INSTRUMENTATION AND ALGORITHMS FOR POSTURE ESTIMATION IN COMPLIANT FRAMED MODULAR MOBILE ROBOTIC SYSTEMS

by

Clinton Leroy Merrell

A thesis submitted to the faculty of
The University of Utah
in partial fulfillment of the requirements for the degree of

Master of Science

Department of Mechanical Engineering

The University of Utah

December 2005

THE UNIVERSITY OF UTAH GRADUATE SCHOOL

SUPERVISORY COMMITTEE APPROVAL

of a thesis submitted by

Clinton Leroy Merrell

This thesis has been read by each member of the supervisory committee and by majority vote has been found to be satisfactory.

	Chair: Mark A. Minor
	Sanford G. Meek
	Daniel O. Adams

THE UNIVERSITY OF UTAH GRADUATE SCHOOL

FINAL READING APPROVAL

To the Graduate Council of the University of Utah:

I have read the thesis of Roy Merrell in its final form and have found that (1) its format, citations, and bibliographic style are consistent and acceptable; (2) its illustrative materials including figures, tables, and charts are in place; and (3) the final manuscript is satisfactory to the supervisory committee and is ready for submission to The Graduate School.

Date

Mark A. Minor
Chair: Supervisory Committee

Approved for the Major Department

Kent S. Udell
Chair/Dean

Approved for the Graduate Council

David S. Chapman
Dean of The Graduate School

ABSTRACT

Posture sensing techniques for Compliant Framed Modular Mobile Robots (CFMMR) are presented in this paper using a new Relative Posture Sensor (RPS) combined with standard sensors in a tiered fusion algorithm. The fusion algorithm uses traditional Kalman filters and rigid axle kinematic models to predict the global posture of each axle. A Covariance Intersection filter then fuses disparate axle data using the RPS. The RPS consists of a compliant frame member instrumented with strain gauges and associated algorithms. Modeling and instrumentation of the RPS for large deflections is the remaining focus. While the RPS can predict relative posture, force, and moment, only posture estimates are used here. Experimental results are derived from over 60 trials operating the robot on high traction carpet, low traction sand, and sand with rugged rocky terrain. Results comparing the RPS combined with a variety of standard sensor configurations on varied terrain demonstrate that the derived posture sensing techniques yield accurate relative posture estimates and posture regulation even on rugged terrain, which is a vast improvement over previous results.

For my wife and daughters – without whose long suffering and patience this would never have come to be.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF SYMBOLS

x	The states of the system.
u_k	The inputs at time k. Not used in this implementation
a	The nonlinear state equations from the kinematic model.
T	The time step.
A	The state matrix, this is the linearized state equations from the kinematic model.
B	The input matrix. Not used in this implementation
Q	The model covariance matrix.
R	measurement covariance matrix. This is a measure of the sensor's noise
P	error covariance matrix.
H	measurement matrix.
z_k	The measurements at time k. This is the sensor outputs.
I	The identity matrix.
e	The distance from the origin to the midpoint of a line between the center point of each of the robots axles.
γ	The orientation of a line drawn between the center point of each of the robots axles.
x	The axle level x coordinate.
y	The axle level y coordinate.
ϕ	The axle level orientation.
V	The axle level translational velocity.
$\dot{\phi}$	The axle level rotational velocity.
$\dot{V}$	The axle level translational acceleration.
$\ddot{\phi}$	The axle level rotational acceleration.

LIST OF ABBREVIATIONS

CFMMR	Compliant Frame Modular Mobile Robot
RPS	Relative Position Sensor
EKF	Extended Kalman Filter
GPS	Global Positioning System
CI	Covariance Intersection
IPEC	Internal Posture Error Correction
DOF	Degrees Of Freedom
IR	Infra redR
MEMS	Micro-Electro-Mechanical Systems
IMU	Inertial Measurement Unit
SLAM	Simultaneous localization And Mapping
ADC	Analog to Digital Converter
PWM	Pulse Width Modulation
DIP	Dual In-line Package
Pot	Potentiometer
NiCad	Nickel-Cadmium
Gyro	Gyroscope

1. INTRODUCTION

A new breed of wheeled mobile robotic systems is the subject this research: Compliant Framed Modular Mobile Robots (CFMMR), Figure 1. This concept is unique in several ways. First, it uses a novel yet simple structure to provide suspension and highly controllable steering capability to mobile robots without adding any additional hardware to the system. This is accomplished by using flexible frame elements to couple

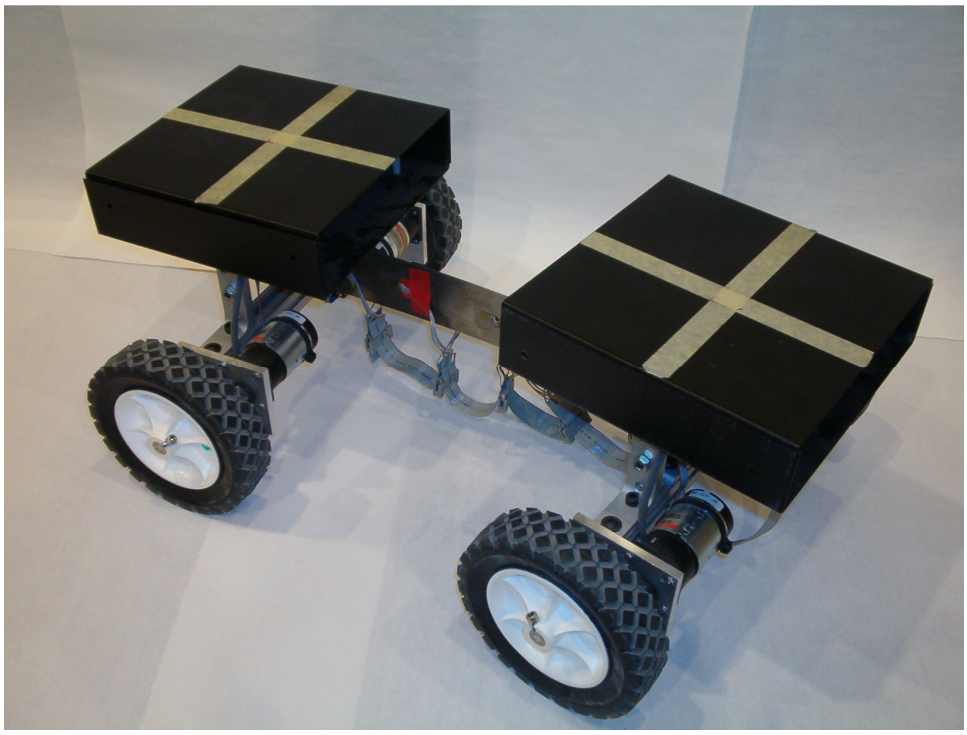

Figure 1. Compliant framed modular mobile robot.

rigid differentially steered axles. In this study, the frame element provides compliant roll and yaw between the axles. Roll provides suspension capability to the system by allowing the axles to deflect in order to accommodate uneven terrain, and yaw allows the axles to independently change heading for advanced steering capability. Steering and maneuvering of the system are thus accomplished via coordinated control of the axles to achieve a desired mobility task, such as path following or stabilization about a point. Since each axle can be steered independently, the system provides enhanced maneuverability in confined environments as well as the capability to control the shape of the frame.

A second and very unique aspect of the CFMMR is its predisposition for modular robotics. Reconfigurable modular robotic systems have been of great interest to researchers during the last decade due to their improved ability to overcome obstacles and perform a variety of tasks using a single hardware platform. To achieve this goal, numerous researchers have devoted their efforts to investigating minimalist homogenous robotic modules that can be assembled in various formats to create sophisticated robotic systems that are highly capable of adapting to uncertainty in their environment. These systems have examined reconfigurable manipulation [1], mobility [2-4], or a combination therein [5, 6]. Module homogeneity is advantageous for a number of reasons. With homogeneous modules system complexity is reduced. Manufacturing is also simplified. When a small set of modules can be reconfigured to address a specific functionality the amount of custom design and manufacturing required to address a new functionality is greatly reduced. This simplification of the manufacturing process will also bring with it the added benefit of cost reduction. An additional benefit can be found in the fact that a

fleet of CFMMR's regardless of configuration will have a part type count similar to a single CFMMR. This has the potential to drastically reduce the parts inventory required by a fleet maintenance program. A similar advantage is realized in the reduction of the amount of training and expertise required on the part of the maintenance staff for a fleet of CFMMR's versus a fleet of more traditional robots that are custom built to a perform various tasks. The modular nature of the system also increases the robustness of the fleet by providing redundancy. The homogeneous nature of the modules provides compact and ordered storage. The improved storage capability also means that more modules can be transported at the same time, reducing logistical constraints to a large deployment. The redundancy and homogeneity also lend themselves to increased adaptability of the systems [7, 8]. If the operating requirements in the field change the robots can easily be reconfigured and optimized for the task at hand. The compliant frame allows this concept to be extended to wheeled mobile robots by allowing a number of different vehicle configurations to be formed from a set of uniform frame and axle modules.

Another unique aspect of this research is the Relative Position Sensor (RPS), used to improve the positional tracking between the front and rear axles. In navigating from point to point, the independent axles and flexible frame allow the robot to control its shape offering an improvement over more traditional rigid vehicles to move over or around many obstacles. Vital to this advanced control is the accurate knowledge of the relative position and orientation of the axles. Even the simplest of navigational maneuvers will result in antagonistic forces exerted between the axles, if the error in relative position and orientation between the axles is too large. Therefore, it is of paramount importance that the posture estimation drift between the two axles be

eliminated. The RPS accomplishes this feat with a package small and light enough to sit on the robot without limiting its full range of motion or impeding its travel in any way. In addition to relative posture estimation, another unique aspect of this sensor is its ability to estimate the forces and moments acting on the beam.

In addition to its utility in this application the RPS has potential for extensibility to structural failure analysis. Any homogeneous beam like structure can be instrumented in an RPS-like fashion. Analysis of the RPS output will provide estimates of beam deflection, forces, and moments at any point along the length of the beam. This technique could be used to improve understanding of typical structural beam member failure modes. A RPS type system could also improve understanding of high velocity impact studies such as automotive crash testing. The vehicle's internal structural members could be instrumented with strain gauges and the RPS algorithm would provide researchers with greater insight into the forces and deflections as the collision occurs.

Another unique aspect of this research is the combination of data fusion methods used to incorporate the data from the RPS, and other sensors that the robot may use. The modular nature of the CFMMR and the novel nature of the RPS result in challenging data fusion problems. This research uses an Extended Kalman Filter (EKF) to fuse data sources on an individual axle module, such as wheel encoders, axle gyroscopes, accelerometers, GPS, etc. Because an individual axle unit has the capability to act autonomously the data fusion system must have a self contained axle level module. The literature is replete with examples of this type of EKF implementation. When multiple axle modules are joined with RPS equipped flexible frame members, a second tier of data fusion is necessary to facilitate data transfer from the RPS to the adjoining axle modules

and between the joined axle modules themselves. The Covariance Intersection (CI) filter strategy is used for the second tier data fusion module. The unknown correlation between data sources as relative posture information is passed back and forth between axle modules and the RPS makes this module of the data fusion algorithm particularly challenging. In order for the EKF to be consistent in its state estimation it is necessary that the correlation between data sources be known. The CI filter assumes complete correlation and is therefore consistent in its estimation in the face of uncertain correlations [9]. All of this must be accomplished in real time without impeding the robot's motion controller [10-12]. A third component of the data fusion structure is the incorporation of Borenstein's Internal Posture Error Correction (IPEC) algorithm [13]. The IPEC algorithm has been incorporated into the CI filter. This hybrid fusion approach provides the odometry error correction ability of the IPEC algorithm while maintaining the covariance propagation of the CI filter. This combination of data fusion techniques is a new approach that is required by the modular nature of the robot and the unique sensing capability of the RPS.

The tiered data fusion system described here may also have extensibility to cooperative and distributed robotics applications. Any system that requires the cooperation of multiple independent systems to accomplish a task can benefit from this data fusion architecture. Examples include cooperative robotic manipulators, cooperative mobile robots, distributed sensor networks, etc.

Chapter 2 contains background information about this robot configuration and how it relates to other relevant research. Chapter 3 discusses the theoretical basis for the RPS and the problems that were overcome in its implementation. Chapter 4 discusses the axle

level Extended Kalman Filter, (EKF), used to fuse the data from all of the axle level sensors. Chapter 5 is a description of the higher level Covariance Intersection (CI) data fusion algorithm that uses the RPS to propagate posture data between axles connected by the RPS and to detect and correct for inaccuracies in an axles posture estimate. Chapter 6 presents the experimental data gathered from the static tests of the RPS. Chapter 7 presents the results of the system integration testing using the complete robotic system. Chapter 8 contains concluding remarks.

2. BACKGROUND

A limited number of compliant vehicles can be found in the literature, and none possess a similar highly compliant frame whose deflection is controlled by coordinated actuation of the wheels. The earliest found reference to compliant vehicles in the literature is a system proposed for planetary exploration that uses compliant members to provide roll and pitch degrees of freedom for suspension capability between the axles [14]. This concept was later extended [15] in a design where the frame of a vehicle was composed of at least one helical spring, but hydraulic cylinders were then to be used to control the deflection. In each of these cases, compliance was introduced for accommodating terrain. More recent research, by Borenstein [16], has introduced compliance for accommodating measurement error and resulting wheel slip occurring between independently controlled axle units on a service robot. This robot is similar in spirit to the compliant framed system in that it allows relative rotation between the axles, but this is provided by rotary joints connected to the ends of a frame with limited prismatic compliance. The system is intended for operation on flat surfaces in industrial service settings. As the author states, the system provides high levels of mobility, but since the axle units are coupled by a relatively nonconforming rigid frame, its ability to maneuver in confined environments is limited [17]. In comparison to the system proposed here, the Borenstein system [16] is likewise not intended for operating on

uneven terrain, but in smoother and flatter manufacturing environments. Other flexible robots use actuated articulated joints to provide similar relative motion between axles; examples include the Marsokhod rover [18] and other six wheeled research rovers with high relative DOF provided between axle modules [19]. The compliant frame mobile robot system proposed here allows independent steering control of the axles with minimal slip but unlike the aforementioned designs does not require any additional hardware or actuators. It does, however, present unique sensing challenges for localizing the relative position of the axles that the aforementioned robots with fixed joint structures do not experience.

Position localization begins with the most basic unit of this system, the single axle. Numerous examples can be found in the literature that use a Kalman type filter to fuse sensor data from a mobile robot. The typical approach is to use incremental encoder readings as the input to the mobile robot kinematic equations. This odometry model is then fused with another independent sensor, such as a gyroscope, inertial sensors, GPS, vision systems, some type of reflective sensor (laser, sonar, ir), or some combination of the above. The goal is to limit the impact of nonsystematic odometry errors [16] and achieve the best possible posture evaluation with a method that eliminates, or at least limits, the drift inherent in the sensors commonly used on mobile robots. This robot platform is no exception. Each axle has a 1024 count per revolution quadrature encoder attached directly to each wheel. In addition, each axle is equipped with a MEMS rate gyroscope that outputs a voltage proportional to the angular rate of rotation in the yaw direction. This gyroscope utilizes a micro machined vibrating quartz tuning fork element to detect the coriolis force. Each axle is also equipped with two accelerometers. These

accelerometers are oriented to be sensitive to the axle x and y coordinates respectively. See Table 1 for a description of each sensor. For a more detailed description of the axle Inertial Motion Unit (IMU) module see [20]. Even though each axle was equipped with each of these sensors, the front axle gyroscope and the accelerometers were not actually used in any of the experiments in this thesis.

Cooperating axle units are connected with a flexible frame element. When this frame element is instrumented to return relative position information about the axles connected it is referred to as a RPS. The closest found example to the RPS is that described by Piedbeouf [21] for the purpose of predicting the endpoint position of a flexible linkage. While the curvature-based sensor model found there is similar to that presented in this paper, the fundamental constraints on the sensor as well as the beam are different. In Piedbeouf's case the instrumented linkage is a link in a fixed robotic manipulator that ideally should be stiff, but fails the stiffness assumption due to its long and slender geometry. For this application Peidbeouf required only endpoint position relative to the root of the linkage for small deflections. The CFMMR is more demanding in that it requires posture estimates in a system that will greatly exceed the common small deflection assumptions, on a regular basis. In addition, the relative position data returned

Table 1

Sensor manufacturers and part numbers.

Sensor	Sensitive Axis	Manufacturer	Part No.
Gyroscope	Yaw	BEI	HZI-90-100A
Accelerometer	X	Honeywell	ASA7002
Accelerometer	Y	Honeywell	ASA7002

must be used to improve the global posture estimate of both of the robots axles, not merely endpoint position relative to root. This global axle posture estimation is further complicated by the kinematic and nonholonomic constraints of the axle motion.

The closest found example to the current research in mobile robotics is Borenstein's OMNImate [16] where the Internal Position Error Correction (IPEC) algorithm is proposed. This algorithm works by detecting which axle has the least trustworthy posture estimate based on odometry and replacing that axle's posture estimate with the sum of the other axles posture estimate plus the relative posture between the two axles calculated from the linkage potentiometers. This seems to be quite good at limiting the effect of odometry errors, but, no attempt is made to maintain variance information about each axle's posture. Variance information can be vital to a path-planning algorithm in deciding whether the robot posture is known with sufficient accuracy to permit close quarters maneuvering around objects. One of the unique contributions of the data fusion architecture presented here is that it allows the propagation of variance information. The first layer of data fusion consists of the axle level EKF combining all of the sensor data from a single axle. The second layer uses a CI filter to combine the data from the EKF update with pose data from adjacent axles and data from the RPS.

The CI filter was proposed by Uhlman and Julier [9]. This filter technique maintains consistent estimates without regard to the correlation between data sources. This is important because the correlation between data sources is often difficult or impossible to calculate. Because of this property the CI filter has been used in attempts to solve the Simultaneous Localization and Mapping, or SLAM problem [22]. In this application, a mobile robot uses its sensory input to create a map of its environment and

then uses the map to localize itself. The sensor data are therefore correlated with the robots internally generated map but it is very difficult to determine the degree of correlation. Recently, Arambel, Rago, and Mehra [23] proposed a CI filter for use in coordinating the motion of a constellation of deep space imaging satellites for the purpose of increasing the sensitivity of a space based sensor over what a single satellite can accomplish. The sensor fusion architecture used in this application is similar to that of this thesis. Both employ an EKF to calculate the states of each axle/satellite individually. Both systems also use a relative position sensor and data from this RPS is fused via a CI filter because of the difficulty in determining the correlation between data source. This application is unique in that it incorporates the Borenstein IPEC algorithm that acts as a gatekeeper and allows only the more trustworthy of the two axles connected by the RPS to propagate it's posture data through the RPS. In addition, in the case of [23], the RPS is not defined; it is merely assumed to exist and all data presented is from simulation. Whereas, in this thesis, the RPS is a real piece of hardware that has been developed solely for this application and results are based on the actual physical system in motion.

3. THE RELATIVE POSITION SENSOR

It is desired to instrument the frame module in such a way that the relative position and orientation of one axle with respect to the other can be detected by sensor data from the flexible frame that is not dependent on the axle sensors in any way. Such information is critical for dynamic stabilization of the robot and for implementing motion planning algorithms. It will also be helpful if the beam sensors relate information about the forces and moments applied by the beam on the axles. To this end the flexible frame is instrumented with strain gauges at regular intervals along its length. In order to extract the required information from these strain gauges it is necessary to derive a basic theoretical foundation showing how a polynomial interpolation of the curvature of the beam can be obtained from the strain gauges as in [21]. With the curvature based model in hand the needed posture and force/moment information can be estimated. The theoretical frame work will then be followed by a discussion of the implementation of the strain gauges.

Beam Model

The following assumptions are used in defining the theoretical basis for the RPS.

1) The beam will be subjected to extreme bending conditions. In the course of maneuvering the end point of the beam will regularly be bent by greater than

40° relative to its root. Therefore, small angle approximations cannot be used.

2) For this research the robot will operate in a planar space. This means that the RPS will only need to detect the relative x, y, and ϕ of one axle to the other. Along with this assumption is the caveat that the ultimate goal is a robot that can operate in rough terrain and the groundwork should be laid to be able to allow expansion of the role of the RPS to detect the relative roll angle between axles as well.
3) The robot controller used will attempt to maneuver the robot such that the beam maintains a state of pure bending.
4) The sensor will consist of a series of strain bridges placed at known locations along the length of the beam. This means that the strain at discrete locations along the beam will be known.

