1996 NASA/ASEE SUMMER FACULTY FELLOWSHIP PROGRAM
JOHN F. KENNEDY SPACE CENTER
UNIVERSITY OF CENTRAL FLORIDA

FOLLOW-THE-LEADER CONTROL FOR THE PIPS PROTOTYPE HARDWARE

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

KSC Colleague - Thomas Lippitt
Robotics

Contract Number NASA-NGT10-52605

August 9, 1996

ABSTRACT

This report summarizes the author's summer 1996 work at NASA Kennedy Space Center in the Advanced Systems Division. The subject was 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. During the summer of 1995, the author worked on the same project, developing a follow-the-leader (FTL) algorithm in graphical simulation which ensures whole-arm collision avoidance by forcing ensuing links to follow the same tip trajectory. The summer 1996 work was to control the prototype PIPS hardware in follow-the-leader mode.

This report summarizes improvements in the algorithm accomplished this summer. Angle-to-length mappings and length-to-LVDT voltage calibrations are presented; these were required for FTL hardware implementation. The algorithm was improved with a general feed-line for FTL (rather than straight out from the zero angles as last summer), reduced iterations for solution convergence, and the inclusion of joint limit checks for trajectories. Teleoperation was developed and implemented as the primary path planning mode for the prototype hardware. In this mode, the operator defines obstacle-free trajectories for the manipulator tip using a hand controller, either off- or on-line. Improvements in the existing low-level *C* code were made to enable FTL motions. *C++* code was developed to run the FTL algorithm on-line; this code was interfaced to the low-level control *C* code. A videotape was produced to document proof-of-concept FTL control of the prototype hardware.

The project was successful in providing FTL control in hardware. The STS-82 payload mockup was used in the lab to demonstrate serpentine motions to avoid obstacles in a realistic environment. Four trajectories are delivered for this payload in this report. A general FTL prototype hardware demonstration capability including teleoperation is the primary accomplishment of the summer.

This ten-page report presents highlights of the thirty-seven-page report delivered to Tom Lippitt of NASA KSC at the end of the project [6]. Please request the full version from the author or Tom Lippitt if desired.

1. INTRODUCTION

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], [3]. 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. [4] and modified by NASA) has eighteen degrees-of-freedom.

The author worked on the same project during the summer of 1995, where a follow-the-leader algorithm was successfully developed and implemented to the KSC serpentine truss manipulator in graphical simulation. Given a obstacle-free trajectory for the manipulator tip , the follow-the-leader algorithm ensures whole arm collision avoidance by forcing each ensuing link to follow the tip. The primary goal of the summer 1996 work was implementation of the follow-the-leader algorithm to the prototype hardware.

This report is organized as follows. Section 2 summarizes improvements in the general FTL algorithm including a general feed-line, teleoperation (human-based path planning), and joint limit checks. Section 3 discusses hardware implementation of the FTL algorithm and STS-82 hardware trajectory simulations. The conclusion follows, including a summary of accomplishments and design lessons learned from simulation.

This report presents project highlights because it is limited to ten pages in length. For detailed reports on two summers' work on this project, please see [5, 6].

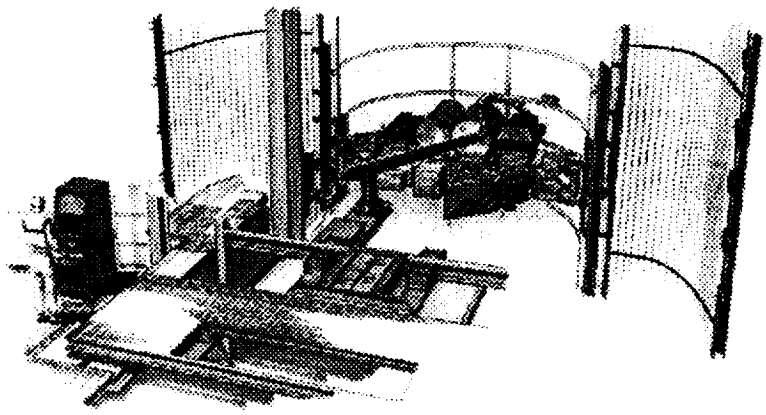

Figure 1.1 PIPS Conceptual Design

2. ALGORITHM IMPROVEMENTS

2.1 General Feed-Line

In last summer's algorithm, the STM was constrained to start follow-the-leader trajectories from the nominal reset position, where all joint angles are zero and the STM configuration is straight out along the prismatic joint. With this limitation, the crane joint θ_2 and the third base joint θ_3 are never used during trajectories. A significantly greater follow-the-leader workspace is enabled if the STM is fed onto the trajectory along general lines in space, rather than just along the prismatic track.

