SYSTEM IDENTIFICATION OF A FREE FLOATING TELEROBOT USING KALMAN FILTERING AND A STEREOSCOPIC VISION SENSOR

A Thesis

by

WILLIAM WYBORN BENSON

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

December 1998

Major Subject: Aerospace Engineering

SYSTEM IDENTIFICATION OF A FREE FLOATING TELEROBOT USING KALMAN FILTERING AND A STEREOSCOPIC VISION SENSOR

A Thesis

By

WILLIAM WYBORN BENSON

Submitted to Texas A&M University
in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

Approved as to style and content by:

Thomas C. Pollock
(Chair of Committee)

John L. Junkins
(Member)

Jo W. Howze
(Member)

Terry Alfriend
(Head of Department)

December 1998

Major Subject: Aerospace Engineering

ABSTRACT

System Identification of a Free Floating Telerobot Using Kalman Filtering and a Stereoscopic Vision Sensor.

(December 1998)

William Wyborn Benson, B.S., Washington and Lee University

Chair of Advisory Committee: Dr. Thomas C. Pollock

A telerobot has been acquired that floats on air bearings and is intended to simulate the dynamics of a spacecraft in a two-dimensional plane. The robot was delivered without a computer, sensors or documentation so an effort has been launched to determine how the apparatus worked and to identify the model parameters associated with mass, moment of inertia and thrust. The robot has been modified to accommodate a laptop as the onboard computer and a unique stereoscopic vision sensor as a navigation system. The unknown model parameters are then identified using both least squares estimation and Kalman filtering.

The unique stereoscopic vision sensor system is based on one-dimensional position sensing diodes (PSD's) and active targets that broadcast a modulated signal. The active targets are mounted at known points in the robot frame of reference and broadcast their signal to the PSD sensors that are stationary in the inertial coordinate frame. This system enables real-time attitude measurements with no moving parts.

The robot's mass, moment of inertia and the forces generated by its thrusters are identified using direct measurements and the well known linear least squares estimation algorithm. Identification using these techniques required experiments specifically designed to characterize the system. Some of the parameters may change over time, so a means of conducting on-line system identification was developed. A Kalman filter was designed which could simultaneously perform state estimation and parameter identification on-line. This technique did not require an experiment specifically designed for identification purposes and could accurately find the unknown model parameters during normal robot maneuvering.

ACKNOWLEDGEMENTS

This research would not have been possible without the generous donation of the telerobot from Lockheed Martin Astronautics in Denver, Colorado. The author would like to thank Dr. Thomas Pollock for his support and counsel during the course of this project. Special thanks are due to Dr. John Junkins for his assistance in working out the algorithm for the Kalman filter. Dr. Declan Hughes has provided invaluable guidance in the construction of the PSD sensor hardware and I have benefited greatly from working with him these past months.

I would also like to thank the many people who assisted me with getting the robot project off of the ground in the first place. Garlan Moreland from NASA JSC was instrumental with his advice on the construction of the air-bearing floor. Jim Jobling of the Nautical Archeology Department was kind enough to loan us an air compressor. Conrad Wilson has been of great assistance by fabricating numerous parts for the robot as well as assisting in their design. Finally, I would like to thank my wife Linda for her patience and support throughout the course of my studies here at Texas A&M.

TABLE OF CONTENTS

Page

LIST OF FIGURES

Page

Page

LIST OF TABLES

TABLE Page

INTRODUCTION

Analysis of spacecraft dynamical behavior normally requires utilization of computer models to simulate the influence of various disturbances and controls. The gift of a free-floating telerobot has afforded the aerospace department an opportunity to conduct experiments that are not limited to computer models. It is a valuable adjunct to the modelling process to validate such models with appropriate ground based experiments. One difficulty associated with experiments of this type is the provision of a low friction environment. Another is the precise measurement of position and attitude of the simulated spacecraft. Both of these problems must be overcome in order to identify the parameters that describe the dynamics of the simulated spacecraft. This thesis addresses the identification of a particular such apparatus. As part of this work the development of the low friction environment and a position attitude sensing system are presented.

The telerobot floats on air bearings, which permit it to float freely over a specially prepared floor. This allows the robot three degrees of freedom (x-translation, y-translation, and rotation). The telerobot is equipped with eight thrusters and a reaction wheel which provide control forces and moments for all degrees of freedom. It comes equipped with an onboard air and power supply so no umbilical is necessary. This permits the robot to float without external coupling.

A system is identifiable and observable only to the degree to which the variables describing its dynamical behavior can be measured. The unique requirements of the

The format used for this thesis is taken from the *AIAA Journal*.

telerobot require a new approach to motion detection that will provide an accurate measurement of its position and orientation. Determining an object's position and orientation can usually be accomplished by using various sensors that must be in physical contact with that object. The intended purpose of the telerobot precludes the use of such devices because it must be free to move independently and no coupling may exist between it and another object. Since there can be no "plumbing" that leads to the robot, a *means of measuring* its motion without touching it must be found. One means of remotely measuring the motion of a body are camera based stereoscopic navigation systems. These systems are useful in situations where the data may be collected and later analyzed. Unfortunately, they present problems for a real-time applications because computational requirements associated with processing video images permits an update rate of only 2 Hz.[1] A suitable alternative to these methods is present with the recent arrival of Position Sensing Diode (PSD) technology.

Real time measurements of angles to a point may be accomplished using a device based upon the PSD. Several previous investigations into the use of such devices in a stereoscopic vision based navigation system have proven that they can accurately determine the position of a point remotely in 3-dimensional space.[2] A simpler version of this concept was used in the development of a 2-dimensional sensing system. In both of the previously mentioned cases, the position of a single point was found by using triangulation from two PSD detectors. The detectors used in these two investigations projected a laser beam at a target and the angle of the reflected light was measured using the PSD. Two angles in a plane to a single target from two separate known points may be used to find the position of the target.

Using the reflected light as the detectable signal limits the range at which the position of the target may be accurately determined to a few meters. It is not necessary to use reflected light if the targets themselves are broadcasting a signal that is detectable by the PSD sensors. This method increases the signal to noise ratio and permits remote sensing at greater ranges than those using reflected laser light. The increased range of the active target PSD depends upon the amount of light transmitted. In addition, targets may be turned on and off in a specific sequence that permits tracking of multiple targets. A number of target's mounted at known positions on the body would allow the system to be used as a navigation sensor for highly accurate measurements of the telerobot's motion at every possible position and orientation.

This investigation focuses on using PSD sensors with active targets to find the position and orientation of an object in real time. This information is then applied towards system identification and feedback control. Mounting multiple active targets on the telerobot and using time multiplexing permits accurate measurement of incident angles from known inertial coordinates to known body fixed coordinates. Using a nonlinear least squares differential correction, the robot's inertial position and location may be recovered.[3]

The first step in the identification process was to define a mathematical model that accurately describes the system. These experiments made use of the autoregressive exogenous variable model (ARX) and the least squares estimation algorithm. The ARX model describes the rotational dynamics of the system. This model assumes that the parameters of the system do not change with time and that the system is linear.[4] In the initial set of experiments the output of the system using a single thruster is measured using an angular rate sensor. The data collected is later analyzed to identify the control influence of that thruster using least squares estimation.

Once the PSD sensor system is installed, other methods of system identification besides least squares may be applied. The least squares estimation technique mentioned previously is not an on-line identification procedure and it requires that a special experiment be conducted to identify the system. It would be much more desirable to identify the unknown model parameters during the course of normal operations. A Kalman filter used in conjunction with a state space model of the system is the proposed method to conduct on-line parameter estimation of the system. The unknown parameters are augmented as additional states to the state space model. If the augmented state space model is observable and controllable, then the Kalman filter can estimate these augmented states in the same manner as any other state that is not directly measured.[5] The information gathered from the PSD sensors can be applied to this Kalman filtering algorithm to produce an accurate estimate of the system parameters. The results gathered from using the least squares technique provides an initial guess for the Kalman filter algorithm.

THE LOCKHEED MARTIN TELEROBOT

General

The vehicle is a tubular steel framework into which are installed mechanical and electronic systems which allow it to glide on air bearing pads over a prepared flat floor area by means of compressed gas thrusters actuated by computer control. In use, the vehicle can be autonomous. Power sources are carried on board, so that no umbilical is required. As received, the vehicle includes eight SCUBA-type compressed air tanks supplying air to the four air bearing pads, four brake pads and eight thrusters through a series of manifolds and solenoid-operated valves. Two banks of lead-acid dry cells provide power to a set of solid state DC-DC converters that in turn power the electronics. Electronics include a standard bus card cage containing cards that perform timing and logic functions in support of the thruster system.

The telerobot mainframe has a base of approximately $4\ \text{ft.}^2$, with a height of about 3.5 ft. Its weight, assembled except for the computer, was measured using a load cell-equipped overhead crane and found to be 850 lb. Fig. 1 is a photograph of the assembled telerobot.

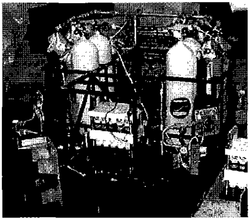

Fig. 1 The Lockheed-Martin Telerobot.

Pneumatic System

Prominent within the frame of the telerobot are eight aluminum SCUBA tanks, four of which are connected by a manifold to air pad, brake and thruster systems. The remaining four tanks are disconnected. Located on a side panel are manually operated pneumatic controls and gauges, including an "extra" set of controls for a reaction wheel which perhaps was air-actuated. A schematic diagram of the pneumatic system showing the control layout is shown in Fig. 2. Air is supplied to the pads by a manually operated regulator, with a pressure of 50 psi. being sufficient for levitation. Separately regulated air is supplied to the thrusters and brakes through solenoid on/off valves.

There are a total of eight thrusters which simply act as apertures in the manifold which may be opened and closed by solenoid on/off valves. The solenoid valves are normally closed and require 24 volts DC to open.

The brakes system consists of four pistons, which are each connected to a brake pad. The brakes are held in the release position by a spring. The pistons receive air pressure from the manifold through a regulator. When pressure is applied to the brakes

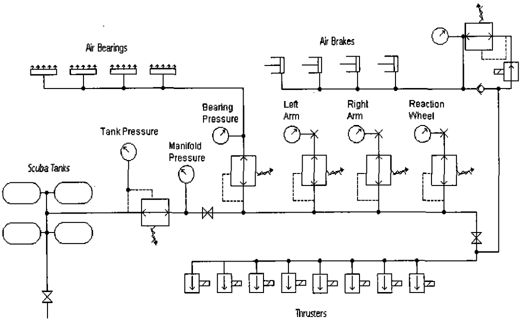

Fig. 2 Schematic of pneumatic system.

the pistons will force the brakes downward into the deployed position. Pressure to the brakes can only be released by closing a solenoid valve. The solenoid valve is normally open and can only be closed by applying 15 volts DC.

The tanks were charged with dry air to 3000 psi. pressure using a SCUBA-rated compressor and dryer loaned to the project by the TAMU Department of Nautical Archeology. All pneumatic circuits were free of leaks, and all valves functioned correctly. One questionable gauge was replaced.

Electronic Systems

The battery packs output 120 volts DC. DC-DC converters are used to provide power for the various on-board systems. The converters are programmed to provide 5 volts DC to the Transistor-Transistor Logic (TTL) control circuits, 15 volts DC to transistors providing on/off switching to the thruster solenoids and 24 volts DC to the thruster solenoids themselves.[6]

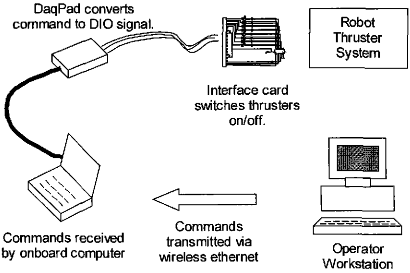
DaqPad converts command to DIO signal.
Robot Thruster System
Interface card switches thrusters on/off.
Commands received by onboard computer
Commands transmitted via wireless ethernet
Operator Workstation

Fig. 3 ***Wireless ethernet control flow.***

The robot was delivered without an onboard computer so it was necessary to find a way to interface with the robot's existing control system architecture. It was determined that a convenient way to control the robot was to use a Toshiba Tecra 500s laptop computer as the robot's onboard computer and a National Instruments DAQpad-MIO-16XE data acquisition board as the interface between the laptop and the electronic systems. The DAQpad comes equipped with eight digital input-output lines (DIO).[7] A control circuit was fabricated which would accept a 3-bit input from the data acquisition board and translate it into a specific command to the thrusters. A flowchart that explains how this system works may be found in Fig. 3.

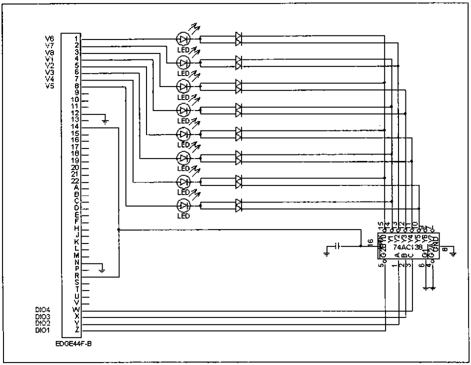

Fig. 4 DAQpad-Robot interface card.

The additional circuit card was fabricated to control the thrusters from the notebook computer (Fig. 4). This was necessary because the A/D board only has eight digital input/output (DIO) lines. Some of the DIO lines cannot be dedicated to thruster control because they are needed for other tasks. The new card uses a 74138 decoder integrated circuit chip that converts a 3-bit input into eight separate commands. The card is programmed to execute a maneuver based on the 3-bit input. The commands are:

no thrust, positive rotation, negative rotation, forward, backward, lateral left, lateral right and deploy brakes. Each output from the 74138 chip will open the appropriate thruster solenoid valves to execute the desired maneuver. The digital inputs and their related commands are listed in Table 1.

Table 1 Thruster commands

DIO Input	Command	Thrusters Activated
000	No thrust	None
001	Rotate Clockwise	V1,V3,V5,V7
010	Rotate Counter-Clockwise	V2,V4,V6,V8
011	Translate Forward	V6,V7
100	Translate Backward	V2,V3
101	Translate Right	V4,V5
110	Translate Left	V1, V8
111	No thrust	None

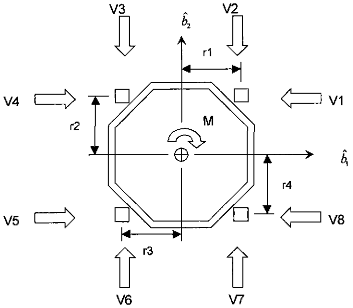

Fig. 5 Robot free-body diagram.

Equations of Motion

A free body diagram of the telerobot may be found in Fig. 5. The inertial reference frame is designated as the $\hat{e}$ frame and the robot or body fixed reference frame is designated as the $\hat{b}$ frame. The well-known coordinate transformation between the two reference frames is defined (see Fig. 6).

$$\begin{Bmatrix} \hat{b}_1 \\ \hat{b}_2 \\ \hat{b}_3 \end{Bmatrix} = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{Bmatrix} \hat{e}_1 \\ \hat{e}_2 \\ \hat{e}_3 \end{Bmatrix} \tag{1}$$

A reaction wheel is located at the geometric center. The center of mass is located off center and is at a previously unknown position due to uneven loading of the support frame. There are eight thrusters that provide translational and rotational control of the robot position on the floor. Viscous damping is present in the form of aerodynamic forces as well as interaction between the air bearings and the layer of air that suspends them. In this model the damping is treated as a lumped parameter that is linear with respect to rotation and translation rates.

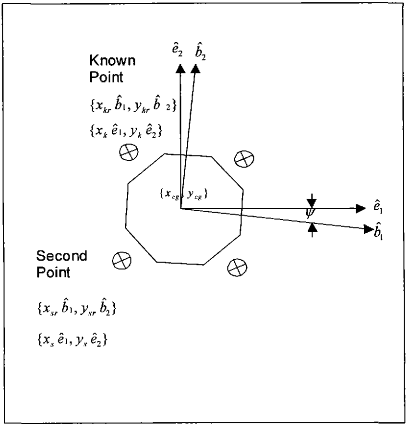
$\hat{e}_2$ $\hat{b}_2$
Known Point
$\{x_{kr}\,\hat{b}_1, y_{kr}\,\hat{b}_2\}$
$\{x_k\,\hat{e}_1, y_k\,\hat{e}_2\}$
$\{x_{cg}, y_{cg}\}$
ψ
$\hat{e}_1$
$\hat{b}_1$
Second Point
$\{x_{sr}\,\hat{b}_1, y_{sr}\,\hat{b}_2\}$
$\{x_s\,\hat{e}_1, y_s\,\hat{e}_2\}$

Fig. 6 Robot coordinate systems.

The robot is free to translate in two dimensions and can rotate about one axis, thus it has three degrees of freedom. The sum of all moments about the $\hat{b}_3, \hat{e}_3$ axis is:

$$\sum M = -\eta\omega + V_1 r_2 - V_2 r_1 + V_3 r_3 - V_4 r_2 + V_5 r - V_6 r_{34} + V_7 r_1 - V_8 r_4 \tag{2}$$

Where η is defined as the rotational linear damping coefficient.

Substituting Eq. (2) into Newton's second law:

$$\ddot{\psi} = \frac{1}{I_n}\left(-\eta\psi + V_1 r_2 - V_2 r_1 + V_3 r_3 - V_4 r_2 + V_5 r - V_6 r_{34} + V_7 r_1 - V_8 r_4\right) \tag{3}$$

The sum of the forces in the $\hat{b}_1$ direction are:

$$\sum F\hat{b}_1 = -\nu\dot{x} - V_1 + V_4 + V_5 - V_8 \tag{4}$$

where ν is defined as a linear translational viscous damping coefficient.