The starting point for the derivation of the RPS algorithm is the strain at discrete locations along the beam, ε_1, ε_2, …, ε_n. With the strain known accurately at discrete locations the next step is to derive a smooth function based upon the strain gauge data that will interpolate the strain along the length of the beam. Choices for this interpolation include cubic spline, polynomial or a rational function. Due to its relative ease of calculation, the polynomial interpolation was chosen. For this interpolation it is necessary to determine the coefficients of a polynomial that relates strain (ε) to location along the beam (x) where the polynomial exactly intersects the strain reported by each strain gauge. This polynomial will have the form of equation (1), where n is equal to the number of discrete locations along the beam at which the strain is known.

$$\varepsilon(x) = a_1 + a_2 x + \cdots + a_{n-1} x^{n-2} + a_n x^{n-1} \tag{1}$$

The coefficients in equation (1) can be determined by solving the system of linear equations in (2) where l_1, … l_n are the discrete locations along the beam at which the strain is known.

$$\begin{bmatrix} 1 & l_1 & \cdots & l_1^{n-2} & l_1^{n-1} \\ 1 & l_2 & \cdots & l_2^{n-2} & l_2^{n-1} \\ 1 & l_3 & \cdots & l_3^{n-2} & l_3^{n-1} \\ \vdots & \vdots & & \vdots & \vdots \\ 1 & l_n & \cdots & l_n^{n-2} & l_n^{n-1} \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \vdots \\ \varepsilon_n \end{bmatrix} \tag{2}$$

The next step is to determine how to use this strain polynomial to calculate the posture of one axle relative to the other without using any small angle approximations. The starting point is the relationship between strain and curvature (κ) and the radius of curvature (ρ) shown in equation , where y is the inflection point for the strain internal to the beam.

$$\frac{1}{\rho(x)} = \kappa(x) = \frac{\varepsilon(x)}{y(x)} \tag{3}$$

Examination of a small segment of the beam (dL), as seen in Figure 2, will show how the relationship in equation (3) can be exploited. First, it must be noted, that in order for (3) to be generally valid, the segment dL must be chosen sufficiently small so that the change

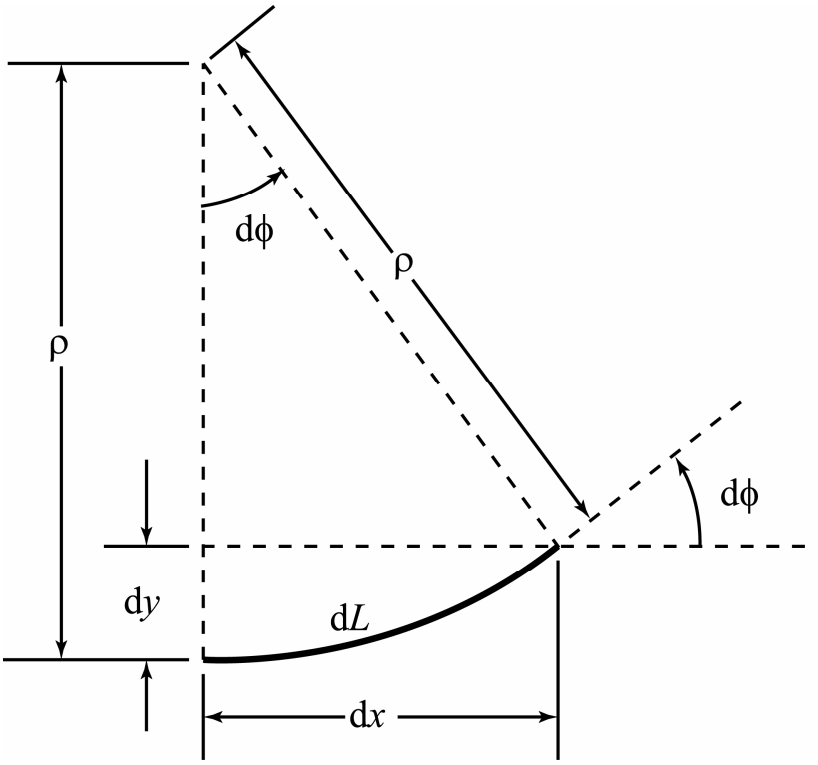

Figure 2. Curvature integration diagram.

in curvature over its length is negligible. When this assumption is met, the segment dL inscribes a circular arc of radius ρ which allows dϕ to be calculated in radians from the arc length equation as in (4).

$$d\phi = \frac{dL}{\rho} \tag{4}$$

Once dϕ is known, trigonometry gives the relationships for dx and dy as shown in equations (5) and (6).

$$dx = \rho \sin\left(d\phi\right) \tag{5}$$

$$dy = \rho\left(1 - \cos\left(d\phi\right)\right) \tag{6}$$

Equations thru (6) are now used in a piecewise integration over the length of the beam to determine the posture (x, y, and ϕ) of the end point of the beam with respect to its root. The beam is broken into segments small enough that the radius of curvature can be considered constant over each segment. At the midpoint of each segment the radius of curvature is determined using equation and then assumed to be constant over the length of each segment. Then equations (4) thru (6) are applied to determine dx, dy, and $d\phi$ for the each segment. Once this is done, the $d\phi$ from each segment is summed to calculate the total angular difference between the endpoint and root of the beam. The application of equations (4) thru (6) assumes that each segment of the beam starts at an angular orientation of 0°. In reality, each segment starts with an angular orientation equal to the sum of $d\phi$'s from all previous segments. In order to calculate the endpoint x and y relative to the beams root, each vector (dx, dy) from each segment must be rotated into alignment with the previous segment. Equation (7) shows how this is accomplished where the vector (dx_r, dy_r) is the rotated position vector and ϕ_p is the sum of each $d\phi$ from all of the previous segments.

$$\begin{bmatrix} dx_r \\ dy_r \end{bmatrix} = \begin{bmatrix} \cos\left(-\phi_p\right) & \sin\left(-\phi_p\right) \\ -\sin\left(-\phi_p\right) & \cos\left(-\phi_p\right) \end{bmatrix} \begin{bmatrix} dx \\ dy \end{bmatrix} \tag{7}$$

After this rotation, the dx_r and dy_r from each segment can be summed to calculate the total change in x and y over the length of the beam.

Because the beam is necessarily clamped to an axle at both of its endpoints it has a small rigid section at each end that needs to be summed in as well. Both of these rigid sections can be handled in a similar manner as the segments from the flexible portion of the beam. The only difference being that, because these segments are rigid $d\phi$ is always zero. After this process of piecewise integration is completed, the result is the calculation of the relative pose (x, y, and ϕ) of the center point of one axle with respect to the other.

The strain polynomial generated by the RPS in calculating the relative posture can also be used to calculate end point forces and moments applied by the axles to the beam. While this information is secondary in importance and not currently used by the robot controller, it is presented here because it could be useful in the future. For this analysis the robot is assumed to be slow enough in its motion that a quasi static analysis of the beam can be made.

The moment at any point x along the beam is directly proportional to the strain at that point in the beam according to the relationship in (8).

$$M(x) = \frac{\varepsilon(x) IE}{y} \tag{8}$$

By this relationship the moment at any point along the beam can be easily calculated as a constant multiple of the strain polynomial in (1). The moments at the extreme ends of the beam can also be extrapolated in the same fashion.

The free body diagram shown in Figure 3 a, defines the boundary conditions and coordinate frame imposed on the beam. Using the free body diagram shown in Figure 3 b, the following static equations of equilibrium can be determined.

$$\sum F_x = F_{x\mathrm{A}} - F_{x\mathrm{B}} = 0, \Rightarrow F_{x\mathrm{A}} = F_{x\mathrm{B}} = F_x \tag{9}$$

$$\sum F_y = F_{y\mathrm{A}} - F_{y\mathrm{B}} = 0, \Rightarrow F_{y\mathrm{A}} = F_{y\mathrm{B}} = F_y \tag{10}$$

$$\sum M_{\mathrm{A}} = -M_{\mathrm{A}} + M_{\mathrm{B}} - F_y L = 0 \tag{11}$$

In equation (11), L is the foreshortened total length of the beam. By solving for F_y in (11), it can be shown that the y component of the boundary condition force is:

$$F_y = \frac{-M_{\mathrm{A}} + M_{\mathrm{B}}}{L} \tag{12}$$

F_{y} can now be determined by calculating the moments at the endpoints of the beam using the relationship in (8) and substituting them into equation (12).

Finding a relationship for F_{x} will be only slightly more difficult. Making a cut in the frame and drawing a new free body diagram, as in Figure 3 c, we can evaluate the following relationships about the forces and moments internal to the frame.

$$\sum F_x = F_{x\mathrm{A}} - P = 0, \Rightarrow P = F_x \tag{13}$$

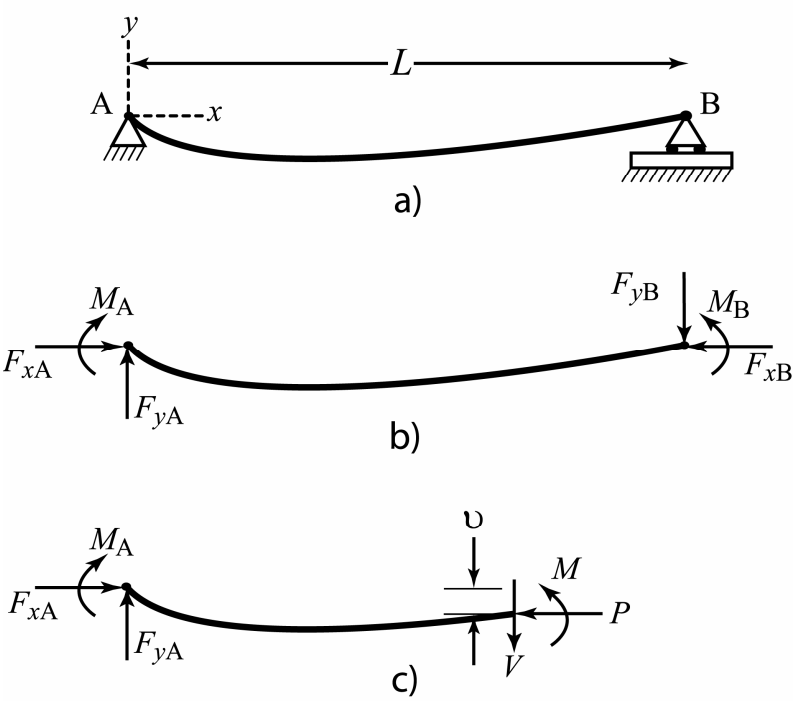

Figure 3. Beam free body diagrams.

$$\sum F_y = F_y - V = 0, \Rightarrow V = F_y \tag{14}$$

$$\sum M_{\mathrm{A}} = -M_{\mathrm{A}} + M - Vx - P\upsilon = 0 \tag{15}$$

Using the relationships for P and V from equations (13) and (14) in equation (15) and solving for F_{x} yields the x component of the boundary condition force.

$$F_x = \frac{M(x) - M_{\mathrm{A}} - F_y x}{\upsilon(x)} \tag{16}$$

The deflection of the beam, $\upsilon(x)$, in (16) can be found by applying the beam integration equations, (4) thru (6), along a subset of the length of the beam x. The result of which will be a relative position vector from one end of the flexible portion of the beam to the arbitrarily chosen midpoint location along the beam. This relative position vector is not expressed in the coordinate frame shown in Figure 3. The relative position vector is expressed in the local frame of the rear axle. In this frame the x-axis is in the direction of the axle's heading which coincides with the beam at its point of attachment to the rear axle. In order to determine the deflection, an appropriate rotation matrix is applied such that the y component of the relative position vector is perpendicular to the line between the attachment points of the beam at either end. After this rotation of the relative position vector the y component will equal the displacement of the beam $\upsilon(x)$. Once $\upsilon(x)$ is known, $M(x)$ and M_A can be calculated from (8) and F_y from (12) which allows F_x to be found from (16).

The ability of the RPS to return endpoint forces and moments is important because the torque required for holding the beam in bending must be provided by wheel traction. The implication of this is that when the robot bends the beam in the course of maneuvering, the traction force available to the wheels is reduced in proportion to the torque required to hold the beam in the bent position. Thus, knowing the forces and moments exerted by the beam on the axles could be important in predicting wheel slip and determining how tight of a turning radius can be afforded over a given terrain. It should also be noted that the ability to estimate these forces and moments is a unique contribution of this research.

Implementation

The implementation of the RPS involved both hardware and software considerations. The electrical requirements include signal amplification, signal conditioning, noise reduction, and signal loss over long transmission wires. The software considerations include: integration step size, and processor time requirements.

The voltage output from the strain bridges is on the order of up to 20mV. This small signal needs to be amplified to take advantage of the Analog to Digital Converter's (ADC) input range of ±10V and thereby limit the measurement error due to the ADC's resolution. The first step in finding a strain bridge amplifier was to check the commercial market to find a suitable unit. The amplifier needed to be small and portable so that it could be placed on the robot and amplify the signal as close to the strain bridges as possible in order to decrease the significance of transmission loss over the 25 foot umbilical cable connecting the robot to the controller. The amplifier also must provide an analog output in the range of the ADC. No commercial package was found that satisfied all of the requirements. Therefore, the decision was made to design and build a custom amplification circuit. A photograph of the amplifying circuit is shown in Figure 4.

The design of strain amplification circuitry begins with the design of a power supply. The robot's motor amplifiers provide a 10V regulated supply but the current is limited to 3mA, which is nowhere near adequate to supply five bridges. The robot also has a 40V high current power supply to the motors. However, no discrete component linear regulator was found that had a maximum input voltage greater than 28V. It is also a concern that the power source used for the Pulse Width Modulation (PWM) amplifiers

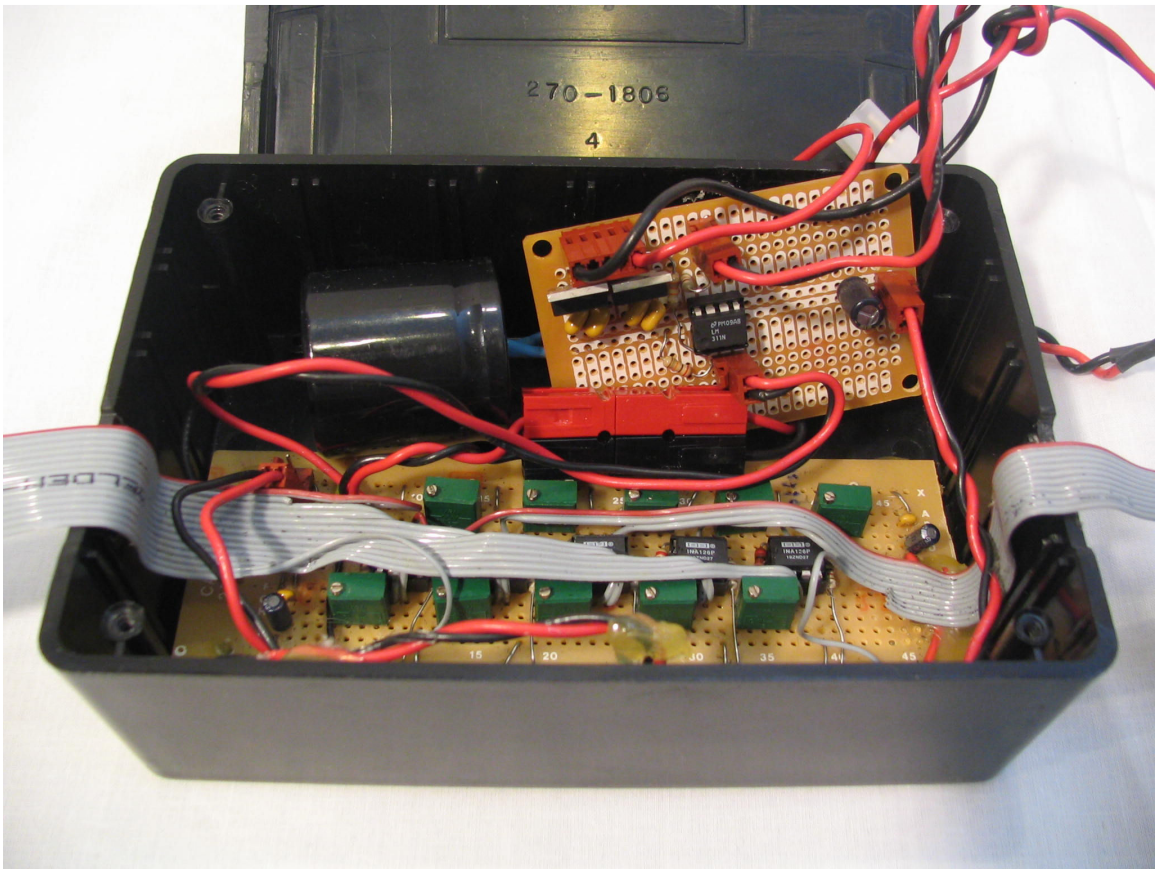

Figure 4. Photograph of the strain bridge amplifying circuit. The small circuit board at the upper right contains the +5V and +10V linear regulators. The top row of potentiometers is used to adjust the reference voltage. The row of integrated circuits is the 5 AD623 differential amplifiers. The bottom row of potentiometers is used to adjust the amplifier gain.

and motors might contain RF noise that would be detrimental to the operation of the sensitive strain bridge sensors.

The solution was to provide the robot with two 7.2V battery packs, of the type used in radio controlled cars, in series to provide a 14.4V clean master power source that is easily regulated to the required 10V and 5V with linear regulators. Both of these batteries were mounted on the rear axle of the robot along with the strain amplification circuitry.

In designing the circuit it was desirable to have an amplifier with a variable gain and the ability to offset the reference voltage in order to zero the bridge. In addition the circuit must be a high quality amplifier with good common mode error rejection qualities. The AD623 instrumentation amplifier was chosen as the centerpiece of the amplification circuit because it provided all of these capabilities in an easy to prototype 8 pin DIP form factor.