Figure 2.1 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 2, 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, as before. However, the initial position for the STM tip, spine point {18}, is determined by a sequence of two rotations: 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:

$$ {}^0_F R = R_X(\beta)R_Y(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\beta & -s\beta \\ 0 & s\beta & c\beta \end{bmatrix} \begin{bmatrix} c\alpha & 0 & s\alpha \\ 0 & 1 & 0 \\ -s\alpha & 0 & c\alpha \end{bmatrix} = \begin{bmatrix} c\alpha & 0 & s\alpha \\ s\alpha s\beta & c\beta & -c\alpha s\beta \\ -s\alpha c\beta & s\beta & c\alpha c\beta \end{bmatrix} \tag{2.1} $$

For all trajectories, the first two points are: ${}^F P_1 = \{0 \;\; 0 \;\; 0\}^T$ and ${}^F P_2 = \{0 \;\; 0 \;\; L_2\}^T$. (Note: ${}^B P_C$ is the vector to the origin of frame $\{C\}$ from the origin of frame $\{B\}$, expressed in the coordinates of frame $\{B\}$). 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 $\{0\}$:

$$ {}^0 P_i = {}^0_F T \, {}^F P_i \tag{2.2} $$

where (see Fig. 2.1):

$$ {}^{0}_{F}T = \begin{bmatrix} c\alpha & 0 & s\alpha & X_S \\ s\alpha s\beta & c\beta & -c\alpha s\beta & 0 \\ -s\alpha c\beta & s\beta & c\alpha c\beta & L_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.3} $$

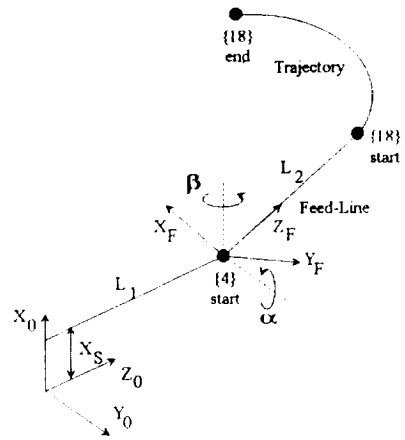

Figure 2.1 General Feed-Line Geometry

2.2 Teleoperation

Teleoperation is the most effective, reliable, and safe method for path planning of the PIPS system. Under teleoperation, a human operator enters commands to a robot system via a hand controller and the system responds. In the context of hyper-redundant serpentine manipulators, the operator enters obstacle-free trajectories for the manipulator tip, step-by-step. The follow-the-leader algorithm ensures that whole arm collision avoidance is maintained, also step-by-step. If any step results in a violation of joint limits, the operator is given the option to try again, but the bad command set is not sent to the manipulator. To extract the manipulator from a teleoperated trajectory, the command history is reversed.

For spatial teleoperation, three-dof input is sufficient, which controls relative XYZ positions. Orientation at the manipulator tip (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 next trajectory point relative to 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. 2.2, the hand controller can be aligned with the manipulator tip video monitor so teleoperation is natural. Input motion is relative to the manipulator tip coordinate frame $\{i\}$. Teleoperation is greatly enhanced by placing two or three cameras in the workspace to provide orthogonal views. In this report, the computer keyboard was used to simulate a virtual hand controller (typing numerical commands). Teleoperation would be much easier using an actual hand controller with proportional readings from each axis.

Figure 2.3 shows the i^{th} teleoperation step where the next trajectory point ${}^{0}\{{}^{0}P_{i+1}\}$ is determined based on the current trajectory point ${}^{0}\{{}^{0}P_{i}\}$ using the following vector-loop-closure equation:

$$ {}^{0}\{{}^{0}P_{i+1}\} = {}^{0}\{{}^{0}P_{i}\} + {}^{0}\{{}^{i}P_{i+1}\} \tag{2.4} $$

(Note: ${}^{A}\left\{{}^{B}P_{C}\right\}$ is the vector to the origin of frame $\{C\}$ from the origin of frame $\{B\}$, expressed in the coordinates of frame $\{A\}$.) 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 (Note: the order of matrix multiplication is opposite from the fixed rotation sequence presented in Section 2.1).