The sum of the forces in the $\hat{b}_2$ direction are:

$$\sum F\hat{b}_2 = -\nu\hat{x} - V_2 - V_3 + V_6 - V_7 \tag{5}$$

Making use of Eq. (1), Eq. (4) and Eq. (5) the sum of all forces in the $\hat{e}$ frame are defined.

$$\sum F\hat{e}_1 = -\nu\dot{x} - \cos\psi V_1 + \sin\psi V_2 + \sin\psi V_3 + \cos\psi V_4 + \cos\psi V_5 - \sin\psi V_6 - \sin\psi V_7 - \cos\psi V_8 \tag{6a}$$

$$\sum F\hat{e}_2 = -\nu\dot{y} - \sin\psi V_1 - \cos\psi V_2 - \cos\psi V_3 + \sin\psi V_4 + \sin\psi V_5 + \cos\psi V_6 + \cos\psi V_7 - \sin\psi V \tag{6b}$$

Substituting Eq. (6) into Newton's second law, the equations of motion become:

$$\ddot{x} = \frac{1}{m}\left(-\nu\dot{x} - \cos\psi V_1 + \sin\psi V_2 + \sin\psi V_3 + \cos\psi V_4 + \cos\psi V_5 - \sin\psi V_6 - \sin\psi V_7 - \cos\psi V_8\right) \tag{7a}$$

$$\ddot{y} = \frac{1}{m}(-\dot{v} - \sin\psi V_1 - \cos\psi V_2 - \cos\psi V_3 + \sin\psi V_4 + \sin\psi V_5 + \cos\psi V_6 + \cos\psi V_7 - \sin\psi V_8) \text{ (7b)}$$

Placed in matrix form, the equations of motion become:

$$\begin{Bmatrix} \ddot{\psi} \\ \ddot{x} \\ \ddot{y} \end{Bmatrix} = \begin{bmatrix} \frac{-\eta}{I} & 0 & 0 \\ 0 & \frac{-v}{m} & 0 \\ 0 & 0 & \frac{-v}{m} \end{bmatrix} \begin{Bmatrix} \dot{\psi} \\ \dot{x} \\ \dot{y} \end{Bmatrix} +$$

$$(8)$$

$$\begin{bmatrix} \frac{r_2 V_1}{I_n} & -\frac{r_1 V_2}{I_n} & \frac{r_3 V_3}{I_n} & -\frac{r_2 V_4}{I_n} & \frac{r_4 V_5}{I_n} & -\frac{r_3 V_6}{I_n} & \frac{r_1 V_7}{I_n} & -\frac{r_4 V_8}{I_n} \\ -\cos\psi \frac{V_1}{m} & \sin\psi \frac{V_2}{m} & \sin\psi \frac{V_3}{m} & \cos\psi \frac{V_4}{m} & \cos\psi \frac{V_5}{m} & -\sin\psi \frac{V_6}{m} & -\sin\psi \frac{V_7}{m} & -\cos\psi \frac{V_8}{m} \\ -\sin\psi \frac{V_1}{m} & -\cos\psi \frac{V_2}{m} & -\cos\psi \frac{V_3}{m} & \sin\psi \frac{V_4}{m} & \sin\psi \frac{V_5}{m} & \cos\psi \frac{V_6}{m} & \cos\psi \frac{V_7}{m} & -\sin\psi \frac{V_8}{m} \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \\ u_8 \end{Bmatrix}$$

Where u_i is 1 or 0 and represents the on or off command to a particular thruster.

LEAST SQUARES ESTIMATION

Theory

The autoregressive exogenous variable model is a simple input-output relationship described by a linear difference equation of the following form.

$$y(t)+a_0y(t-1)+a_1(t-2)+\ldots+a_{n_a}y(t-n_a)=b_0u(t-1)+b_1u(t-2)+\ldots+b_{n_b}u(t-n_b)+e(t) \quad (9)$$

This is called an equation error model since the white noise term *e(t)* enters as a direct term in the difference equation. If the unknown parameters are

$$\theta=\left[a_0,a_1,\ldots,a_{n_a},b_0,b_1,\ldots,b_{n_a}\right] \qquad (10)$$

and the following terms are introduced

$$A(y)=1+a_0y(t-1)+a_1y(t-2)+\ldots+a_{n_a}y(t-n_a) \qquad (11a)$$

$$B(u)=1+b_0u(t-1)+b_1u(t-2)+\ldots+b_{n_a}u(t-n_a) \qquad (11b)$$

then the resulting transfer function becomes

$$G(y,\theta)=\frac{B(y)}{A(y)}+\frac{e(t)}{A(y)} \qquad (12)$$

Where *A(y)* is the AR or autoregressive part and *B(y)* is the exogenous value part.

The equations of motion Eq. (8) are decoupled with respect to the states. Therefore, the effect of the thrusters on the rotational mode of the robot can be tested and measured accurately with only a rate sensor. In order to get an accurate measure of the thruster force, each thruster may be tested individually and the control influence for that thruster may be found using the ARX model and least squares estimation. Consider the angular rate equation of motion for the robot with a single thruster:

$$\ddot{\psi} = \frac{-\eta}{I}\dot{\psi} + \frac{V_i r_i}{I}u, \tag{13}$$

Which in discrete form becomes,

$$\frac{\dot{\psi}_{k+1} - \dot{\psi}_k}{\Delta t} = \frac{-\eta}{I}\dot{\psi}_k + \frac{V_i r_i}{I}u_k \tag{14}$$

This can be manipulated to give the standard ARX predictor model:

$$\dot{\psi}_{k+1} = (1 - \frac{\eta \Delta t}{I})\dot{\psi}_k + \frac{V_i r_i \Delta t}{I}u_k \tag{15}$$

Or, in the more general form,

$$y_{k+1} = a_0\, y_k + b_0 u_k \tag{16}$$

The parameters a_o and b_o must be found using some identification process and then the values for V_{ij} and η may be determined. Given a batch of data of the form:

$$Z^N = [\dot{\psi}(1), u(1), \dot{\psi}(2), u(2), \ldots.., \dot{\psi}(N), Z(N)] \tag{17}$$

If this vector of known parameters is defined as:

$$\varphi(k) = \begin{bmatrix} \dot{\psi}(k) \\ u(k) \end{bmatrix}, k = 1..N \tag{18}$$

and the vector of estimated parameters is defined as:

$$\hat{\theta}(k) = \begin{bmatrix} \hat{a}_0(k) \\ \hat{b}_0(k) \end{bmatrix}, k = 1..N \tag{19}$$

Then the predicted estimate of the output y can be defined as:

$$\hat{y} = \varphi(k)^T \theta(k) \tag{20}$$

A residual for the estimate is defined as:

$$V(\theta,N)=\sum_{k=1}^{N}\frac{1}{2}\left(\hat{\psi}(k)\ -\phi(k)^{T}\theta(k)\right)^{2} \tag{21}$$

The object of the parameter estimation is to minimize the residual.[4] The parameter estimate that minimizes the error is:

$$\hat{\theta}^{LS}=\left[\frac{1}{N}\sum_{k=1}^{N}\phi(k)\phi(k)^{T}\right]^{-1}\frac{1}{N}\sum_{k=1}^{N}\phi(k)y(k) \tag{22}$$

Which, is expressed in this specific case by the following equation.

$$\begin{Bmatrix}\hat{a}_0\\ \hat{b}_0\end{Bmatrix}=\left[\frac{1}{N}\sum_{k=1}^{N}\begin{Bmatrix}\dot{\psi}(k)\\ u(k)\end{Bmatrix}\begin{bmatrix}\dot{\psi}(k) & u(k)\end{bmatrix}\right]^{-1}\frac{1}{N}\sum_{k=1}^{N}\begin{Bmatrix}\dot{\psi}(k)\\ u(k)\end{Bmatrix}y(k) \tag{23}$$

The Identification Process

As mentioned previously, the rotational mode in Eq. (8) is decoupled from the other two modes, thus experiments to identify parameters specific to rotational motion can be conducted independently. Moreover, control inputs may be entered individually, which allows the system representation to be reduced to single input, single output (SISO) equation. The parameters of interest are the damping ratio η and the thrust for each thruster at a given manifold pressure Vi. The moment of inertia and centroid of the robot were found in a separate test and are listed in Table 2.[8]

The rotational mode was selected to define the thrusters because a reliable sensor to measure angular rate was immediately available and thrusters could be tested individually. Prior to each test, the surface of the floor was cleaned using a shop-vac and wiped down with denatured alcohol to remove any dust or debris that would interfere with the operation of the air bearings. All thrusters except for the individual of interest were disconnected. The pneumatic system was charged to a pressure of 1600 psi. or greater.

To ensure consistent results, the same location on the floor was used as the initial position of the robot for each test. Each thruster was tested at an air bearing pressure of 60 psi. and the given manifold pressure was varied from 100 psi. to 200 psi. At each successive test the manifold pressure was increased at an increment of 20 psi. This would characterize the relationship between overall manifold pressure and the actual thrust.

Results and Discussion

The identification experiments yielded the input-output data that was processed using least squares estimation. Once the discrete time state-space parameters were identified, the values for η and Vi found using Eq. (23). Table 8 - Table 15 in Appendix A display the discrete-time and continuous time parameters for each thruster at a given manifold pressure.

Table 2 Robot parameters

Moment of Inertia (slug-ft)	47.8
Mass (slugs)	26.4
V1 Moment Arm (ft.)	1.88
V2 Moment Arm (ft.)	1.84
V3 Moment Arm (ft.)	2.05
V4 Moment Arm (ft.)	1.88
V5 Moment Arm (ft.)	2.00
V6 Moment Arm (ft.)	2.05
V7 Moment Arm (ft.)	1.84
V8 Moment Arm (ft.)	2.00

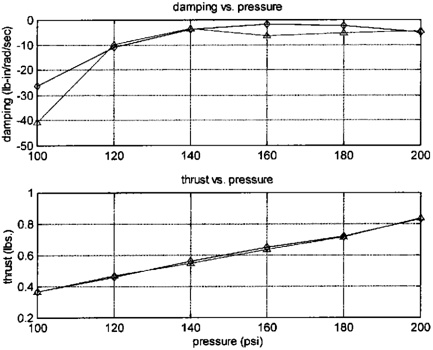

Fig. 7 V1 thrust curve.

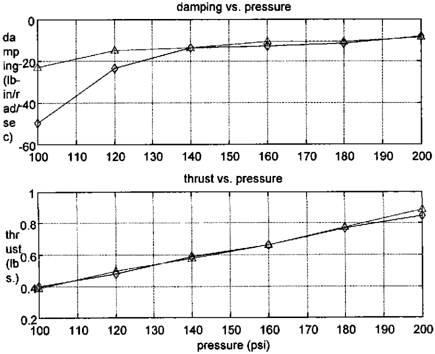

Fig. 8 V2 thrust curve.

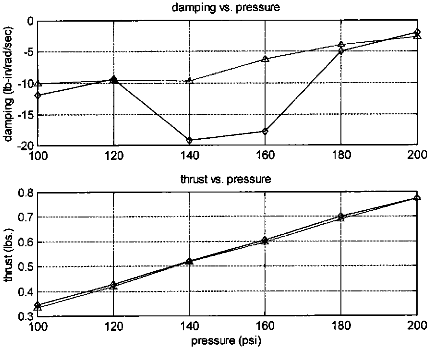

Fig. 9 V3 thrust curve.

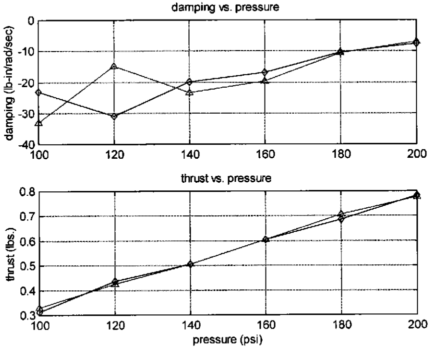

Fig. 10 V4 thrust curve.

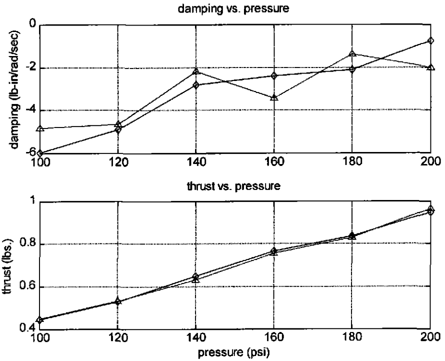

Fig. 11 V5 thrust curve.

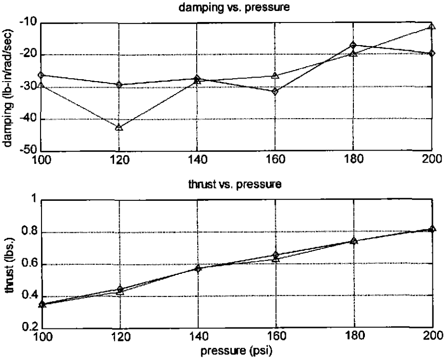

Fig. 12 V6 thrust curve.

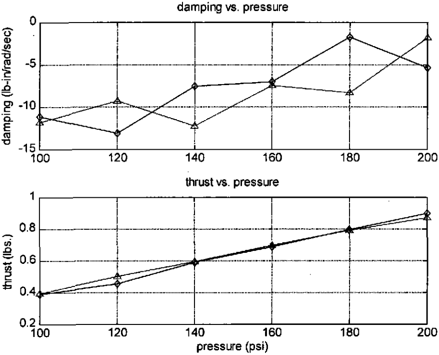

Fig. 13 V7 thrust curve.

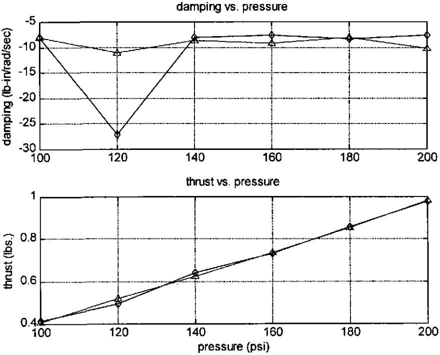

Fig. 14 V8 thrust curve.

As is evident from Fig. 7 - Fig. 14, there is a clear linear relationship between the pressure applied to the thrusters and the output. At every experiment it was necessary to manually set the input pressure, so variations from this linearity are almost certainly due to operator error rather than some nonlinearity in the system. The imprecise nature of the pressure controls makes an on-line identification process desirable. The data gathered from these experiments provides a good initial guess for such an algorithm.

The erratic values obtained for damping are probably due to the fact that the floor is not perfectly clean, flat or smooth and intermittent coulomb and viscous damping is bound to occur. The magnitude of the damping is sufficiently small that it can be taken into account as an unmodeled disturbance. Feedback control laws are designed to deal with such disturbances so this should not provide an obstacle to further experiments.

REAL TIME ORIENTATION AND POSITION SENSING

The previous discussion dealt with determining the robot's parameters using only an angular rate sensor to gather the input-output data. A sensor that can determine the robot's position and orientation accurately in real time is required for better experiments. A device based upon the position sensing diode was designed for this purpose and was installed on the robot for the purposes of navigation and parameter estimation.

PSD Sensor Description

A PSD averages the light intensity along a photo-sensitive strip and returns two currents indicative of the centroid of the energy distribution. These position sensors have an advertised analog resolution of better than one part in a million and have a very small position non-linearity. The photosensitive strip is 30mm long and is fitted in a casing manufactured by DMK Engineering.[9]

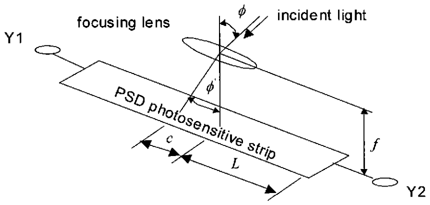

Fig. 15 Illustration of the incident light projection onto the photosensitive strip.

When the incident light strikes the PSD, two currents (Y1 and Y2 in Fig. 15) are produced and they are related to the centroid of the incident light by Eq. (24), (where *c* is the centroid). Each current is converted to a voltage by an amplifier circuit (see Fig. 16). A normalized voltage that is proportional to the position of the centroid of the incident light is generated by the following equation.[2]

$$v = \frac{c}{L} = \frac{Y2 - Y1}{Y2 + Y1} \tag{24}$$

The circuits in Fig. 16 convert the PSD currents into voltages. The first stage of the preamplifier are transimpedence opamp circuits which convert the current into a voltage. The second and third stages amplify these voltages at a gain of 680 (at 38.4 Khz., the modulation frequency).

Since the PSD is sensitive to light other than that generated by the active targets, it is necessary to modulate the output of these targets and pass the output signal from the PSD through a filter so that only this modulated signal is measured. The circuits in Fig. 17 are two eighth order bandpass filters centered at 38.4 Khz. frequency (one per PSD channel). Each opamp in the circuit represents poles and zeroes in the transfer function of the system. The first two opamps of each channel act as fourth order Butterworth high-pass filters. These filters have zeroes at the origin and poles shortly below 38.4 Khz. Following these are fourth order Butterworth low-pass filters comprised of two opamps with poles slightly above 38.4 Khz. and zeroes at infinity. Between these pole frequencies lies the pass band of the bandpass filter.

After the bandpass filter eliminates the majority of energy other than that due to the modulated signal of interest, the signals are put through rectifiers and low-pass filters (see Fig. 18), which bring the modulated signals down to baseband format (i.e. DC to a cut-off frequency). The cut-off frequency for this system is 1 Khz. These processed PSD signals are then sampled by the computer and applied to Eq. (24) to produce a normalized voltage proportional to the incident angle of the light striking the wide angle lens. A flow chart representing the signal processing of the PSD sensor is illustrated in Fig. 19.