In designing the gain required for the amplifying circuit the simplest approach is to set the gain for all five of the amplifying circuits to the same constant value. This approach would provide satisfactory performance if the beam was always bent in a mode that is close to pure bending and all of the bridges report similar strain reading. However, it was desirable to test the RPS in all three bending modes [24] shown in Figure 5. In these more complex bending modes the strain along the length of the beam is not constant. The gain was set so that each bridges amplifier could be tuned to provide an output swing of no more than ±4.5V at the worst case scenarios of each of the three primary bending modes. In addition to gain requirements it was necessary to use the smallest potentiometer possible that would provide the desired gain range. This was needed in order to reduce the sensitivity of the gain adjustment potentiometer thereby making it easier to dial in the appropriate gain and also reduce the systems sensitivity to potentiometer jostling that could change the gain of the system and disturb the calibration of the strain bridge. To this end, the fixed resistor R1 was used in series with the 25 turn trimming potentiometer R2 as shown in Figure 6. This arrangement provides a gain range of 144 to 501

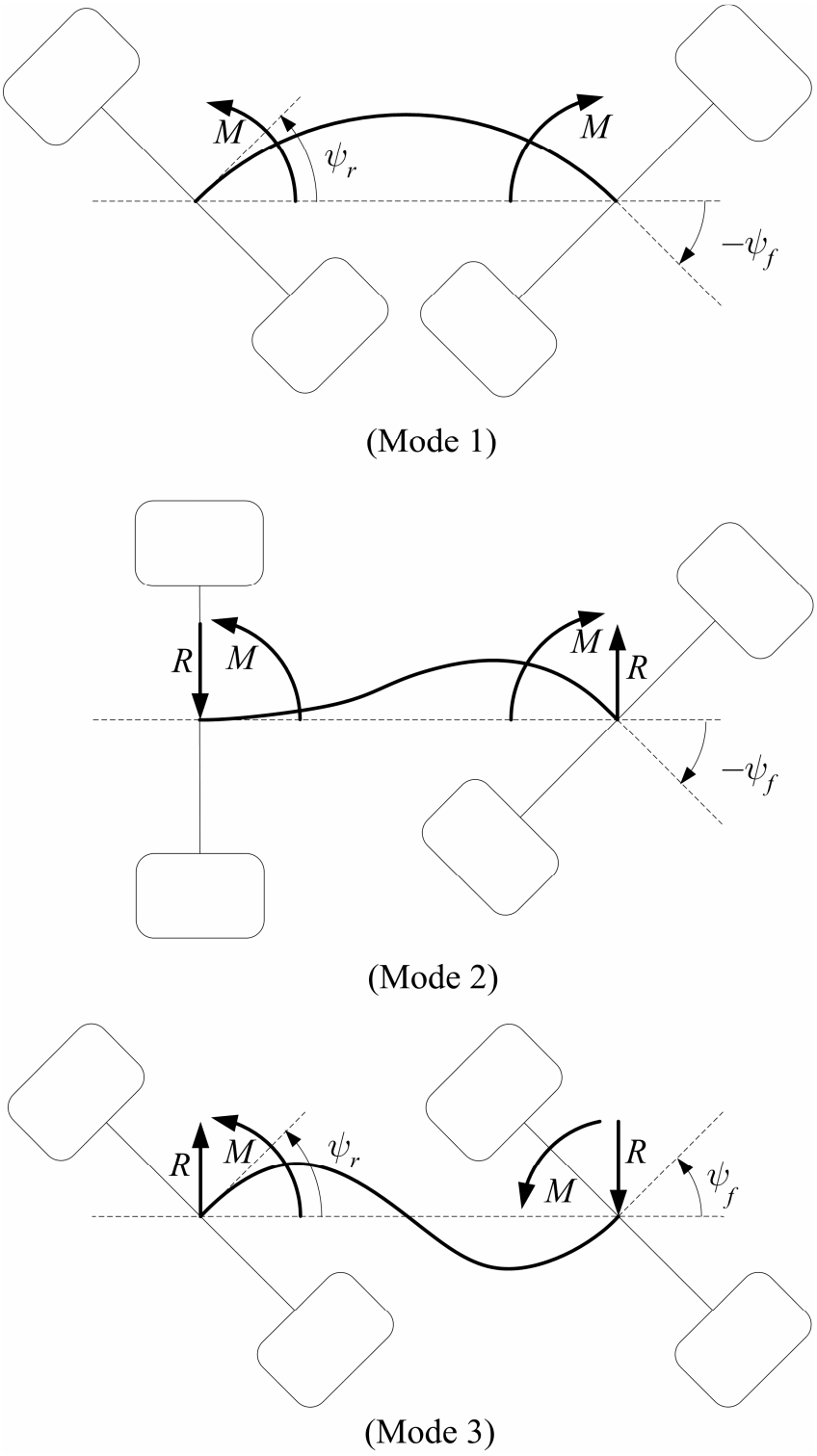

Figure 5. The bending modes of the Flexible Frame.

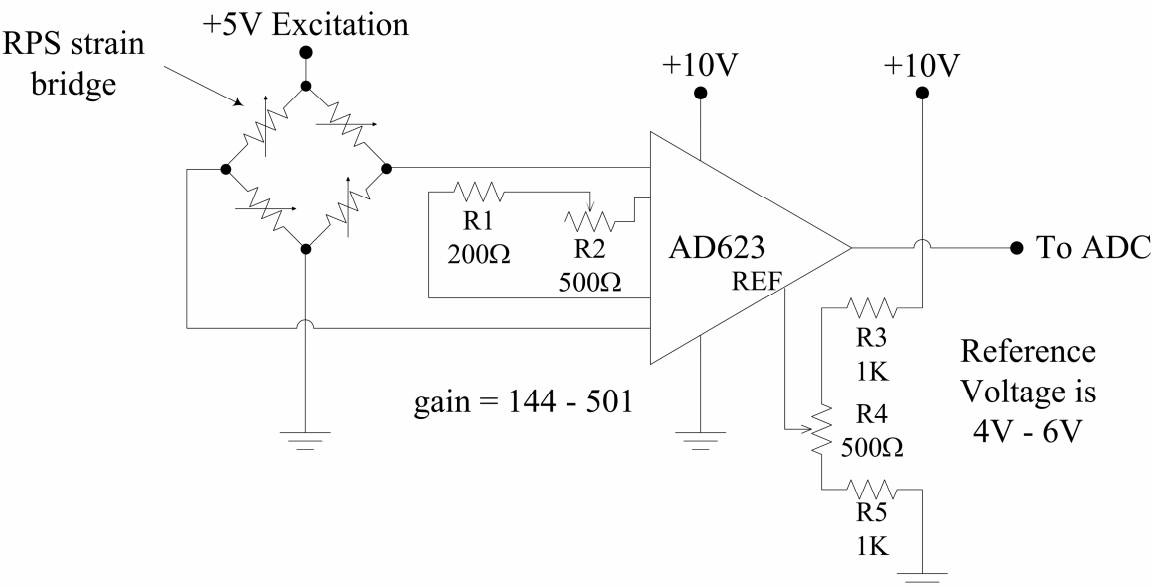

Figure 6. Schematic of a single strain bridge amplifier.

If a ground reference is sent to the AD623 the post amplification signal of ±4.5V will run below the negative input of the 0V to 10V chip supply. Therefore an offsetting reference of ~5V is required. By adding the multiturn potentiometer R4, this reference can also provide the strain bridge zeroing voltage. The reference voltage divider needs to give a large enough voltage swing such that the strain bridge can be zeroed, but, also be as insensitive as possible to provide precise calibration and limit the damage done by unintentional turning of the potentiometer. The sum value of all resistors in the voltage divider should be large to limit current drain on the batteries. The resistors should not be so large as to introduce unnecessary noise into the reference signal. The final design uses the 25 turn 500Ω potentiometer R4 with the center pick as the reference voltage in between R3 and R5 of 1K each. This design produces a reference voltage that can be varied between 4V and 6V.

The strain bridges are located on the beam near the four motors used to drive the robot. This presents a challenge since the sensitive strain sensors are located in a very electrically noisy environment. While the motor noise could not be entirely eliminated, it was reduced significantly by using shielded cabling and locating the strain amplifier as close to the sensors as possible given the space constraints on the robot.

In addition to the electrical design, the software algorithms presented their own set of challenges. The RPS algorithm involves a piecewise integration along the length of the beam that requires the beam to be split into n segments. The number of steps n used in the RPS algorithm must be large enough to provide an accurate integration. On the other hand, a given robot controller must operate at some minimum speed for optimal stability and a large n could adversely affect the time step of the system. In order to achieve the best possible trade off between accuracy and system speed, experiments were conducted to determine the minimum required step size in the path wise integration of equations (4) thru (6) and to develop a relationship between the value of the number of integration steps n and the processor time required to implement the RPS algorithm at the chosen number of steps. It should be noted that algorithm speed experiments are processor specific, and the results should not be used if the robot is controlled by a processor other than the dSpace 1103 real time processor used in all of the experiments in this thesis.

For experiments involving the RPS, a test stand similar in dimension to the RPS frame module was constructed, see Figure 7. It consists of a pinned connection at each end of the beam, with a linear bearing at one end to allow changes in separation distance

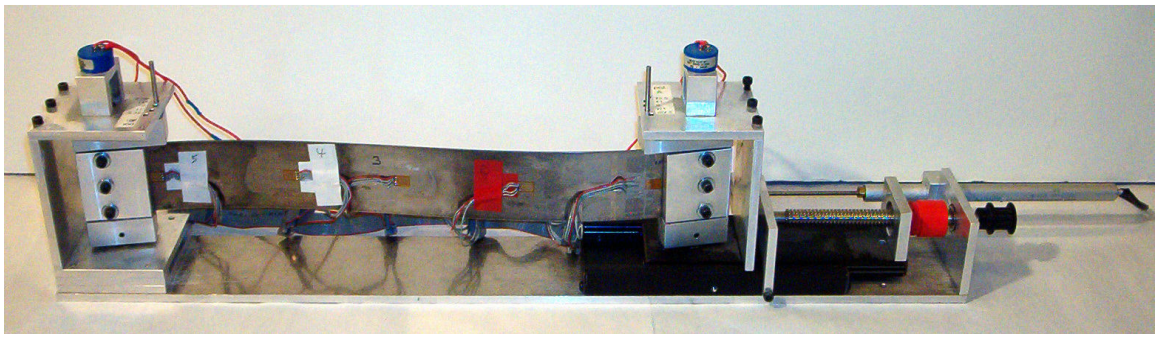

Figure 7. The FFMMR showing the strain gauges of the RPS.

between the pivots. The angular and prismatic deflection of the endpoints was instrumented via a 16bit data acquisition system and highly linear potentiometers.

The first experiment was to determine the effect of changing the number of integration steps, *n*, on the accuracy of the RPS algorithm. With the beam deflected, 10 seconds of calibrated strain data along with the corresponding relative position output from the fixture potentiometers was recorded. This strain data was then used in the RPS algorithm with the resulting relative pose estimate being subtracted from the relative pose measurement returned by the fixture potentiometers to arrive at the error in the RPS algorithm pose estimate. This process was repeated a number of times with n being set to the values on the interval of 1 thru 20, 100, 1000, and 10,000. This process was then repeated for each bending mode. Figure 8, Figure 9, and Figure 10 show the results.

In each of the charts in Figure 8 thru Figure 10 mode 1 refers to the case where the beam was deflected in pure bending to the extreme positions allowed by the beam where the A end was pinned at 22.5° and the B end at -22.5°. For mode 2 the boundary conditions were A at 0° and B at 22.5°. For mode 3 A and B were pinned to 22.5°. See

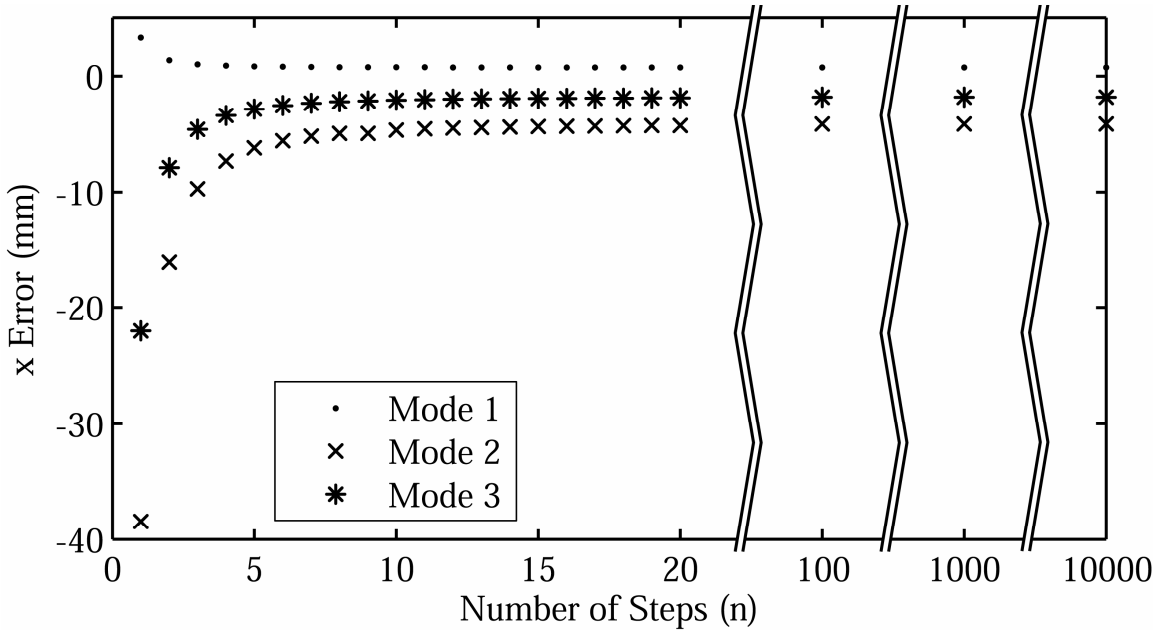

Figure 8. Error in the x component of the RPS integration algorithm as the number of integration steps is increased from 1 to 10000

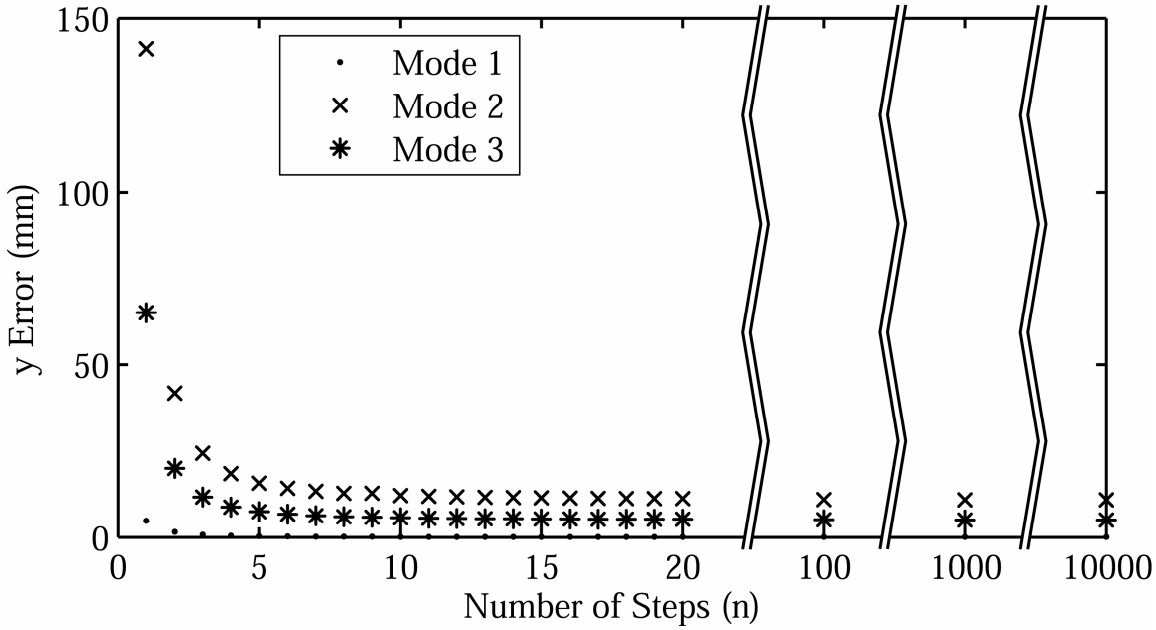

Figure 9. Error in the y component of the RPS integration algorithm as the number of integration steps is increased from 1 to 10000

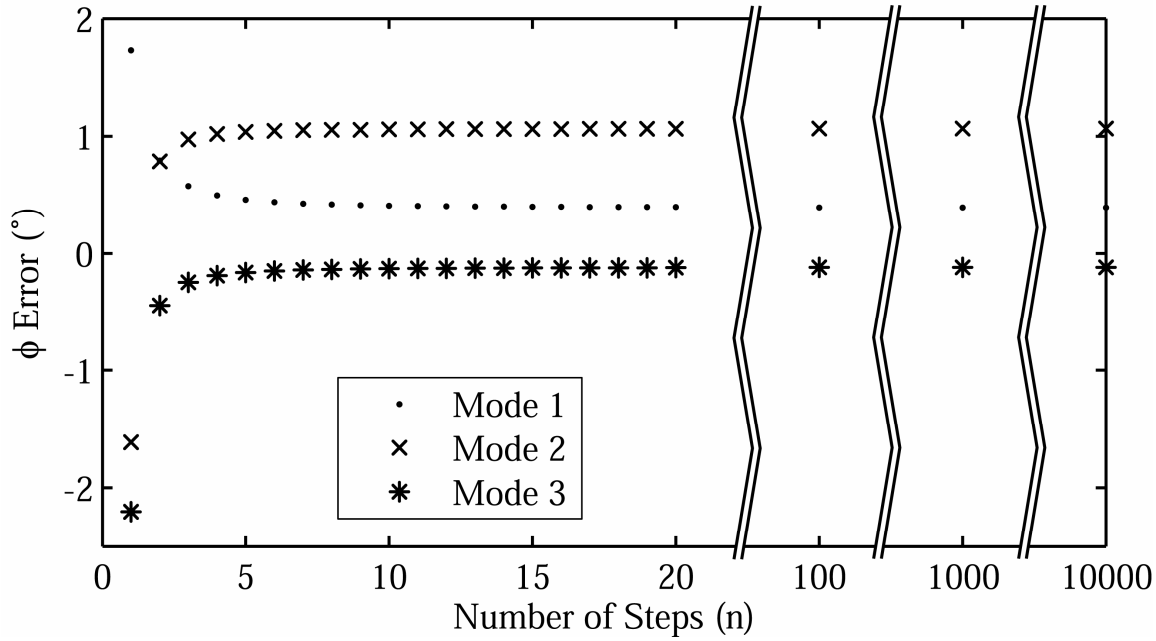

Figure 10. Error in the ϕ component of the RPS integration algorithm as the number of integration steps is increased from 1 to 10000

for a schematic of the fixture showing the location and orientation of the A and B ends of the beam. All of the n versus error plots follow a similar pattern with a very rapid drop off in error for n = 1 to 5 followed by a very slow decrease in error thereafter. Based on this information a minimum n of 5 should be used.

The second experiment involved increasing the number of integration steps *n* for a given time step until the algorithm can no longer be run in real time. This process is repeated at several time steps to determine the relationship between *n* and the processor time required by the RPS. The results of this test are shown in Figure 12. The solid line intersecting the data points is a linear curve fit. The equation corresponding to the curve fit is shown in equation (17).

$$\text{time (ms)} = 0.0025n + 0.018 \tag{17}$$

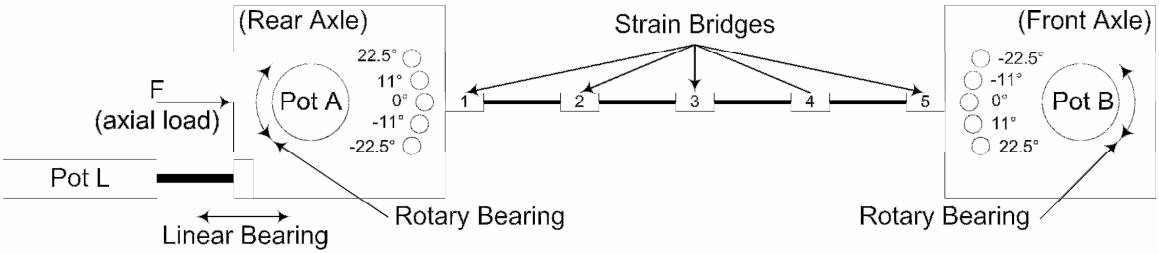

Figure 11. RPS test fixture schematic (top view)

In Figure 12 the 1ms mark represents the desired time step for the robot controller. If the processing time required by the robot controller, the axle EKF's, and the CI filter is known then the upper bound of n can be estimated with the graphic in Figure 12.

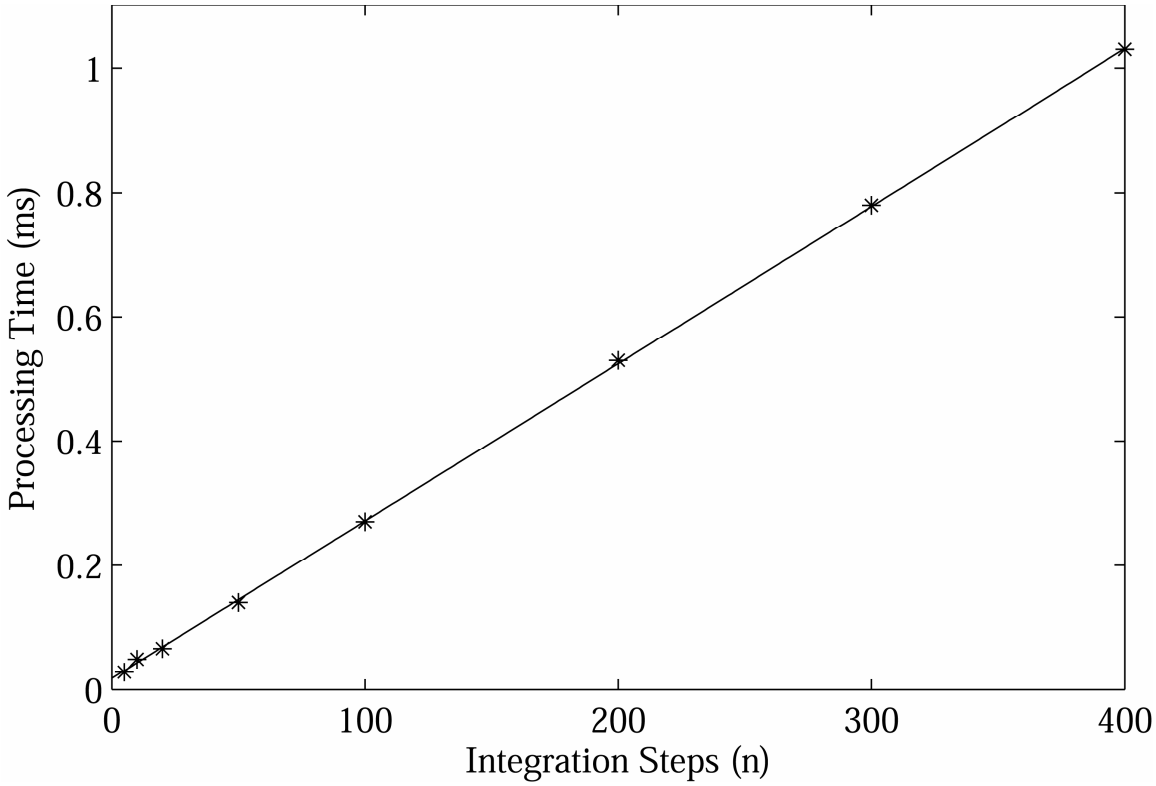

Figure 12. The RPS processing time required as n is varied.