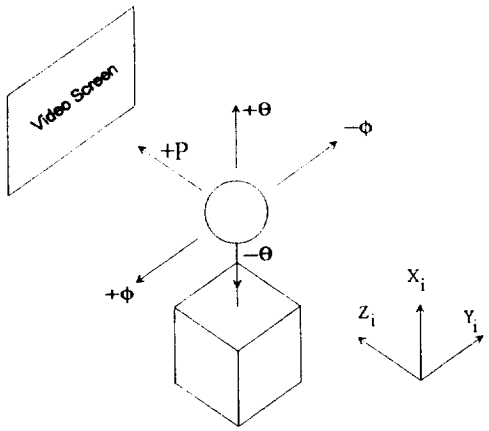

Figure 2.2 Teleoperation Hand Controller

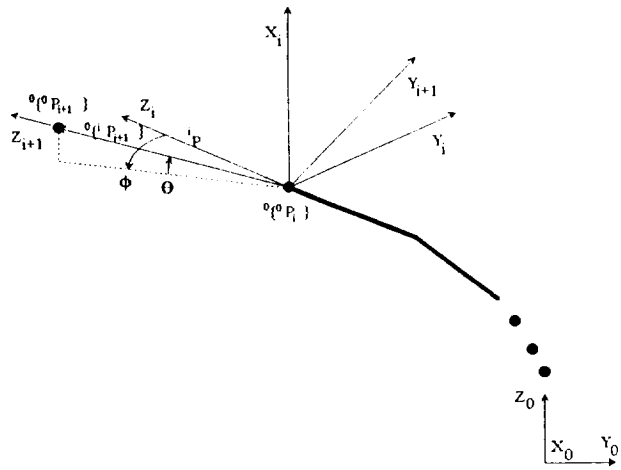

Figure 2.3 i^{th} Teleoperation Step

The relative vector ${}^{0}\left\{{}^{i}P_{i+1}\right\}$ for Eq. 3.6 is found from:

$$ {}^{0}\left\{{}^{i}P_{i+1}\right\}={}^{0}_{i}R\,{}^{i}\left\{{}^{i}P_{i+1}\right\} \tag{2.5} $$

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

$$ {}^{i}\left\{{}^{i}P_{i+1}\right\}=R_X(\phi)R_Y(\theta)\,{}^{i}P=\begin{Bmatrix} Ps\theta \\ -Pc\theta s\phi \\ Pc\theta c\phi \end{Bmatrix} \tag{2.6} $$

The rotation matrix forms are given in Eq. 2.1. In Eq. 2.5, the rotation matrix ${}^{0}_{i}R$ must be initialized to the starting orientation, at the second trajectory point (which is the first point the tip can reach). If the starting point is the nominal straight reset configuration, ${}^{0}_{1}R=I_3$. If a general feed-line is used, the initial orientation matrix is ${}^{0}_{1}R={}^{0}_{F}R$, given in Eq. 2.1. The rotation matrix ${}^{0}_{i}R$ must be updated after each successful teleoperation input as follows, to prepare for the next input step:

$$ {}^{0}_{i}R\rightarrow{}^{0}_{i+1}R={}^{0}_{i}R\,{}^{i}_{i+1}R \tag{2.7} $$

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

$$ {}^{i}_{i+1}R=R_X(\phi)R_Y(\theta) \tag{2.8} $$

2.3 Initial Q_i

As discussed in Section 2.2, the first step in calculating a follow-the-leader step is to shape the manipulator to the trajectory from tip to base. This is accomplished by intersecting manipulator segment link spheres of radii Q_i with the trajectory straight-line segments. Due to the offset universal joint structure of the prototype hardware, the intersect/inverse position kinematics computation is iterative. In last summer's work, the initial Q_i were taken as average values for generality [5]. This summer it was realized that if the initial manipulator configuration is straight (straight out or with a general feed-line), the initial Q_i values should be the maximum possible, $Q_i = \Delta_i + S_{i+1}$ because this leads to the exact solution the first iteration. Therefore, this improvement was made in the algorithm; the data is summarized in Table A.2.