The active targets must be controlled so that they generate enough light to produce large PSD signals, but not so large as to saturate the following opamps, Otherwise, the PSD receivers will saturate at close range or the output will attenuate as distance increases. This will reduce sensor accuracy. The four targets must also each be turned on and off in a sequence so that when the sensors are receiving their signal, the data acquisition code can distinguish which target is active. It is the ability to distinguish the incident angles that correspond to a specific target that provides the information necessary to find the robot's orientation and position. The circuit in Fig. 20 performs this task. It recieves a 4-bit digital command from the data acquisition board which indicates which target should be activated. This circuit also contains a digital clock which drives a sine wave generator chip. The output from this chip is sent to an analog multiplexor. The analog multiplexor sends this signal to one of four LED drive circuits depending upon the command from the data acquisition board. A commanded voltage sets the amplitude of this modulated signal before it reaches the multiplexor.[10]

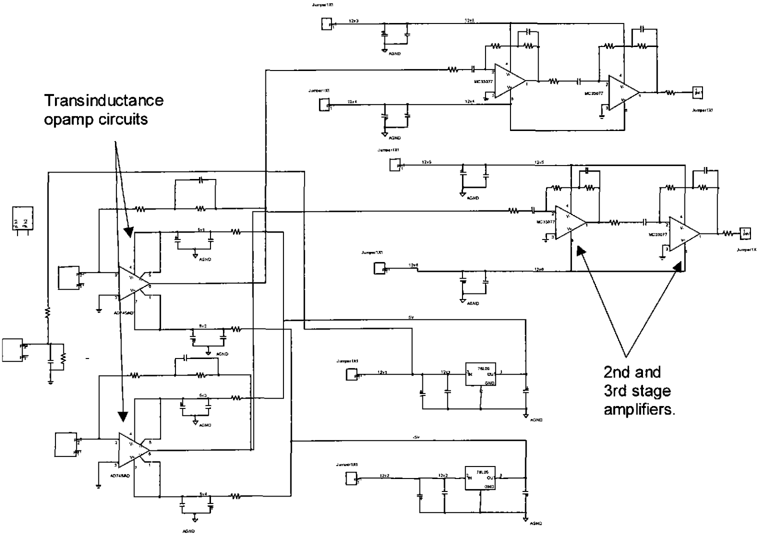

Fig. 16 PSD preamplifier.

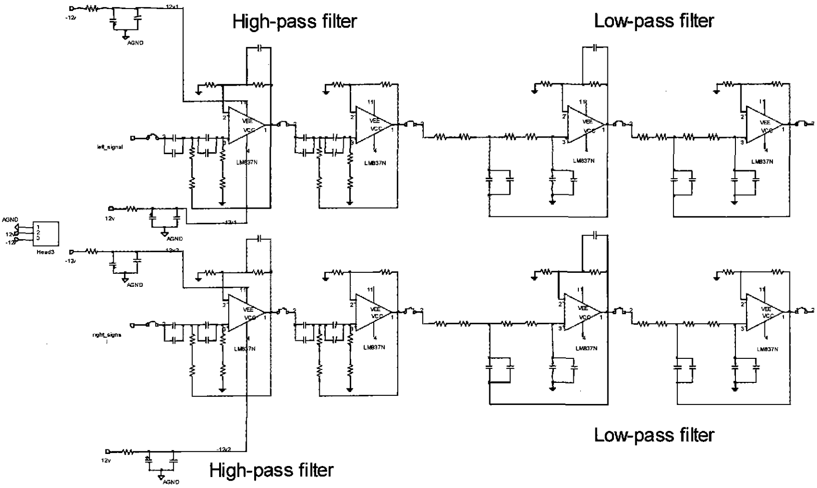

Fig. 17 Bandpass filter.

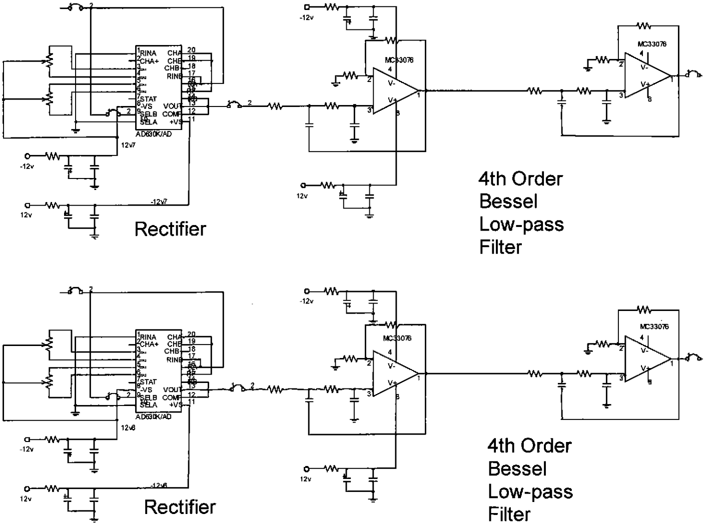

Fig. 18 Rectifier and low-pass filter.

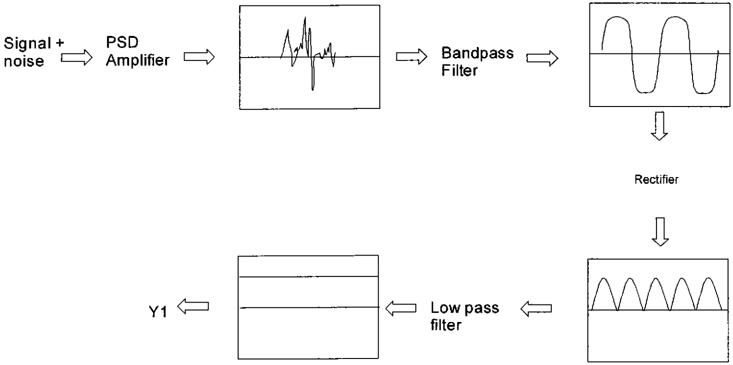

Fig. 19 PSD signal flow chart.

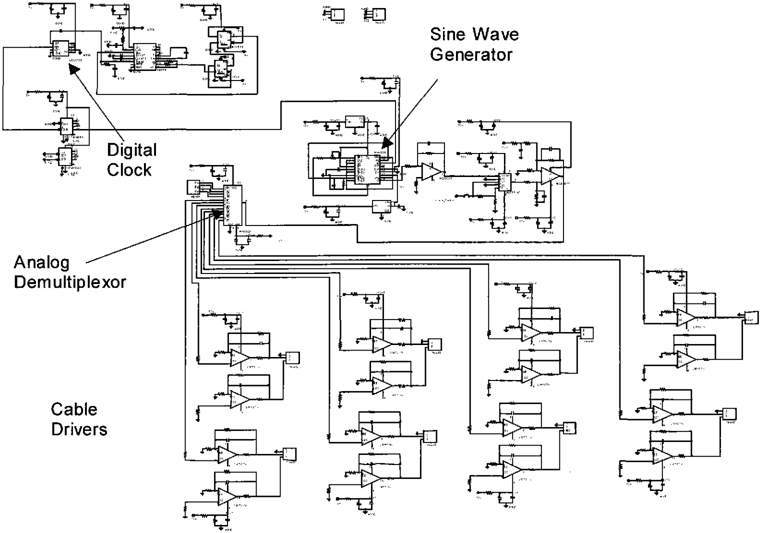

Fig. 20 Target control circuit.

PSD Calibration

The calibration process establishes the exact relationship between the normalized voltage output from the receiving circuit and the incident angle ϕ. An accuracy of 0.1 degrees or better is desired.

From Fig. 15 it can be seen that the relationship between c and ϕ' is

$$\frac{c}{f} = \tan\phi' \tag{25}$$

After dividing Eq. (25) by L and substituting into equation Eq. (24) the relationship between v and $\phi^{'}$ is found.[2]

$$v = \frac{f}{L}\tan\phi' \tag{26}$$

Since the PSD sensor field of view extends over a 100 degree arc, it cannot be assumed that the relationship between ϕ and ϕ' is a linear one. It is therefore necessary to conduct a sweep of all possible values of ϕ over the field of view and at the same time measure the normalized output voltage v.

Calibration was accomplished by setting the PSD sensor on the Contraves air bearing so that the focus of the lens at ϕ=0 was located at or near the hub of the air bearing. An active target consisting of a single bright LED (3000 millicandles max) was placed on an optical table opposite the air bearing and its altitude was adjusted so that it was the same height as the lens nominal axis. From previous experiments it has been determined that the near linear range of the PSD and lens extends over a 90 degree arc.[2] Since it is desirable to calibrate over as much of the linear range as possible, the active target was set up such that when $\phi \cong 0$, the air bearing was set at zero degrees. Fig. 21 is an illustration of the calibration apparatus.

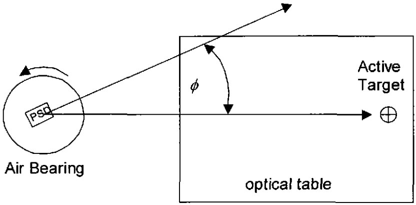

Fig. 21 PSD calibration experimental setup.

Once the PSD sensor and active target were established in the desired location, the air bearing was programmed to rotate from –50 degrees to 50 degrees at .1 degree increments. The output *v* was measured at each increment and the resulting relationship between *v* and $\tan\phi$ was established using a Chebyshev polynomial curve fit.[11] Fig. 22 shows the monotonic relationship between the normalized voltage and the incident angle tangents.

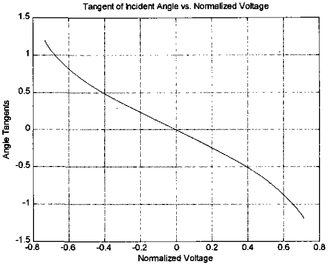

Fig. 22 Incident angle tangents vs. normalized voltage.

The normalized voltage appears to be a linear function with respect to incident angle as seen in Fig. 23. The relationship becomes nonlinear at view angles beyond 45 degrees. The output of the Y1 and Y2 channels on the PSD receiver circuit are due to the signal amplitude feedback control law that attempts to keep the maximum output of the two PSD channels (after amplification) at 7 volts by controlling the amplitude of the modulated signal. As one channel or the other becomes dominant, it will receive the most energy and therefore is the determining factor in the control of the modulated signal. It is for this reason that as the light incident angle crosses 0 degrees, one channel remains at approximately 7 volts, while the other drops below.

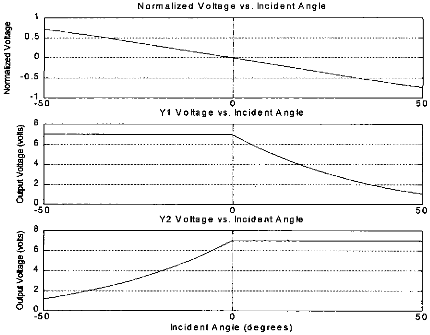

Fig. 23 Left, right and normalized voltages vs. incident angle.

Fig. 24 shows the standard deviations at a sampling rate τ of 10 Khz. The signal to noise ratio μ for the normalized voltage may be calculated by taking the signal range and dividing it by the standard deviation.

$$\mu = \frac{\max(v) - \min(v)}{mean(std\_deviations)} \tag{27}$$

Taking the information from Fig. 24 and using it in Eq. (27) the signal to average noise ratio at τ =10 Khz. is determined to be 687.3. The standard deviation and μ are a minimum at $\phi = 0$ and a maximum $\phi = \pm 50$. This is because the light aperture becomes smaller as the incident angle increases. For the experiments conducted later in this section, a sampling rate of $\tau_1 = 50$ Hz. was used by averaging 200 samples at 10 Khz. The signal to noise ratio μ_1 for 50 Hz. was then calculated to be 9727.7 using Eq. (28).

$$\mu_1 = \mu\sqrt{\frac{\tau}{\tau_1}} \tag{28}$$

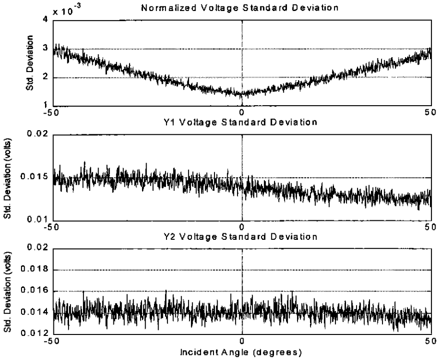

Fig. 24 PSD output standard deviations.

Establishing the Inertial Coordinate Frame

To determine the position of the robot from angle data acquired from the PSD sensors, it is necessary to determine the position and orientation of the sensors themselves in an inertial coordinate frame. This inertial coordinate frame is completely arbitrary and may be established anywhere with respect to the sensors themselves. All that is required is a series of reference points that have known locations in the inertial coordinate frame.[12]

In the experiments that follow, a small optical table mounted on the Contraves air bearing (Fig. 25) was used as a source of known points. This optical table has a 16 in. by 21 in. grid of ¼ in. holes that are precisely spaced 1 in. apart. The air-bearing table may be locked in place so that it provides a rigid platform for calibration purposes. The active target is moved to each hole and the view angles θ_{1i} and θ_{2i} are measured to each known point located at coordinate $\{x_i, y_i\}$. For situations where an optical table is not conveniently available, another technique may be used. A bar that has holes precisely drilled along its length may be used in the same way as the small optical table. The bar must be rigidly mounted so that it remains in the same position as a target is moved from point to point (Fig. 26).

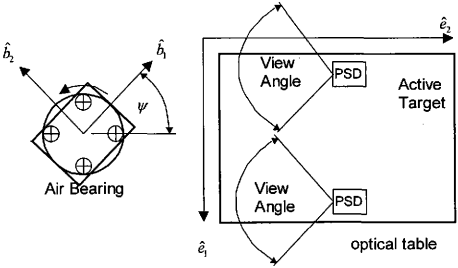

Fig. 25 Experimental setup for triangulation tests.

Once the angle data is gathered it may be analyzed to determine the position of the sensors using a nonlinear least squares differential correction. The residuals are defined as

$$\{f\} = \begin{Bmatrix} f_{11} \\ f_{21} \\ f_{12} \\ f_{22} \\ \vdots \\ f_{1n} \\ f_{2n} \end{Bmatrix}_{n=20} \tag{29}$$

Where,

$$f_{1i}(x_i, y_i, \theta_{1i}, x_{1c}, y_{1c}, \theta_{1c}) = \tan(\theta_i + \theta_{1c}) - \frac{x_i - x_{1c}}{y_i - y_{1c}} \tag{30a}$$

$$f_{2i}(x_i, y_i, \theta_{2i}, x_{2c}, y_{2c}, \theta_{2c}) = \tan(\theta_i + \theta_{1c}) - \frac{x_{2c} - x_i}{y_i - y_{2c}} \tag{30b}$$

From the well known property

$$f(q + \Delta q) = f(q) + \frac{\partial f(q)}{\partial q}\Delta q \tag{31}$$

Where,

$$q = \{x_{1c}, y_{1c}, \theta_{1c}, x_{2c}, y_{2c}, \theta_{2c}\} \tag{32}$$

it is possible to find the correction based upon each successive guess q.

$$\Delta q = -\left(\left[\frac{\partial f(q)}{\partial q}\right]^T \left[\frac{\partial f(q)}{\partial q}\right]\right)^{-1} \left[\frac{\partial f(q)}{\partial q}\right]^T f(q) \tag{33}$$

An Euler correction is used to update the current sensor position estimate.

$$q_{k+1} = q_k + \Delta q_k \tag{34}$$

If the result of Eq. (34) is then plugged back into Eq. (30) and the process is repeated, the correction should cause the next result of Eq. (33) to move closer to zero. Repeated iterations will cause the residual to converge to zero and the resulting value of q at convergence is the actual position of the PSD sensors.[5]

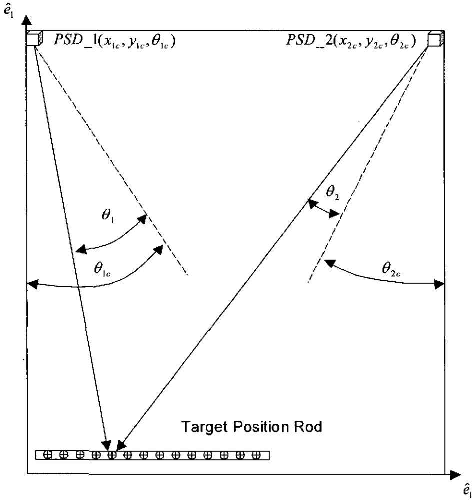

Fig. 26 PSD inertial coordinate frame.

In order to verify that the algorithm described above would converge to the actual PSD positions, a simulation was performed to test it. View angles were generated and input into the algorithm. The results may be found in Fig. 27. The algorithm converges to the proper values in less than 10 iterations with great accuracy.

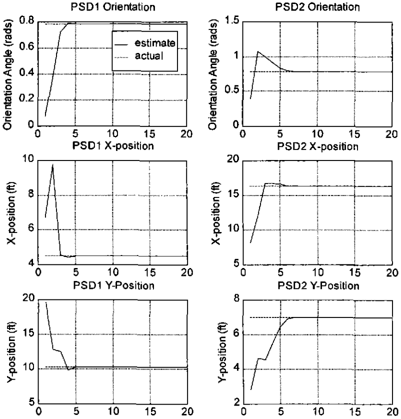

Fig. 27 Determination of PSD positions using simulated values.

Triangulation

Fig. 28 shows the robot located somewhere on the air-bearing floor. It is mounted with four active LED targets located at each corner. Two PSD receivers are located at the upper corners of the air-bearing floor and are in the same plane as the LED targets. The position of the receivers is known in terms of the variables $x_{1c}, y_{1c}, \theta_{1c}, x_{2c}, y_{2c}, \theta_{2c}$. The output from each receiver is the incident angle of the line of sight (LOS) to each visible target in the form of its tangent $\tan\theta_{jk}$, where *j* corresponds to the PSD and *k* corresponds to the target number.