4. AXLE LEVEL DATA FUSION

In the algorithm proposed here, each axle module fuses the data from its wheel encoders and gyroscope, using a kinematic model based Extended Kalman Filter[25]. A Kalman filter is an algorithm that produces optimal estimates of a linear system's states with access only to current and past measurements. This estimate is optimal in the least squares sense. The Extended Kalman Filter is an extension of the Kalman filter to a nonlinear system and provides sub optimal state estimation. The purpose of the Extended Kalman Filter, in this application, is to fuse the data from the axle level position sensors with the state predictions of the system model in order to provide, as near as possible with a nonlinear system, optimal estimation of the true system states.

The EKF propagates two vital pieces of information; the system states x and the covariance matrix P. The systems states are the important parameters of the system, such as positions, velocities, and accelerations. The covariance matrix represents the uncertainty associated with the EKF estimates of the system states as well as the correlations between the states. The EKF has two primary components, the time update and the measurement update. In general terms the time update is a prediction step. The state estimate from the previous time step is used to predict the state and covariance values at the current time step. The measurement update, again in very general terms, involves 3 steps. The first step uses the previously computed prediction of the covariance

to calculate a Kalman gain equation K. The gain matrix is used with the state prediction and the current measurements to calculate the final state estimate for the time step. Lastly the gain matrix and the predicted covariance are used to calculate the final covariance estimate for the time step. The Kalman filter requires a state model, which includes the uncertainties inherent to the model. The general form of this model is shown in (18).

$$x_k = a\left(x_{k-1}, u_{k-1}, w_{k-1}, T\right) \text{ where } w_{k-1} \sim N\left(0, q_{k-1}\right) \tag{18}$$

The filter also requires a measurement model including the noise inherent to the sensors. The general form of this model is shown in (19).

$$z_k = h\left(x_k, v_k, T\right) \text{ where: } v_k \sim N\left(0, r_k\right) \tag{19}$$

In (18) u is the input to the system. In (18) and (19) x is the system states, w and v are process noise variables, and the $N\left(i, j\right)$ notation means that u and v are normal distributions with a mean of i and a variance of j. The Kalman filter also works under the assumption that u and v are uncorrelated.

The time update and measurement update of the EKF implementation and their corresponding matrix equations are embodied in (20) thru (24). In the following paragraphs each of the five EKF equations and their associated parameters will be explained in detail.

The time update equations.

$$\hat{x}_k^- = a\left(\hat{x}_{k-1}, 0, 0, T\right) \tag{20}$$

$$P_k^- = AP_{k-1}A^T + Q \tag{21}$$

The measurement update equations.

$$K_k = P_k^- H^T \left(HP_k^- H^T + R\right)^{-1} \tag{22}$$

$$\hat{x}_k = \hat{x}_k^- + K_k\left(z_k - H\hat{x}_k^-\right) \tag{23}$$

$$P_k = \left(I - K_k H\right)P_k^- \tag{24}$$

In equation (20) x is the states of the system. The "^" means that it is an estimate of the true states. The "¯" indicates that it is a prediction of the states at the current time step. The subscript $_k$ indicates the current time step and $_{k\text{-}1}$ indicates the previous time step. The function a, is the nonlinear model of the system. T is the time step used. During the time update the EKF output states from the previous time step are used in the nonlinear state equations to predict the current value of the system states. This can be thought of as a system observer or a virtual sensor that predicts the value of each state at

the start of the filter cycle. In this implementation the 7 axle states shown in equation (25) are maintained by the EKF.

$$x = \begin{bmatrix} x & y & \phi & V & \dot{\phi} & \dot{V} & \ddot{\phi} \end{bmatrix}^T \tag{25}$$

In (25) x, y, and ϕ are the axle's global position and orientation. V and $\dot{\phi}$ are the translational and rotational velocity of the axle. The two remaining states are the translational and rotational acceleration of the axle. See Figure 13 for a schematic of the axle states.

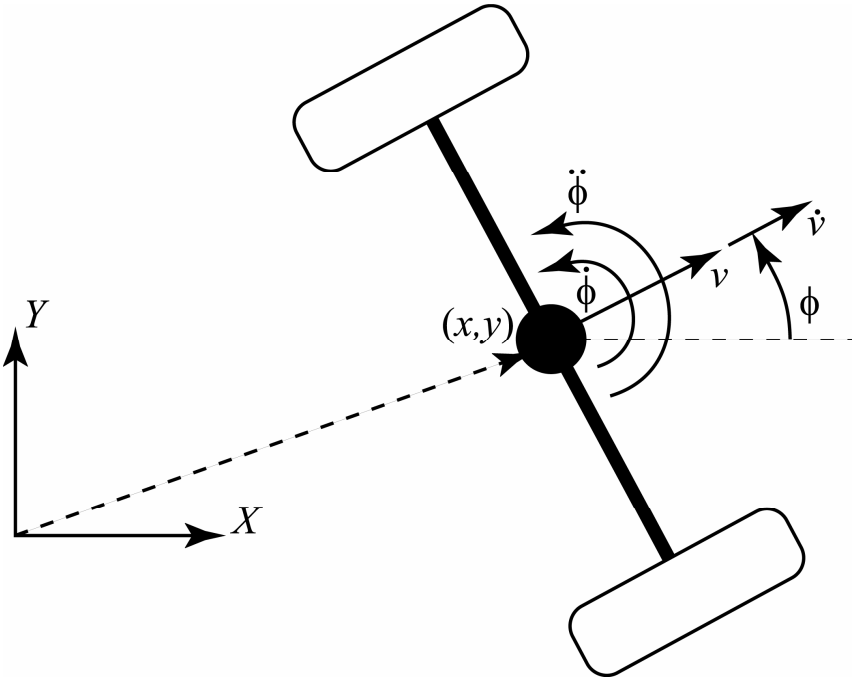

Figure 13. The seven single axle EKF states.

The nonlinear system of equations used to model the system is shown in (26).

$$\begin{bmatrix} x_k \\ y_k \\ \phi_k \\ v_k \\ \dot{\phi}_k \\ \dot{v}_k \\ \ddot{\phi}_k \end{bmatrix} = a\left(x_{k-1}, 0, 0, T\right) = \begin{bmatrix} x_{k-1} \\ y_{k-1} \\ \phi_{k-1} \\ V_{k-1} \\ \dot{\phi}_{k-1} \\ \dot{V}_{k-1} \\ \ddot{\phi}_{k-1} \end{bmatrix} + \begin{bmatrix} \left(V_{k-1}T + \frac{\dot{V}_{k-1}T^2}{2}\right)\cos\left(\phi_{k-1} + \frac{1}{2}\left(\dot{\phi}_{k-1}T + \frac{\ddot{\phi}_{k-1}T^2}{2}\right)\right) \\ \left(V_{k-1}T + \frac{\dot{V}_{k-1}T^2}{2}\right)\sin\left(\phi_{k-1} + \frac{1}{2}\left(\dot{\phi}_{k-1}T + \frac{\ddot{\phi}_{k-1}T^2}{2}\right)\right) \\ \dot{\phi}_{k-1}T + \frac{\ddot{\phi}_{k-1}T^2}{2} \\ \dot{V}_{k-1}T \\ \ddot{\phi}_{k-1}T \\ 0 \\ 0 \end{bmatrix} \quad (26)$$

This model is similar to the kinematic model found in [25], the difference being that (26) employs constant acceleration states instead of constant velocities. The two velocity terms in (26) are simply acceleration X time applied to both the translational and rotational velocity terms. A figure will help with the derivation of the posture states x, y and ϕ. Figure 14 shows the path traveled by a single axle over one time step. The length of the path and the change in the axle's orientation over a time step is greatly exaggerated for increased clarity in deriving the kinematic model in (26). The primary assumption employed in the derivation is that the time step is chosen short enough relative to the working velocities of the axle that the difference in length between the straight line vector δs' and the path the axle actually traveled δs is negligible. Given this assumption the length of $\delta s'$ can be calculated from (27).

$$\delta s' = \dot{v}_{k-1}T + \frac{\ddot{v}_{k-1}T^2}{2} \quad (27)$$

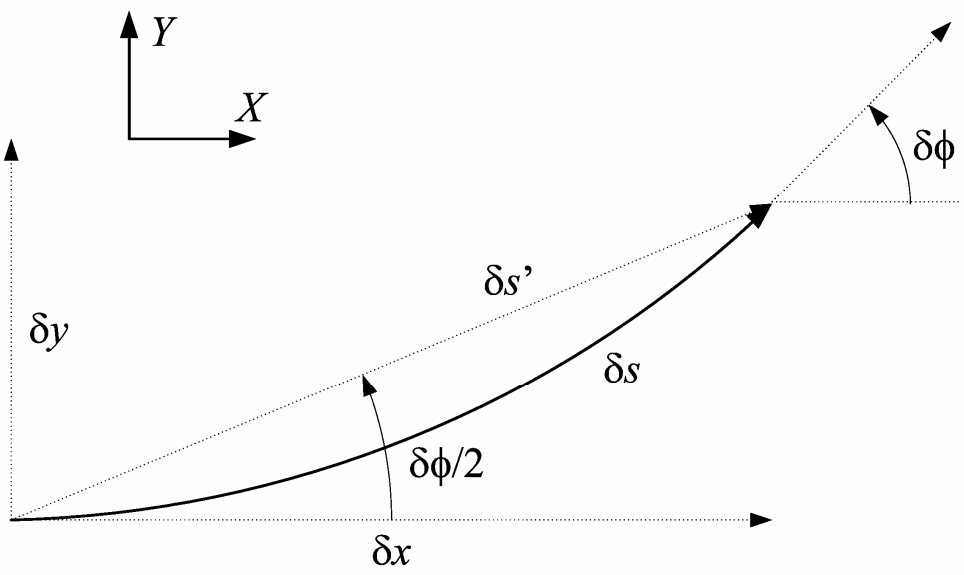

Figure 14. The kinematics of axle motion over a single time step.

The change in orientation of the axle can be predicted from the angular velocity and acceleration terms calculated by the EKF in the previous time step as in (28).

$$\delta\phi_k = \dot{\phi}_{k-1}T + \frac{\ddot{\phi}_{k-1}T}{2} \tag{28}$$

If the assumption that δs is a circular arc holds then the angle between the global x axis and the line δs' can be calculated by (29).

$$\phi_{k-1} + \frac{1}{2}\left(\dot{\phi}_{k-1}T + \frac{\ddot{\phi}_{k-1}T^2}{2}\right) \tag{29}$$

δx and δy can now be predicted as the x and y components of the change in position vector δs' (30).

$$\begin{aligned}\delta x &= \left(v_{k-1}T + \frac{\dot{v}_{k-1}T^2}{2}\right)\cos\left(\phi_{k-1} + \frac{1}{2}\left(\dot{\phi}_{k-1}T + \frac{\ddot{\phi}_{k-1}T^2}{2}\right)\right) \\ \delta y &= \left(v_{k-1}T + \frac{\dot{v}_{k-1}T^2}{2}\right)\sin\left(\phi_{k-1} + \frac{1}{2}\left(\dot{\phi}_{k-1}T + \frac{\ddot{\phi}_{k-1}T^2}{2}\right)\right)\end{aligned} \tag{30}$$

The system of equations in (26) makes some assumptions. The primary assumption is that there is no wheel slip. For this research no attempt has been made to model wheel slip directly at this level. The RPS is the primary wheel slip detector and corrector. Another assumption is that the time step T, is sufficiently small that any changes in the states during a single time step can be neglected. This is required so that the velocities and accelerations used to calculate the position and velocity states in (26) can be considered the average over the interval T. Another item of interest that merits explanation is the constant acceleration component of (26). The acceleration is a function of the motor inputs and the dynamics of the system. For the purposes of this EKF this part of the system was deemed too complex to model. Practically speaking the robot is slow and the system velocities are directly measured by the wheel encoders and gyroscope. These facts make the acceleration terms less critical. The important point is that much of the model error will come from the combination of two sources. First, the unmodeled wheel slip, and second, the constant acceleration terms in the model.

Model error is important in the implementation of equation (21). The purpose of this equation is to predict the covariance matrix, P^-, associated with the states of the system. The superscript "$^-$" denotes this as a prediction and not the final output of the filter. A represents the linearized system of equations.

$$A = \left.\frac{\partial a}{\partial x_k}\right|_{x_k = \hat{x}_k, w_k = 0} \tag{31}$$

Q is the covariance associated with the uncertainties in the state model a. Getting back to the wheel slip discussion in the previous paragraph; the unmodeled wheel slip is not directly dealt with in this research, but, the constant acceleration terms in (26) can be used to advantage by proper tuning of the Q matrix. The idea is that when the robot encounters a slick spot and one of the wheels suddenly slips, the affected wheel will often experience a dramatic acceleration. If this acceleration can be shown to be significantly greater than the normal system accelerations then it can be attenuated to some degree by the constant acceleration model in (26). This is done by adjusting the two acceleration components of the Q matrix so that the speed with which a change in acceleration is incorporated into the system states is fast enough not to affect the normal system accelerations but slow enough to dampen the quick accelerations characteristic of a slipping wheel. The Q matrix actually used is shown in (32).

$$Q = \text{diagonal}\begin{bmatrix} 0.001 & 0.001 & 0.001 & 0.1 & 0.1 & 10 & 10 \end{bmatrix} \tag{32}$$

The P matrix is initially set to a diagonal matrix where all of the diagonal terms are 0.001. This initial matrix then quickly converges to the actual variances of the system. This matrix is updated at the end of both the time and measurement updates. Although not currently implemented, the P matrix could be used as a guide for how close the robot can get to obstacles in its path without running the risk of actually hitting them.

Following the Time Update is the Measurement Update. The Measurement Update consists of equations (22) thru (24). The purpose of (22) is to calculate the Kalman gain matrix *K*. The K matrix is an m X n matrix where the number of columns is equal to the number of sensor inputs and the number of rows is equal to the number of states. The K matrix is a measure of the relative difference in trust between the time update state prediction and the measurements. It also allows a mapping of the measurements to each of the system states. It is by the K matrix in equation (23) that the EKF can estimate the acceleration states, even though the model assumes constant acceleration and the acceleration states are not directly measured. The H matrix is a mapping of the states to the measurements and R_k is the covariance matrix associated with the measurements z_k. *H* can be found by taking the Jacobian of the measurement model *h* as in (33).

$$H = \left.\frac{\partial h}{\partial x_k}\right|_{x_k = \hat{x}_k, v_k = 0} \tag{33}$$

The measurements include the left and right encoder counts, and the gyroscope voltage. The H matrices used for the front and rear axles respectively are shown in (34) and (35)

$$H = \begin{bmatrix} 0 & 0 & 0 & \frac{C_E T}{2\pi r} & \frac{C_E BT}{2\pi r} & 0 & 0 \\ 0 & 0 & 0 & \frac{C_E T}{2\pi r} & \frac{-C_E BT}{2\pi r} & 0 & 0 \end{bmatrix} \tag{34}$$

$$H = \begin{bmatrix} 0 & 0 & 0 & \frac{C_E T}{2\pi r} & \frac{C_E BT}{2\pi r} & 0 & 0 \\ 0 & 0 & 0 & \frac{C_E T}{2\pi r} & \frac{-C_E BT}{2\pi r} & 0 & 0 \\ 0 & 0 & 0 & 0 & C_G & 0 & 0 \end{bmatrix} \tag{35}$$

C_E is the number of quadrature encoder counts in a single revolution of a wheel, r is the radius of the tire, B is the robot's wheel base, and C_G is the inverse gyroscope output to rotational velocity calibration coefficient. Table 2 contains the actual values used in the H matrix.

In this implementation the measurements are assumed to be independent and as such R is a diagonal matrix with the variance of each sensor along the diagonal as in (36).

$$R = \text{diagonal}\left(0.005, 0.005, 0.005\right) \tag{36}$$

Table 2

Parameter values used in the H matrix

Variable	C_E	T	r	B	C_G
Value	1024 counts	0.01 s	0.073 m	0.343 m	1.39 rad/s/V

The purpose of (23) is to calculate the final state estimate for the current time step. The $H\hat{x}_k^-$ term is a mapping of the states to the measurements and amounts to a prediction of the measurements based on the state prediction calculated in (20). The term in parenthesis, $\left(z_k - H\hat{x}_k^-\right)$ is known as the residual and is the difference between the predicted and actual measurements. The change in states for the time step is calculated by multiplying the residual by the Kalman gain *K*.

The purpose of the final EKF equation (24) is to calculate the final value of the state covariance matrix P. This is a function of *K*, *H* and P^-.

5. CI UPDATE

The axle level EKF's do not consider any data from the RPS. In order to bring together all of the data sources on the robot, a 2nd tier of data fusion is needed, see Figure 15. Bringing the RPS into the data fusion equation is complicated by the fact that the RPS has no new information in terms of the global coordinate frame; it is merely an observation of the relative pose (x, y, and ϕ) of the two axles. As such, if the beam data is fused with the axle data in such a way as to decrease the variance of the axle pose, as in a traditional Kalman filter, then such an estimate is nonconservative. In other words it purports to improve the estimate of the axle posture – because it decreases the variance associated with it – but, in fact, it cannot do this because the beam merely observes the relative position and orientation of the two axles and knows nothing about the absolute posture of either axle in the global coordinate frame.