In last summer's work, to achieve a spine point error tolerance of 0.1", two iterations were required at the initial step and only one thereafter since the Q_i values are continuously updated in the inverse kinematics solution. With the improvement in initial Q_i, only one iteration is required to achieve the error tolerance 0.1".

2.4 Joint Limit Checks

Last summer joint limit checks were based on plotting angular data resulting from the follow-the-leader data and inspecting to see if any limits are violated. This process was facilitated by bounding the graphs with the appropriate joint limit bounds, but it was not effective. This summer joint limit checks have been implemented in MATLAB and *C++* code in terms of logic statements. This is used in both automated (input *XYZ* data) and teleoperated modes. The variables **lolim** and **hilim** contain hardware joint angular (and d_1) limits which were derived from the actual measured limits on LVDT voltage and ball-screw actuator length. If joint limits are violated during simulation, the array **lims** is filled: with 0 if the corresponding joint does not meet a limit, and with 1 if the corresponding joint exceeds its limit. Then the array **jons** is displayed which contains the numerical values of the bad step, which of course cannot be sent to the hardware.

3. HARDWARE IMPLEMENTATION

3.1 Calibration

The follow-the-leader algorithm results in STM joint angle histories, eighteen joint commands for each input *XYZ* trajectory point. The STM hardware accepts eighteen LVDT voltage commands to drive each stepper motor / gear box / ball screw actuator combination. The LVDT voltage is the feedback sensor to ensure each joint moves to the commanded location. Therefore, a sequence of two transformations is required: 1) Joint angle to joint length mapping; and 2) Joint length to LVDT voltage calibration. The mathematics for these transformations is presented in the next two sub-sections. Both require extensive physical and electronic measurements, which are presented in appendices.

3.1.1 Angle-to-Length Mapping

This section presents the angle-to-length mapping required for each of the eighteen STM joints.

3.1.1.1 Prismatic Joint 1. The first joint is already a sliding joint, hence no mapping is required.

3.1.1.2 Crane Joint 2. Given a general θ_2, the corresponding L_2 must be calculated (see Figure 3.1):

$$L_2 = \sqrt{P_X^2 + P_Y^2} \tag{3.1}$$

The parameters for Eq. 3.1 are given below:

$$P_X = C + E\cos\beta$$
$$P_Y = -D + E\sin\beta$$

$$E = \sqrt{A^2 + B^2} \qquad \beta = \theta_2 + \phi_2 + 90^\square \qquad \phi_2 = \tan^{-1}\left(\frac{A}{B}\right)$$

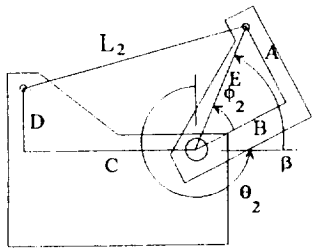

Figure 3.1 Crane Joint Model

3.1.1.3 Odd serpentine joints 3 through 17. The odd serpentine joints 3, 5, 7, 9, 11, 13, 15, and 17 have the same structure, called a type "A" box in [4]. In a type "A" Box, positive change in L_i corresponds to a positive change in θ_i (see Figure 3.2). At the nominal $\theta_i = 0$ position, the following relationships hold:

$$L_i = L - \delta \qquad \theta_0 = \tan^{-1}\left(\frac{L}{H}\right) - \tan^{-1}\left(\frac{\delta}{H}\right) \tag{3.2}$$