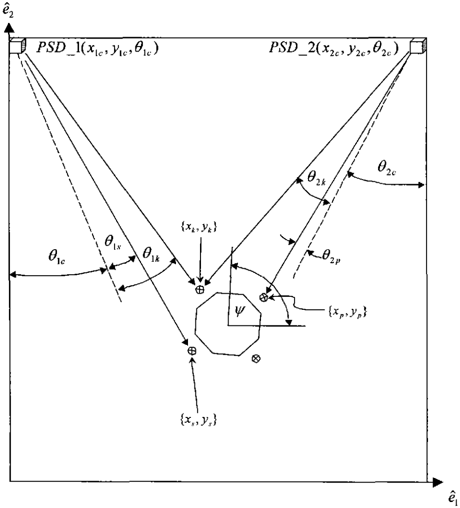

Fig. 28 Robot triangulation.

Assuming that a given target is visible to both receivers (such as the point $\{x_k, y_k\}$ in Fig. 28), the following relationship may be established.

$$y_k = \frac{y_{1c}\tan(\theta_{1c} + \theta_{1k}) + y_{2c}\tan(\theta_{2c} + \theta_{2k}) - x_{2c} + x_{1c}}{\tan(\theta_{1c} + \theta_{1k}) + \tan(\theta_{2c} + \theta_{2k})} \tag{35a}$$

$$x_k = \tan(\theta_{1c} + \theta_{1k})\left[y_{1c} - \frac{y_{1c}\tan(\theta_{1c} + \theta_{1k}) + y_{2c}\tan(\theta_{2c} + \theta_{2k}) - x_{2c} + x_{1c}}{\tan(\theta_{1c} + \theta_{1k}) + \tan(\theta_{2c} + \theta_{2k})} \right] + x_{1c} \tag{35b}$$

With the establishment of a single known point, the position of all the other active targets and the centroid of the robot may be determined if the orientation can be found. Under most circumstances, at least two active targets will be visible to both detectors and the orientation may be found by taking the arctangent of the difference between the $\hat{e}_2$ coordinates over the difference of the $\hat{e}_1$ coordinates of the two known points. However, there are also many cases where there will only be one point that is visible to both receivers and each receiver will be able to see a second target, but not the same target.

An algorithm must be developed that can find the position of the robot under all conditions. The example in Fig. 28 shows that only one target is visible to both sensors, and that each sensor can see a separate additional target. The governing equations for the inertial frame location of the active targets in terms of the incident angle tangents are as follows.

$$\tan(\theta_{1c} + \theta_{1s}) = \frac{x_s - x_{1c}}{y_{1c} - y_s} \tag{36a}$$

$$\tan\left(\theta_{1c}+\theta_{1k}\right)=\frac{x_k-x_{1c}}{y_{1c}-y_k} \tag{36b}$$

$$\tan\left(\theta_{2c}+\theta_{2k}\right)=\frac{x_{2c}-x_k}{y_{2c}-y_k} \tag{36c}$$

$$\tan\left(\theta_{2c}+\theta_{2p}\right)=\frac{x_{2c}-x_p}{y_{2c}-y_p} \tag{36d}$$

The location of the robot center position and orientation in terms of the active target inertial and body-fixed coordinates is defined in the following expressions.

$$\begin{Bmatrix} x_s \\ y_s \end{Bmatrix}=\begin{Bmatrix} x_{cg} \\ y_{cg} \end{Bmatrix}+\left[C\right]^T\begin{Bmatrix} x_{sb} \\ y_{sb} \end{Bmatrix} \tag{37a}$$

$$\begin{Bmatrix} x_k \\ y_k \end{Bmatrix}=\begin{Bmatrix} x_{cg} \\ y_{cg} \end{Bmatrix}+\left[C\right]^T\begin{Bmatrix} x_{kb} \\ y_{kb} \end{Bmatrix} \tag{37b}$$

$$\begin{Bmatrix} x_p \\ y_p \end{Bmatrix}=\begin{Bmatrix} x_{cg} \\ y_{cg} \end{Bmatrix}+\left[C\right]^T\begin{Bmatrix} x_{pb} \\ y_{pb} \end{Bmatrix} \tag{37c}$$

Where $\left[C\right]$ is the coordinate transformation established in Eq. (1).

Substituting Eq. (37) into Eq. (36)

$$\tan\left(\theta_{1c}+\theta_{1s}\right)=\frac{x_{cg}+x_{sb}\cos(\psi)-y_{sb}\sin(\psi)-x_{1c}}{y_{1c}-y_{cg}-x_{sb}\sin(\psi)-y_{sb}\cos(\psi)} \tag{38a}$$

$$\tan\left(\theta_{1c}+\theta_{1k}\right)=\frac{x_{cg}+x_{kb}\cos(\psi)-y_{kb}\sin(\psi)-x_{1c}}{y_{1c}-y_{cg}-x_{kb}\sin(\psi)-y_{kb}\cos(\psi)} \tag{38b}$$

$$\tan\left(\theta_{2c}+\theta_{2k}\right)=-\frac{x_{cg}+x_{kb}\cos(\psi)-y_{kb}\sin(\psi)-x_{2c}}{y_{2c}-y_{cg}-x_{kb}\sin(\psi)-y_{kb}\cos(\psi)} \tag{38c}$$

$$\tan\left(\theta_{2c}+\theta_{2p}\right)=-\frac{x_{cg}+x_{pb}\cos(\psi)-y_{pb}\sin(\psi)-x_{2c}}{y_{2c}-y_{cg}-x_{pb}\sin(\psi)-y_{pb}\cos(\psi)} \tag{38d}$$

Using Eq. (38) a nonlinear least squares differential correction similar to that discussed previously, where the residual *f* is defined as

$$f=\left\{\begin{array}{c}\tan\left(\theta_{1c}+\theta_{1s}\right)-\dfrac{x_{cg}+x_{sb}\cos(\psi)-y_{sb}\sin(\psi)-x_{1c}}{y_{1c}-y_{cg}-x_{sb}\sin(\psi)-y_{sb}\cos(\psi)}\\ \tan\left(\theta_{1c}+\theta_{1k}\right)-\dfrac{x_{cg}+x_{kb}\cos(\psi)-y_{kb}\sin(\psi)-x_{1c}}{y_{1c}-y_{cg}-x_{kb}\sin(\psi)-y_{kb}\cos(\psi)}\\ \tan\left(\theta_{2c}+\theta_{2k}\right)+\dfrac{x_{cg}+x_{kb}\cos(\psi)-y_{kb}\sin(\psi)-x_{2c}}{y_{2c}-y_{cg}-x_{kb}\sin(\psi)-y_{kb}\cos(\psi)}\\ \tan\left(\theta_{2c}+\theta_{2p}\right)+\dfrac{x_{cg}+x_{pb}\cos(\psi)-y_{pb}\sin(\psi)-x_{2c}}{y_{2c}-y_{cg}-x_{pb}\sin(\psi)-y_{pb}\cos(\psi)}\end{array}\right\} \tag{39}$$

The robot coordinates may be found given the incident angle measurements and an initial guess. An attempt was made to invert Eq. (38) and solve for the robot coordinates directly, but the resulting equations were extremely cumbersome and computationally expensive. It was also determined that due to the triangulation's sensitivity to small errors in the incident angles, an approximate solution that would iterate in search of a best compromise between conflicting errors would be better than an exact solution that was highly sensitive to those errors.

Fig. 29 shows that a least squares algorithm will converge to the actual robot coordinates in less than 10 iterations. Since Eq. (36) and Eq. (37) contain 10 equations and 9 unknowns, the system of equations is overspecified. This will allow the algorithm to search for the "best" possible solution, which should make it robust in the presence of errors in the angle readings. Of course it is not necessary to limit the algorithm to only four angles. If more angles are available, then they may also be included.

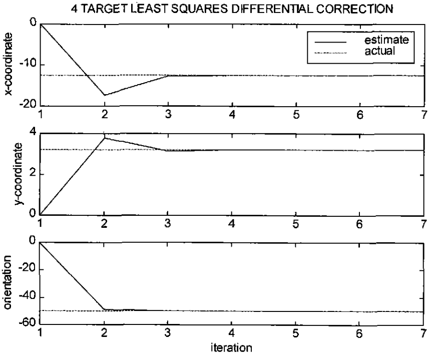

Fig. 29 Least squares differential correction for the four target solution.

Active Target Design

There were two design goals that had to be met for the active targets. A bright light source was necessary because signal to noise ratio had to be maximized. The signal also had to be evenly distributed over a view angle wide enough that both sensors would receive the signal. Fig. 30 - Fig. 33 show the relationship between the signal strengths versus the viewing angle for four types of LED's. The current through each LED was maintained at a constant and the output from the sensor was measured. The "IR LED" is a basic 5mm round infrared LED with a moderately wide view angle. The Hammamatsu 2168 is a small wide angle red LED that is normally used for transmitting signals. The Hammamatsu 3882 has four very small infrared LED's that provide a very wide view angle and a stronger signal than the 2168. The T4 round is a very bright, highly directional LED which is mounted in a standard 10mm diameter package. A comparison of the four LEDs is summarized in Table 3. The T4 round LED's were selected because of their brightness (3000 millicandles max.) and relatively low cost. It was deemed that the highly directional quality of this particular LED could be compensated for with the addition of a device that would disperse the light.

Table 3 Comparison of various light emitting diodes

LED Type	Signal Strength (mV)	View Arc (deg.)	Price
T4 Round	361.65	15	$.73 ea.
IR Round	93.02	5	$1.09 ea.
Hammamatsu 2168	3.856	60	$24.09 ea.
Hammamatsu 3882	11.276	60	$48 ea.

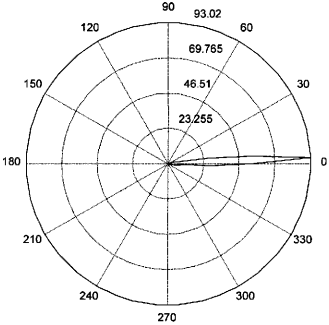

Fig. 30 IR LED view angle.

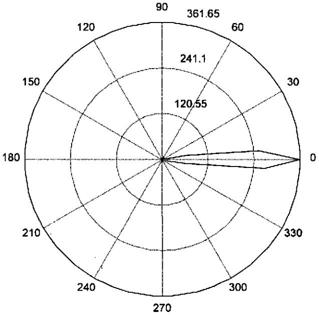

Fig. 31 T4 Round view angle.

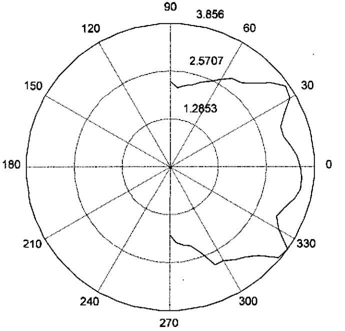

Fig. 32 Hammamatsu 2168 view angle.

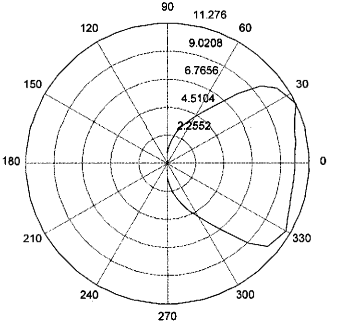

Fig. 33 Hammamatsu 3882 view angle.

The LED target itself is a 3.25 in. diameter cylinder with fourteen T4 round bright LED's distributed around 270 degrees of its perimeter (see Fig. 34). The size of the cylinder was selected so that the circuit board for each target could be mounted inside. A belt of 3 millimeter diameter glass rods was placed around the outside of the active target so that the light emitted by the LED's would be evenly dispersed in a plane coincident with that of the PSD sensors.

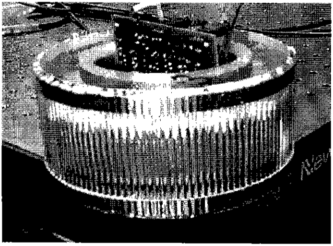

Fig. 34 Active LED Target.

The desired output from the active targets would be a light distribution similar to the Hammamatsu 3882 but with greater intensity. This even distribution is desired so that errors due to assymetry in the target light distribution are minimized. Fig. 35 shows the distribution from one of the four targets. The light distribution varies considerably over the view angle. This occurs despite the use of a lens placed around the target to spread out the signal. Experiments showed that this assymetry in light distribution caused up to a 5 in. error in target location when using the triangulation algorithm. The targets could be significantly improved by making them smaller and selecting LED's with a wider light distribution.

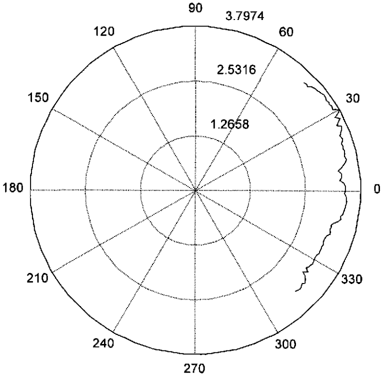

Fig. 35 Light distribution from active target.

By comparing the commanded output to the LED targets at various ranges, a general relationship between the sensor output and the distance from target to sensor was found and is illustrated in Fig. 36. The "uncontrolled" output is the amplitude that the sensor would have if the target was placed at full power and was unregulated. The PSD sensor will saturate at a distance of approximately 13.4 ft. or less if the target intensity is not regulated. The signal will attenuate to below 7 volts at a range of over 17.6 ft.. even if the commanded amplitude is at maximum. At the greatest possible range of 24 ft. (the length of the air-bearing floor), the sensor output will be 3.77 volts. Since the signal to noise ratio at a sampling rate of 50 Hz. is 9727:1, at the increased distance of 24 ft. the signal to noise ratio will be 3670:1. This will still provide the necessary accuracy of 0.1 degrees.

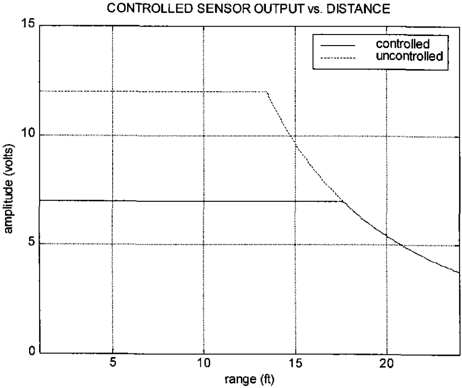

Fig. 36 PSD signal to noise ratio.

Experimental Verification

In order to determine the effectiveness of the PSD sensor system for robot navigation, it is necessary to test this system in a controlled environment where the accuracy of this system may be verified. Using an optical table, a Contraves air bearing, and a computer equipped with a data acquisition board, such an experiment was conducted.

The active targets were set up on a small optical table mounted on the Contraves air bearing. The optical table acted as the body-fixed coordinate frame and it provided a convenient way to get the exact locations of the targets within that frame. The two PSD sensors were set up on the larger optical table and they were oriented so that they could observe the Contraves air bearing. A photo of the experimental setup is in Fig. 37.

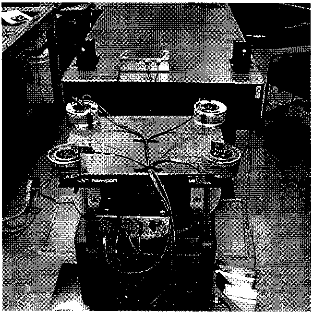

Fig. 37 Photo of experimental apparatus.

Prior to testing, an inertial coordinate frame was established in the manner described on page 36. A total of eighteen points were surveyed and the results were used in the aforementioned algorithm to compute the exact locations of the two PSD sensors in the inertial frame of reference. The survey data and the results are tabulated in Table 4 and Table 5 respectively.

As a preliminary step to determine how well the sensors tracked an individual target, the first set of tests were conducted with one target mounted at the corner of the optical table mounted to the air bearing. The system would then have to track a target moving in a circular trajectory. The air bearing could be commanded to rotate at a fixed rate, so the performance of the system in tracking targets at several different speeds could be evaluated.

Table 4 Inertial frame setup data

Angle θ_1	Angle θ_1	Angle θ_2	Angle θ_2	Target Coordinates	
(degrees)	(radians)	(degrees)	(radians)	X (in.)	Y(in.)
6.1	0.106465084	25.7	0.448549618	0	0
9.5	0.165806279	22.8	0.397935069	4	0
12.9	0.225147474	19.8	0.345575192	8	0
16.1	0.28099801	16.65	0.29059732	12	0
19.3	0.336848546	13.3	0.232128791	16	0
22.2	0.387463094	9.9	0.172787596	20	0
7.1	0.123918377	28.85	0.503527489	0	8
11	0.191986218	25.65	0.447676953	4	8
14.75	0.257436065	22.4	0.390953752	8	8
18.4	0.321140582	18.9	0.329867229	12	8
21.8	0.380481777	15.2	0.265290046	16	8
25.15	0.438950307	11.3	0.197222205	20	8
8.3	0.144862328	32.05	0.559378025	0	15
12.6	0.219911486	28.8	0.502654825	4	15
16.85	0.294087979	25.2	0.439822972	8	15
20.85	0.363901149	21.35	0.372627795	12	15
24.65	0.430223661	17.3	0.301941961	16	15
28.3	0.493928178	13	0.226892803	20	15

Table 5 PSD coordinates

SENSOR	PSD1	PSD2
Orientation (radians)	.0233	.0107
X-Position (inches)	-8.4967	32.1040
Y-Position (inches)	65.2052	64.8940

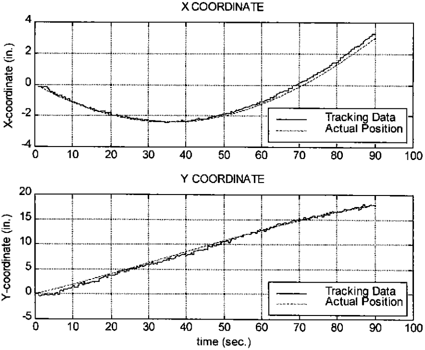

Fig. 38 Tracking one target at a sampling rate of 50 Hz.