The solution to this problem can be found in the Covariance Intersection filter strategy. This filter does not produce nonconservative estimates from data with unknown correlations [9]. Simulations were conducted to verify the superiority of the CI filter when compared to a Kalman filter. The simulation consisted of two axles connected by an RPS. The axles used an EKF to estimate the axle states with the covariance matrix initialized to a diagonal matrix with the value 0.01 at each diagonal element. The axles were configured with wheel encoders only, no gyroscopes. All states for both axles were

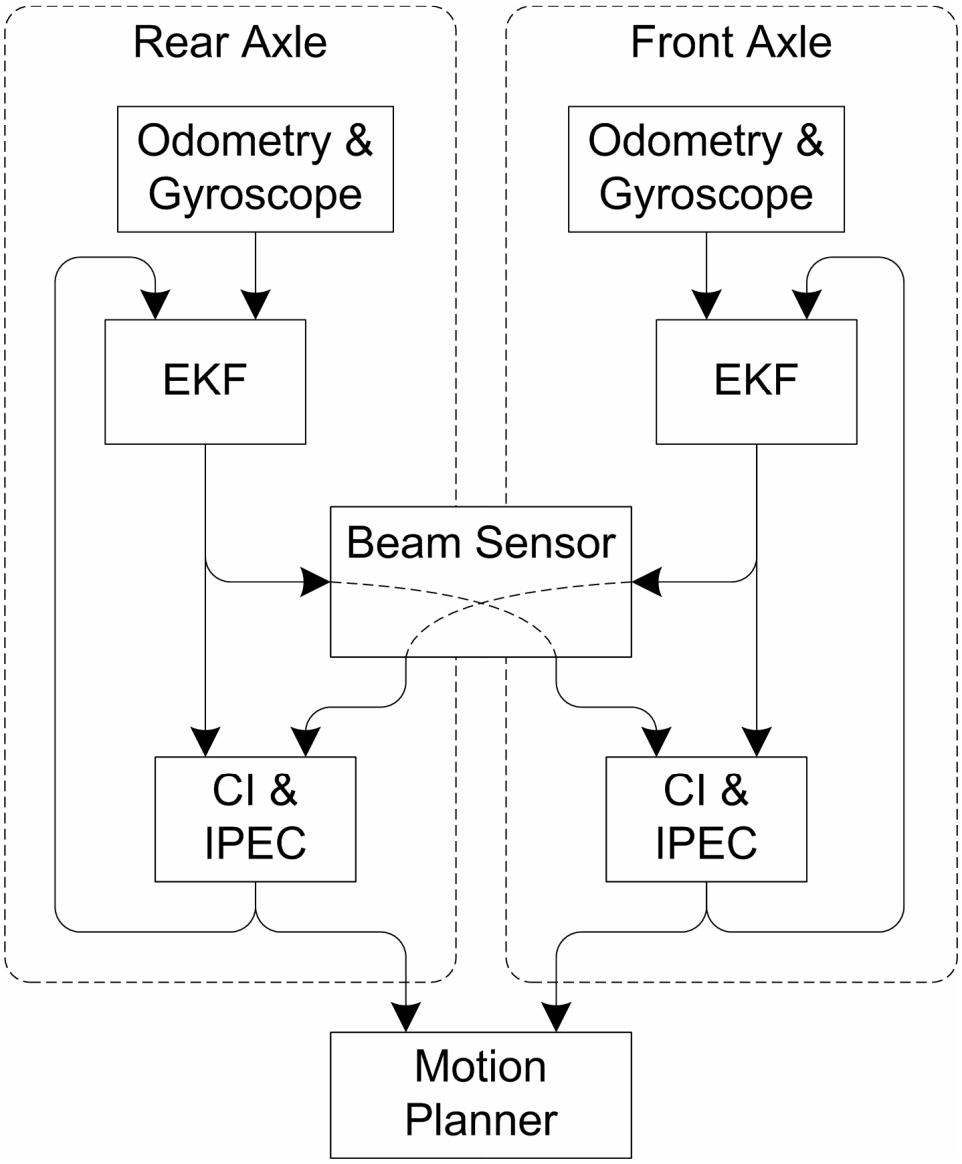

Figure 15. The tiered data fusion structure

initialized to zero. The simulated sensors, (wheel encoders, and RPS strain bridges), were all set to report zero with the addition of a small amount of zero mean normally distributed noise. The contradictory state of both axles initially being at the origin should be corrected by the top tier data fusion algorithm. Two simulations were conducted. The first used a CI filter as the top tier data fusion device and the second used a Kalman filter.

Figure 16 is a simulation of the robot in which a CI filter is used as the top tier data fusion method. Even though both axles are initialized to zero, the CI filter quickly converges and adjusts the *x* position of both axles such that they are separated by the distance of the beam apart. The lines above and below the front and rear axle positions are the *x* location plus and minus a standard deviation that has been calculated from the covariance matrix. The standard deviations never drop below their initial values, but

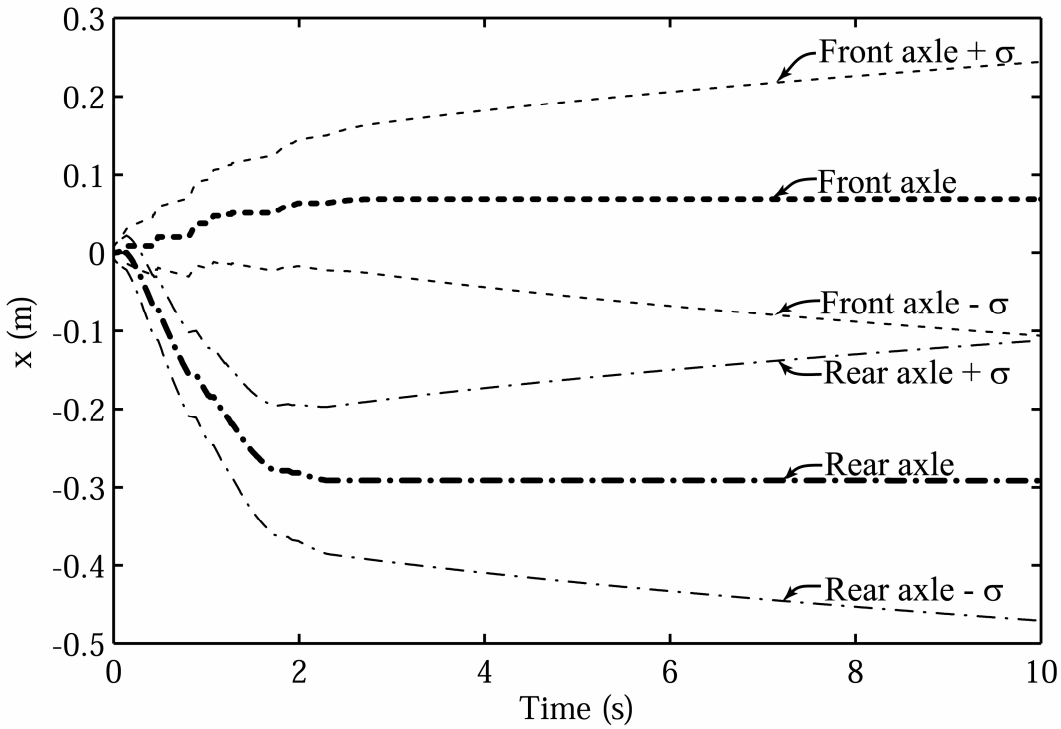

Figure 16. Using CI as the top tier data fusion method.

steadily increase, as is expected in the absence of a direct and independent measurement of the *x* position state.

In contrast, Figure 17 shows the first six hundredths of a second of a simulation using the same initial conditions with a Kalman filter as the top tier data fusion method. As can be seen the initial covariances are very small. In addition, Figure 18 shows the covariances remain so small that they are indistinguishable from the mean axle position for the duration of the simulation. Because the Kalman filter believes the RPS and encoder measurements to be independent measurements with respect to the global coordinate frame the covariances are slashed with each measurement update, causing the filter to ignore new measurement input at the axle level. This behavior results in the

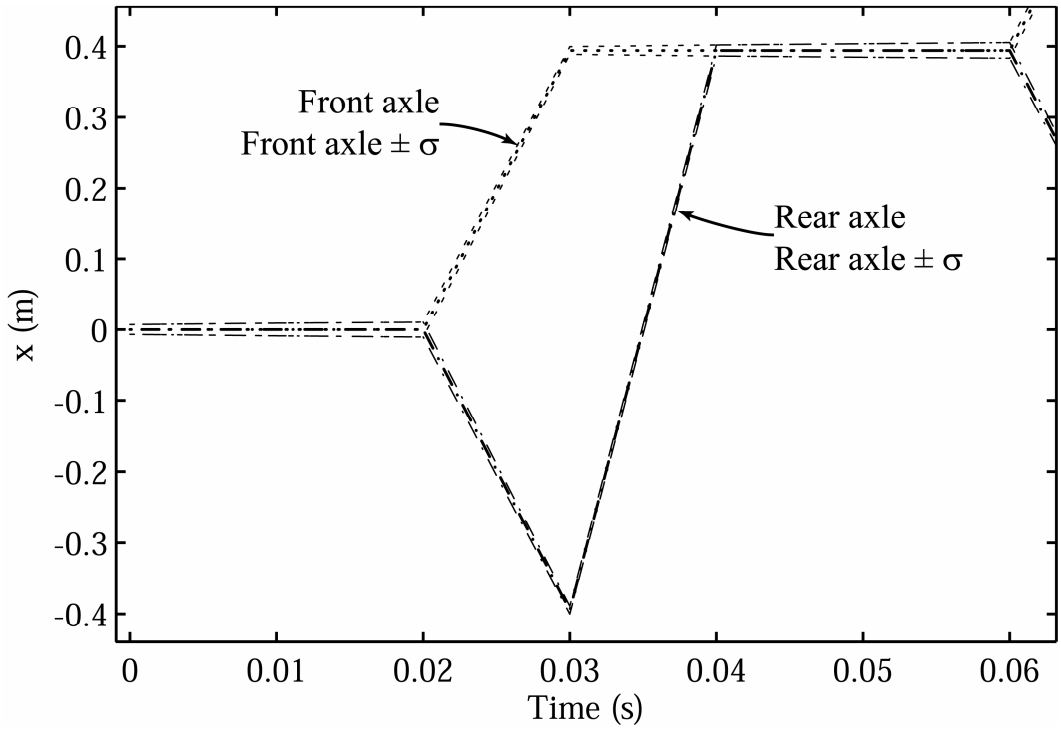

Figure 17. Detail of using a Kalman filter as the top tier data fusion method.

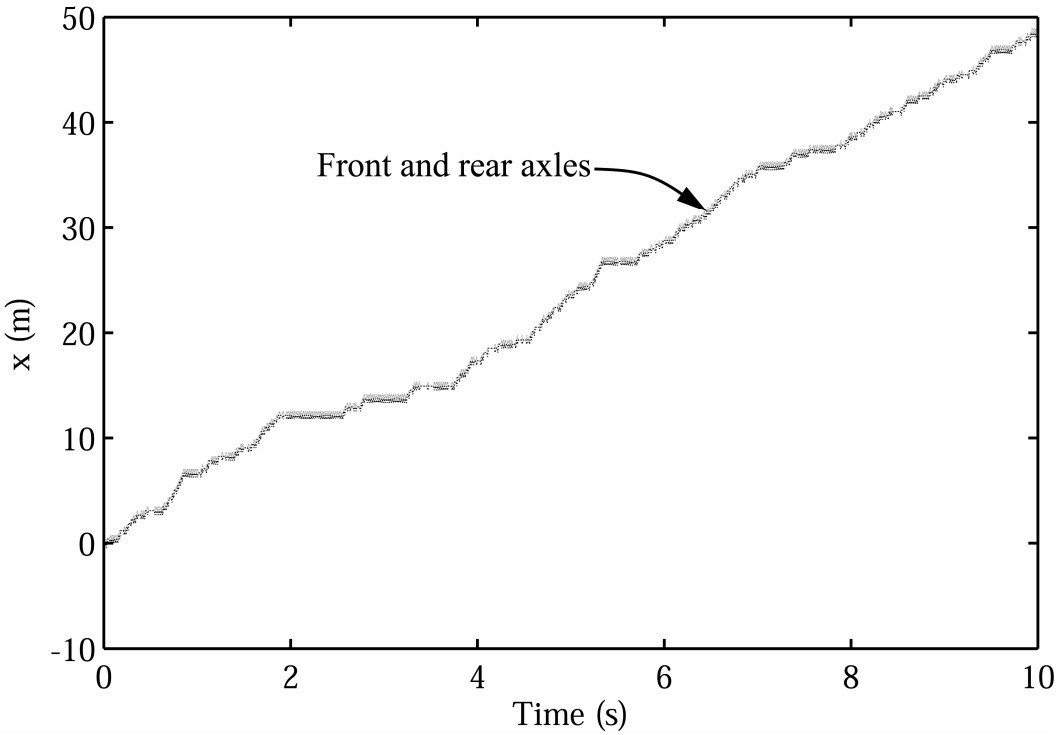

Figure 18. Using a Kalman Filter as the top tier data fusion method.

rapid divergence of the filter shown in Figure 18. This is clearly undesirable and conclusively shows the superiority of the CI filter for this application.

The Covariance Intersection (CI) algorithm is stated as,

$$C = \left(wA^{-1} + (1-w) B^{-1} \right)^{-1}$$
$$c = C\left(wA^{-1}a + (1-w) B^{-1}b \right) \tag{37}$$

where $\{a, A\}$ and $\{b, B\}$ represent the {mean, variance} associated with datasets, and $\{c, C\}$ likewise represents the fusion of $\{a, A\}$ and $\{b, B\}$. The weighting factor, w, is chosen to minimize a parameter of the resulting covariance matrix such as the

determinant or the trace. In this application w is chosen to minimize the trace. Thus the CI filter provides us with the ability to use the frame module to reduce odometric error and maintain consistent covariance estimates.

In calculating the *w* parameter for the CI filter both accuracy and system limitations had to be taken into account. It was desired to limit the number of calculations so as not to impact system performance. Experimentation revealed that for this system *w* often changes very rapidly. To accommodate this fact a three part hybrid approach is used to estimate *w*. In this approach *C* is calculated a maximum of 10 times for each time step, with the value of w that minimizes the trace of *C* being used as the final result. The first six calculations are a brute force approach in which *C* is calculated at *w* values of 0, 0.2, 0.4 0.6, 0.8, and 1. This is followed by a more dynamic approach in which *C* is calculated at the *w* used in the previous time step and at two additional points a small $\pm\delta w$ on either side of the *w* used in the previous time step. The final calculation is made by constructing a polynomial from three of the previously calculated values of *w* that meet the criteria of being adjacent, and producing the smallest values of the trace of *C* (assuming that they bracket the minimum trace). By taking the derivative of the resulting polynomial and setting it equal to zero, the *w* that minimizes the polynomial can be found. The *w* that is finally used in the output of the CI filter is the *w* from the previous ten calculations that produces the smallest value of the trace of *C*.

The state vector used in the CI filter structure was composed of a single axle's position and orientation states (*x*, *y*, and ϕ), since these are the only states that the RPS has the capacity to distinguish.

Immediately after the axle level EKF, the posture of the front axle relative to the rear axle is calculated from the strain gauge outputs on the RPS. This vector can then be added to the pose vector of the rear axle that was just returned by the Kalman filter to produce an estimate of the front axle. Likewise, a pose estimate of the rear axle can be obtained by subtracting the relative position vector of the beam from the front axle. These two new position estimates can then be fused with the pose estimates generated by the Kalman filter using the CI fusion algorithm. In order to more accurately correct for odometry errors, an adaptation of Borenstein's IPEC algorithm is also used to limit the CI update to correct only the axle that has the least accurate odometry.

This fusion scheme is readily expandable to robot configurations that have multiple axles and beam elements. In this situation, every axle would estimate its pose using a Kalman filter and the sensors it has on board. Then, the IPEC algorithm is used to determine if any of the RPS outputs from neighboring axles is considered trustworthy. If so, then any number of trustworthy pose estimates extrapolated from adjoining axles can be fused together with the pose estimate from the axle's native sensor suite. This has the advantage that different axles can carry different sensors, yet all of the axles can benefit from the pose information provided. For example, if only one axle in a chain is equipped with a GPS, every axle will benefit from the absolute pose information returned by the GPS as the sensor data is propagated from one axle to the next via the integrated RPS sensors linking each axle.

After application of the CI filter, the relative error in position between one axle and its neighbors is limited to the resolution of the beam as a sensor. This is true regardless of the amount of drift associated with the sensors on each individual axle. This property

is essential if the robot is to carry out any but the briefest laboratory maneuvers. Without the RPS in the loop, the relative error between the axles would drift apart until the motion controller could no longer function.

6. STATIC TESTING OF THE RPS

The static testing of the beam was conducted on the fixture described in the implementation sub section of Chapter 3, 'The Flexible Frame Module as a Sensor'. For the static testing, the flexible portion of the beam itself was 0.3464 m in length with strain gauges mounted at (0.0090, 0.1736, 0.3380) m along the beam. At each end, the beam was cantilevered to the pivots to emulate the actual axles of the robot. A string was attached to the side of the fixture that allows prismatic motion. This string was stretched across the work bench and over a pulley. By hanging weights on the string a force could be applied to the beam along the line between the center point of both axles, in order to emulate the situation of nonzero F_x. See Figure 19 for a schematic of the test setup. The endpoints of the beam were then deflected -22.5°, -11°, 0°, 11°, and 22.5°. At each of

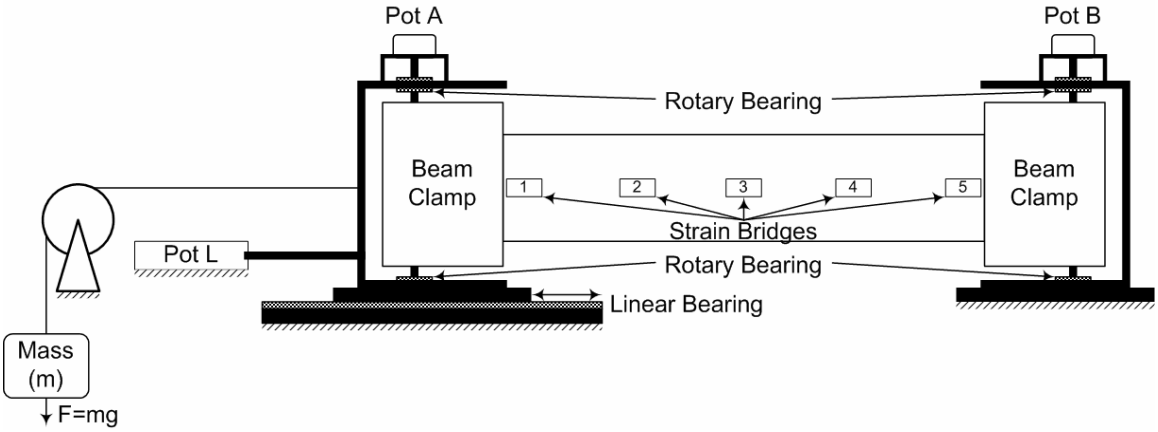

Figure 19. RPS test fixture schematic (side view)

these deflections, longitudinal loads of 0 N, 1.0 N, 2.0 N, and 3.9 N were applied to determine the effects of F_x. Due to the small amount of friction in the joints of the beam fixture, the beam comes to rest at a slightly different location if released slowly from tension than when released from compression. For this reason, for each data point of this experiment, the RPS output was recorded after releasing the beam from both tension and compression and then averaged. The resulting calculations of the relative x, y, and ϕ describe the point B in the coordinate frame of A as shown in Figure 20. For each test the potentiometers on the frame were used to calculate the true relative pose of the beam. The RPS algorithm output was then subtracted to calculate the error of each test. In addition a mean normalized error was calculated. The mean error of all tests at a given angular deflection was normalized to provide a nondimensional error reference. The x component error was normalized to the absolute value of the true error in y. The y component error was normalized to the value of the y component of the RPS output. Finally the ϕ error was normalized to the true angular deflection of the beam.

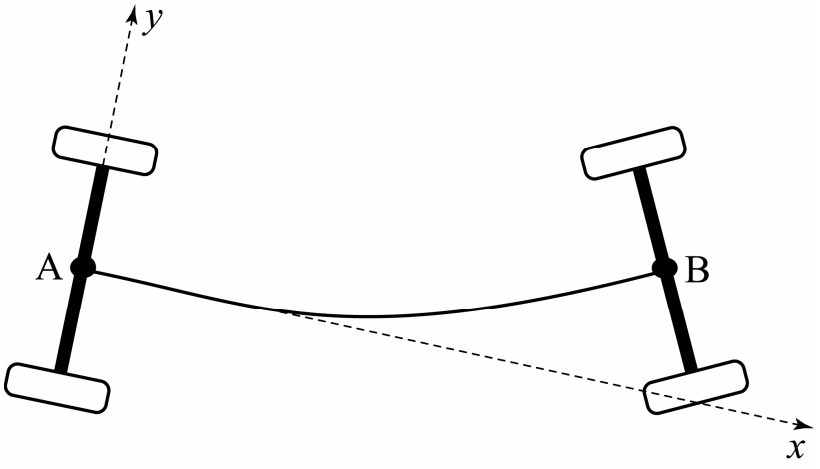

Figure 20. The pose of B in the coordinate frame of A.

Results and Discussion

Figure 21, Figure 22, and Figure 23 show the error associated with the *x*, *y*, and ϕ components of the relative position vector returned by the RPS algorithm. In the plots that follow, the absolute error is displayed as vertical bars. Above the bars for each non zero deflection angle is a percentage that represents the normalized mean error at that deflection.