Given a general θ_i, the corresponding L_i can be calculated using Law of Cosines:

$$L_i = \sqrt{L_V^2 + H_V^2 - 2L_V H_V \cos(\theta_0 + \theta_i)} \tag{3.3}$$

where: $L_V = \sqrt{L^2 + H^2}$ $\qquad H_V = \sqrt{H^2 + \delta^2}$

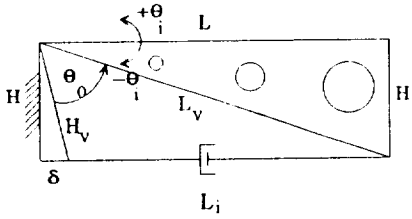

Figure 3.2 "A" Box Joint Model

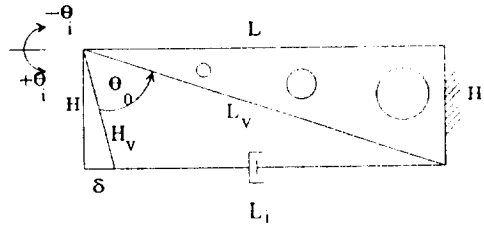

Figure 3.3 "B" Box Joint Model

3.1.1.4 Even serpentine joints 4 through 18. The even serpentine joints 4, 6, 8, 10, 12, 14, 16, and 18 have the same structure, called a type "B" box in [4]. In a type "B" Box, positive change in L_i corresponds to a negative change in θ_i (see Figure 3.3). At the nominal $\theta_i = 0$ position, Eq. 3.2 still holds. Given a general θ_i, the corresponding L_i can be calculated using Law of Cosines:

$$L_i = \sqrt{L_V^2 + H_V^2 - 2L_V H_V \cos(\theta_0 - \theta_i)} \tag{3.4}$$

3.1.2 Length-to-Voltage Calibration

This section presents the length-to-voltage calibration required for each of the eighteen STM joints. There are two different joint categories: 1) Motion base joints 1 and 2; and 2) Serpentine joints 3 through 18.

3.1.2.1 Motion base joints 1 and 2. Since the LVDT devices are linear, a linear calibration was performed between the minimum and maximum conditions. Given a general L_i, the LVDT voltage V_i is calculated using a linear equation for hardware feedback control.

$$V_i = V_{MIN} + \left(\frac{V_{MAX} - V_{MIN}}{L_{MAX} - L_{MIN}} \right) \left(L_i - L_{MIN} \right) \tag{3.5}$$

3.1.2.2 Serpentine joints 3 through 18. The calibrations for these joints are similar to Eq. 3.5. However, in order to ensure that the nominal STM zero position gives the measured nominal voltage V_0, a half-side calibration is used.

$$V_i = V_{MIN} + \left(\frac{V_0 - V_{MIN}}{L_0 - L_{MIN}} \right) \left(L_i - L_{MIN} \right) \tag{3.6}$$

3.2 Operational Scenario

There are several scenarios under which the hardware may be operated in follow-the-leader (FTL) mode. This section briefly discusses the options; please see [6] for more information.

The low-level control *C* code was developed by NASA and modified in this project. This code enables feedback control on each joint to achieve commanded LVDT voltages singly or in combinations.

Off-line MATLAB code was developed to program follow-the-leader motions, complete with graphics and animation. This code allows teleoperation, computation of *XYZ* trajectories, and input of externally-generated *XYZ* trajectories. The fidelity of the graphics is sufficient to avoid obstacles in the real hardware. The follow-the-leader algorithm generates histories of d_l and joint angles, which are mapped to lengths and the lengths are calibrated to equivalent LVDT voltages for each stepper motor / gearbox / ball screw actuator combination. The MATLAB code writes ASCII files to disk; the **VLT.DAT** or **TRAJ.DAT** files can be downloaded to the operational hardware *C* code to execute MATLAB-generated trajectories.

High-level follow-the-leader *C++* code was developed by graduate assistant James Mayhew to run the MATLAB code on-line. This code interfaces to the modified NASA low-level *C* code.

3.3 STS-82 Hardware Simulation

This section discusses hardware demonstration of follow-the-leader trajectories developed for the STS-82 payload, which is scheduled to fly in February, 1997. The hardware setup includes the eighteen-dof STM, Shuttle pallet, and mock-up STS-82 payload. The Shuttle pallet and STS-82 payload were modeled in MATLAB, along with the spine of the STM.

Four follow-the-leader trajectories were developed off-line using the MATLAB code 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. The four trajectories are named below, along with output trajectory and voltage command data files.