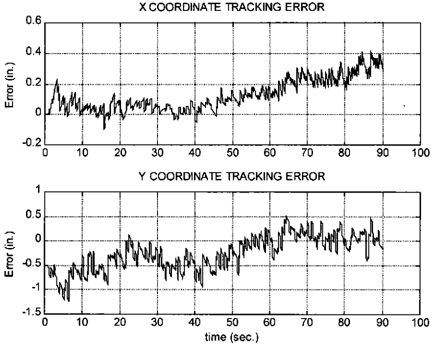

Fig. 39 Tracking error for a single target at a sampling rate of 50 Hz.

Fig. 38 and Fig. 39 show the tracking data and tracking error for one of the tests. In all cases the x position was tracked with greater accuracy than the y position. This was due to the location of the target with respect to the position of the PSD sensors. Fig. 40 shows how an error in the measured incident angle effected the triangulation on a single target. The "error diamond" elongated in the $\hat{e}_2$ direction as the target's distance from the sensors increased in that same direction. Conversely, as that distance decreased, the error diamond compressed and triangulation became more sensitive in the $\hat{e}_1$ direction. Since in this experiment the target was always farther away than the optimal 45 degree center, the error diamond due to bias and noise in the angle readings effected the y position reading more than the x position reading. In spite of these errors, the system was still tracking the location of the target to within an inch of the actual position and in most cases to within .5 inches.

The next step in the evaluation process was to use two targets and determine how well the system could track orientation in addition to position. In this case the body-fixed coordinate reference frame was again established at the corner of the optical table to simulate tracking the robot as it translates and rotates across the floor.

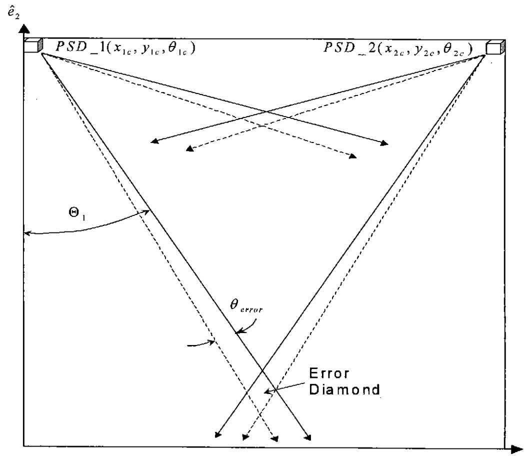
$\hat{e}_2$
$PSD\_1(x_{1c}, y_{1c}, \theta_{1c})$
$PSD\_2(x_{2c}, y_{2c}, \theta_{2c})$
Θ_1
θ_{error}
Error Diamond

Fig. 40 Triangulation sensitivity due to incident angle error.

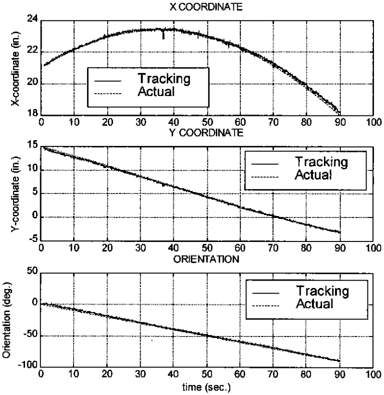

Fig. 41 Tracking two targets at a sampling rate of 50 Hz.

Fig. 41 and Fig. 42 show the tracking data and error respectively for the two-target case. It was more accurate than the single target case because the each of the two targets canceled out the errors caused by the asymmetrical shape of the other. During the course of these experiments, it was discovered that when using a two targets, their relative assymetry merited using both of them to establish the inertial reference coordinate frame. If only one target was used, a large error would result. The spike found in the data for this experiment was due to an interrupt in the processing of the data caused by the operating system and is not a sudden burst of electronic noise.

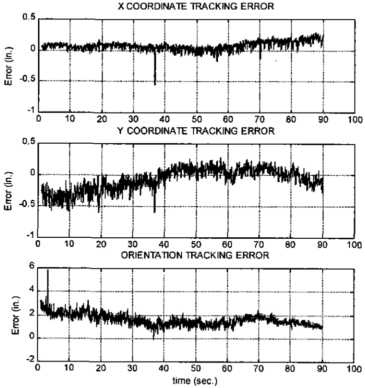

Fig. 42 Tracking error for two targets at a sampling rate of 50 Hz.

Fig. 43 and Fig. 44 show the tracking data and error respectively when four targets were used to recover the robot orientation and position. There was a marginal improvement in error over the two-target case, but this was probably due to the fact that the sampling rate was less and the oversampling was greater in the four-target case. It was also discovered that the least squares estimation was computationally expensive and therefore could not be run at a rate greater than 10 Hz. while maintaining system stability.

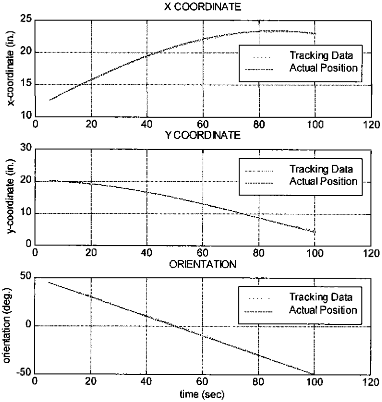

Fig. 43 Tracking four targets at 10 Hz.

Despite these drawbacks however, the four-target solution is more desireable for a number of reasons: 1.) There is more information available and therefore an error in one angle will have less of an effect on the accuracy. 2.) There is less likelihood of a poorly conditioned configuration (such as in the two-target case where one target may be viewed as directly behind another from one of the sensors). 3.) There are existing brackets around the robot where four targets may be mounted. Mounts do not yet exist for a two-target configuration.

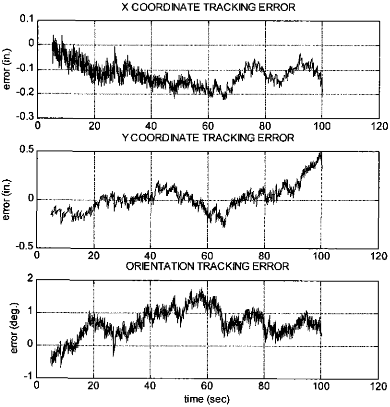

Fig. 44 Tracking error for four targets at 10 Hz.

KALMAN ESTIMATOR DESIGN AND SIMULATION

The PSD sensor is capable of providing position and orientation data in real time to a control algorithm, however the translational and rotational rates are also of interest in this experiment. Another issue is that the model parameters associated with mass, mass moment of inertia and thrust will change with each experiment and with time. The PSD sensors are also susceptible to electronic noise. The extended Kalman filter is an effective means of dealing with all of these problems.

The Extended Kalman Filter Computational Algorithm

The underlying theory behind the extended Kalman filter is well documented in other texts and will not be discussed here.[13] The general algorithm is introduced to make the later discussion about parameter identification more clear. The general discrete time model for a linear dynamical system is described as:

$$x(k+1) = \Phi(k+1,k)x(k) + \Gamma(k+1,k)w(k) \tag{40a}$$

$$z(k) = H(k)x(k) + v(k) \tag{40b}$$

Where the standard assumptions and definitions are

$$E\{x(0)\} = m_x(0); E\{w(k)\} = 0, \quad \mathit{for} \quad \mathit{all} \quad k$$

$$\begin{aligned}
&E\{v(k+1)\} = 0, \quad \mathit{for} \quad \mathit{all} \quad k \\
&\mathrm{cov}\{x(0), x(0)\} \equiv P_x(0) \\
&\mathrm{cov}\{w(k), w(j)\} \equiv Q(k)\delta_{jk} \\
&\mathrm{cov}\{v(k), v(j)\} \equiv R(k)\delta_{jk} \\
&\mathrm{cov}\{x(0), w(k)\} \equiv 0, \quad \mathit{for} \quad \mathit{all} \quad k \\
&\mathrm{cov}\{x(0), v(j)\} = 0, \quad \mathit{for} \quad \mathit{all} \quad j \\
&\mathrm{cov}\{w(k), v(j)\} = 0, \quad \mathit{for} \quad \mathit{all} \quad j,k
\end{aligned} \tag{41}$$

Where *w(k)* and *v(k+1)* are zero mean, white noise sequences uncorrelated with each other and with the initial condition random vector x(0). *w(k)* is often referred to in the literature as "process noise" and *v(k+1)* as "measurement noise". δ_{jk} is the Kronecker delta function.

The more general nonlinear version of the dynamics described in Eq. (40) may be written as

$$\dot{x}(t) = f(x(t),t) + G(x(t),t)w(t) \tag{42a}$$

$$z(t_i) = h[x(t_i),t_j] + v(t_i) \quad for \quad i = 0,1,.... \tag{42b}$$

Where *f(x(t),t)* is an n-vector and *G(x(t),t)* is an n x r-matrix of nonlinear functions in *x(t)* and *t*. $z(t_i)$ is an m-dimensional measurement vector and $h[x(t_i),t_j]$ is an m-dimensional vector of nonlinear functions in $x(t_i)$ and $t_{i.}$ By analogy with the linear model written above, the following relationships are assumed

$$E\{x(t_0)\} = m_0(0); E\{w(t)\} = 0, \quad for \quad all \quad t > t_0$$

$$\begin{aligned} &\operatorname{cov}\{w(k),w(j)\} \equiv Q(t)\delta(t-\tau) \\ &\operatorname{cov}\{v(t_i),v(t_j)\} \equiv R(i)\delta_{ij} \\ &\operatorname{cov}\{x(t_0),w(t)\} \equiv 0, \quad for \quad all \quad t > t_0 \\ &\operatorname{cov}\{x(t_0),v(t_i)\} = 0, \quad for \quad all \quad t > t_0 \\ &\operatorname{cov}\{w(t),v(t_i)\} = 0, \quad for \quad all \quad t > t_0 \end{aligned} \tag{43}$$

Using the above equations and assumptions, the extended Kalman filter problem is as follows: Given a measurement sequence $Z(k) \equiv \{z(t_0), z(t_1),..., z(t_k)\}$, find an estimator to provide estimates of *x(t)*, denoted by $\hat{x}(t / t_k)$.

The computational sequence to solve this problem is:

1. Extrapolate the state estimate to time t_1. The equation

$$\dot{\hat{x}}(t/t_0) = f(\hat{x}(t/t_0), t) \tag{44}$$

is integrated on the interval [t_0, t_1] with initial conditions $\hat{x}(t_0/t_0)$. The result at the end of the integration is $\hat{x}(t_1/t_0)$.

2. Extrapolate the covariance matrix to time t_1. The equation is

$$P(t_{k+1}/t_k) = \Phi(t_{k+1}, t_k)P(t_k/t_k)\Phi^T(t_{k+1}, t_k) + Q(t_k) \tag{45}$$

to use this equation, the state transition matrix $\Phi(t_{k+1}, t_k)$ and the process noise matrix *Q(t_0)* must be computed.

The state transition matrix is formed using the Jacobian of the equations of motion

$$F(t) = \left.\frac{\partial f}{\partial x}\right|_{x=\hat{x}(t,t_0)} \tag{46}$$

where $\hat{x}(t_0/t_0)$ is a result from step 1. The matrix differential equation

$$\frac{\partial\Phi(t,t_0)}{\partial t} = F(t)\Phi(t,t_0) \tag{47}$$

is integrated on the interval [t_o, t_1] with initial conditions $\Phi(t_0,t_0) = I$. The result is $\Phi(t_1,t_0)$ which is then plugged into equation (41)

3. The state estimate at time t_1 must be updated to account for the measurement at time t_1. The update equation is

$$\hat{x}(t_1/t_1) = \hat{x}(t_1/t_0) + K(t_1)\tilde{z}(t_1/t_0) \tag{48}$$

where

$$K(t_1) = P(t_1/t_0)H^T(t_1)[H(t_1)P(t_1/t_0)H^T(t_1) + R(t_1)]^{-1} \tag{49}$$

and the measurement residual is

$$\tilde{z}(t_1/t_0) = z(t_1) - h(\hat{x}(t_1/t_0), t_1) \tag{50}$$

$H(t_1)$ is computed from

$$H(t_1) = \left.\frac{\partial h}{\partial x}\right|_{x=\hat{x}(t_1/t_0)} \tag{51}$$

where $\hat{x}(t_1/t_0)$ is available from step 1. Using $P(t_1/t_0)$ from step 2, all required quantities are now collected to compute the Kalman gain *$K(t_1)$*. The residual is computed Eq. (50), where the new measurement *$z(t_1)$* is assumed to be available. The result of this state update is $\hat{x}(t_1/t_1)$.

4. The covariance matrix at time t_1 is updated using the equation

$$P(t_1/t_1) = [I - K(t_1)H(t_1)]P(t_1/t_0) \tag{52}$$

All required quantities are available from the steps listed above and the result is the updated covariance matrix $P(t_1/t_1)$.

The above sequence establishes the computational framework from which the algorithm that provides parameter and state estimation for the robot is derived.[14]

Derivation of the Kalman Estimator for the Robot Case

The Kalman filter is an effective means of estimating an m-dimensional state vector x when not all of the states are available for measurement and when noise is present in the system. Since there are additional parameters of interest that are unknown and may change with time, it is logical to augment the state equations with additional states that represent these unknown parameters. If the system is still observable, then it is possible for an extended Kalman filter to estimate these parameters just as if they were an unknown position or rate.

Since the damping associated with the robot dynamics was shown to be a small random force when compared with the magnitude of the control thrust, it is neglected in the modified equations of motion below. Any disturbance associated with random contact between the air bearings and the floor surface can be accounted for as model uncertainty. An appropriate Q matrix in the Kalman filter will allow it to deal with any such disturbance. Dropping the damping terms from Eq. (8) and substituting the unknown parameters $b_{2i} = \frac{V_i}{m}$ and $b_{2i-1} = \frac{r_i V_i}{I_n}$ the equations of motion reduce to

$$\begin{Bmatrix} \ddot{\psi} \\ \ddot{x} \\ \ddot{y} \end{Bmatrix} = \begin{bmatrix} b_1 & -b_3 & b_5 & -b_7 & b_9 & -b_{11} & b_{13} & -b_{15} \\ -b_2 \cos\psi & b_4 \sin\psi & b_6 \sin\psi & b_8 \cos\psi & b_{10} \cos\psi & -b_{12} \sin\psi & -b_{14} \sin\psi & -b_{16} \cos\psi \\ -b_2 \sin\psi & -b_4 \cos\psi & -b_7 \cos\psi & b_8 \sin\psi & b_{10} \sin\psi & b_{12} \cos\psi & b_{14} \cos\psi & -b_{16} \sin\psi \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \\ u_8 \end{Bmatrix} \quad (53)$$

If the unknown parameters b_{2i} and the control inputs are each assigned a state, then the nonlinear differential equations become

$$\begin{Bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \\ \dot{x}_7 \\ \vdots \\ \dot{x}_{30} \end{Bmatrix} = \begin{Bmatrix} x_2 \\ -x_7x_9 + x_{10}x_{12} - x_{13}x_{15} + x_{16}x_{18} - x_{19}x_{21} + x_{22}x_{24} - x_{25}x_{27} + x_{28}x_{30} \\ x_4 \\ -x_8x_9\sin x_1 + x_{11}x_{12}\cos x_1 - x_{14}x_{15}\cos x_1 - x_{17}x_{18}\sin x_1 + x_{20}x_{21}\sin x_1 - x_{23}x_{24}\cos x_1 + x_{26}x_{27}\sin x_1 + x_{29}x_{30}\cos x_1 \\ x_6 \\ x_8x_9\sin x_1 + x_{11}x_{12}\cos x_1 - x_{14}x_{15}\cos x_1 - x_{17}x_{18}\sin x_1 + x_{20}x_{21}\sin x_1 - x_{23}x_{24}\cos x_1 + x_{26}x_{27}\sin x_1 + x_{29}x_{30}\cos x_1 \\ 0 \\ \vdots \\ 0 \end{Bmatrix} \quad (54)$$

where,

$x_1 = \psi$

$x_2 = \dot{\psi}$

$x_3 = x$

$x_4 = \dot{x}$

$x_5 = y$

$x_6 = \dot{y}$

$x_7 = b_1$

$x_8 = b_2$

$x_9 = u_1$

$x_{10} = b_3$

$x_{11} = b_4$

$x_{12} = u_2$

$x_{13} = b_5$

$x_{14} = b_6$

$x_{15} = u_3$

$x_{16} = b_7$

$x_{17} = b_8$

$x_{18} = u_4$

$x_{19} = b_9$

$x_{20} = b_{10}$

$x_{21} = u_5$

$x_{22} = b_{11}$

$x_{23} = b_{12}$

$x_{24} = u_6$

$x_{25} = b_{13}$

$x_{26} = b_{14}$

$x_{27} = u_7$

$x_{28} = b_{15}$

$x_{29} = b_{16}$

$x_{30} = u_8$

Numerical Simulation and Results

To verify that the extended Kalman filter would be capable of state estimation for this problem, it was simulated using MATLAB. The parameters used to simulate the dynamic model of the system were taken directly from the data found in the previous experiments listed in Table 8 - Table 15. It was assumed that the robot was operating at a manifold pressure of 200 psi. The parameters used in the simulation are in Table 6.

Table 6 Robot model parameters

Thruster	Parameter b_{2i}	Parameter b_{2i-1}
V1	.0317	.0329
V2	.0337	.0342
V3	.0293	.0331
V4	.0294	.0305
V5	.0365	.0403
V6	.0310	.0351
V7	.0329	.0334
V8	.0373	.0412

Since the intent of the extended Kalman filter is to identify the system parameters during “normal” operations, the experimental identification methodology previously discussed using least squares is not applicable here. It is not possible to disconnect all the thrusters except for the one of interest and measure the system’s response. The simulation will instead put the robot through a series of maneuvers using all the thrusters just as it would be during operations not specifically dedicated to system identification. The "robot waltz" manuever sequence is listed in **Table 7**.

Table 7 Robot maneuver sequence

Manuever	Elapsed Time
Rotate Clockwise	3 seconds
Rotate Counter-Clockwise	3 seconds
Move Forward	3 seconds
Move Backward	3 seconds
Move Left	3 seconds
Move Right	3 seconds

Eq. (54) is integrated forward in time using the Runge-Kutta method with a time step of .01 seconds. The measurable values of $\{x_{1,}x_2,x_3,x_5\}$ are fed to the Kalman filter at every tenth time step to simulate a sampling rate of 10 Hz. The state estimates associated with position and rate are illustrated in Fig. 45. The residual is illustrated in Fig. 46. The estimates associated with the unknown parameters are illustrated in Fig. 47 - Fig. 54.

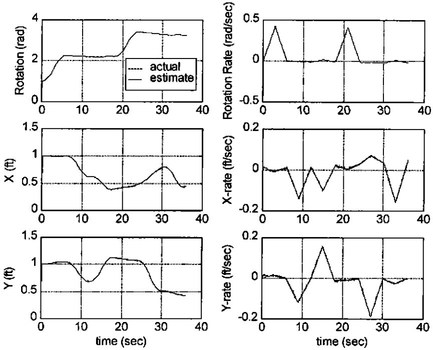

Fig. 45 Robot motion as interpreted by the Kalman filter.

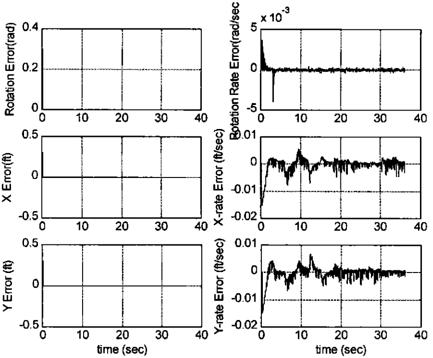

Fig. 46 State estimate error.

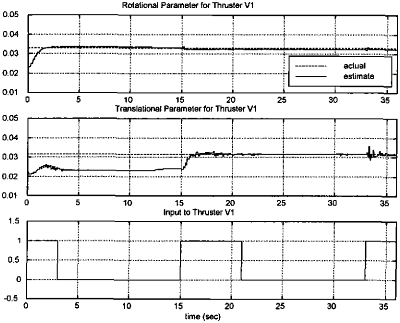

Fig. 47 Identification of V1 thruster parameters.

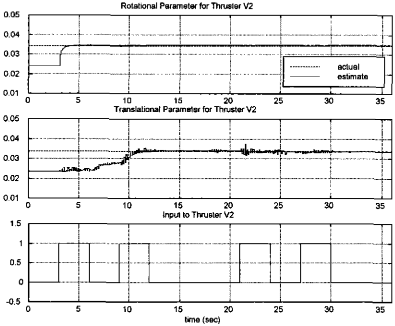

Fig. 48 Identification of V2 thruster parameters.

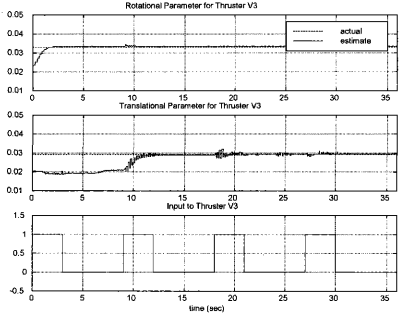

Fig. 49 Identification of V3 thruster parameters.

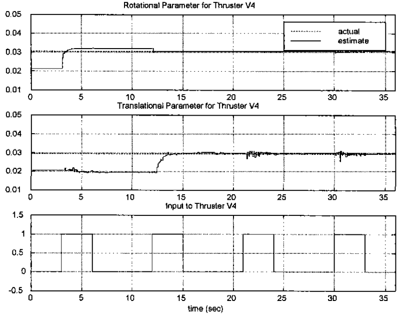

Fig. 50 Identification of V4 thruster parameters.

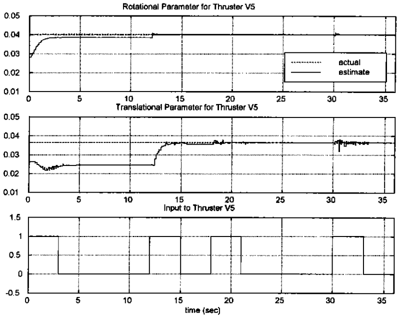

Fig. 51 Identification of V5 thruster parameters.

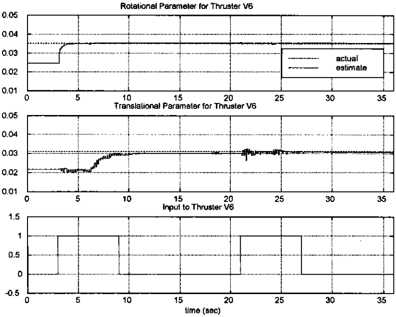

Fig. 52 Identification of V6 thruster parameters.

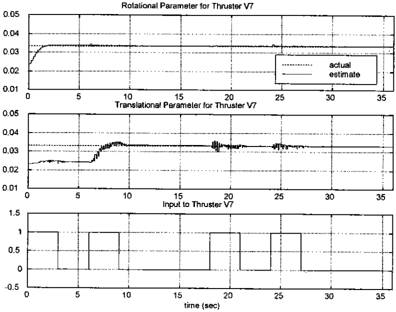

Fig. 53 Identification of V7 thruster parameters.

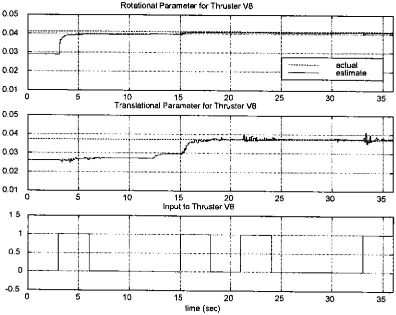

Fig. 54 Identification of V8 thruster parameters.

The Kalman filter estimates corresponding to the unknown parameters converge to the correct values after 18 seconds of the "robot waltz" and remain at or near these values for the remainder of the time that the robot is maneuvering. In Fig. 45 it can be observed that the estimated dynamic states closely match the simulated values throughout the maneuver despite a 30% error introduced on the initial conditions.

A random error in the measured values was introduced in the form white Gaussian noise to simulate the electronic interference observed in the calibration of the PSD sensors. This random error did cause some oscillation in the identification process, but the oscillations were about the desired values. In all but two of the thrusters this oscillation was significantly attenuated by the time the maneuver had ended. It is interesting to note that this oscillation was largest when a thruster initially went from an "off" state to an "on" state. This indicates that the Kalman filter was momentarily "surprised" by the sudden change in one of the parameters, but was able to recover sufficiently to give accurate rate and position readings as well as to bring the unknown parameter back to its correct value.

In cases run where no noise was present the Kalman filter performed even better than in the situation illustrated in Fig. 47 - Fig. 54. The deviation from nominal found in the state estimation also occurred in the noise free case, but the filter was able to recover and cause the estimates to converge to the correct value. An example of the noise free case may be found in Fig. 55.

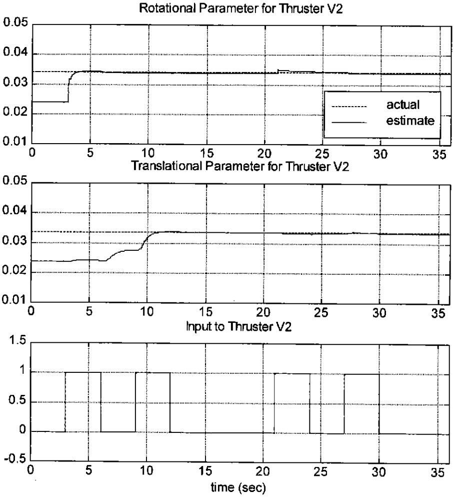

Fig. 55 Noise free identification of thruster V2.

CONCLUSIONS AND RECOMMENDATIONS

A means of navigating a robot using a remote sensing system is developed for the purposes of system identification and control. Using a nonlinear least squares differential correction, the robot orientation and position may be recovered from measured incident angle data. The effect of errors in the incident angle data on the triangulation algorithm was reduced as more targets became visible. More numerous targets also meant more computational load on the system and reduced the sampling rate to 10 Hz. A real-time operating system or a faster platform is necessary to increase the sampling rate to 50 Hz. or better.

An extended Kalman filter was developed to estimate the rotational and translational rates as well as unknown parameters in the system. Data collected using the PSD sensors may be used with the Kalman filter to conduct system identification. It was discovered that the unknown parameters associated with rotation were easy for the algorithm to identify, but translational parameters took longer to converge. The estimates were also sensitive to sudden changes in the system brought on by thruster switching, but the estimates would still reconverge to the true values. Damping will be a small but significant force in the real system, and since it always acts in opposition to the direction of motion, it cannot be considered to be a white noise random process. When the Kalman filter is applied to the robot, an additional unknown parameter associated with the damping coefficient should be introduced.

Future work should include a redesign of the active targets to make them smaller and more uniform so that partial occlusion of the targets will cause less error. Replacing the active targets with retro reflective patches illuminated by a scanning laser should also be investigated to determine if any advantage can be gained using that method. Further investigations into applications for a 3-dimensional navigation system should also yield interesting results, particularly for spacecraft applications.

REFERENCES

[1] Ho, C.J. and McClamroch, N.H., "Automatic Spacecraft Docking Using Computer Vision-Based Guidance and Control Techniques," *Journal of Guidance, Control, and Dynamics,* Vol. 16, No. 2, 1993, pp. 281-288.

[2] Schaub, H., "Real-Time Stereo Vision Using Laser Scanning and Position Sensitive Photodetectors: Analytical and Experimental Results," MS Thesis, Texas A&M University, College Station, TX, 1994.

[3] Junkins, J.L., Gadhok, J., Belur, R., and Kealey, L., "A Novel G&C Technology Transfer: Automated Notetaking in the Classroom of the Future." *17th Annual AAS Guidance and Control Conference*, Keystone, CO, 1994.

[4] Ljung, L. *System Identification: Theory for the User,* Prentice Hall, Inc., Englewood Cliffs, NJ, 1987.

[5] Junkins, J.L. *An Introduction to Optimal Estimation of Dynamical Systems,* Sijthoff & Noordhoff International Publishers, Alphen aan den Rijn, Netherlands, 1978

[6] Pollock, T.C., Vadali, S.R., Junkins, J.L., Benson, W.W., and Kline, E.M., "Autonomous Control Strategies for Multibody Spacecraft," Interim Report to Lockheed Martin Co. Texas A&M University, College Station, TX, 1997.

[7] *DAQpad-MIO-16XE User Manual*, National Instruments Corporation, Austin, TX 1996.

[8] Morgan, S., *Aero 305 Lab Manual,* TEES Copy Center, College Station, TX, 1988, pp. 115-137.

[9] *SVS-100 User's Manual*, DMK Engineering, Rancho Palos Verdes, CA 1995.

[10] Hughes, D., Personal communication, Texas A&M University, College Station, TX, 1998.

[11] Press, W.H., Teukolsky, S.A., Vetterling, W.T., and Flannery, B.P., *Numerical Recipes in C,* Cambridge University Press, New York, 1992.

[12] Junkins, J.L., Personal communication, Texas A&M University, College Station, TX, 1998.

[13] LeMay, J.L. and Brogan W.L., *Kalman Filtering at UCSD Extension, December 8-12, 1986,* St. Joseph Sciences, San Diego, CA, 1986.

[14]Ward, D.T., Nonlinear Dynamics class notes, Department of Aerospace Engineering, Texas A&M University, Personal Collection, W. Benson, 1997.

[15]Moreland, G., Personal communication, Johnson Space Center, NASA, Clear Lake, TX, 1997.

APPENDIX A

Table 8 ARX parameters and resulting thrust for thruster V1

Manifold Pressure	A_1 (data set 1)	B_1 (data set 1)	η (data set 1)	*V3* (data set 1)	A_1 (data set 2)	B_1 (data set 2)	η (data set2)	*V3* (data set2)
(psi.)			Lb.-in.-sec/rad	lbs.			lb.-in.-sec/rad	lbs.
100	0.9914	0.0027	-26.3935	0.3654	0.9868	0.0027	-40.8525	0.3679
120	0.9964	0.0034	-11.0164	0.4618	0.9968	0.0034	-9.8115	0.4694
140	0.9987	0.0042	-3.7869	0.5633	0.9989	0.0041	-3.4426	0.5506
160	0.9995	0.0049	-1.6639	0.6521	0.9979	0.0048	-6.4262	0.6395
180	0.9992	0.0054	-2.4098	0.7232	0.9968	0.0053	-5.2213	0.7181
200	0.9984	0.0062	-4.8197	0.8323	0.9991	0.0062	-4.3033	0.8374

Table 9 ARX parameters and resulting thrust for thruster V2

Manifold Pressure	A_1 (data set 1)	B_1 (data set 1)	η (data set 1)	*V3* (data set 1)	A_1 (data set 2)	B_1 (data set 2)	η (data set2)	*V3* (data set2)
(psi)			Lb.-in.-sec/rad	lbs.			lb.-in.-sec/rad	lbs.
100	0.9837	0.0029	-49.9181	0.3975	0.9925	0.0028	-23.0656	0.3897
120	0.9923	0.0035	-23.4672	0.4806	0.9951	0.0036	-14.8033	0.4962
140	0.9955	0.0042	-13.7131	0.5845	0.9955	0.0042	-13.7705	0.5767
160	0.9959	0.0048	-12.6230	0.6573	0.9965	0.0048	-10.6148	0.6599
180	0.9962	0.0056	-11.4754	0.7664	0.9965	0.0056	-10.6148	0.7742
200	0.9973	0.0062	-8.2623	0.8495	0.9972	0.0064	-8.3771	0.8885

Table 10 ARX parameters and resulting thrust for thruster V3

Manifold Pressure	A_1 (data set 1)	B_1 (data set 1)	η (data set 1)	*V3* (data set 1)	A_1 (data set 2)	B_1 (data set 2)	η (data set2)	*V3* (data set2)
(psi)			Lb.-in.-sec/rad	lbs.			lb.-in.-sec/rad	lbs.
100	0.9961	0.0028	-11.9344	0.3454	0.9967	0.0027	-10.0984	0.3338
120	0.9970	0.0034	-9.3525	0.4271	0.9969	0.0033	-9.6393	0.4178
140	0.9938	0.0041	-19.1639	0.5205	0.9968	0.0041	-9.7541	0.5182
160	0.9942	0.0048	-17.8443	0.6045	0.9980	0.0047	-6.1967	0.5952
180	0.9984	0.0056	-4.9918	0.7025	0.9987	0.0055	-3.9016	0.6909
200	0.9994	0.0062	-1.9508	0.7726	0.9991	0.0062	-2.6967	0.7726

Table 11 ARX parameters and resulting thrust for thruster V4

Manifold Pressure	A_1 (data set 1)	B_1 (data set 1)	η (data set 1)	*V3* (data set 1)	A_1 (data set 2)	B_1 (data set 2)	η (data set2)	*V3* (data set2)
(psi)			Lb.-in.-sec/rad	lbs.			lb.-in.-sec/rad	lbs.
100	0.9925	0.0023	-23.0656	0.3121	0.9893	0.0024	-33.1640	0.3299
120	0.9900	0.0032	-30.8689	0.4365	0.9952	0.0031	-14.7459	0.4263
140	0.9936	0.0037	-19.8525	0.5050	0.0024	0.0037	-23.3525	0.5075
160	0.9945	0.0044	-16.9836	0.6065	0.9936	0.0044	-19.6803	0.6065
180	0.9966	0.0051	-10.3853	0.6877	0.9965	0.0052	-10.7295	0.7080
200	0.9975	0.0057	-7.5738	0.7816	0.9977	0.0057	-7.0000	0.7765

Table 12 ARX parameters and resulting thrust for thruster V5

Manifold Pressure	A_1 (data set 1)	B_1 (data set 1)	η (data set 1)	*V3* (data set 1)	A_1 (data set 2)	B_1 (data set 2)	η (data set2)	*V3* (data set2)
(psi)			Lb.-in.-sec/rad	lbs.			lb.-in.-sec/rad	lbs.
100	0.9981	0.0035	-5.9672	0.4436	0.9984	0.0035	-4.8197	0.4484
120	0.9984	0.0042	-4.8771	0.5295	0.9985	0.0042	-4.6475	0.5319
140	0.9991	0.0051	-2.8115	0.6463	0.9993	0.0049	-2.1803	0.6296
160	0.9992	0.0060	-2.4098	0.7656	0.9989	0.0059	-3.4426	0.7560
180	0.9993	0.0066	-2.1230	0.8371	0.9996	0.0065	-1.3770	0.8300
200	0.9997	0.0074	-0.7459	0.9492	0.9993	0.0075	-2.0082	0.9635

Table 13 ARX parameters and resulting thrust for thruster V6

Manifold Pressure	A_1 (data set 1)	B_1 (data set 1)	η (data set 1)	*V3* (data set 1)	A_1 (data set 2)	B_1 (data set 2)	η (data set2)	*V3* (data set2)
(psi)			Lb.-in.-sec/rad	lbs.			lb.-in.-sec/rad	lbs.
100	0.9914	0.0028	-26.2787	0.3547	0.9904	0.0028	-29.4344	0.3477
120	0.9904	0.0036	-29.2623	0.4457	0.9860	0.0035	-42.6885	0.4294
140	0.9911	0.0046	-27.2541	0.5741	0.9908	0.0047	-28.1721	0.5787
160	0.9897	0.0053	-31.6148	0.6581	0.9913	0.0050	-26.5656	0.6301
180	0.9944	0.0059	-17.0984	0.7398	0.9935	0.0059	-19.9672	0.7374
200	0.9936	0.0065	-19.6803	0.8121	0.9962	0.0066	-11.4754	0.8191

Table 14 ARX parameters and resulting thrust for thruster V7

Manifold Pressure	A_1 (data set 1)	B_1 (data set 1)	η (data set 1)	*V3* (data set 1)	A_1 (data set 2)	B_1 (data set 2)	η (data set2)	*V3* (data set2)
(psi)			Lb.-in.-sec/rad	lbs.			lb.-in.-sec/rad	lbs.
100	0.9964	0.0028	11.0738	-0.3897	0.9962	-0.0029	11.8197	0.3975
120	0.9957	0.0033	13.0820	-0.4546	0.9970	-0.0036	9.2377	0.5014
140	0.9975	0.0042	7.5738	-0.5897	0.9960	-0.0043	12.2213	0.5949
160	0.9977	0.0050	7.0000	-0.6884	0.9976	-0.0050	7.4590	0.6962
180	0.9994	0.0058	1.7213	-0.7975	0.9973	-0.0057	8.3197	0.7897
200	0.9982	0.0064	5.3934	-0.8963	0.9994	-0.0062	1.7787	0.8677

Table 15 ARX parameters and resulting thrust for thruster V8

Manifold Pressure	A_1 (data set 1)	B_1 (data set 1)	η (data set 1)	*V3* (data set 1)	A_1 (data set 2)	B_1 (data set 2)	η (data set2)	*V3* (data set2)
(psi)			Lb.-in.-sec/rad	lbs.			lb.-in.-sec/rad	lbs.
100	0.9973	0.0032	-8.0328	0.4156	0.9974	0.0031	-8.0328	0.4082
120	0.9911	0.0038	-27.0820	0.4942	0.9963	0.0040	-11.1312	0.5188
140	0.9973	0.0049	-8.0902	0.6418	0.9972	0.0048	-8.5492	0.6221
160	0.9975	0.0056	-7.6312	0.7303	0.9970	0.0056	-9.2377	0.7352
180	0.9972	0.0066	-8.3771	0.8557	0.9974	0.0065	-7.9754	0.8532
200	0.9975	0.0075	-7.6312	0.9786	0.9966	0.0075	-10.2131	0.9836

APPENDIX B

AIR BEARING FLOOR DESIGN AND CONSTRUCTION

The project was assigned space in a former Texas Instruments factory building now owned by Texas A&M University. The space was partitioned from adjoining areas with a chain link fence eight feet in height. TAMU Physical Plant provided electrical service and low-pressure (125-psi) compressed air service. Within this area was poured an epoxy floor measuring 4.88 m (16 ft) by 7.31 m (24 ft).

In an effort that echoed the prior experience of Lockheed Martin, a number of epoxy products were purchased in gallon quantities. These products were chosen on the basis of recommendations from suppliers located using the Thomas Register and similar sources. The two most important parameters were the elapsed time from mixing to curing ("pot-life") and viscosity. A low viscosity fluid that would not harden for several hours was desired so that it could flow easily and level to within .0005 in. Test pours of each product were made into frames of roughly 4 ft. by 4-ft. size. Results were not promising. From conversations with Garlan Moreland[15] of NASA JSC it was learned that JSC had also poured an ultra-flat floor. Their material of choice was the same as that used at Lockheed Martin. Unfortunately, this product was removed from the market prior to the JSC effort. Koch Corrosion Control Co. received the contract to provide the materials for JSC, and a successful pour was made. JSC offered us their surplus materials (Techni-plus EP 60.250 SL) for testing. Results were promising.

The existing floor in the factory building was surveyed and it was determined that elevations varied up to .91 in. over the proposed air bearing floor (ABF) area.

Calculations were made to determine the quantity of epoxy required to level the existing floor and add an additional .25 in. of elevation to the surface. The total exceeded 150 gallons at $80/gallon of Techni-plus. A more cost-effective means of preparing the floor for epoxy application had to be found.

Further consultation with NASA revealed that a concrete underlayment to improve the grade of the existing floor was necessary to reduce costs and provide a solid base for the ABF. After researching various techniques and materials, it was determined that Ardex would offer the best solution. Ardex is a self-levelling concrete with polymer additives, which can level to within .0125 in. if properly applied. Several preliminary steps were necessary before the Ardex could be installed.

The existing floor was mechanically cleaned using a process known as shot-peening. A shot-peening machine scours a surface by projecting minute lead pellets at high velocity onto the surface and simultaneously vacuuming the expended shot and resulting debris. The shot is separated from the debris and recycled through the machine for a subsequent use. Concrete Cleaning Inc. was contracted and the desired results were obtained. Wooden forms were erected around the area where the Ardex was to be installed.

Jay South, a licensed Ardex dealer, was contracted to install the Ardex underlayment. The Ardex was mixed using a portable mixer and was completely installed in less than 20 minutes. Using wooden dowels as elevation benchmarks, the Ardex was levelled using a squeegee. The Ardex underlayment was allowed to cure and was surveyed to determine the quantity of epoxy required. After curing, the Ardex was found to be flat and level to within .25 in., but had a single peak that was .33 in. above datum. It was determined that a .25 in. tolerance was acceptable and that the single peak could be ground down to the datum. Concrete Cleaning Inc. returned to shot-peen the Ardex and level off the peak.

After the Ardex was shot-peened and levelled, it was resurveyed and calculations were made to determine the necessary quantity of epoxy to level the Ardex and add an additional .125 in. of elevation. Despite its low viscosity and long pot-life, the EP-60 would not flow more than several feet, so a detailed method of application had to be developed. It was found that the best way to apply the epoxy would be to survey the floor into 4 ft. by 4 ft. grid squares and find the necessary volume of epoxy to raise each area to the desired datum. The volume was converted into weights for easy measurement.

The actual application process started with installing metal edging around the ABF perimeter to protect the epoxy and separating the area into discrete grid squares using .75 in. by 3 in. rubber belting as dams. A representative from Koch Corrosion Control was present to supervise the mixing of the epoxy and to lend technical assistance as necessary. The process was broken down into teams of individuals responsible for mixing, measuring and pouring the epoxy into the individual grid squares. After the adjacent grid squares were filled with epoxy the dams were removed to permit the epoxy to self-level and smooth.

The epoxy was permitted to cure for several days before an assessment was made. Several areas of uncured epoxy were observed. After consultation with Koch, it was determined that epoxy was not properly mixed prior to pouring. According to instructions from Koch, three to four minutes of mixing using a .75 in. drill and a mixing paddle were necessary to properly mix all the resin and hardener. It was found that because of the special nature of this product, additional mixing time should be added to ensure a proper reaction. Despite this problem, the ABF was level to the desired tolerance of .0005 in. The only remaining course of action was to remove the uncured areas of epoxy and apply a uniform .01625 inch layer of EP-60 to the existing floor. Koch agreed to sell the necessary epoxy at reduced price. The additional layer was applied using squeegees and the resulting floor hardened to the desired strength and texture.

The telerobot was placed on the ABF and it appeared to move smoothly over the surface. There were several small peaks where it would "hang", but these were easily removed using 200 grit sandpaper. The ABF installation process was deemed a success.

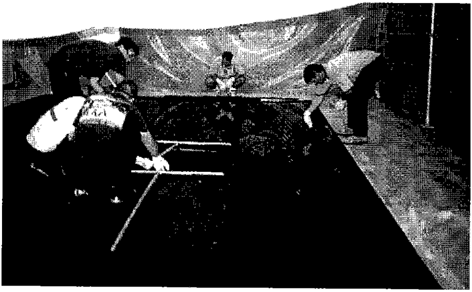

Fig. 56 Air bearing floor installation.

APPENDIX C

SOURCE CODE

This is the Matlab code LSE.m used to analyze the input-output data taken during the least squares identification process.

```
%% LSE.m (Least Squares Est%imate)
%% Written By:  Bill Benson February 28, 1998
%% This is a matlab code that finds the unknown parameters of the
%% robot using the least squares method.
%% The program will prompt the user for the data file (in *.*.wk1)
format
%% and it will analyze the data and return the ARX parameters.

clear all
  filename=input('What file to search?','s')

rng=[1 1 230 6];
A=wk1read(filename,1,1,rng);% Search the spreadsheet file

quit=length(A);
deltaT=sum(A(:,4))/quit;% find the average sampling time

RN=[0 0;0 0];FN=[0 0]';% initialize the process

for step=2:quit,
   phi(1,step)=A(step-1,2);
   phi(2,step)=A(step-1,1);
   RNtemp=phi(:,step)*phi(:,step)';
   FNtemp=phi(:,step)*A(step,2);
   RN=RN+RNtemp;FN=FN+FNtemp;
 end
RN=RN/309;FN=FN/309;
('The estimated discrete parameters are:')
theta=inv(RN)*(FN)

('The estimated continuous time parameters are')
% convert from discrete to continuous time
Ac=log(theta(1))/deltaT
Bc=inv(deltaT +.5*Ac*deltaT^2 +1/6*Ac^2*deltaT^3)*theta(2)
time=(1:step)*deltaT;
('time constant')
deltaT
clear phi
('or maybe')
phi(2:229,1)=A(1:228,2);
phi(2:229,2)=A(1:228,1);
K=inv(phi'*phi)*phi';
```

```
y=A(1:229,2);
thetaK=K*y

%Find the confidence interval
v=y-phi*thetaK;
m=length(v);
Rv=cov(v);
Rtk=K*Rv*K';
sigma=sqrt(diag(Rtk))
('The confidence Interval')
('High Limit')
high=thetaK+sigma;
Ach=log(high(1))/deltaT
Bch=inv(deltaT +.5*Ach*deltaT^2 +1/6*Ach^2*deltaT^3)*high(2)
('Low Limit')
low=thetaK-sigma;
Acl=log(low(1))/deltaT
Bcl=inv(deltaT +.5*Acl*deltaT^2 +1/6*Acl^2*deltaT^3)*low(2)
yhat=phi*thetaK;
figure(1)
subplot(211)
plot(A(:,3),A(:,2));grid;title('Angular Rate vs. Time');
xlabel('time (secs)');ylabel('Angular Rate (rad/sec)');
subplot(212)
plot(A(:,3),A(:,1));grid;title('Input vs. Time');
xlabel('time (secs)');ylabel('Input (on/off)');

figure(2)
subplot(211)
plot(A(:,3),yhat,'b-',A(:,3),A(:,2),'g-.');grid;title('Angular Rate vs.
Time');
xlabel('time (secs)');ylabel('Angular Rate
(rad/sec)');legend('predicted','actual');
subplot(212)
plot(A(:,3),A(:,1));grid;title('Input vs. Time');
xlabel('time (secs)');ylabel('Input (on/off)');
```

This is the Matlab code PSDsetup.m that determines the inertial position of the PSD sensors.

```
%% This m-file is designed to find the actual position of the
photodetectors
%% PSDsetup.m
%% Written by: Bill Benson June 25, 1998
%%
%% This program analyzes the data collected during the inertial frame
set-up
%% process and determines the location of the PSD sensors in the
inertial
%% coordinate frame.
%% a numerical non-linear least squares
clear all
```

```
%%initialize the actual values

z(1,1)= 0 %.1*Theta1c;
z(2,1)=16 %*x1c;
z(3,1)= 86 %1.9*y1c;

z(4,1)= 0 %.5*Theta2c;
z(5,1)=55 % *x2c;
z(6,1)= 86 %.4*y2c;
k=1;

% data is entered manually here
theta1=[];
theta2=[];
x=[];
y=[];

for k=1:40,%number of iterations
for I = 1:length(theta1),% generate Jacobian matrix
   temp2=2*I;
   temp1=2*I-1;
   % evaluate the error at each point

   f(temp1)=tan(z(1,k) + theta1(I)) - (x(I)-z(2,k))/(z(3,k)-y(I));
   f(temp2)=tan(z(4,k) + theta2(I)) - (z(5,k)-x(I))/(z(6,k)-y(I));

   A(temp1,1)=1+tan(z(1,k)+theta1(I))^2;%df1/dtheta1c
   A(temp1,2)=1/(z(3,k)-y(I));%df1/dx1c
   A(temp1,3)=(x(I)-z(2,k))/(z(3,k)-y(I))^2;%df1/dyc1
   A(temp2,4)=1+tan(z(4,k)+theta2(I))^2;%df2/dtheta2c
   A(temp2,5)=-1/(z(6,k)-y(I));%df2/dxc2
   A(temp2,6)=(z(5,k)-x(I))/(z(6,k)-y(I))^2;%df2/dyc2

end

%% compute the psuedoinverse
deltax=-pinv(A)*f'

%% update the guess
z(:,k+1)=z(:,k)+deltax;

end

% display the results
z(1,k)
z(2,k)
z(3,k)
z(4,k)
z(5,k)
z(6,k)

% plot the results
figure(1)
subplot(321)
```

```
plot(z(1,:));
subplot(322)
plot(z(2,:));
subplot(323)
plot(z(3,:));
subplot(324)
plot(z(4,:));
subplot(325)
plot(z(5,:));
subplot(326)
plot(z(6,:));
```

This Matlab code simulates the Kalman Estimator.

```
%% ExtKalman.m
%% Written by: Bill Benson, July 2, 1998
%% This program simulates the identification of unknown model
parameters
%% of the robot using an extended Kalman filter with a state-space
matrix augmented
%% by additional states that represent the unknown parameters.

clear all
quit = 3600; %number of time steps
deltaT = .01; %Discrete Sample Interval
h=.01; %Runge Kutta interval
j=1; %EKF iterate

x(:,1)=[1 0 1 0 1 0 .0329 .0317 1 .0342 .0337 0 .0331 .0293 1 .0305
.0294 0 .0403 .0365 1 .0351 .0310 0 .0334 .0329 1 .0412 .0373 0]';
%initial conditions of actual state
xhat(:,1)=.7*x(:,1);xhat(9,1)=x(9,1);xhat(9,1)=x(9,1);xhat(12,1)=x(12,1
);xhat(15,1)=x(15,1);
xhat(18,1)=x(18,1);xhat(21,1)=x(21,1);xhat(24,1)=x(24,1);xhat(27,1)=x(2
7,1);xhat(30,1)=x(30,1);
phi=eye(9); %initial conditions of the state transition matrix

P=eye(30);
phi=eye(30);
Q=eye(30);
H=[1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
   0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
   0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
   0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
   0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
   0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
   0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
   0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0;
   0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0;
   0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0;
   0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0;
   0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1];
```

```
R=eye(12)*.01;
tset=1; %thruster select
z(:,1)=H*x(:,1);
for I=1:quit;
   %  The robot waltz
   if (rem(I,300) == 0),
      tset=tset+1;
      if (tset == 7),
         tset = 1;
      end
      %rotate clockwise
      if (tset==1),
         x(9,I)=1;
         x(12,I)=0;
         x(15,I)=1;
         x(18,I)=0;
         x(21,I)=1;
         x(24,I)=0;
         x(27,I)=1;
         x(30,I)=0;
      end
      %rotate counter-clockwise
      if (tset==2),
         x(9,I)=0;
         x(12,I)=1;
         x(15,I)=0;
         x(18,I)=1;
         x(21,I)=0;
         x(24,I)=1;
         x(27,I)=0;
         x(30,I)=1;
      end
      %move forward
      if (tset==3),
         x(9,I)=0;
         x(12,I)=0;
         x(15,I)=0;
         x(18,I)=0;
         x(21,I)=0;
         x(24,I)=1;
         x(27,I)=1;
         x(30,I)=0;
      end
      %move backwards
      if (tset==4),
         x(9,I)=0;
         x(12,I)=1;
         x(15,I)=1;
         x(18,I)=0;
         x(21,I)=0;
         x(24,I)=0;
         x(27,I)=0;
```

```
            x(30,I)=0;
         end
         %move right
         if (tset==5),
            x(9,I)=0;
            x(12,I)=0;
            x(15,I)=0;
            x(18,I)=1;
            x(21,I)=1;
            x(24,I)=0;
            x(27,I)=0;
            x(30,I)=0;
         end
         %move left
         if (tset==6),
            x(9,I)=1;
            x(12,I)=0;
            x(15,I)=0;
            x(18,I)=0;
            x(21,I)=0;
            x(24,I)=0;
            x(27,I)=0;
            x(30,I)=1;
         end
      end

      %%simulate the robot trajectory
      deltax1(:,I)=DelX(deltaT,x(:,I));
      dx2=x(:,I)+.5*deltax1(:,I);
      deltax2(:,I)=DelX(deltaT,dx2);
      dx3=x(:,I)+.5*deltax2(:,I);
      deltax3(:,I)=DelX(deltaT,dx3);
      dx4=x(:,I)+deltax3(:,I);
      deltax4(:,I)=DelX(deltaT,dx4);
      x(:,I+1)=x(:,I) + (deltax1(:,I) + 2*deltax2(:,I) + 2*deltax3(:,I) +
   deltax4(:,I))/6;

      j=I;

      %integrate the estimate forward in time
      dhat1(:,I)=DelX(h,xhat(:,I));
      dh2=xhat(:,I)+.5*dhat1(:,I);
      dhat2(:,I)=DelX(h,dh2);
      dh3=xhat(:,I)+.5*dhat2(:,I);
      dhat3(:,I)=DelX(h,dh3);
      dh4=xhat(:,I)+dhat3(:,I);
      dhat4(:,I)=DelX(h,dh4);

      xhat(:,I+1)=xhat(:,j) + (dhat1(:,j) + 2*dhat2(:,j) + 2*dhat3(:,j) +
   dhat4(:,j))/6;

      %find the state transition matrix
      A=jacob(xhat(:,j));
      dphi1=h*(A*phi);
```

```
    A=jacob(dh2);
    dphi2=h*(A*(phi+.5*dphi1));
    A=jacob(dh3);
    dphi3=h*(A*(phi+.5*dphi2));
    A=jacob(dh4);

    dphi4=h*(A*(phi+dphi3));
    phi=phi+(dphi1 + 2*dphi2 + 2*dphi3 +dphi4)/6;

    if (rem(I,10)==0),

       z(:,I+1)=H*x(:,I+1);
       %Run Extended Kalman Filter
       P=phi*P*phi' + Q;
       K=P*H'*inv(H*P*H' + R);
       xhat(:,j+1)=xhat(:,j+1)+K*(z(:,I+1)-H*xhat(:,j+1));
       P=(eye(30)-K*H)*P;
       ERROR(:,I)=(x(:,I+1)-xhat(:,I+1));
       phi=eye(30);
    end

end
time=(1:quit)*deltaT;
figure(1)
subplot(321)
plot(time,x(1,1:quit),'--',time,xhat(1,1:quit),'-');ylabel('Rotation
(rad)');
legend('actual','estimate');
subplot(322)
plot(time,x(2,1:quit),'--',time,xhat(2,1:quit),'-');ylabel('Rotation
Rate (rad/sec)');
subplot(323)
plot(time,x(3,1:quit),'--',time,xhat(3,1:quit),'-');ylabel('X (ft)');
subplot(324)
plot(time,x(4,1:quit),'--',time,xhat(4,1:quit),'-');ylabel('X-rate
(ft/sec)');
subplot(325)
plot(time,x(5,1:quit),'--',time,xhat(5,1:quit),'-');ylabel('Y (ft)');
xlabel('time (sec)');
subplot(326)
plot(time,x(6,1:quit),'--',time,xhat(6,1:quit),'-');ylabel('Y-rate
(ft/sec)');
xlabel('time (sec)')

orient tall

figure(2)
subplot(311)
plot(time,x(7,1:quit),'--b',time,xhat(7,1:quit),'-r');grid
on;axis([0,quit*h,.01,.05]);
title('Rotational Parameter for Thruster
V1');legend('actual','estimate')
subplot(312)
```

```
plot(time,x(8,1:quit),'--b',time,xhat(8,1:quit),'-r');grid
on;axis([0,quit*h,.01,.05]);
title('Translational Parameter for Thruster V1');
subplot(313)
plot(time,x(9,1:quit),'--b',time,xhat(9,1:quit),'-r');grid
on;axis([0,quit*h,-.5,1.5])
title('Input to Thruster V1');xlabel('time (sec)');
orient tall
```