Several observations can be made that apply to all three plots. In spite of the difficulty in predicting the endpoint of a beam that is capable of deflections in excess of 45°, the RPS sensor performs quite well. Applying a force along a line between the attachment points of the beam does not seem to be detrimental to the accuracy of the RPS

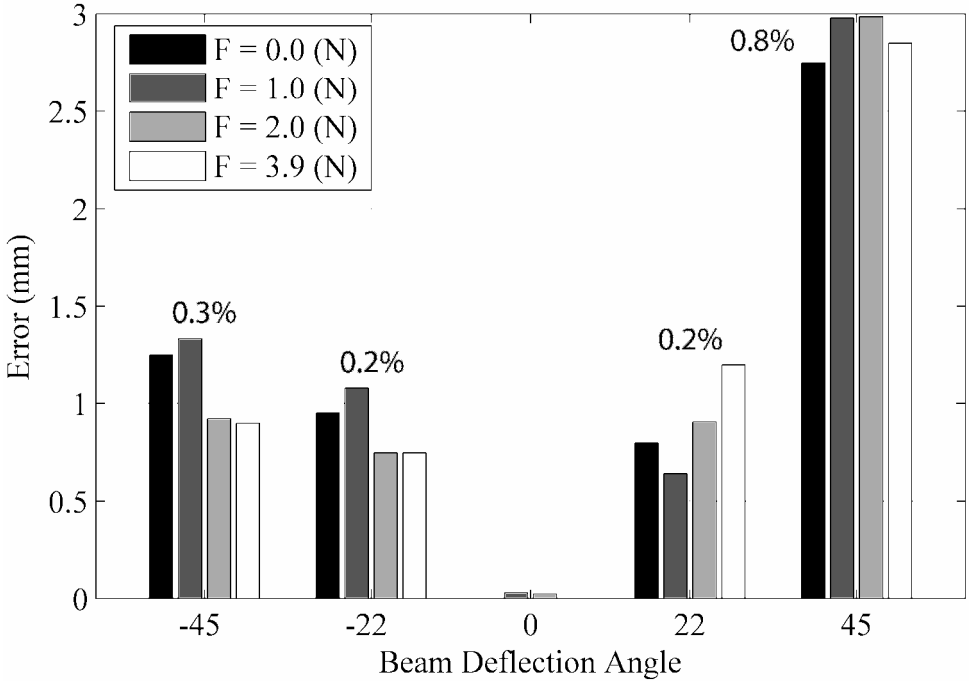

Figure 21. The x component of the RPS error. The percent values are the mean error normalized to the absolute value of the true value of the RPS deflection in y.

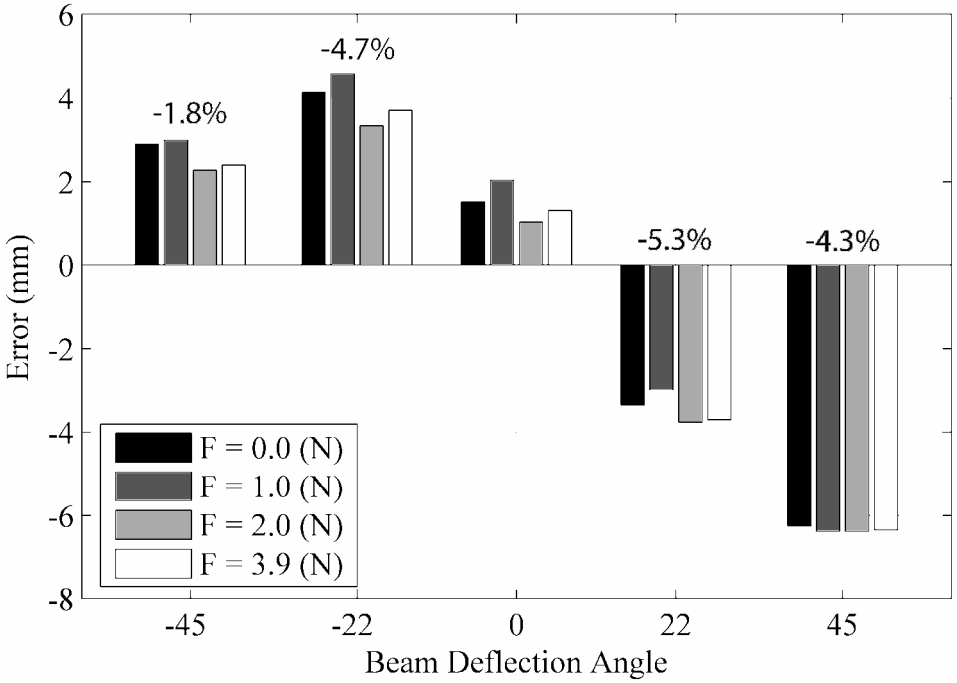

Figure 22. The y component of the RPS error. The percent values are the mean error normalized to the absolute value of the true value of the RPS deflection in y.

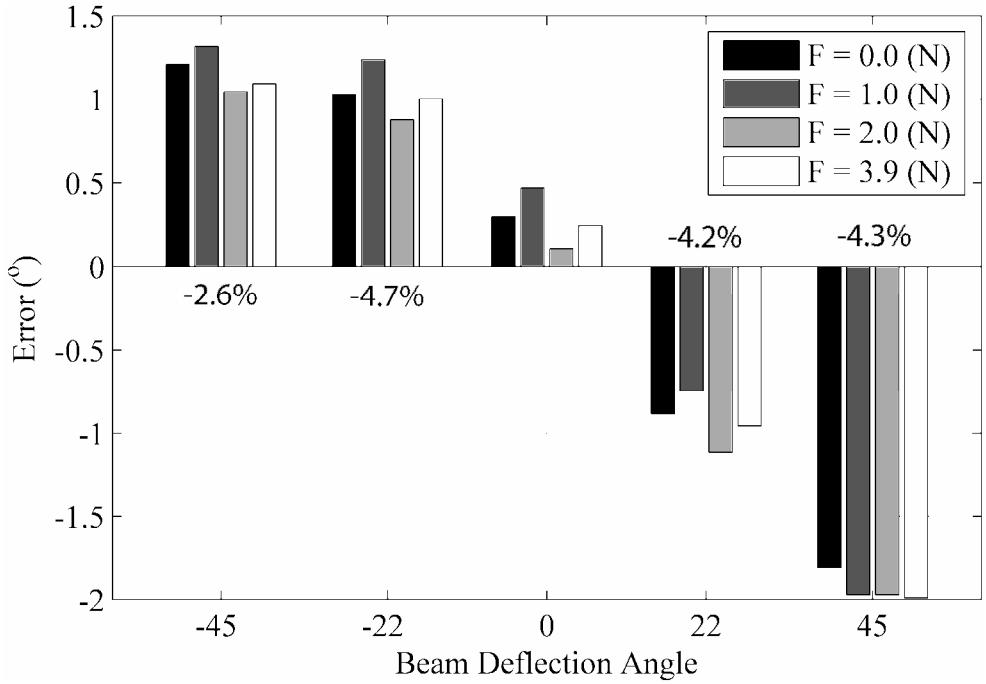

Figure 23. The ϕ component of the RPS error. The percent values are the mean error normalized to the absolute value of the true value of the RPS deflection in ϕ.

algorithm. It is clear from Figure 21 thru Figure 22 that a small, up to 4 N, force acting to separate the axles does not significantly degrade the ability of the RPS to determine the relative posture of the two axles. Small relative position errors between the two axles will have the effect of generating antagonistic forces between the axles. It is good to know that the RPS is robust to this type of disturbance, as it will likely be common in practical application.

The RPS error increases with larger deflections and is somewhat symmetric with respect to the zero deflection angle point. This suggests that perhaps a better calibration method could be developed. The current calibration method relies on an individual calibration of each strain gauge that is applied to the raw voltage that is measured across a particular strain gage. Perhaps an algorithmic level calibration could be applied in addition. Such a calibration would be applied to the output of the RPS algorithm. In addition, the normalized data shows that the relative position errors are likely to be linearly proportional to the beam deflection. This suggests that it may also be advantageous to make the calibration dynamically proportional to the estimated beam deflection. Even if an algorithmic calibration proved useful when the beam is primarily in operating in a mode 1 pure bending regime, this second level of calibration may have deleterious effects on RPS accuracy when the beam is operating in bending modes 2 or 3 (see Figure 5). Whether an algorithmic level RPS calibration would prove to increase accuracy has not been investigated and is left to future work.

The symmetry of the error about the zero angular deflection point is broken by the measured RPS error when the beam was deflected to -45°. The RPS error at a -45° deflection is approximately half what it should be for the symmetry found in the error at

the remainder of the deflection angles to hold at -45°. It is possible that the spring steel has a permanent strain that causes this aberration. One indication of this is the fact that when the beam is placed on a flat surface it does not sit flat. Instead, the beam has a slight twist to it and one of the corners sits several millimeters off of the surface. It is also possible that this effect could be due to errors in the calibration process or errors in measuring the location of the strain bridges. In any case, the precise reason for this anomaly is unknown and left for future work.

The biggest question found in the results of static RPS testing is whether the up to 5% error is acceptable. Without the beam, the relative error between front and rear axles is likely to grow without bound. In practical application on a mobile robot without the RPS algorithm small navigation errors will accumulate without bound until the robot can no longer function. In comparison, with the RPS in place the relative error between axles will be limited to the error of the RPS. With this in mind an error bounded by 5% of the relative deflection in y seems to be a bargain in comparison to an unbounded error without the RPS, as long as the robot motion controller can tolerate the 5%.

Figure 24 shows the RPS prediction error in calculating the value of a tension force applied on a line between the attachment points of the beam at each end as the angular deflection changes. The errors in the force prediction are large. Each test at 45 and -22 resulted in a negative error, while 3 out of 4 of the force tests at a deflection of 22 and -45 produced positive errors. All of the tests using a 4N force produced negative errors.

Figure 25 shows the same force data that has been normalized to the actual force applied to the beam. The normalized data shows 80% errors when attempting to measure a 1N force. The errors decrease some but are still large at 2N and 4N.

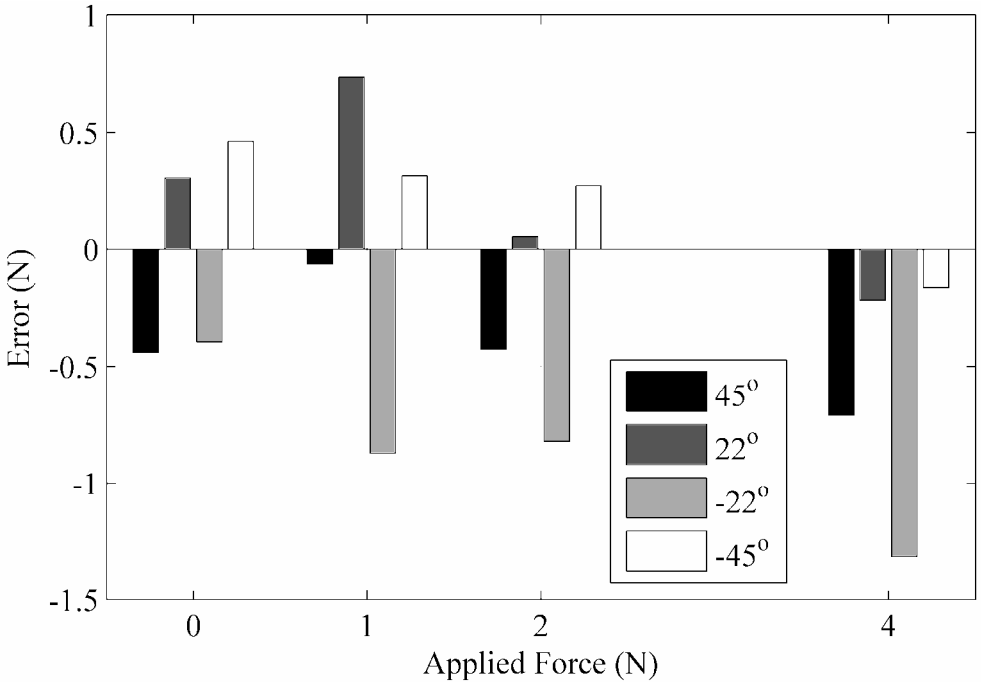

Figure 24. Absolute errors in longitudinal force.

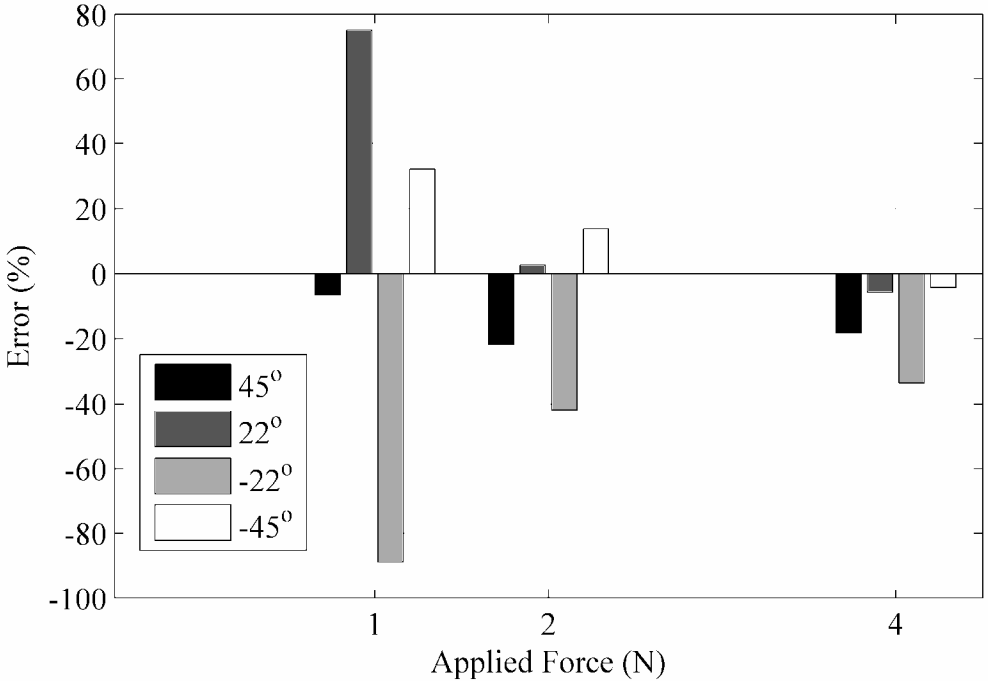

Figure 25. Percent errors in longitudinal force.

The RPS predicted force averages close to -10% overall with a very wide standard deviation. It is unclear why the performance is so poor. Because force estimation is of secondary importance no attempt has been made to optimize it. The normalized plot shows a slight improvement in performance as the force increases. It may be possible that the force estimation capabilities of the RPS can be realized only at high relative forces. This effect has not been investigated and is left to future work. Why the force errors for deflections of -22° and 45° are negative and the errors for the remaining deflections are positive is unknown. It is possible that there may have been some form of operator error when conducting the tests. For instance, the small amount of friction in the beam fixture causes the beam to come to rest at a slightly different position when released from tension rather than compression. If the operator was not careful about averaging measurements after releasing slowly from both tension and compression this type of one sided error would be the result. Because there are no current plans to use force data on the robot this phenomenon has not been fully investigated. If force data is to be applied in some meaningful way to aid in robot navigation, more work will need to be done to refine the force output to achieve a more accurate result.

7. TESTING OF THE RPS AND DATA FUSION ON THE ROBOT

The robot was tested on three surfaces of increasing difficulty. The first surface was a tightly knit closed loop carpet that provided excellent traction and very little slipping. This surface was used as the benchmark. The second surface consisted of sand at a uniform thickness of 10 mm spread on top of a plastic sheet. This surface resulted in a large amount of wheel slip. The final surface consisted of the sand from the previous testing with the addition of 10 mm to 20 mm thick rocks scattered in the path of the robot at approximately 70 mm intervals. These rocks are sand stone in origin with a rough surface to provide the robot with traction as it climbs over them. The size of the rocks was chosen to provide a significant obstacle when compared to the diameter of the wheels but not be so large as to be a barrier.

In the tests the robot motion controller provided point stabilization [24]. The robot was started at a distance of -1m in y and -1.55m in x from the origin. The controller caused the robot to travel an s-shaped path making first a left then a right turn as it tried to converge to the origin. Tests were done on each surface using four different sensor configurations. The sensor configurations were odometry only, odometry in conjunction with a gyroscope on the rear axle only, and the previous two configurations with the addition of the RPS and the added CI data fusion layer.

At the end of each run the position and orientation of each axle was measured using a tape measure and a grid system of strings stretched just above the height of the robot. The actual final pose of the robot could then be compared to the pose as reported by the robot.

It is desired to show the improvement in relative off tracking afforded by the RPS. Therefore, the difference between the measured and sensed ending relative pose of each axle is reported. Specifically the x, y, and ϕ pose of the front axle in the reference frame of the rear axle as shown in Figure 20. The 95% confidence interval, based on at least five trials per test is also reported along with the mean. The confidence interval is included because in some of the trials the differences among trials, although large, is centered close to zero. It is also desired to show an improvement in the final global pose estimate of the robot. To show the improvement in final global pose estimation e is reported, which is the distance from the origin to the mid point between both axles. The variable γ is the orientation of a line drawn between the center points of both axles. See Figure 26 for a schematic of e and γ.

It should be noted that the rear axle housed two 7.2 Volt NiCad batteries that are the source of power for the on-board strain gauge amplification circuitry required by the strain gauges on the beam. This extra weight significantly increased the normal force of the tires against the driving surface with a corresponding increase in traction.

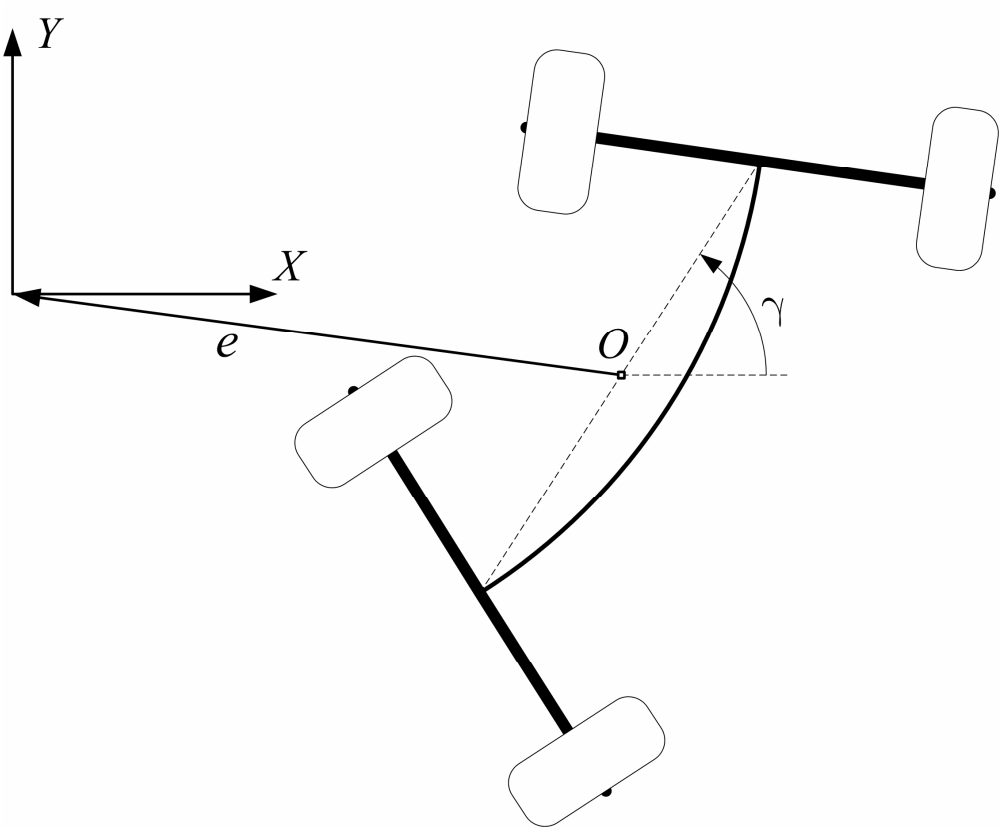

Figure 26. The global pose variables *e* and γ.

Results and Discussion

Table 3 summarizes the results for all tests. In each data field the top number is the mean of all trials and the bottom number is the 95% confidence interval. The data from Table 3 are also represented graphically over the next five pages as a bar chart for each column of Table 3. The bars represent the error between robot ground truth and the pose information generated from the sensors on the robot. The error bars represent a 95% confidence interval.