1) *Remove WFPC SIPE Box Plastic Sheeting Task*	**TRAJ1_96.DAT**	**VLT1_96.DAT**
2) *Inspect Load Isolation System Strut Task*	**TRAJ2_96.DAT**	**VLT2_96.DAT**
3) *Inspect Keel Fitting Task*	**TRAJ3_96.DAT**	**VLT3_96.DAT**
4) *Retrieve Witness Plates Task*	**TRAJ4_96.DAT**	**VLT4_96.DAT**

A three minute videotape was produced by the PI to highlight the summer's accomplishments and demonstrate proof-of-concept follow-the-leader hardware control of the prototype STM hardware. Trajectory 3 is the featured trajectory and the motion is time-lapsed. Three final trajectories were developed and appear after the

narrated portion of the video. Figures 3.4 and 3.5 are photographs of the initial and final STM configurations for **TRAJ1_96.DAT**, respectively.

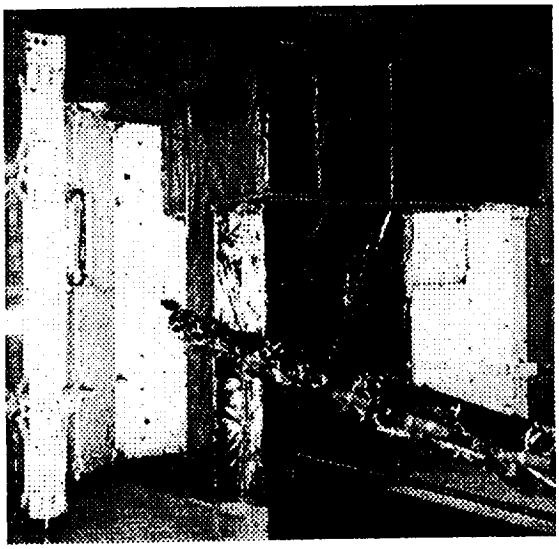

Figure 3.4 Initial STM Configuration

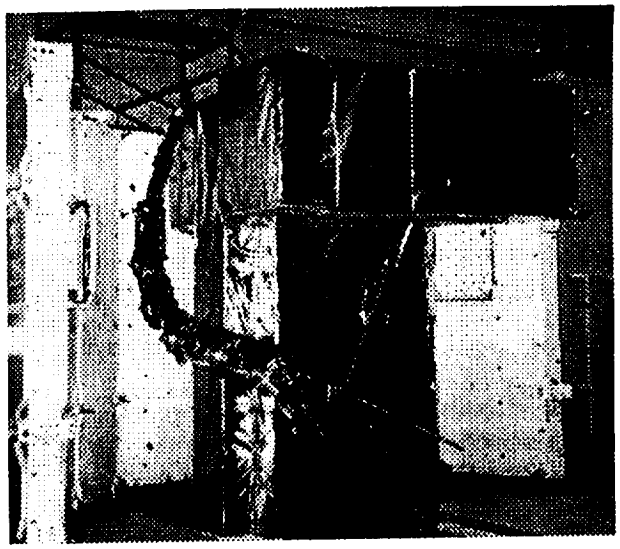

Figure 3.5 Final STM Configuration

4. CONCLUSION

This report highlights the PI's summer 1996 project in FTL hardware control of the prototype STM hardware. During last summer's project the PI developed the FTL algorithm and demonstrated it in graphical simulation. This year's project successfully accomplished FTL control in hardware, and several improvements were made to the algorithm in addition. Given an obstacle-free trajectory for the hyper-redundant manipulator tip, the FTL algorithm assures whole-arm collision avoidance for the entire trajectory by forcing all ensuing links to follow the tip.

Accomplishments for the summer 1996 project are as follows.

- Follow-the-leader (FTL) proof-of-concept demonstration in hardware
- Mapping and calibration from joint angles to LVDT voltages
- Determination and avoidance of hardware joint limits
- Upgrade FTL algorithm to include general feed-line and better convergence
- Implementation of teleoperation as primary path planning mode
- Adaptation of off-line MATLAB code to on-line C++ code
- Hardware simulation of STS-82 payload follow-the-leader trajectories
- Demonstration capability for future FTL tasks
- Videotape to document results and demonstrate hardware trajectories

The summer 1996 project was successful in demonstrating FTL trajectories in hardware. One product from this project over the past two summers is a list of design lessons learned from the prototype STM hardware and control system. These lessons should form part of the specifications for the final PCR PIPS hardware.

- 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.