This mfile is a subroutine of ExtKalman.m and it contains the nonlinear differential equations of motion.

```
function[DX]=DelX2(deltaT,x)
%%This evaluates the non-linear equations
sx1=sin(x(1));cx1=cos(x(1));
DX=deltaT*[x(2);
   (-x(15)*x(7)*x(21)/x(20)+x(16)*x(8)*x(22)/x(20)-
x(16)*x(9)*x(23)/x(20)+x(17)*x(10)*x(24)/x(20)-
x(17)*x(11)*x(25)/x(20)+x(18)*x(12)*x(26)/x(20)-
x(18)*x(13)*x(27)/x(20)+x(15)*x(14)*x(28)/x(20));
   x(4);
   (-sx1*x(7)*x(21)/x(19)+cx1*x(8)*x(22)/x(19)-cx1*x(9)*x(23)/x(19)-
sx1*x(10)*x(24)/x(19)+sx1*x(11)*x(25)/x(19)-
cx1*x(12)*x(26)/x(19)+sx1*x(13)*x(27)/x(19)+cx1*x(14)*x(28)/x(19));
   x(6);
   (cx1*x(7)*x(21)/x(19)+sx1*x(8)*x(22)/x(19)-
sx1*x(9)*x(23)/x(19)+cx1*x(10)*x(24)/x(19)-cx1*x(11)*x(25)/x(19)-
sx1*x(12)*x(26)/x(19)+cx1*x(13)*x(27)/x(19)-sx1*x(14)*x(28)/x(19));
   0;
   0;
   0;
   0;
   0;
   0;
   0;
   0;
   0;
   0;
   0;
   0;
   0;
   0;
   0;
   0;
   0;
   0;
   0;
   0;
   0;
   0];
```

This mfile is a subroutine of ExtKalman.m and it computes the Jacobian matrix [A].

```
function[A]=jacob2(x);
%%This function evaluates the Jacobian of the nonlinear equations
A=zeros(28);
sx1=sin(x(1));cx1=cos(x(1));
A(1,2)=1;
A(2,7)=-x(15)*x(21)/x(20);
A(2,15)=(x(14)*x(28)-x(7)*x(21))/x(20);
A(2,21)=-x(15)*x(7)/x(20);A(2,8)=x(16)*x(22)/x(20);
A(2,16)=(x(8)*x(22)-x(9)*x(23))/x(20);
A(2,22)=x(8)*x(16)/x(20);
A(2,9)=-x(16)*x(23)/x(20);
A(2,23)=-x(16)*x(9)/x(20);
A(2,10)=x(17)*x(24)/x(20);
A(2,24)=A(17)*x(10)/x(20);
A(2,17)=(x(10)*x(24)-x(11)*x(25))/x(20);
A(2,11)=-x(17)*x(25)/x(20);
A(2,25)=x(17)*x(11)/x(20);
A(2,12)=x(18)*x(26)/x(20);
A(2,26)=x(18)*x(12)/x(20);
A(2,18)=(x(12)*x(26)-x(13)*x(27))/x(20);
A(2,13)=-x(18)*x(27)/x(20);
A(2,27)=-x(18)*x(13)/x(20);
A(2,14)=x(15)*x(28)/x(20);
A(2,28)=x(15)*x(14)/x(20);
A(2,20)=(x(15)*x(7)*x(27)-x(16)*x(8)*x(22)+x(16)*x(9)*x(23)-
x(17)*x(10)*x(24)+x(17)*x(11)*x(25)-
x(18)*x(12)*x(26)+x(18)*x(13)*x(27)-x(15)*x(14)*x(28))/x(20)^2;
A(3,4)=1;
A(4,1)=(-cx1*x(7)*x(21)/x(19)-
sx1*x(8)*x(22)/x(19)+sx1*x(9)*x(23)/x(19)-
cx1*x(10)*x(24)/x(19)+cx1*x(11)*x(25)/x(19)+sx1*x(12)*x(26)/x(19)+cx1*x
(13)*x(27)/x(19)-sx1*x(14)*x(28)/x(19));
A(4,7)=-sx1*x(21)/x(19);
A(4,21)=-sx1*x(7)/x(19);
A(4,8)=cx1*x(22)/x(19);
A(4,22)=cx1*x(8)/x(19);
A(4,9)=-cx1*x(23)/x(19);
A(4,23)=-cx1*x(9)/x(19);
A(4,10)=-sx1*x(24)/x(19);
A(4,24)=-sx1*x(10)/x(19);
A(4,11)=sx1*x(25)/x(19);
A(4,25)=sx1*x(11)/x(19);
A(4,12)=-cx1*x(26)/x(19);
A(4,26)=-cx1*x(12)/x(19);
A(4,13)=sx1*x(27)/x(19);
A(4,27)=sx1*x(13)/x(19);
```

```
A(4,14)=cx1*x(28)/x(19);
A(4,28)=cx1*x(14)/x(19);
A(4,19)=(sx1*x(7)*x(21)-cx1*x(8)*x(22)+cx1*x(9)*x(23)+sx1*x(10)*x(24)-
sx1*x(11)*x(25)+cx1*x(12)*x(26)-sx1*x(13)*x(27)-
cx1*x(14)*x(28))/x(19)^2;
A(5,6)=1;
A(6,1)=(-sx1*x(7)*x(21)/x(19)+cx1*x(8)*x(22)/x(19)-
cx1*x(9)*x(23)/x(19)-sx1*x(10)*x(24)/x(19)+sx1*x(11)*x(25)/x(19)-
cx1*x(12)*x(26)/x(19)-sx1*x(13)*x(27)/x(19)-cx1*x(14)*x(28)/x(19));
A(6,7)=cx1*x(21)/x(19);
A(6,21)=cx1*x(7)/x(19);
A(6,8)=sx1*x(22)/x(19);
A(6,22)=sx1*x(8)/x(19);
A(6,9)=-sx1*x(23)/x(19);
A(6,23)=-sx1*x(9)/x(19);
A(6,10)=cx1*x(24)/x(19);
A(6,24)=cx1*x(10)/x(19);
A(6,11)=-cx1*x(25)/x(19);
A(6,25)=-cx1*x(11)/x(19);
A(6,12)=-sx1*x(26)/x(19);
A(6,26)=-sx1*x(12)/x(19);
A(6,13)=cx1*x(27)/x(19);
A(6,27)=cx1*x(13)/x(19);
A(6,14)=-sx1*x(28)/x(19);
A(6,28)=-sx1*x(14)/x(19);
A(6,19)=(-cx1*x(7)*x(21)-sx1*x(8)*x(22)+sx1*x(9)*x(23)-
cx1*x(10)*x(24)+cx1*x(11)*x(25)+sx1*x(12)*x(26)-
cx1*x(13)*x(27)+sx1*x(14)*x(28))/x(19)^2;
```

This C code finds the orientation and position of the robot.

```
/*********************************************************************
*triangul.c                                                          *
*Written by:  Bill Benson, July 28, 1998                             *
*                                                                    *
*This code recovers the robot orientation and position from the      *
*available line of sight angles from the two PSD sensors.  The code is*
*flexible and can adjust to use from 4 to 8 available target angles.  *
*                                                                    *
*********************************************************************/

#include <analysis.h>
#include <ansi_c.h>
#include "mat_lib.h"
#include <math.h>
#define PI 3.14159265359

double* triangul12(double Psd_Pos[3][2],double Loc_tar[2][4],double
Tar_tan[4][2],double guess[3])
{
double X[4],Y[4],den,Or[2];
double **A,**Atran,**AtranA,**invAtranA,**psuedo,**F;
double **deltax;
double* output;
int i,j,k,I,J,mul,status,dud,ntar,ptar,visible=0,LOS[9][2];
double TstarR[2][1],TntarR[2][1],Tntar[2],Cmat[2][2];
double Tht[4][2],ITht,ztemp[3];
double norm[2],dx[3]={.0001,.0001,.0001};
double tgttan[2];
double* coord;
double Tp[2],Tn[2],Lsn,Dspn,d;
double c,s,cn,sn,phi,max,min,ORS[2][1],cp,sp,sptemp,cptemp;

for (j=0;j<4;j++) {
      for (i=0;i<2;i++){
      if (Tar_tan[j][i] != 0.0) {
            /*search the tangents to see how many and which targets are
visible*/
            visible++;
            LOS[visible][0]=j;
            LOS[visible][1]=i;
            }
            Tht[j][i]=(Psd_Pos[2][i]+Tar_tan[j][i])/(1-
Psd_Pos[2][i]*Tar_tan[j][i]);
      }
      Y[j]=Tht[j][0];/*store tangents for later sorting */
      X[j]=Tht[j][1];

}
/* setup the Jacobian matrices, size varies according to number of
visible targets */
A=dmatrix(1,visible,1,3);
invAtranA=dmatrix(1,3,1,3);
F=dmatrix(1,visible,1,1);
deltax=dmatrix(1,3,1,1);


/*check to see if initial guess is totally absurd, if it is reset*/

if (guess[0] >= (double)40.0)
      guess[0] = (double)0.0;

if (guess[0] <= (-(double)40.0))
      guess[0] = (double)0.0;

if (guess[1] >= (double)40.0)
      guess[1] = (double)0.0;

if (guess[1] <= (-(double)40.0))
      guess[1] = (double)0.0;

if (guess[2] >= 2*PI)
      guess[2] = (double)0.0;

if (guess[2] <= -(-2*PI))
      guess[2] = (double)0.0;
```

```
for (k=1;k<=20;k++) {
         cp=cos(guess[2]);
         sp=sin(guess[2]);

         ztemp[0]=guess[0]+dx[0];
         ztemp[1]=guess[1]+dx[1];
         ztemp[2]=guess[2]+dx[2];

         cptemp=cos(ztemp[2]);
         sptemp=sin(ztemp[2]);

         for (i=1;i<=visible;i++) {
          I=LOS[i][0];
          /*compute the vector f and the numerical Jacobian matrix for
PSD1*/
            F[i][1]=(guess[0]+Loc_tar[0][I]*cp-Loc_tar[1][I]*sp-
Psd_Pos[0][0])/(Psd_Pos[1][0]-guess[1]-Loc_tar[0][I]*sp-
Loc_tar[1][I]*cp)-Tht[I][0];

          A[i][1]=-(F[i][1]-((ztemp[0]+Loc_tar[0][I]*cp-
Loc_tar[1][I]*sp-Psd_Pos[0][0])/(Psd_Pos[1][0]-guess[1]-
Loc_tar[0][I]*sp-Loc_tar[1][I]*cp)-Tht[I][0]))/dx[0];

            A[i][2]=-(F[i][1]-((guess[0]+Loc_tar[0][I]*cp-
Loc_tar[1][I]*sp-Psd_Pos[0][0])/(Psd_Pos[1][0]-ztemp[1]-
Loc_tar[0][I]*sp-Loc_tar[1][I]*cp)-Tht[I][0]))/dx[1];

            A[i][3]=-(F[i][1]-((guess[0]+Loc_tar[0][I]*cptemp-
Loc_tar[1][I]*sptemp-Psd_Pos[0][0])/(Psd_Pos[1][0]-guess[1]-
Loc_tar[0][I]*sptemp-Loc_tar[1][I]*cptemp)-Tht[I][0]))/dx[2];


            if (LOS[i][1]==1) {
                  /*compute the vector f and the numerical Jacobian
matrix for PSD2*/
                  F[i][1]=-(guess[0]+Loc_tar[0][I]*cp-Loc_tar[1][I]*sp-
Psd_Pos[0][1])/(Psd_Pos[1][1]-guess[1]-Loc_tar[0][I]*sp-
Loc_tar[1][I]*cp)-Tht[I][1];

                  A[i][1]=-(F[i][1]-(-(ztemp[0]+Loc_tar[0][I]*cp-
Loc_tar[1][I]*sp-Psd_Pos[0][1])/(Psd_Pos[1][1]-guess[1]-
Loc_tar[0][I]*sp-Loc_tar[1][I]*cp)-Tht[I][1]))/dx[0];

                  A[i][2]=-(F[i][1]-(-(guess[0]+Loc_tar[0][I]*cp-
Loc_tar[1][I]*sp-Psd_Pos[0][1])/(Psd_Pos[1][1]-ztemp[1]-
Loc_tar[0][I]*sp-Loc_tar[1][I]*cp)-Tht[I][1]))/dx[1];

                  A[i][3]=-(F[i][1]-(-(guess[0]+Loc_tar[0][I]*cptemp-
Loc_tar[1][I]*sptemp-Psd_Pos[0][1])/(Psd_Pos[1][1]-guess[1]-
Loc_tar[0][I]*sptemp-Loc_tar[1][I]*cptemp)-Tht[I][1]))/dx[2];

            }//end of if
          }//end of for
          dprint_mat(A,visible,3);
          printf("\n");
          /*compute the pseudoinverse*/

      Atran=dtranspose(A,4,3);
      AtranA=dmult_mat(Atran, A,3,4,3);
      dmat_inver(3,AtranA,invAtranA);
      psuedo=dmult_mat(invAtranA, Atran,3,3,4);
      deltax=dmult_mat(psuedo,F,3,4,1);

          /*update the guess*/

          guess[0]=guess[0]-deltax[1][1];
          guess[1]=guess[1]-deltax[2][1];
          guess[2]=guess[2]-deltax[3][1];
          }

output=malloc(sizeof(double)*3);

/* reset the matrices for next sample time*/
free_dmatrix(A,1,visible,1,3);
free_dmatrix(Atran,1,3,1,visible);
free_dmatrix(AtranA,1,3,1,3);
free_dmatrix(invAtranA,1,3,1,3);
free_dmatrix(psuedo,1,3,1,visible);
free_dmatrix(deltax,1,3,1,1);
free_dmatrix(F,1,visible,1,1);


output[0]=guess[0];
output[1]=guess[1];
output[2]=guess[2];
return output;

}
```

VITA

William Wyborn Benson
295 Beacon Street (1st Floor Apartment)
Somerville, MA 02143-3509

Major Field: Aerospace Engineering

Biographical Information:

Personal Data: William Benson was born on November 17, 1969 in Summit, N.J. On June 3, 1995, he married Linda Jane Duncan. He was commissioned as a Second Lieutenant in the United States Army in June 1992 and served on active duty in Germany, Kuwait and Bosnia until September 1996. He has accepted a position as a systems engineer at Raytheon Electronic Systems in Tewksbury, Massachusetts.

Education: William Benson attended high school at Hackettstown High School in 1988, and received an Army Scholarship at Washington and Lee University. He graduated Cum Laude with a Bachelor of Science degree in Physics-Engineering in June 1992.