Table 3

The error between robot ground truth and sensor driven pose estimate

Test Type	Carpet				Sand				Rocks on Sand			
	No Beam		With Beam		No Beam		With Beam		No Beam		With Beam	
	No Gyro.	With Gyro	No Gyro	With Gyro	No Gyro	With Gyro	No Gyro	With Gyro	No Gyro	With Gyro	No Gyro	With Gyro
xRel (mm)	185 ±40.6	326 ±31.1	30.1 ±7.94	-15.7 ±3.77	399 ±201	-103 ±125	11.4 ±2.08	-0.41 ±2.52	-768 ±594	-270 ±160	14.9 ±10.6	37.0 ±20.8
yRel (mm)	-359 ±32.7	165 ±11.8	-15.2 ±2.86	53.5 ±3.79	-24.6 ±290	-549 ±182	28.7 ±14.7	33.9 ±2.57	90.9 ±307	-1111 ±214	6.86 ±5.76	45.6 ±46.3
phiRel (°)	-11.4 ±4.98	2.53 ±9.47	0.82 ±0.52	4.38 ±0.40	-2.38 ±10.2	-26.5 ±8.37	12.9 ±5.88	13.5 ±2.95	-270 ±248	-49.6 ±27.9	1.27 ±4.14	1.77 ±16.3
e (mm)	51.9 ±15.2	-4.60 ±4.42	-49.1 ±7.75	-78.8 ±5.93	-447 ±125	114 ±79.7	45.6 ±34.0	38.5 ±41.9	-131 ±182	500 ±182	43.4 ±64.6	96.7 ±67.2
γ (°)	-13.4 ±3.42	93.5 ±2.55	-8.94 ±0.41	-7.22 ±1.25	128 ±69.9	-77.2 ±24.9	-4.84 ±6.42	0.68 ±2.23	-6.57 ±27.0	-42.0 ±5.98	-0.49 ±0.84	-13.6 ±7.20

Figure 27 shows the error in the relative x distance between the axles for all trials. Note the large decrease in both the mean and confidence interval of the error in the tests that used the RPS when compared with those that did not. The addition of the gyroscope without the RPS increased the error in this metric on the high traction surface while decreasing the error on the low traction surfaces. The effect is very likely due to the fact that for this experiment the gyro bias, or gyroscope output offset, was assumed to be constant when in reality it is generally modeled as a first order Markov process. This phenomenon is also commonly referred to as gyroscope drift. The gyroscope data sheet states that the gyro bias stability over operating environments is <4.5 °/s. The implication is that for a high traction surface the odometry estimate of axle rotation rate is better than the gyroscope estimation due to the statistical variation in gyro bias with the result that

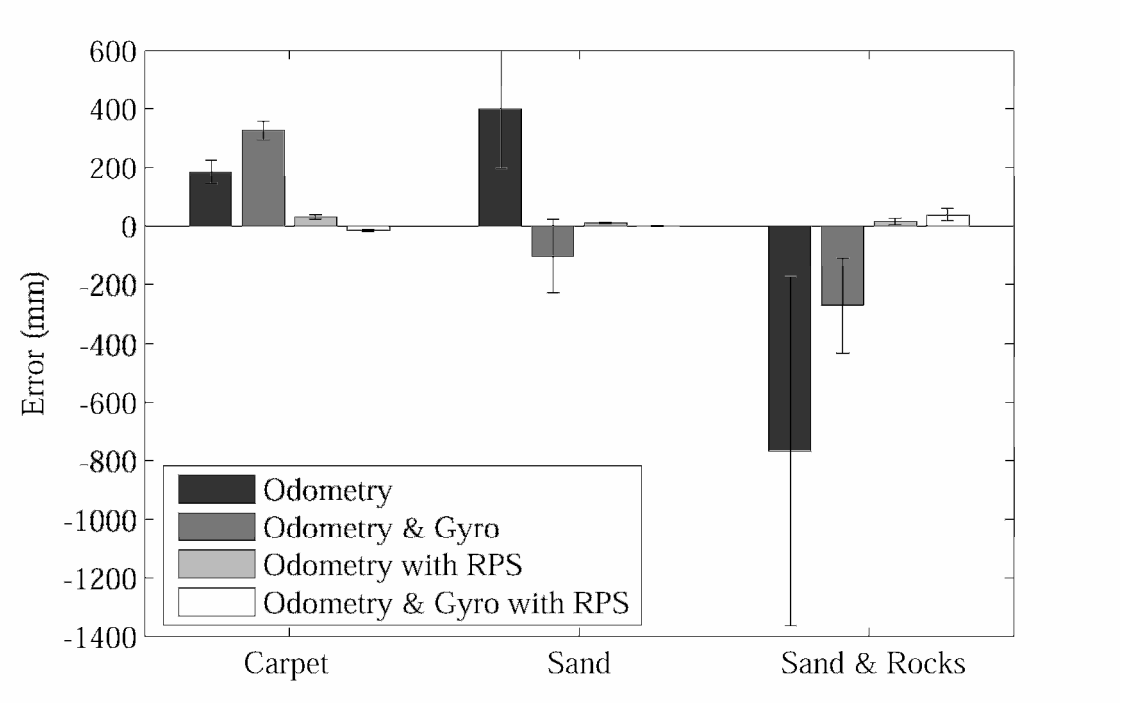

Figure 27. x error in relative axle pose

the gyroscope degrades performance on the carpet surface. On the other hand, when maneuvering on a low traction surface such as sand the gyroscope rotation rate estimate is superior to the rotation rate estimate from odometry, even with the gyro bias effects, and therefore improves the robot performance on the sand surfaces.

Figure 28 shows the error in the relative y distance between the axles for all trials. Note the reduction in error, with the addition of the RPS, for all of the surfaces. Also, the addition of the rear axle gyroscope seems detrimental to this metric, especially in the tests on sand and sand with rocks without the RPS. In order to understand why this metric is degraded by the gyroscope it is important to consider how this metric is formed. The relative position metrics are all based on the posture of the front axle in the coordinate frame of the rear axle (see Figure 20). Because of this the relative y metric is highly

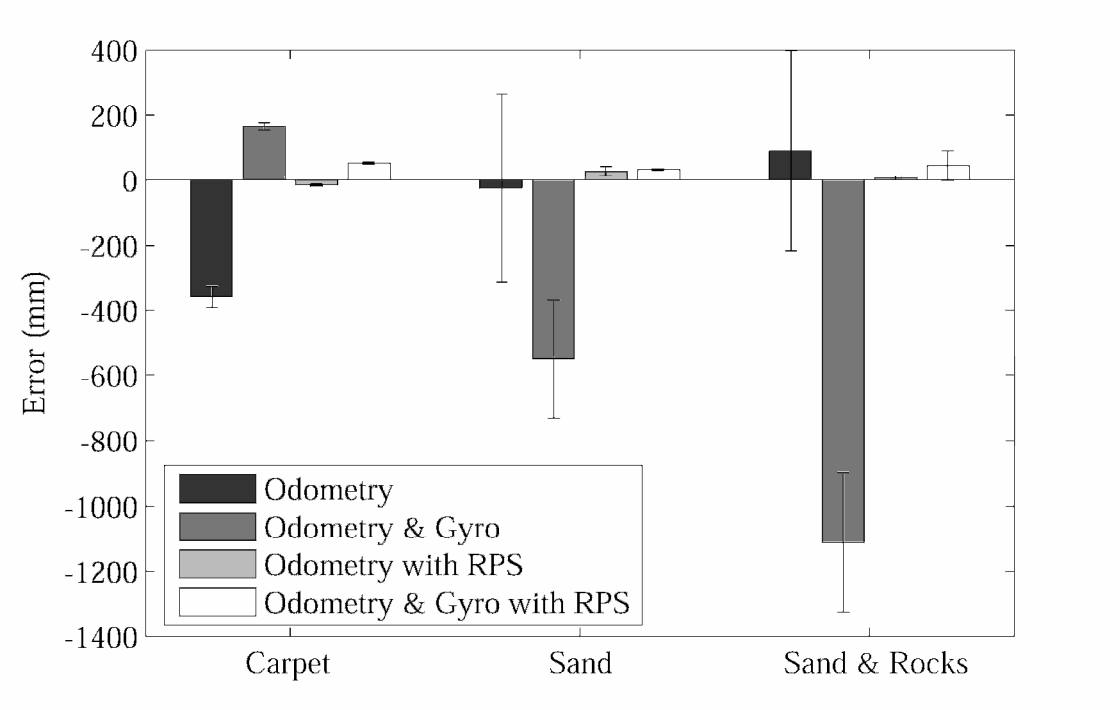

Figure 28. y error in relative axle pose

sensitive to perturbations in the orientation of the rear axle. Any effect that the gyroscope has on the orientation of the rear axle will be amplified by this metric's sensitivity to perturbations in the orientation of the rear axle.

Figure 29 shows the error in the relative orientation between the axles for all trials. For this metric, robot performance on carpet and sand was similar. This makes sense because the robot's motion controller is trying to control both axles in a bending mode 1 configuration. At the same time, wheel slippage will have a tendency to move the robot closer to mode 1 bending, because this bending mode requires smaller traction forces to maintain its shape. For this reason the relative orientation metric is less sensitive to low traction surfaces than the previous metrics. There is a striking difference in the orientation error with and without the RPS on the sand with rocks surface. This can be

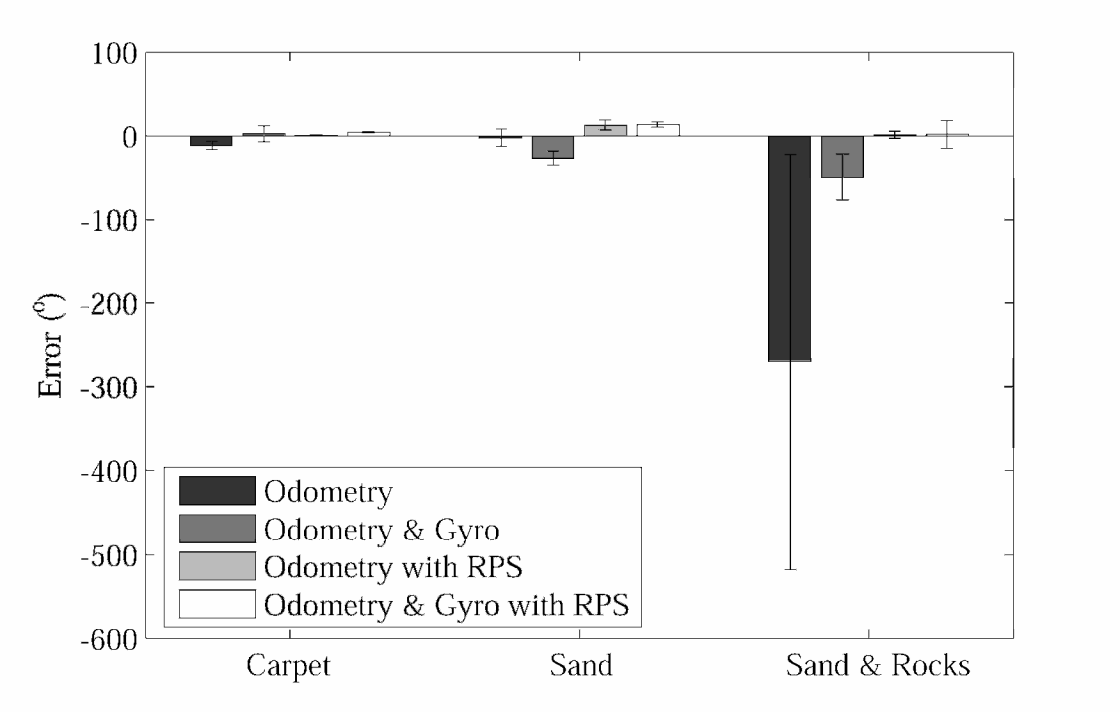

Figure 29. phi error in relative axle pose

attributed to the large disruption created by the rocks, in some cases the rear axle came to a complete stop on a rock. It can also be seen than even with the gyro bias problem the gyroscope significantly improved the performance of the robot on the sand with rocks surface.

Figure 30 shows the error in *e* for all trials (see Figure 20). This metric is degraded slightly by the addition of the RPS on carpet. This is most likely due to the approximate 5% error in the RPS estimates. If the distance traveled on carpet were to be increased it is likely that the error in this metric, for odometry only, would increase proportionally while the error with the RPS would remain bounded. The error in e is also improved significantly by the RPS on both sand and carpet. Again, the addition of the gyroscope is

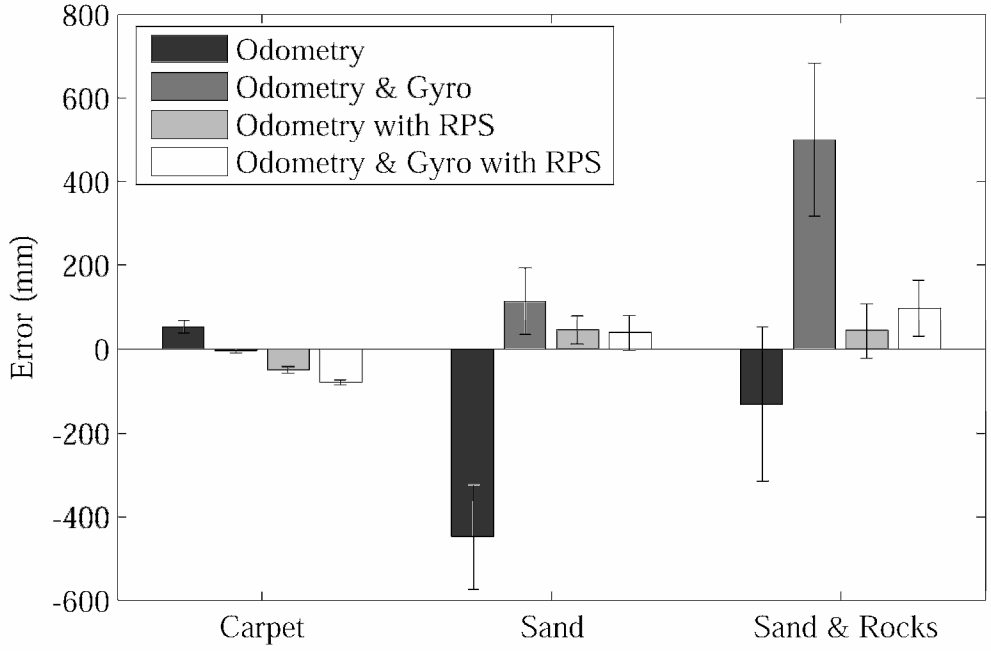

Figure 30. Error in *e*, the distance from the origin to the midpoint of a line connecting the midpoint of both axles.

inconclusive.

Figure 31 shows the error in γ for all trials (see Figure 20). This metric was improved by the addition of the RPS for all surfaces. The general trend for the addition of the gyroscope to increase the error in this metric is likely due to the gyro bias.

In addition to the table and charts of the various error metrics, video footage was taken of the majority of the tests. Still frames from video at 3, 8, 21 and 45 seconds from representative tests of each surface and sensor configuration were assembled. These still frames are overlaid with the estimate of the robots path as recorded by the sensors and data fusion architecture. A dashed line was used to represents the estimated path followed by the rear axle and a solid line was used to illustrates the estimated path followed by the front axle. Barely visible in the frames are two white lines. One runs horizontally across the top of the frames and the other runs vertically down the right side.

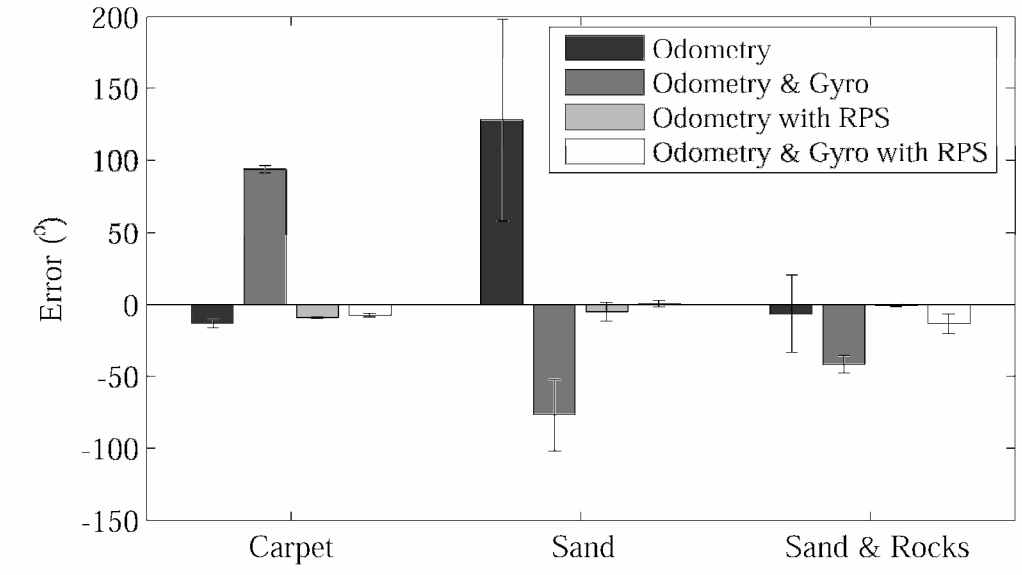

Figure 31. Error in γ, the global orientation of a line between the midpoint of both axles.

The intersection of these lines represents the location that the point stabilization controller is trying to drive the center point of the rear axle. Simultaneously the controller is trying to drive the front axle to a point on the horizontal line to the right of the line intersection in the frames.

Figure 32 shows still frames from testing on carpet with the RPS with and without the gyroscope on the rear axle. No video was taken of the carpet testing without the RPS. Because all of the tests on carpet were nearly identical regardless of sensor configuration the lack of video footage is not detrimental to the overall analysis of robot performance.

Note the difference in orientation of a line drawn through the center point of each axle in the 45 s frames in Figure 32. The large change in this orientation agrees with the data shown in Table 3 and Figure 31.

Figure 33 shows progressive still frames from representative tests of all four sensor variations on the sand surface. The first two rows of still frames show the robot's progression with odometry only and odometry with rear axle gyroscope. In these two sensor configurations the navigation pose of both axles, but more particularly the front axle quickly drifts away from the ground truth pose. This divergence is caused by the large amount of wheel slip due to the low traction sand surface. The rear axle does not drift as much because of the increased traction force due to the weight of the batteries and RPS support electronics carried by that axle. Because the front axle drifts at a greater rate than the rear the relative pose between the two axles suffers greatly. The degradation of the relative pose initiates antagonistic forces between the two axles as the robot motion controller delivers control commands to axles that have a relative ground truth pose that is very different from the navigation pose. Further more, the antagonistic forces between

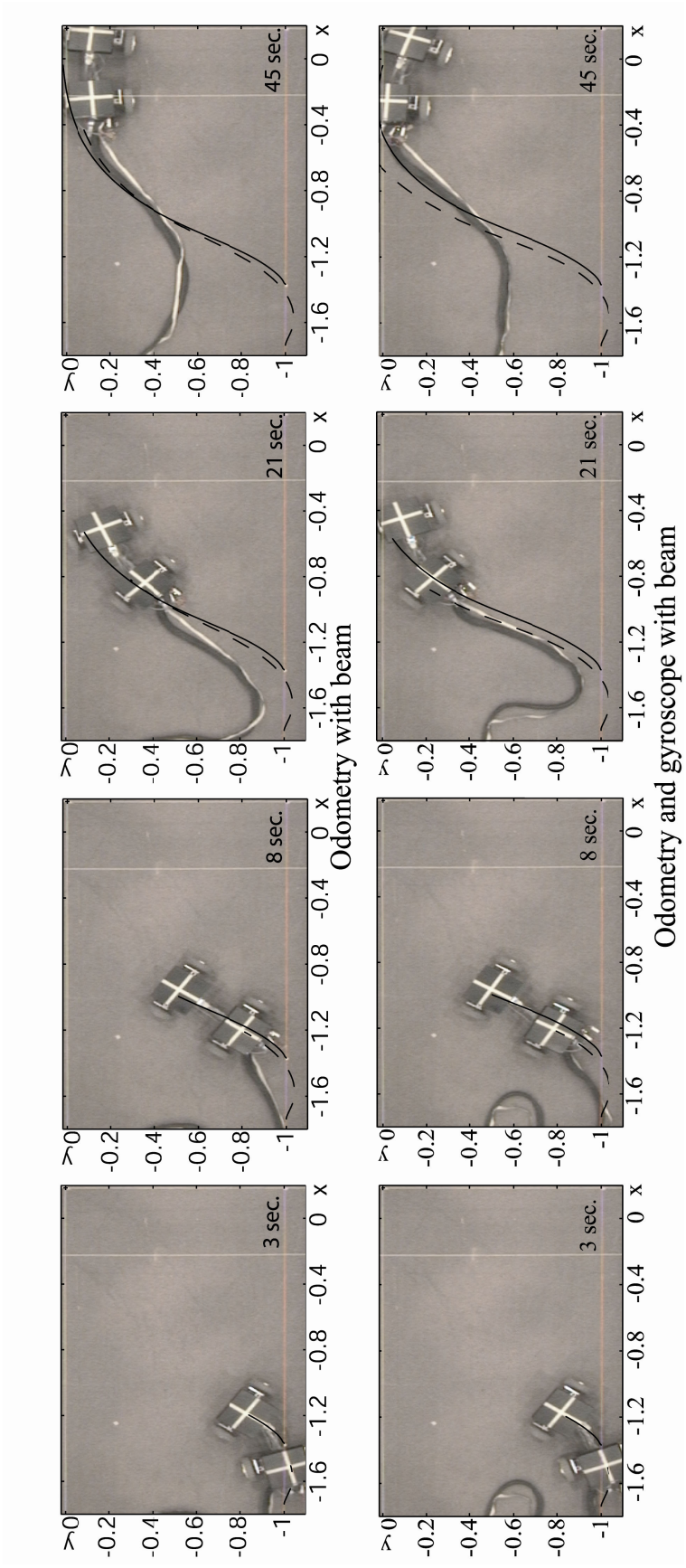

Figure 32. Testing on carpet.