- All joint limits should be increased.
- The motion base must have more range in three dimensions.
- Servo controlled actuators should be used so all motors can reach their goals simultaneously.
- The hardware must be lighter yet also stiffer.
- Actuation redundancy should be provided in the event of joint failures.
- LVDT voltage noise must be reduced.

5. 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] Herman, H., and Schempf, H., "Serpentine Manipulator Planning and Control for NASA Space-Shuttle Payload Servicing", CMU-RI-TR-92-10, Carnegie-Mellon University, October, 1992.

[4] 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.

[5] Williams, R.L., II, "Follow-the-Leader Algorithm for the Payload Inspection and Processing System", Final Report, 1995 NASA/ASEE Summer Faculty Fellowship Program, NASA Kennedy Space Center, August, 1995.

[6] Williams, R.L., II, "Follow-the-Leader Control for the Payload Inspection and Processing System Prototype Hardware", Final Report delivered to Tom Lippitt, 1996 NASA/ASEE Summer Faculty Fellowship Program, NASA Kennedy Space Center, August, 1996.

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

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

1996 Research Reports

NASA/ASEE Summer Faculty Fellowship Program

REPORT AUTHORS:

Dr. Mustafa A.G. Abushagur - University of Alabama-Huntsville

Mr. Randy K. Buchanan - Pittsburg State University (Kansas)

Dr. Luz M. Calle - Randolph-Macon Woman's College (Virginia)

Dr. Guillermo Colon - University of Puerto Rico-Mayaguez

Dr. Roger G. Ford - St. Mary's University (Texas)

Dr. Isaac Ghansah - California State University-Sacramento

Dr. David G. Jenkins - University of Illinois-Springfield

Dr. Khaled A. Kamel - University of Louisville (Kentucky)

Dr. Timothy G. Kotnour - University of Central Florida

Dr. Samuel P. Kozaitis - Florida Institute of Technology

Dr. David Kozel - Purdue University-Calumet (Indiana)

Dr. Jerome P. Lavelle - Kansas State University

Dr. Rasiah Loganantharaj - University of Southwestern Louisiana

Dr. Mark B. Moldwin - Florida Institute of Technology

Dr. Robert E. Peale - University of Central Florida

Dr. Marvin J. Pitts - Washington State University

Dr. Rodney G. Roberts - Florida A&M University-Florida State University College of Engineering

Dr. John M. Russell - Florida Institute of Technology

Dr. Ryan D. Stansifer - Florida Institute of Technology

Dr. Madjid Tavana - La Salle University (Pennsylvania)

Dr. Jonathan E. Whitlow - Florida Institute of Technology

Dr. Robert L. Williams - Ohio University

EDITORS:

Dr. Roger Johnson - University of Central Florida

Mr. Gregg Buckingham - John F. Kennedy Space Center

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	October 1996	Contractor Report - Summer 1996

4. TITLE AND SUBTITLE

1996 Research Reports
NASA/ASEE Summer Faculty Fellowship Program

5. FUNDING NUMBERS

NASA Grant
NGT10-52605

6. AUTHOR(S)

See attached list

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

University of Central Florida
Orlando, Florida 32816-2450
John F. Kennedy Space Center
Kennedy Space Center, Florida 32899

8. PERFORMING ORGANIZATION REPORT NUMBER

NASA CR-202756

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

National Aeronautics and Space Administration
Washington, D.C. 20546

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT

Unclassified - Unlimited
Subject Category 99

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This document is a collection of technical reports on research conducted by the participants in the 1996 NASA/ASEE Summer Faculty Fellowship Program at the Kennedy Space Center (KSC). This was the twelfth year that a NASA/ASEE program has been conducted at KSC. The 1996 program was administered by the University of Central Florida in cooperation with KSC. The program was operated under the auspices of the American Society for Engineering Education (ASEE) with sponsorship and funding from the Office of Educational Affairs, NASA Headquarters, Washington, D.C. and KSC. The KSC Program was one of nine such Aeronautics and Space Research Programs funded by NASA in 1996. The NASA/ASEE Program is intended to be a two-year program to allow in-depth research by the University faculty member. The editors of this document were responsible for selecting appropriately qualified faculty to address some of the many problems of current interest to NASA/KSC.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Research and Technology	290
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL