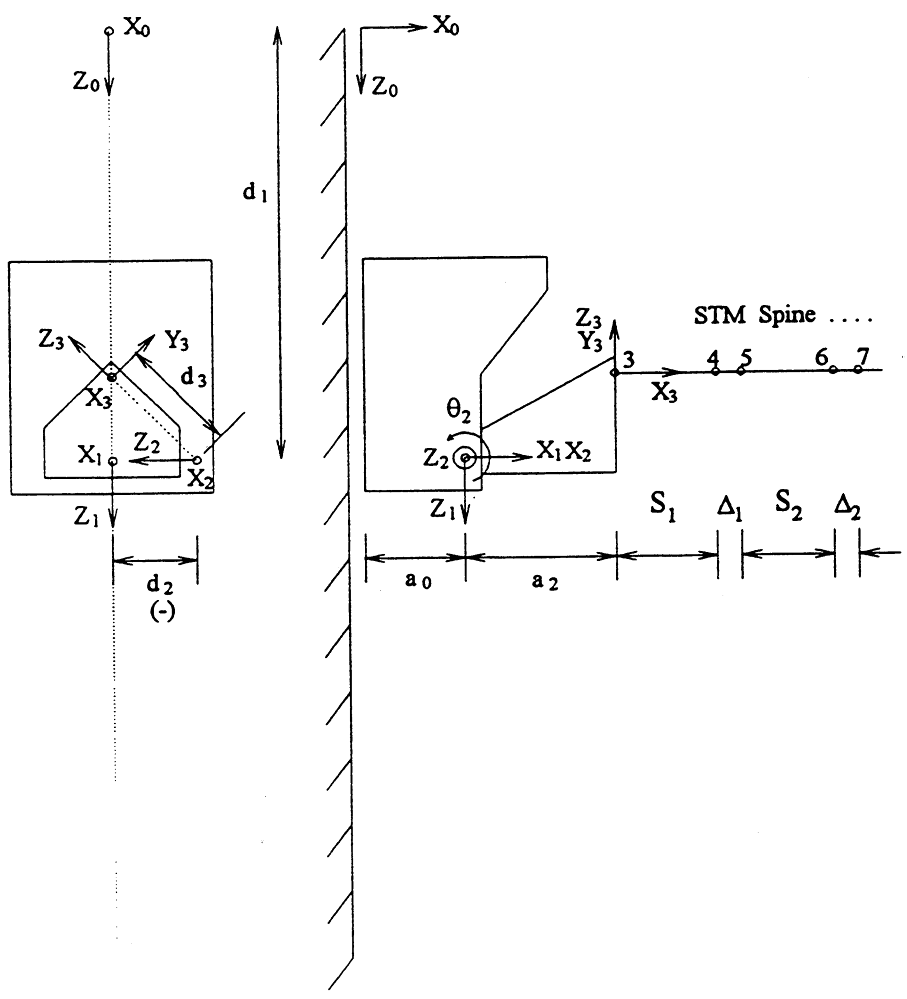


Figure 2.1 Motion Base for Prototype STM

The STM is a sixteen-dof tetrahedral variable geometry truss (VGT) as shown in Fig. 2.2. The rotation axes are orthogonal, rather than spaced by 120^o as in the standard tetrahedron VGT from the literature [10]. The corners of the solid tetrahedra along the manipulator from base to tip, opposite the linear actuators, are called the spine of the STM. Figure 2.2 is mirror-image of the as-built hardware. Figure 2.3 depicts the prototype hardware geometry.

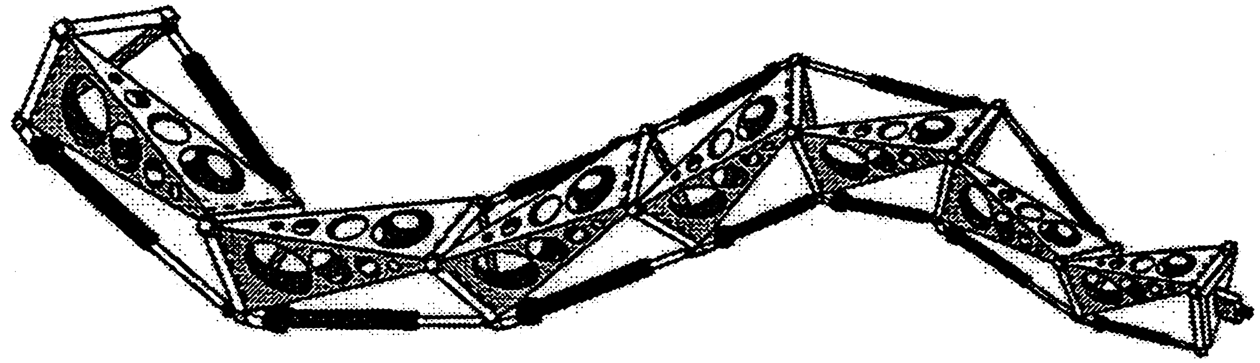


Figure 2.2 Prototype STM

Figure 2.3 shows the kinematic diagram for the STM. Views A and B are "flattened" about the spine. There are sixteen linear actuators $L_3, L_4, \ldots, L_{18}$, controlling the angles $\theta_3, \theta_4, \ldots, \theta_{18}$ about axes $Z_3, Z_4, \ldots, Z_{18}$. All X axes are along the spine. Figure 4 presents the zero position for all STM angles. The linear actuator to angular motion relationship is not as simple as shown in Fig. 2.3. The detailed geometry is documented in [3]. The Foster-Miller "Box" notation is given. This shows that the taper along the arm is discrete (three large boxes, three medium boxes, and two small boxes). An alternate "Segment" numbering for the STM modules is introduced in this report for generality, as shown. 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 DH 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 bridging segment pairs is an offset universal joint. The module lengths Q_i are discussed in Section III.

2.2 FORWARD KINEMATICS TRANSFORMATIONS

The standard forward kinematics transformation calculates the position and orientation (pose) of the manipulator tip coordinate frame relative to the manipulator base given the joint variables and manipulator geometry. In this report, the motion base and STM are taken to be one eighteen-dof manipulator. The Denavit-Hartenberg parameters (Craig convention, [11]) describing the kinematic geometry are given in Appendix A. These parameters are substituted into the following equation to yield the homogeneous tranformation matrices relative neighboring frames [11].

$$ {}^{i-1}_{\ i}T = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i c\alpha_{i-1} & c\theta_i c\alpha_{i-1} & -s\alpha_{i-1} & -d_i s\alpha_{i-1} \\ s\theta_i s\alpha_{i-1} & c\theta_i s\alpha_{i-1} & c\alpha_{i-1} & d_i c\alpha_{i-1} \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} {}^{i-1}_{\ i}R \end{bmatrix} & & & \left\{ {}^{i-1}P_i \right\} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.1} $$

As shown in Eq. 2.1, a homogeneous transformation matrix is composed of a rotation matrix representing the orientation and a position vector the translation between two frames. An artificial fourth row is added to maintain proper mathematical operations. Given all ${}^{i-1}_{i}T$, the forward kinematics transformation is:

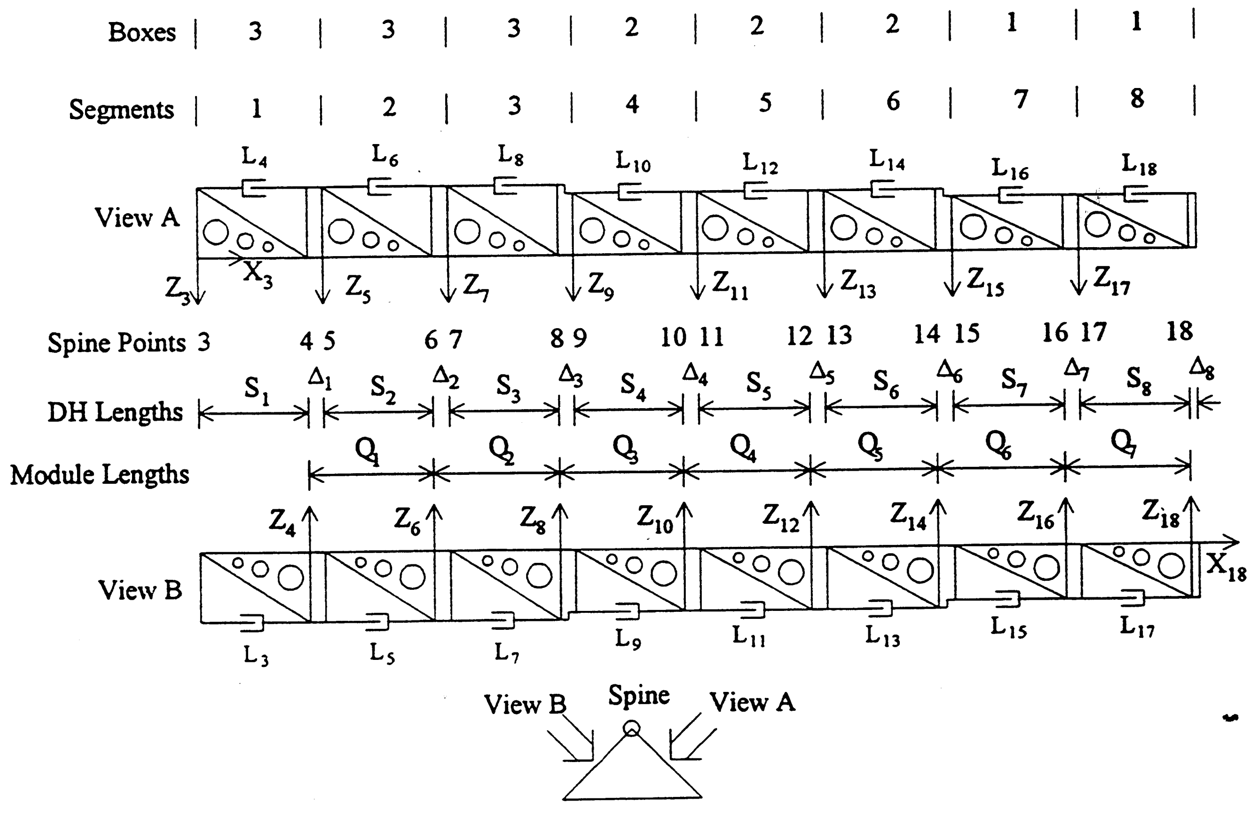
$$ {}^{0}_{tip}T = {}^{0}_{1}T\,{}^{1}_{2}T\,{}^{2}_{3}T \cdots {}^{17}_{18}T\,{}^{18}_{tip}T , \tag{2.2}$$

where:

$$ {}^{18}_{tip}T = \begin{bmatrix} & [I] & & \left\{ {}^{18}P_{tip} \right\} \\ 0 & 0 & 0 & 1 \end{bmatrix} . \tag{2.3}$$

Computation of the forward kinematics transformation is accomplished recursively, and yields a unique result. The inverse kinematics solution and graphical animation presented later requires intermediate rotation matrices and position vectors, respectively, obtained from:

$$ {}^{0}_{k}T = {}^{0}_{1}T\,{}^{1}_{2}T\,{}^{2}_{3}T \cdots {}^{k-2}_{k-1}T\,{}^{k-1}_{k}T , \qquad 1 \le k \le 18 . \tag{2.4}$$



Boxes | 3 | 3 | 3 | 2 | 2 | 2 | 1 | 1 |
Segments | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
View A
L4 L6 L8 L10 L12 L14 L16 L18
Z3 X3 Z5 Z7 Z9 Z11 Z13 Z15 Z17
Spine Points 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
DH Lengths S1 Δ1 S2 Δ2 S3 Δ3 S4 Δ4 S5 Δ5 S6 Δ6 S7 Δ7 S8 Δ8
Module Lengths Q1 Q2 Q3 Q4 Q5 Q6 Q7
Z4 Z6 Z8 Z10 Z12 Z14 Z16 Z18 X18
View B
L3 L5 L7 L9 L11 L13 L15 L17
View B
Spine
View A

Figure 2.3 Kinematic Diagram for Prototype STM

III. FOLLOW-THE-LEADER ALGORITHM

3.1 GENERAL FOLLOW-THE-LEADER ALGORITHM

A general, modular follow-the-leader (FTL) algorithm for hyper-redundant serpentine manipulators was developed and implemented in computer simulation. It was used to simulate FTL motion for the prototype PIPS hardware. 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 developed FTL algorithm is based on an extension of the method in [3]. In that work, the serpentine manipulator had fewer freedoms; the algorithm has been extended to include serpentine motion for each manipulator segment and a general treatment for the motion base. The current algorithm is adaptable to different manipulator and motion base kinematics with a minimum of code changes.

The FTL algorithm moves the manipulator tip along the given trajectory from the start to the end. The trajectory is described by a set of *XYZ* points, a piecewise linear discretization of the continuous trajectory. There is no limitation on the trajectory discretization compared to the size of segment lengths. The FTL algorithm controls the locations of spine points along the manipulator. A minimum of three dof are necessary in the motion base to place the base-most spine point at general 3D points. The motion base is used to feed the STM into the trajectory. Two dof are required to position one point relative to another along the given trajectory and separated by a fixed link distance. If the STM is composed of true universal joints (no offset between the orthogonal joint pairs) every spine point can be placed on the trajectory. For general STMs consisting of offset revolute joints, only every other spine point can be placed on the trajectory. The FTL algorithm controls only the STM spine; therefore, the path planning algorithm must provide trajectories with sufficient clearance to allow collision-free motion considering the manipulator dimensions about the spine.

To place the STM 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 continues until the base-most spine point is placed on the trajectory. For this purpose, the trajectory is appended with the straight line from the fixed base to the first trajectory point. 2) Inverse position kinematics then calculates the required joint values, from the base to the tip.

For zero-offset STMs the above steps are sufficient. However, for STMs 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 an average value 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 process is repeated for each point on the trajectory. At each step, the joint values are saved. Smooth serpentine motion may be obtained by providing a fine trajectory discretization. For operation, the off-line joint values are downloaded to the real-time STM controller. Retraction of the STM along the same obstacle-free trajectory is accomplished by reversing the joint values array.

Figure 3.1 gives a flow chart for the FTL process. The inner iteration loop drives the spine points error to a user-defined tolerance, with a maximum number of iterations. The outer loop drives the tip along the trajectory. The following sections present the detailed FTL algorithm for the prototype PIPS hardware. However, the development is modular so that only specific manipulator kinematics and spine points must be changed for different STMs. FTL derivations are presented for trajectory generation, fitting the manipulator to the trajectory, and inverse kinematics solutions. The FTL discussion concludes with presentation of simulated results.

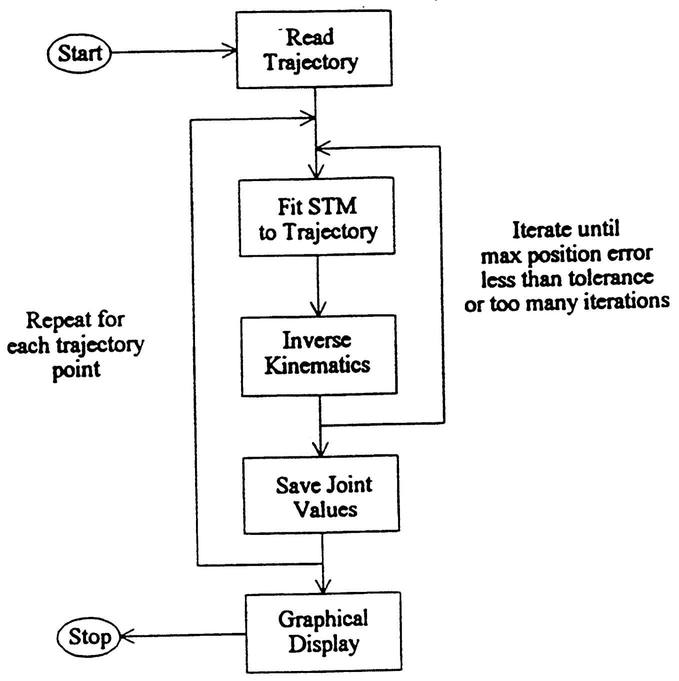

Start
Read Trajectory
Fit STM to Trajectory
Inverse Kinematics
Save Joint Values
Graphical Display
Stop
Repeat for each trajectory point
Iterate until max position error less than tolerance or too many iterations

Figure 3.1 Follow-the-Leader Flowchart

3.2 TRAJECTORY GENERATION

This step will eventually be accomplished with a path planning algorithm. Currently, an obstacle-free trajectory for the manipulator tip is determined manually and discretized by computer. Four trajectories were developed to demonstrate the FTL algorithm. These were developed relative to a shuttle pallet and generic cylindrical payload to display varied motion.

The laboratory hardware is currently arranged so the prismatic joint is vertical, as shown in Figure 2.1. This should be changed so that the prismatic joint is horizontal, feeding the STM into a vertical Shuttle bay pallet. For this case, the motion base crane joint value would be $\theta_2 = -90°$ so that the STM is aligned with the prismatic joint direction. The home position for the STM is the minimum d_1, with $\theta_3 = \theta_4 = \ldots = \theta_{18} = 0$. The trajectory must start at the manipulator tip in this home position. The middle and end trajectory points must be reachable subject to joint limits.

3.3 FIT STM TO TRAJECTORY

This section presents the first basic FTL component, fitting the hyper-redundant manipulator to the given trajectory. The method is general and will be applied to the specific hardware in Section II.

3.3.1 SPINE POINTS

Given an STM to control in FTL mode, the first step is to determine which points to control as the spine points. Since three dof are required to position a point independently in 3D space, 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. Spine point *i* is the origin of coordinate frame $\{i\}$. Since two dof are required to position a point a given distance from another point, along a given trajectory, every second spine point following 4 can be fit to the trajectory. Therefore, 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 combined into modules to place each ensuing spine point. The motion base is pictured in Fig 2.1 and the STM in Figs. 2.2 and 2.3. Figure 3.2 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* and *O* stand for even and odd, respectively).

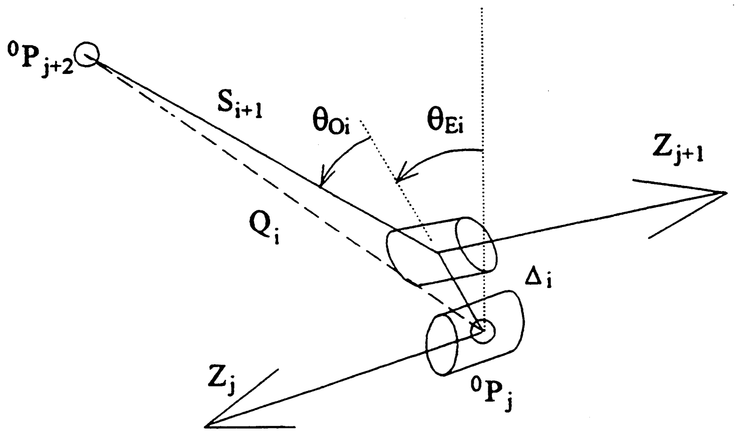


Figure 3.2 General Prototype STM Module

The intermediate spine point *j*+1 cannot be placed in general on the given trajectory. The fixed DH 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 3.1 (from Table A.1).

Table 3.1 DH Parameters for Module *i*

joint	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}, as derived by forward kinematics.

$$\left\{ \begin{Bmatrix} {}^jP_{j+2} \\ 1 \end{Bmatrix} \right\} = {}^j_{j+1}T \left\{ \begin{Bmatrix} {}^{j+1}P_{j+2} \\ 1 \end{Bmatrix} \right\} = {}^j_{j+1}T(\theta_{Oi}) \begin{Bmatrix} S_{i+1} \\ 0 \\ 0 \\ 1 \end{Bmatrix} \tag{3.1}$$

$$ {}^{j}_{j+1}T = \begin{bmatrix} c\theta_{Oi} & -s\theta_{Oi} & 0 & \Delta_i \\ 0 & 0 & -1 & 0 \\ s\theta_{Oi} & c\theta_{Oi} & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.2} $$

$$ {}^{j}P_{j+2} = \begin{Bmatrix} \Delta_i + S_{i+1}c\theta_{Oi} \\ 0 \\ S_{i+1}s\theta_{Oi} \end{Bmatrix} \tag{3.3} $$

$$ Q_i(\theta_{Oi}) = \left\| {}^{j}P_{j+2} \right\| = \sqrt{\Delta_i^2 + 2\Delta_i S_{i+1} c\theta_{Oi} + S_{i+1}^2} \tag{3.4} $$

Equation 3.4 is used to update the seven module lengths when iterating to reduce the maximum spine point position error (see Fig. 3.1). The Q_i are updated after the inverse kinematics solution.

Table 3.2 summarizes the spine point paradigm for the prototype hardware, giving module number, segments composing each module, spine points of the module end and start, fixed DH lengths, joint variables, and variable module length. The first row of Table 3.2 gives the motion base information; the seven STM modules follow.

Table 3.2 STM Modules

Module i	Segments	Spine Points	DH Lengths	Joint Variables	ModuleLength
-	Motion Base/1	4 / 0	-	d_1,θ_2,θ_3	-
1	1 / 2	6 / 4	Δ_1,S_2	θ_4,θ_5	Q_1
2	2 / 3	8 / 6	Δ_2,S_3	θ_6,θ_7	Q_2
3	3 / 4	10 / 8	Δ_3,S_4	θ_8,θ_9	Q_3
4	4 / 5	12 / 10	Δ_4,S_5	θ_{10},θ_{11}	Q_4
5	5 / 6	14 / 12	Δ_5,S_6	θ_{12},θ_{13}	Q_5
6	6 / 7	16 / 14	Δ_6,S_7	θ_{14},θ_{15}	Q_6
7	7 / 8	18 / 16	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 camera pointing. For the prototype STM hardware, the following relates module index i and spine point index j: $j = 2(i+1)$.

3.3.2 INTERSECT ROUTINE

3.3.2.1 The Process

To fit a general 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 trajectory. The module lengths initialize at their average values (see Table A.2) and are updated 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.

This process is pictured for the prototype STM hardware in Fig. 3.3. In addition to the basic process, detail for module 6 is shown to demonstrate that the intermediate point 15 cannot be placed on the trajectory.

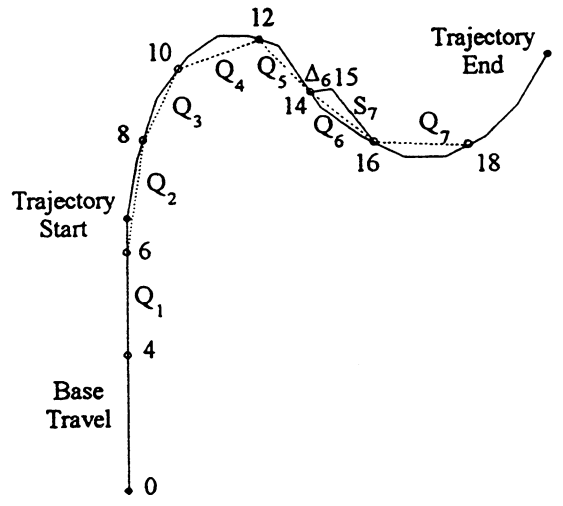


Figure 3.3 Fit STM to Given Trajectory

3.3.2.2 Intersection of Sphere with Line Segment

For the basic process of fitting the STM to a given trajectory described above, the intersection of a sphere with a 3D line segment must be solved repeatedly. This section presents the mathematics for this solution. The author thanks Sean for a fruitful discussion in this area.

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

$$\begin{aligned} x &= at + P_{1x} & a &= P_{2x} - P_{1x} & & t = 0 @ P_1 \\ y &= bt + P_{1y} & b &= P_{2y} - P_{1y} & & t = 1 @ P_2 \\ z &= ct + P_{1z} & c &= P_{2z} - P_{1z} & & \end{aligned} \tag{3.5}$$

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

$$(x - x_c)^2 + (y - y_c)^2 + (z - z_c)^2 = r^2 \tag{3.6}$$

The intersection of the sphere surface and line segment is found by substituting *x,y,z* from Eq. 3.5 into Eq. 3.6, yielding:

$$At^2 + Bt + C = 0 \tag{3.7}$$

$$\text{where:}\quad \begin{aligned} A &= a^2 + b^2 + c^2 \\ B &= 2(ad + be + cf) \\ C &= d^2 + e^2 + f^2 - r^2 \end{aligned} \qquad \begin{aligned} d &= P_{1x} - x_c \\ e &= P_{1y} - y_c \\ f &= P_{1z} - z_c \end{aligned} \tag{3.8}$$

Solve *t* using the quadratic formula, $t_{1,2} = \frac{-B \pm \sqrt{B^2 - 4AC}}{2A}$; then evaluate *x,y,z* from Eqs. 3.5. 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.

3.4 INVERSE POSITION KINEMATICS SOLUTION

Given an STM fit to a given trajectory, this section presents the second basic FTL component, inverse position kinematics. The solution of this problem yields the joint values given the spine points. Only *XYZ* positions are satisfied, because the trajectory must give the nominal orientation at the STM tip. The intermediate orientations are used in the solution but not controlled independently. First, the motion base solution is presented, followed by the general solution which applies to each STM module *i*.

3.4.1 MOTION BASE SOLUTION

The motion base inverse position problem is: Given ${}^0P_4$, calculate d_1, θ_2, θ_3. The associated DH parameters are the first three lines in Table A.1. The kinematic diagram is shown in Fig. 2.1. Equation 3.9 gives the transformation relating the variables to the given information.

$${}^0P_4 = {}^0_3T\,{}^3P_4 = {}^0_1T(d_1)\,{}^1_2T(\theta_2)\,{}^2_3T(\theta_3)\begin{Bmatrix} S_1 \\ 0 \\ 0 \end{Bmatrix} \tag{3.9}$$

This equation expands to Eq. 3.10, where ${}^0P_4$ is given.

$${}^0P_4 = \begin{Bmatrix} X \\ Y \\ Z \end{Bmatrix} = \begin{Bmatrix} a_0 + c\theta_2(S_1c\theta_3 + a_2) - Ks\theta_2(S_1s\theta_3 + d_3) \\ d_2 - K(S_1s\theta_3 - d_3) \\ d_1 - s\theta_2(S_1c\theta_3 + a_2) - Kc\theta_2(S_1s\theta_3 + d_3) \end{Bmatrix} \qquad K = \frac{\sqrt{2}}{2} \tag{3.10}$$

The solution of Eq. 3.10 follows the order: 1) θ_3; 2) θ_2; 3) d_1. From the Y component of Eq. 3.10:

$$\theta_3 = \sin^{-1}\left(\frac{d_2 + Kd_3 - Y}{KS_1}\right) \tag{3.11}$$

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. 3.10 yields:

$$E\cos\theta_2 + F\sin\theta_2 + G = 0 \qquad \begin{aligned} E &= S_1 c\theta_3 + a_2 \\ F &= -K(S_1 s\theta_3 + d_3) \\ G &= a_0 - X \end{aligned} \tag{3.12}$$

Equation 3.12 is solved using the tangent half-angle substitution:

$$t = \tan\left(\frac{\theta}{2}\right) \quad \cos\theta = \frac{1-t^2}{1+t^2} \quad \sin\theta = \frac{2t}{1+t^2} \tag{3.13}$$

Substituting Eq. 3.13 into Eq. 3.12, simplifying, solving t with the quadratic formula, and using $t = \tan\left(\frac{\theta}{2}\right)$ yields two valid solutions for θ_2, given in Eq. 3.14. The solution chosen for FTL control is the value closest to the nominal STM home position, $\theta_2 = -90^\circ$.

$$\theta_{2_{1,2}} = 2\tan^{-1}\left(\frac{-F \pm \sqrt{E^2 + F^2 - G^2}}{G - E}\right) \tag{3.14}$$

Given the θ_3 and θ_2 solutions, the Z component of Eq. 3.10 solves the prismatic joint variable. There is one solution for each θ_2, θ_3 pair.

$$d_1 = Z + s\theta_2(S_1 c\theta_3 + a_2) + Kc\theta_2(S_1 s\theta_3 + d_3) \tag{3.15}$$

3.4.2 MODULE *i* SOLUTION

When the solution for the first three joint 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 in Section 3.3.1. 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:

$${}^0P_{j+2} - {}^0P_j = {}^0_jR\,{}^jP_{j+2} \tag{3.16}$$

Equation 3.16 is rearranged to show the dependence on the unknown joint angles.

$$ {}^{0}P_{j+2} - {}^{0}P_{j} = {}_{j-1}^{0}R \, {}_{j}^{j-1}R(\theta_{Ei}) \, {}^{j}P_{j+2}(\theta_{Oi}) \tag{3.17} $$

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

$$ {}_{j-1}^{0}R^{T}\left({}^{0}P_{j+2} - {}^{0}P_{j}\right) = \begin{Bmatrix} x \\ y \\ z \end{Bmatrix} = {}_{j}^{j-1}R(\theta_{Ei}) \, {}^{j}P_{j+2}(\theta_{Oi}) \tag{3.18} $$

The vector ${}^{j}P_{j+2}(\theta_{Oi})$ was given in Eq. 3.3. The rotation matrix expressing the unknown θ_{Ei} is given below:

$$ {}_{j}^{j-1}R(\theta_{Ei}) = \begin{bmatrix} c\theta_{Ei} & -s\theta_{Ei} & 0 \\ 0 & 0 & 1 \\ -s\theta_{Ei} & -c\theta_{Ei} & 0 \end{bmatrix} \tag{3.19} $$

Substituting these values into Eq. 3.18 yields:

$$ \begin{Bmatrix} x \\ y \\ z \end{Bmatrix} = \begin{Bmatrix} c\theta_{Ei}\left(\Delta_i + S_{i+1}c\theta_{Oi}\right) \\ S_{i+1}s\theta_{Oi} \\ -s\theta_{Ei}\left(\Delta_i + S_{i+1}c\theta_{Oi}\right) \end{Bmatrix} \tag{3.20} $$

In Eq. 3.20, 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. 3.21.

$$ \frac{z}{x} = \frac{-s\theta_{Ei}\left(\Delta_i + S_{i+1}c\theta_{Oi}\right)}{c\theta_{Ei}\left(\Delta_i + S_{i+1}c\theta_{Oi}\right)}; \qquad \theta_{Ei} = a\tan2(-z, x) \tag{3.21} $$

θ_{Oi} is solved from a ratio of the y to the x component; again, *atan2* is used.

$$ \frac{y}{\dfrac{x}{c\theta_{Ei}} - \Delta_i} = \frac{S_{i+1}s\theta_{Oi}}{S_{i+1}c\theta_{Oi}}; \qquad \theta_{Oi} = a\tan2\left(y, \frac{x}{c\theta_{Ei}} - \Delta_i\right) \tag{3.22} $$

This completes the general solution for θ_{Ei}, θ_{Oi}. Once the motion base solution is known, this general solution is applied to the seven modules, substituting the appropriate lengths from Tables 3.2 and A.2. This process results in the values for joint angle pairs θ_4, θ_5 through θ_{16}, θ_{17}, from the formulas for θ_{Ei}, θ_{Oi}. As previously discussed, $\theta_{18} = 0$. Following the inverse kinematics solution, forward kinematics is used to calculate the errors between the desired and actual spine point locations. If the maximum error is too large, the module lengths are updated using the new joint angles and Eq. 3.4 and another iteration is made.

3.5 RESULTS

This section discusses FTL algorithm simulation results for the prototype PIPS hardware. This effort was accomplished using MATLAB, C++, and IGRIP.

3.5.1 MATLAB SIMULATION

MATLAB simulation software was used to develop, implement, and test the FTL algorithm. This code executes the steps in Fig. 3.1 for the prototype PIPS hardware, including graphical animation of the motion base and STM spine. Many trajectories were tested in the Shuttle pallet environment with a nominal cylindrical payload. Four trajectories (developed manually) are given in Appendix C.

The MATLAB code is modular and adaptable to different PIPS hardware designs by changing the forward and inverse kinematics modules. The code was delivered by floppy disk to Gabor Tamasi. Also on the disk are data files for the four trajectories (TRAJi.M, $i = 1,2,3,4$) and for the joint values results from the FTL algorithm (FTLi.M, $i = 1,2,3,4$). Appendix B presents more detail on the MATLAB code. This code and data is also available by calling the author at Ohio University, (614) 593-1096. Appendix D gives graphics resulting from the MATLAB FTL simulation, for trajectory four.

3.5.2 C++ IMPLEMENTATION

The author supported implementation of the FTL algorithm in C++ for the operational hardware.

3.5.3 IGRIP SIMULATION

A high-fidelity, kinematically-correct computer graphics model of the prototype PIPS hardware was developed using IGRIP software. Carey Cooper of Intergraph Corporation performed the IGRIP modeling based on information compiled by the author from reference [3], mechanical drawings of the prototype hardware, and the hardware itself. This simulation is used to demonstrate the FTL algorithm in a virtual laboratory. The four trajectories mentioned above were transferred to IGRIP. Joint angle histories, created by the MATLAB code for each trajectory, were downloaded to the simulation. A fictitious payload was added to the Shuttle pallet environment to simulate whole arm obstacle avoidance using the four trajectories and FTL algorithm. A videotape was produced to demonstrate the IGRIP FTL simulation. Figure D.5 shows the IGRIP model, where the STM is at the end of the fourth trajectory.

IV. CONCLUSION

4.1 Design Lessons Learned

When simulating motion of a robotic hardware system, one always learns lessons on how to improve the design. These issues are not always clear to the developers of hardware. Therefore, because the PCR PIPS hardware is to be developed soon, the following list of design lessons from the prototype hardware is presented.

- STM joint offsets should be zero.
- The motion base translational travel should be equal to the STM length.
- The simplest FTL algorithm results from equal STM link lengths.
- The FTL simulation should be used to design the final system kinematics.

The following list is more specific to the prototype hardware.

- The motion base must have more range in three dimensions.
- The first STM joints allow the STM to be driven into the prismatic track when $\theta_2 = -90^{\circ}$.
- Box 1 encoders do not allow full joint motion.
- The sensor skin also restricts joint motion.

4.2 Concluding Remarks

This report presents a summary of the author's summer 1995 work at NASA Kennedy Space Center. The general area is path planning for the Payload Inspection and Processing System (PIPS). Specifically, a follow-the-leader algorithm (FTL) was developed, implemented, tested, and demonstrated using computer graphics. The FTL algorithm ensures whole-arm collision avoidance for the PIPS manipulator, given an obstacle-free trajectory for the tip link. FTL limits the overall manipulator workspace, but the resulting reliable obstacle avoidance in a sensitive environment justifies the limitation.

The FTL algorithm is derived in general and implemented in modular code. This report focuses on the prototype PIPS hardware, but extension of the method to the PCR hardware requires changes only to the manipulator kinematics. In addition to the FTL algorithm documentation, this report gives kinematic information for the prototype hardware, derived and compiled from various sources. Extensive kinematics modeling, both forward and inverse, are presented to support the FTL algorithm.

Preliminary investigations were undertaken into the general path planning problem. The PIPS path planning algorithm will be accomplished in the IGRIP environment. There are two preliminary recommendations for determining obstacle-free trajectories for the camera at the manipulator tip: 1) Human-assisted path planning using IGRIP; 2) Automated path planning using built-in IGRIP functions. If existing algorithms solve the problem, there is no need to re-invent the wheel. The author proposes to return to implement path planning results to the hardware during summer 1996.

V. REFERENCES

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

[2] Pasch, K., "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 Kennedy Space Center, FL, August, 1990.

[3] Snyder, M., "Self-Contained Deployable Serpentine Truss (SCDST) for Prelaunch Access of Space Shuttle Orbiter Payloads", NAS-1794-FM-9323-651, Final Report, Contract NAS 10-11794, NASA Kennedy Space Center, FL, October, 1993.

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

[5] Lee, T.S., Ohms, T., and Hayati, S., "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, March, 1994.

[6] Krogh, B.H., "A Generalized Potential Field Approach to Obstacle Avoidance Control", Proceedings of the SME Conference on Robotics Research, Bethlehem, PA, August, 1984.

[7] Lozano-Perez, T., "A Simple Motion Planning Algorithm for General Robot Manipulators", *IEEE Journal of Robotics and Automation*, Vol. RA-3, No. 3, June, 1987, pp. 224-238.

[8] Gupta, K.K., "Fast Collision Avoidance for Manipulator Arms: A Sequential Search Strategy", Proceedings of the IEEE International Conference on Automation and Robotics, 1990, pp. 1724-1729.

[9] Nakamura, Y., *Advanced Robotics*, Redundancy and Optimization, Addison-Wesley Publishing Co., Inc., Reading, MA, 1991.

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

[11] Craig, J.J., *Introduction to Robotics: Mechanics and Control*, Addison-Wesley Publishing Co., Inc., Reading, MA, 1988.

APPENDIX A. DENAVIT-HARTENBERG PARAMETERS FOR THE PROTOTYPE PIPS HARDWARE

The Denavit-Hartenberg (DH) parameters (Craig convention, [11]) for the eighteen-dof prototype PIPS hardware are given in Table A.1. The units are degrees for angles and inches for lengths. The first three rows are for the motion base (Fig. 2.1), while rows four through eighteen represent the serpentine truss manipulator (Figs 2.2 and 2.3). The variables are d_1, the prismatic joint variable, and $\theta_2, \theta_3, \ldots, \theta_{18}$, the STM spine joint angles.

Table A.1 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	-90	S_3	0	θ_8
9	90	Δ_3	0	θ_9
10	-90	S_4	0	θ_{10}
11	90	Δ_4	0	θ_{11}
12	-90	S_5	0	θ_{12}
13	90	Δ_5	0	θ_{13}
14	-90	S_6	0	θ_{14}
15	90	Δ_6	0	θ_{15}
16	-90	S_7	0	θ_{16}
17	90	Δ_7	0	θ_{17}
18	-90	S_8	0	θ_{18}

The values for the general parameters in Table A.1 are given below.

$a_0 = 11.250$
$a_2 = 8.908$

$d_1 = \text{var.}$
$d_2 = -5.005$
$d_3 = 7.078$

$S_1 = S_2 = S_3 = 20.035$
$S_4 = S_5 = S_6 = 16.100$
$S_7 = S_8 = 12.000$

$\Delta_1 = \Delta_2 = 1.250$
$\Delta_3 = \Delta_6 = 1.500$
$\Delta_4 = \Delta_5 = 1.188$
$\Delta_7 = 0.813$
$\Delta_8 = 0.763$

Table A.2 gives specific data for the seven Modules *i* (Segment *i-i+1*) for the prototype STM; see Table 3.2. Modules 1,2 and 4,5 are identical. For each module, the DH lengths are given (see Table A.1). Also, information on the variable module length Q_i is included:

$$Q_{iAVG} = \frac{Q_{iMAX} + Q_{iMIN}}{2} \qquad \text{and} \qquad \Delta Q_i = Q_{iMAX} - Q_{iMIN}$$

are given for each module. Figure A.1, a planar view of Fig. 3.2, shows the definitions for Q_{iMAX} and Q_{iMIN}. The last column in Table A.2 gives the values of θ_{Oi} corresponding to Q_{iAVG}. Equation 3.4 was used for the last three columns of Table A.2.

Table A.2 Prototype STM Module Data

Module *i*	Segment *i-i+1*	Δ_i	S_{i+1}	Q_{iAVG}	ΔQ_i	θ_{OiAVG}
1,2	1-2,2-3	1.250	20.035	21.206	0.158	±21.1°
3	3-4	1.500	16.100	17.508	0.185	±21.1°
4,5	4-5,5-6	1.188	16.100	17.214	0.149	±21.1°
6	6-7	1.500	12.000	13.410	0.180	±21.1°
7	7-8	0.813	12.000	12.762	0.102	±21.1°

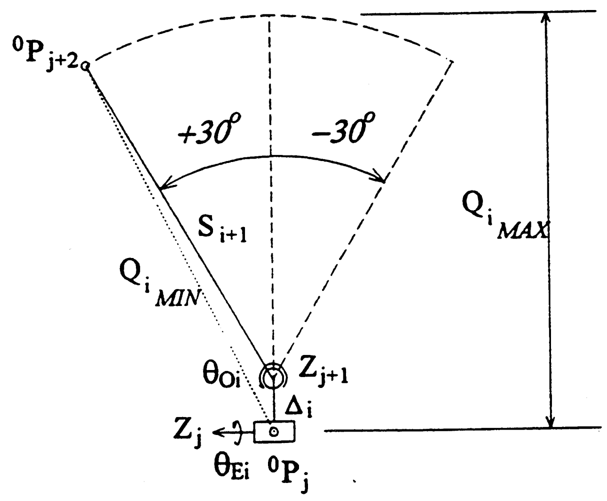


Figure A.1 STM Module i

The range of the prismatic joint, derived from mechanical drawings, is:

$$24.9" \le d_1 \le 114.5" .$$

Nominal limits for all joint angles are ±30°. More detailed limits are published in [3]. Precise limits for the NASA-modified STM must be determined.

APPENDIX B. MATLAB CODE FOR THE FOLLOW-THE-LEADER ALGORITHM

MATLAB code for the follow-the-leader algorithm was delivered to Gabor Tamasi at the conclusion of the fellowship period. The format was ASCII m-files on a WINDOWS floppy disk. This code was also delivered and explained to Mike Sklar of McDonnell Douglas, who is developing a graphical user interface for PIPS. The FTL flowchart was given in Figure 3.1. MATLAB is an interpretive language. The main routine is:

FTL.M.

Additional m-files called from this main routine are:

PALPAY.M,
TRAJ1.M, TRAJ2.M, TRAJ3.M, TRAJ4.M,
INTRSECT.M,
INVKIN.M,
STMPLOT.M.

MATLAB functions, with their calling routines in parentheses, are:

SPHRLINE.M	**(INTRSECT.M),**
DHFUN.M	**(INVKIN.M, STMPLOT.M),**
TANHALF.M	**(INVKIN.M).**

Variables used throughout the code:

traj	Trajectory points to follow.
pt	Point array from intersect routine.
alp, a, d, th	DH parameters.
d1, th2, th3, . . ., th18	Joint values from inverse kinematics.
jnt	Above joint values in one array.

Throughout the code, the following notation is used, with reference to [11].

tji : $^{i}T_{j}$ Homogeneous transformation matrix giving the position and orientation of frame $\{j\}$ with respect to frame $\{i\}$, expressed in $\{i\}$ coordinates.

rji : $^{i}R_{j}$ Orthonormal rotation matrix giving the orientation of frame $\{j\}$ with respect to frame $\{i\}$, expressed in $\{i\}$ coordinates.

pji : $^{i}P_{j}$ Position vector from the origin of frame $\{i\}$ to the origin of frame $\{j\}$, expressed in $\{i\}$ coordinates.

APPENDIX C. TRAJECTORY AND FTL DATA FILES

This appendix presents trajectory data files developed for the follow-the-leader simulation. These data are calculated and inserted into the traj variable by TRAJ1.M, TRAJ2.M, TRAJ3.M, and TRAJ4.M, respectively. All data are given with respect to frame {0} coordinates. The joint values from the follow-the-leader algorithm corresponding to these trajectory data files are FTL1.DAT, FTL2.DAT, FTL3.DAT, and FTL4.DAT. All TRAJi.DAT and FTLi.DAT files were delivered with the MATLAB code as mentioned in Appendix B.

Table C.1 TRAJ1.DAT

i	X	Y	Z
1	16.255	0.000	0.000
2	16.255	0.000	174.902
3	16.255	0.000	179.902
4	16.255	0.000	184.902
5	16.255	0.000	189.902
6	16.255	-0.656	196.139
7	16.255	-2.594	202.104
8	16.255	-5.729	207.536
9	16.255	-9.926	212.196
10	16.255	-15.000	215.883
11	16.255	-20.729	218.434
12	16.255	-26.864	219.738
13	16.255	-33.136	219.738
14	16.255	-39.271	218.434
15	16.255	-45.000	215.883

Table C.2 TRAJ2.DAT

i	X	Y	Z
1	16.255	0.000	0.000
2	16.255	0.000	174.902
3	15.255	0.000	179.902
4	13.255	0.000	184.902
5	10.255	0.000	189.902
6	7.255	-0.656	196.139
7	4.255	-2.594	202.104
8	1.255	-5.729	207.536
9	-1.745	-9.926	212.196
10	-4.745	-15.000	215.883
11	-7.745	-20.729	218.434
12	-10.745	-26.864	219.738
13	-13.745	-33.136	219.738
14	-15.745	-39.271	218.434
15	-16.745	-45.000	215.883

Table C.3 TRAJ3.DAT

i	X	Y	Z
1	16.255	0.000	0.000
2	16.255	0.000	174.902
3	17.255	0.000	179.902
4	19.255	0.000	184.902
5	22.255	0.000	189.902
6	25.255	-0.656	196.139
7	28.255	-2.594	202.104
8	31.255	-5.729	207.536
9	34.255	-9.926	212.196
10	37.255	-15.000	215.883
11	40.255	-20.729	218.434
12	43.255	-26.864	219.738
13	45.255	-33.136	219.738
14	46.255	-39.271	218.434
15	46.255	-45.000	215.883

Table C.4 TRAJ4.DAT

i	X	Y	Z
1	16.255	0.000	0.000
2	16.255	0.000	174.902
3	17.209	0.000	182.149
4	20.006	0.000	188.902
5	24.456	0.000	194.701
6	30.255	0.000	199.151
7	37.008	0.000	201.948
8	44.255	0.000	202.902
9	51.761	-0.988	202.902
10	58.755	-3.885	202.902
11	64.761	-8.494	202.902
12	69.370	-14.500	202.902
13	72.267	-21.494	202.902
14	73.255	-29.000	202.902

APPENDIX D. TRAJECTORY 4 RESULTS

D.1 SHUTTLE PALLET, CYLINDRICAL PAYLOAD, AND TRAJECTORY 4

Figure D.1 shows the Shuttle environment modeled in MATLAB. The 3D trajectory four is shown, consisting of two quarter circles in orthogonal planes. A planar cross section of the Space Shuttle pallet and a cylindrical payload are also shown.

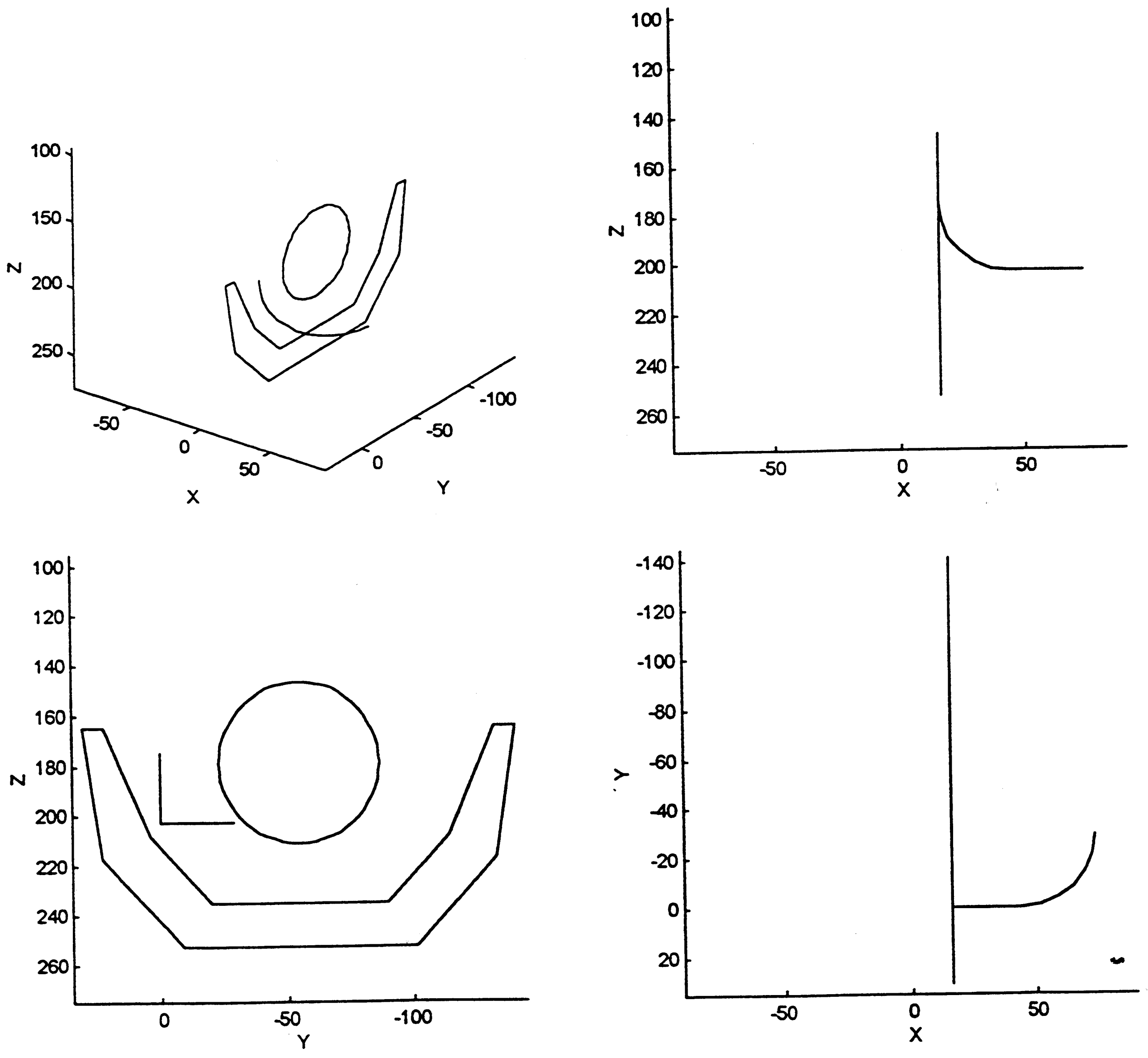


Figure D.1 Pallet, Payload, Trajectory 4

D.2 FOLLOW-THE-LEADER SIMULATION

Figures D.2 show the STM in FTL simulation. The figures progress from home position (Fig. D.2a) through two intermediate positions, and the end of trajectory four (Fig. D.2d). The appearance of interference of the STM and pallet is an illusion; trajectory four enables collision-free motion.

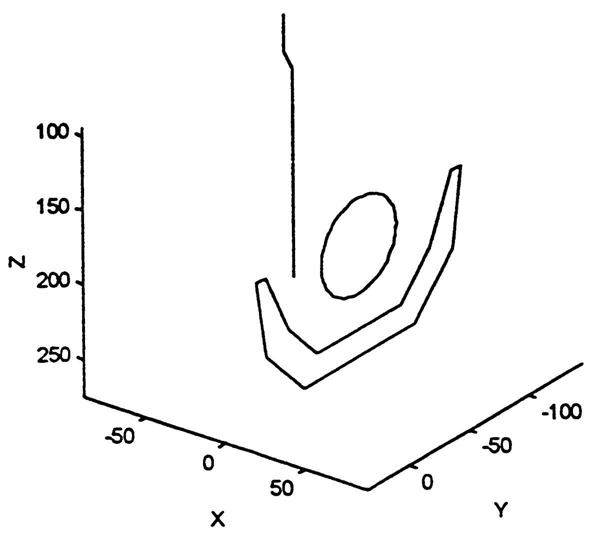


Figure D.2a

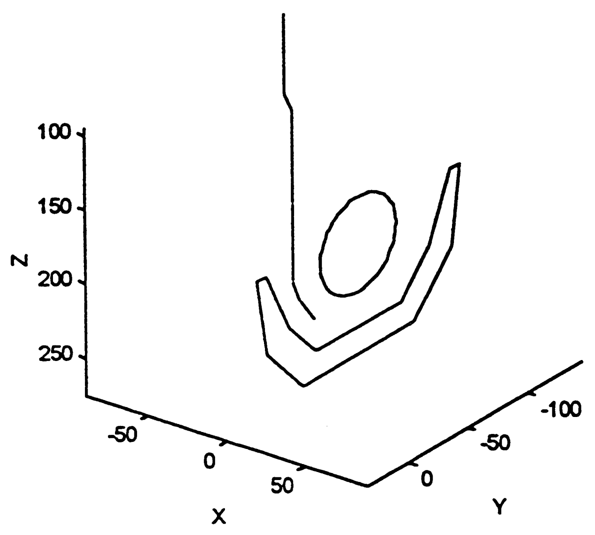


Figure D.2b

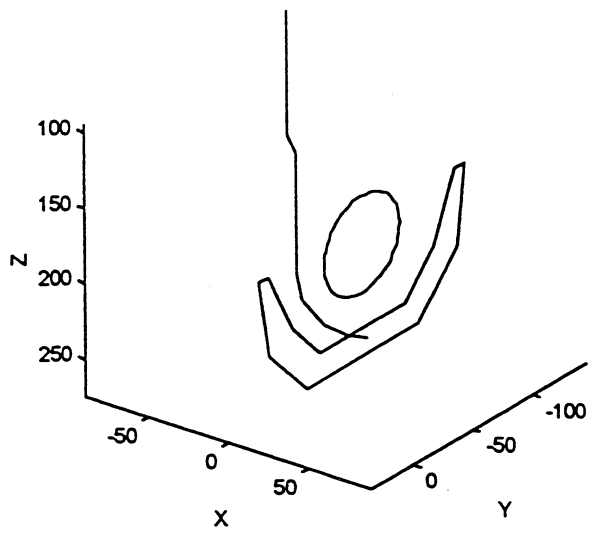


Figure D.2c

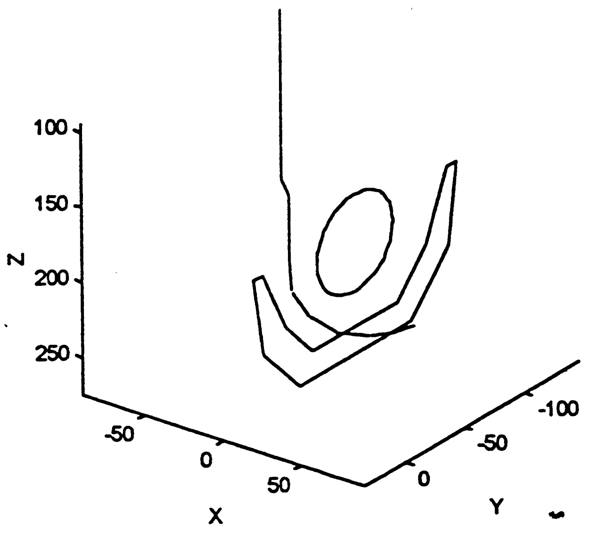


Figure D.2d

Figure D.2 FTL Simulation for Trajectory 4

D.3 JOINT VALUE RESULTS

Figures D.3 show the STM joint values from the FTL algorithm for trajectory four. The motion base values are shown, followed by the seven Module *i* values (Segments *i-i+1*).

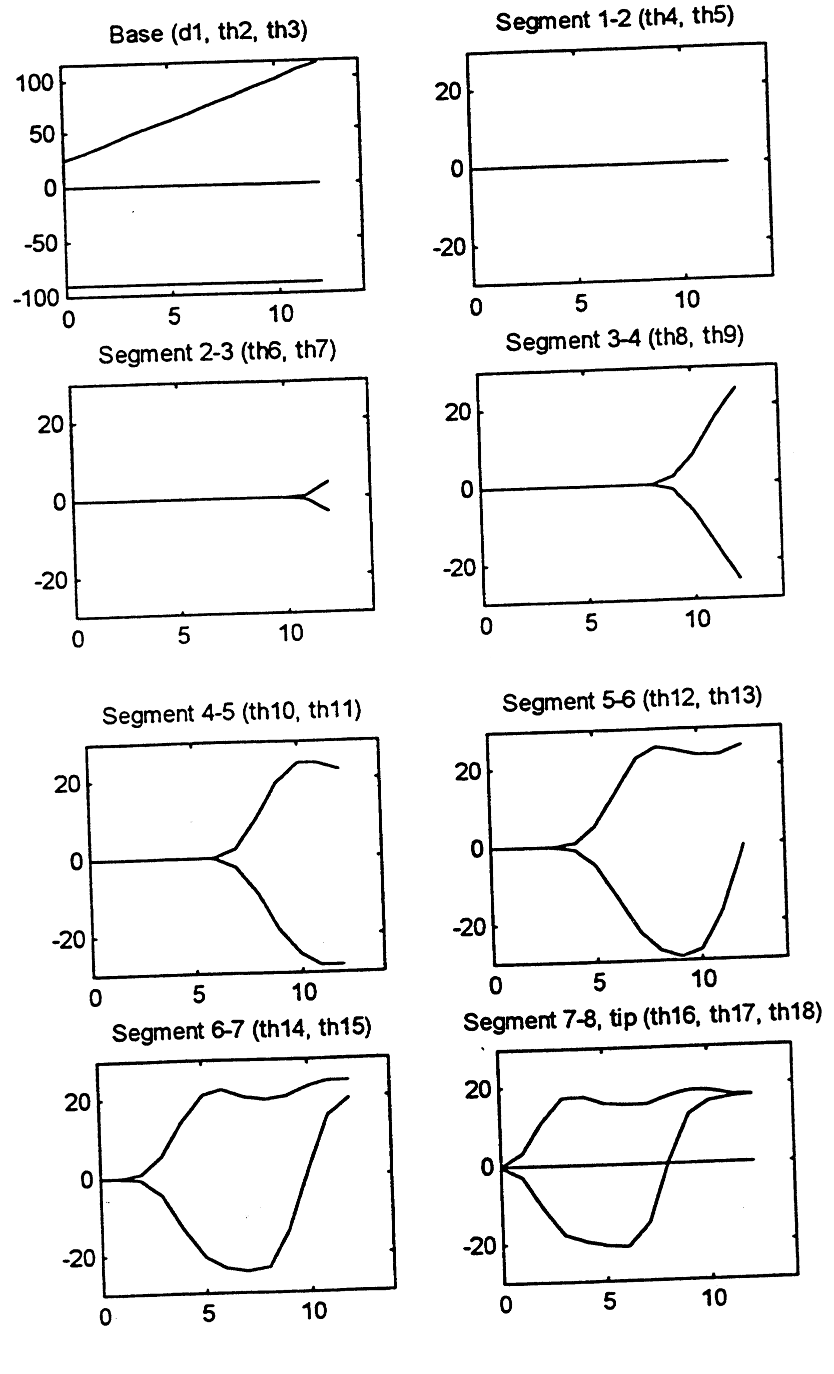


Figure D.3 FTL Joint Values

D.4 ITERATION RESULTS

Figures D.4 show the FTL algorithm convergence history for trajectory four. Two iterations are required for the first trajectory point, and one iteration thereafter to achieve an error tolerance of one-tenth of an inch (the horizontal line in Fig. D.4a). Since there is only one iteration after the initial step, the initial and final manipulator errors are identical. The error tends to increase as the manipulator traverses the trajectory. The error condition checks all spine points for convergence; the maximum for trajectory four was the tip error.

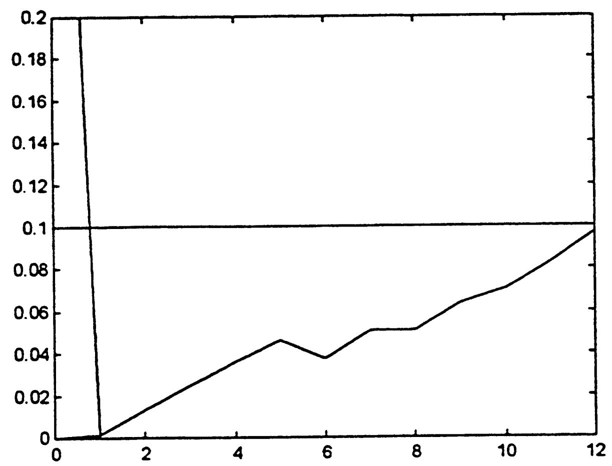


Figure D.4a FTL Tip Error

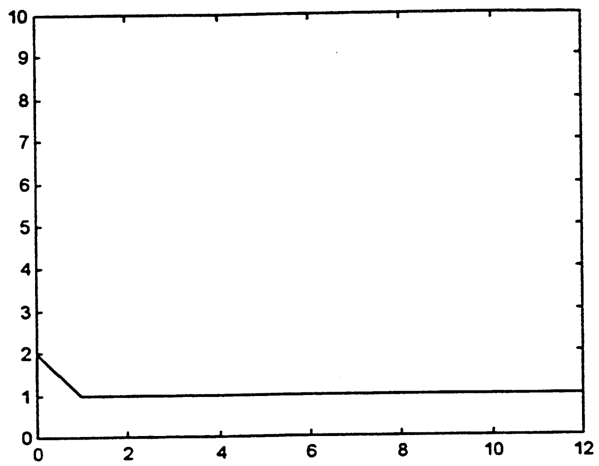


Figure D.4b FTL Iterations

D.5 IGRIP GRAPHICS

Figure D.5 shows the STM at the end of trajectory four, from the IGRIP model. This figure is the same as Fig. D.2d, with a different view angle, modified cylindrical payload, and solid model graphics.

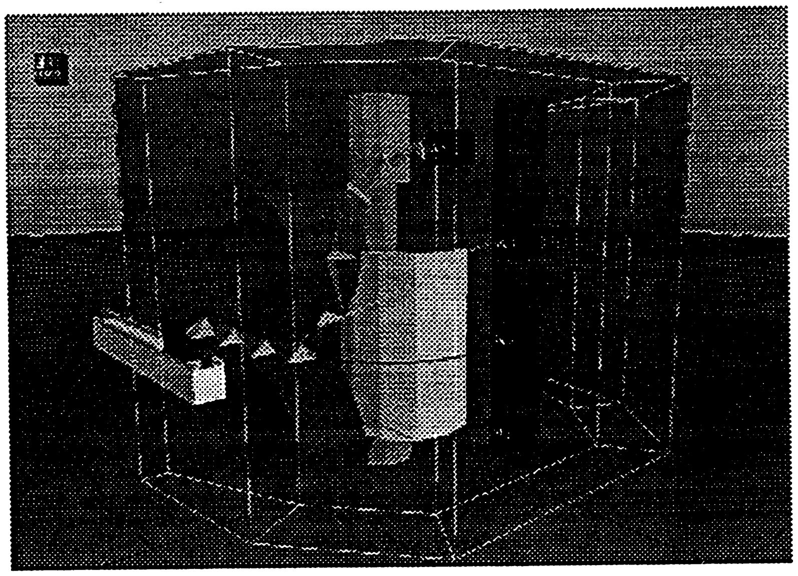


Figure D.5 IGRIP Model at Trajectory 4 End

ATTACHMENT - REPORT DOCUMENTATION PAGE - ITEM #6 (Authors)

1995 Research Reports

NASA/ASEE Summer Faculty Fellowship Program

REPORT AUTHORS:

Dr. Alfred S. Andrawis - South Dakota State University

Dr. William V. Brewer - Jackson State University (Mississippi)

Dr. John R. Cannon - University of Central Florida

Dr. Guillermo Colon - University of Puerto Rico-Mayaguez

Dr. Kemal Efe - University of Southwestern Louisiana

Dr. Gerald W. Evans - University of Louisville (Kentucky)

Dr. Gregory S. Forbes - The Pennsylvania State University

Dr. Raghvendra R. Gejji - Western Michigan University

Dr. Lucille A. Giannuzzi - University of Central Florida

Dr. Barbara H. Glasscock - California State Polytechnic University

Dr. Rhyn H. Kim - University of North Carolina-Charlotte

Dr. Gwendolyn D. Mitchell - Auburn University (Alabama)

Dr. Horacio A. Mottola - Oklahoma State University

Dr. Robert E. Peale - University of Central Florida

Dr. Rafael A. Perez - University of South Florida

Dr. Allen L. Rakow - Colorado State University

Dr. Dan E. Tamir - Florida Institute of Technology

Dr. Madjid Tavana - La Salle University (Pennsylvania)

Dr. Parveen F. Wahid - University of Central Florida

Mr. Pao-lien Wang - University of North Carolina-Charlotte

Dr. Jonathan E. Whitlow - Florida Institute of Technology

Dr. Robert L. Williams - Ohio University

EDITORS:

Dr. E. Ramon Hosler - University of Central Florida

Mr. Gregg Buckingham - John F. Kennedy Space Center