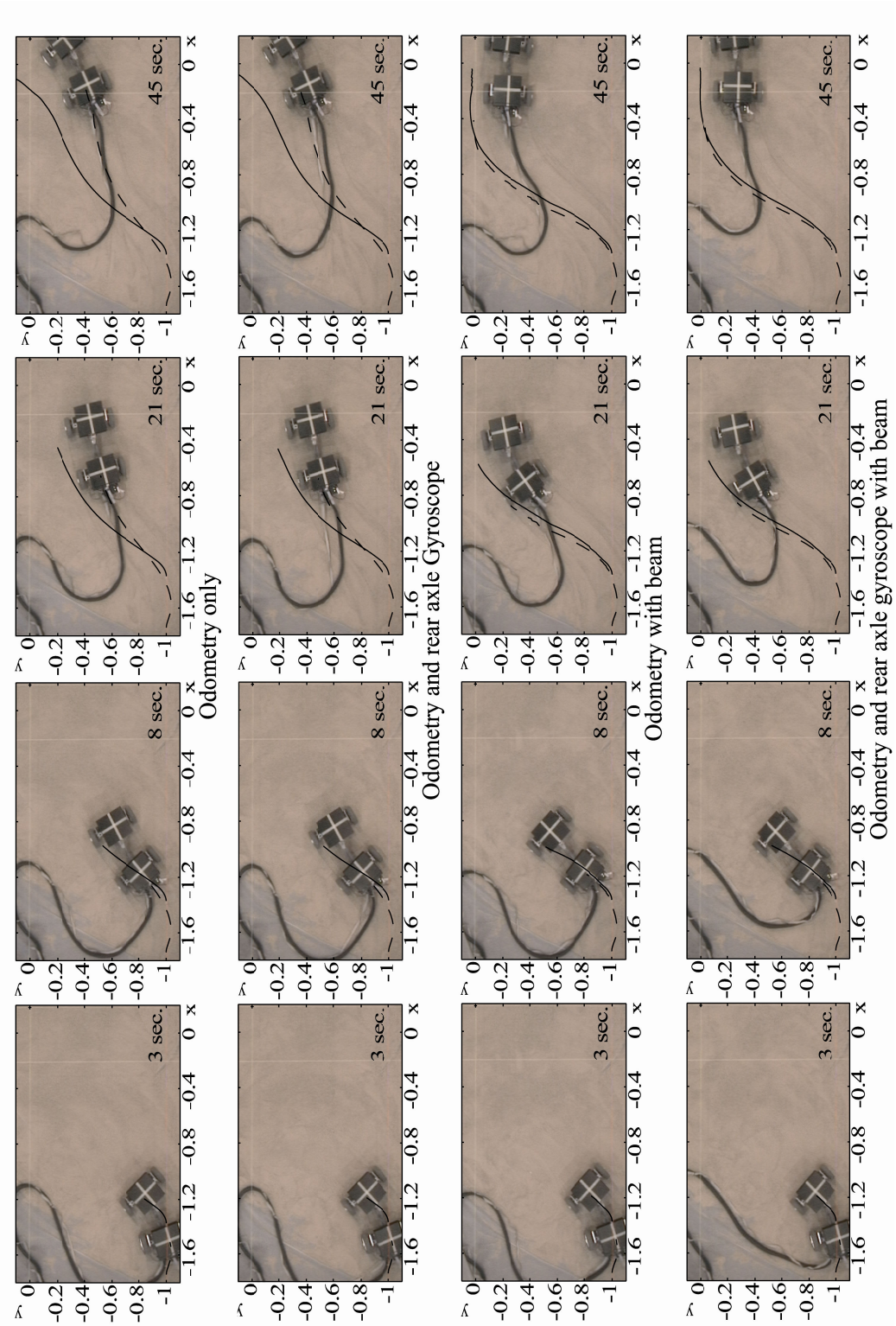

Figure 33. Testing on a 10 mm thick sand surface

the axles increases the rate of navigation pose drift compounding the problem. Finally,with the loss of fidelity between the axles the motion controller losses its ability to drive the navigation states to the origin. The result being that control authority is very rapidly lost, as can be seen in the figure.

The third and fourth rows of still frames show the results with the addition of the RPS algorithm to the sensor suite. In these two series the front and rear axles track each other very accurately. Because the RPS algorithm continually corrects the relative pose between the axles a high degree of fidelity is maintained between the front and rear axles. This allows the robot motion controller to maintain effective control authority regardless of the fact that the navigation states of the robot are still drifting away from the ground truth. This fidelity between the front and rear axle is required by the motion controller for stability, and is the primary benefit of the RPS and associated algorithms.

Figure 34 shows progressive still frames from representative tests of all four sensor variations on the sand and rocks surface. Note the more jagged appearance of all the navigation state lines due to the rocks. In the first three rows of still frames the rear axle got stuck on a rock and came to a complete stop. The front axle in these frames continues to spin its wheels in the sand but never manages to proceed any further. In the first two rows of still frames, without the RPS, the trace of where the front axle thinks it is continues to progress even though the axle is actually stuck in place. This is a more extreme case of the potential difference in drift rates between front and rear axles than was seen in Figure 33. In the third row of frames, with the RPS in use, the front axle trace stops in place at the same time that the rear axle gets stuck. This occurs even though the front axle was spinning its wheels for the entire time it was stuck in place.

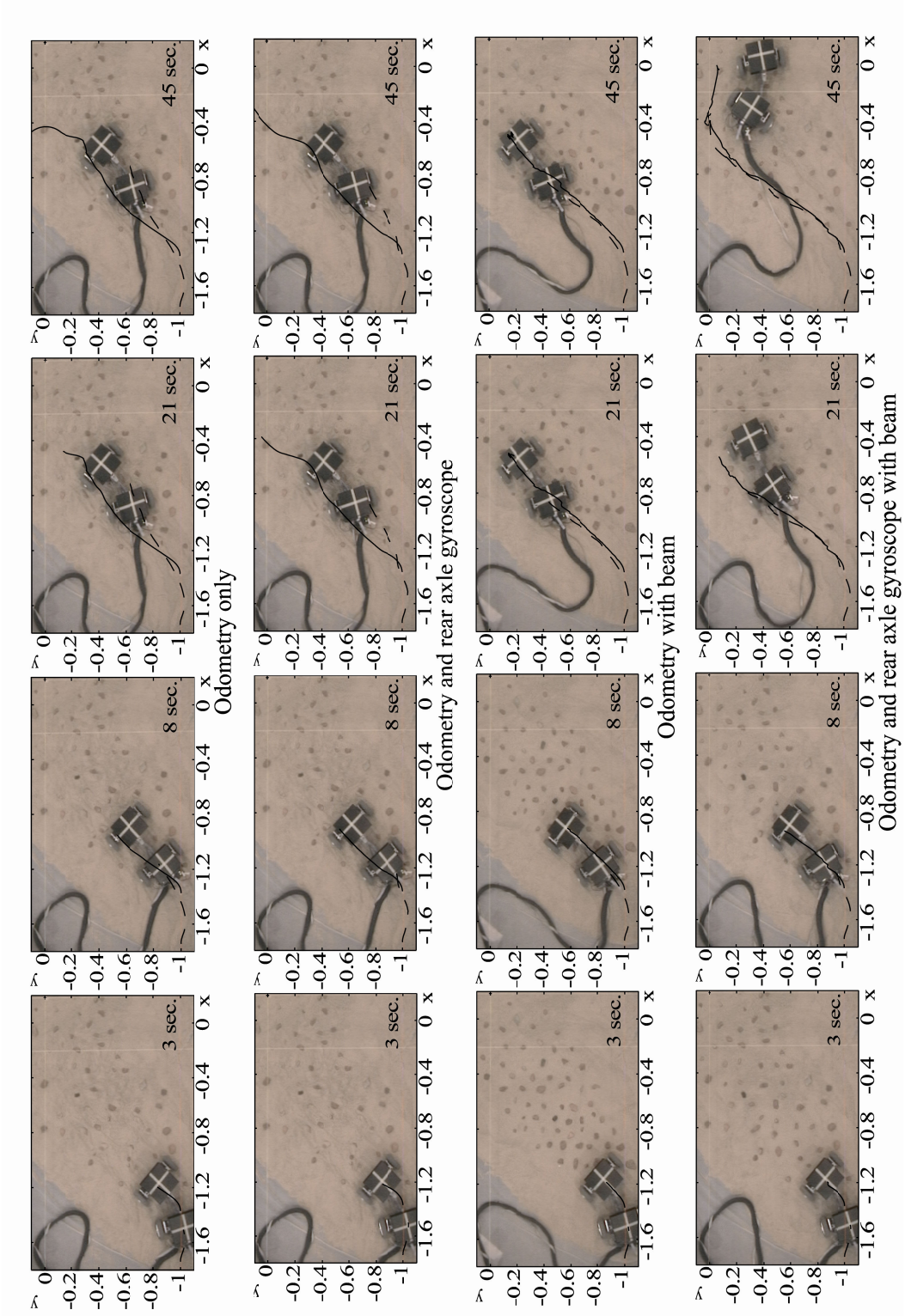

Figure 34. Testing on 10 mm thick sand with rocks.

This is a very effective demonstration of the ability of the RPS to maintain fidelity between the front and rear axles. This type of performance can be extrapolated to many extreme circumstances. For example, if one of the axles were to be kicked out of place in mid maneuver the RPS would detect the radical change in posture and correct the odometry allowing the motion controller to maintain control authority. This behavior makes the robot very robust to not only surface conditions but also to much of the chaos in a dynamic environment. In the fourth row of still frames the robot performance on rocks is very similar to its performance on carpet, except that the axle traces are less smooth due to the rocky surface. The high degree of congruency between the front and rear axles seen in the final set of still frames in Figure 34 can be achieved only through the use of the RPS and associated data fusion algorithms.

In summary, odometry works well for short runs on high traction surfaces. However, it is probable that for longer tests and certain that on lower traction surfaces the performance of odometry degrades substantially. For this reason it is recommended the odometry be augmented with an additionally posture sensing mechanism.

The addition of the gyroscope to odometry is conflicting in its contribution. It improved some metrics on some surfaces while degrading others. It seems clear that the gyro bias was a significant factor in the performance of this sensor. Improvements must be made in the implementation of the gyroscope before it can make a positive contribution to estimating the posture of the robot. The most obvious improvement would be to use a better gyroscope. The gyro bias stability parameter of this gyroscope is several orders of magnitude larger than the best gyroscopes in current use today. Another possibility would be to compensate for the gyro bias in some manner. One common

compensation method is to include the gyro bias stability as a state in the Kalman filter and attempt to estimate its current value online. Other possibilities include schemes to weight the gyroscope information more when the robot is maneuvering quickly and therefore more likely to be suffering from wheel slip with the corresponding reduction in the accuracy of odometry. Another option would be an algorithm to calculate real time wheel traction force requirements in order to detect when the wheels are more likely to slip due to increased traction requirements. This information could be used to increase the relative weighting of the gyroscope when the wheels are more likely to be slipping due to insufficient traction force. In any case, more work needs to be done before the inclusion of the gyroscope as a permanent addition to the sensor suite can be justified.

The RPS improved performance in nearly all cases. This sensor proved to be robust to seemingly catastrophic environmental conditions, such as a stalled axle on rocky terrain. It also appears to have the ability to place an upper bound on the relative position error between the two axles. This property is an essential ingredient in providing the robot the ability to maintain fidelity between the axles during long term maneuvers. The ability of the RPS seems to be only slightly effected by the addition of the gyroscope. It seems that the RPS corrections overwhelm most of the detrimental effects of the gyro bias. It is recommended that any implementation of this robot configuration include the RPS with its associated data fusion algorithms.

8. CONCLUSION

A sensor fusion strategy for compliant framed modular mobile robots based upon traditional Kalman Filtering and Covariance Intersection filtering is presented. Key to the later is the modeling and instrumentation of the frame module as an internal sensor for determining relative axle module posture. Experimental results have been presented that demonstrate the effect on posture estimation afforded by the four sensor suites tested. Results show that odometry alone, while accurate on high traction surfaces for short distances cannot be relied upon to maintain fidelity between the axles on low traction surfaces. Improvements need to be made in the implementation of the gyroscope before it can be relied upon to any degree. The RPS sensor was shown to effectively bound the error in relative posture between the axles to the error inherent to the sensor. This ability was proven even in the face of seemingly catastrophic events such when one axle gets grounded on an obstacle and the other axles wheels spin out of control. This property of the sensor makes it a critical addition to the robots sensor suite.

Future work should include long distance testing to be certain that the error bounding characteristics of the RPS sensor continue to hold under the duress of a long term maneuver. In addition to improving the gyroscope implementation it may be worthwhile to investigate a two degree of freedom IMU by mounting accelerometers sensitive to both the axles' x and y coordinates. This proposal could also be extended to

a full six degree of freedom IMU on each axle to aid in navigation of three dimensional rough terrain. Another obvious sensor addition would be a GPS receiver on one of the axles. It would be interesting to determine to what degree the RPS could propagate a GPS update from one axle to the other. The RPS itself could also be extended to detect rotations in roll between the two axles. This information could prove useful when navigating rough terrain. It may also prove useful to gain better insight into the beam force estimation capabilities of the RPS and investigate its use in improving robot motion planning.

In conclusion the beam sensor in combination with the IPEC algorithm and the CI filter does perform well. In fact, the RPS system essentially makes the robot robust to surface conditions.

APPENDIX

S-FUNCTION USED IN THE RPS IMPLEMENTATION

```
/*  File    : csfunc.c
 *  Abstract:
 *
 *      Example C-file S-function for defining a continuous system.
 *
 *      x' = Ax + Bu
 *      y  = Cx + Du
 *
 *      For more details about S-functions, see
simulink/src/sfuntmpl_doc.c.
 *
 *  Copyright 1990-2001 The MathWorks, Inc.
 *  $Revision: 1.8 $
 */

#define S_FUNCTION_NAME sfunRelPose5g4ord2
#define S_FUNCTION_LEVEL 2

#include "simstruc.h"
#include "math.h"


#define U(element) (*uPtrs[element])  /* Pointer to Input Port0 */

static real_T thick   = 3.4000e-004; /* nominal point of internal shear
inflection */
static real_T Lrigid0 = 0.022225;  /* rigid section from center of rear
axle to start of flexible beam */
static real_T LrigidF = 0.022225;  /* rigid section from start of
flexible beam to center of front axle */

// flexible portion of beam endpoints and strain gauge locations
//   starting at rear axle, sg1,sg2, ... , front axle
static real_T Lvect[7]={ 0.0, 0.0054, 0.0913, 0.1749, 0.2538, 0.3451,
0.35 };
     real_T cVect[7];

static real_T polyFind[5][5]={ {  1.138379743695124e+000, -
2.793030781484927e-001,  2.269282559937683e-001, -1.023402723908596e-
001,  1.633535085046015e-002 } ,
                               { -2.676134582013608e+001,
5.522954686050930e+001, -4.606096900088158e+001,
2.095451375523562e+001, -3.361745794727260e+000 } ,
```

```
                                {  2.140503309026899e+002, -
6.635074116337664e+002,  7.672194129509087e+002, -
3.822065763333901e+002,  6.444424411355794e+001 } ,
                                { -7.041356083435730e+002,
2.630901081793879e+003, -3.655502071904744e+003,
2.120904975172102e+003, -3.921683767176635e+002 } ,
                                {  8.139355084309009e+002, -
3.376413092651278e+003,  5.255178366740569e+003, -
3.439119466794389e+003,  7.464186842741965e+002 } ,
                                };

/*====================*
 * S-function methods *
 *====================*/

/* Function: mdlInitializeSizes
===============================================
 * Abstract:
 *    The sizes information is used by Simulink to determine the S-
function
 *    block's characteristics (number of inputs, outputs, states, etc.).
 */
static void mdlInitializeSizes(SimStruct *S)
{
    ssSetNumSFcnParams(S, 0);  /* Number of expected parameters */
    if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S)) {
        return; /* Parameter mismatch will be reported by Simulink */
    }

    ssSetNumContStates(S, 0);
    ssSetNumDiscStates(S, 0);

    if (!ssSetNumInputPorts(S, 1)) return;
    ssSetInputPortWidth(S, 0, 5);
    ssSetInputPortDirectFeedThrough(S, 0, 1);

    if (!ssSetNumOutputPorts(S, 1)) return;
    ssSetOutputPortWidth(S, 0, 3);

    ssSetNumSampleTimes(S, 1);
    ssSetNumRWork(S, 0);
    ssSetNumIWork(S, 0);
    ssSetNumPWork(S, 0);
    ssSetNumModes(S, 0);
    ssSetNumNonsampledZCs(S, 0);

    /* Take care when specifying exception free code - see
sfuntmpl_doc.c */
    ssSetOptions(S, SS_OPTION_EXCEPTION_FREE_CODE);
}



/* Function: mdlInitializeSampleTimes
=========================================
 * Abstract:
 *    Specifiy that we have a continuous sample time.
 */
static void mdlInitializeSampleTimes(SimStruct *S)
{
    ssSetSampleTime(S, 0, INHERITED_SAMPLE_TIME);
```

```
    ssSetOffsetTime(S, 0, 0.0);
}



#define MDL_INITIALIZE_CONDITIONS
/* Function: mdlInitializeConditions
========================================
 * Abstract:
 *    Initialize both continuous states to zero.
 */
static void mdlInitializeConditions(SimStruct *S)
{

}



/* Function: mdlOutputs
=======================================================
 * Abstract:
 *      y = Cx + Du
 */
static void mdlOutputs(SimStruct *S, int_T tid)
{
  real_T            *y    = ssGetOutputPortRealSignal(S,0);
  InputRealPtrsType uPtrs = ssGetInputPortRealSignalPtrs(S,0);
  real_T            a=0, b=0, c=0, d=0, e=0;  /* curvature polynomial
coefficients */
  real_T            accum = 0.0;
  int_T             i=0, j=0, n=0;
  real_T            dx=0, dy=0, dPsi=0;
  real_T            L=0, dL=0.00035, dLprev=0, dLmin=0.0007002,
dLmax=0.03501;
  real_T            Dcl=0; // change in curvature over the prev time
step
  real_T            DclMax=15300, DclMM=0; // max change in curv per unit
length
  real_T            curv=0, curvPrev=0;
  real_T            xx=0, yy=0, psi=0, psiP=0;

  UNUSED_ARG(tid); /* not used in single tasking mode */

  for (i = 1; i < 6; i++)
  {
    cVect[i] = U(i-1)/thick;
  }

  for (i = 1; i < 6; i++)
  {
    accum += polyFind[0][i-1]*cVect[i];
  }
  a = accum;
  accum = 0.0;
  for (i = 1; i < 6; i++)
  {
    accum += polyFind[1][i-1]*cVect[i];
  }
  b = accum;
  accum = 0.0;
  for (i = 1; i < 6; i++)
```

```
  {
    accum += polyFind[2][i-1]*cVect[i];
  }
  c = accum;
  accum = 0.0;
  for (i = 1; i < 6; i++)
  {
    accum += polyFind[3][i-1]*cVect[i];
  }
  d = accum;
  accum = 0.0;
  for (i = 1; i < 6; i++)
  {
    accum += polyFind[4][i-1]*cVect[i];
  }
  e = accum;

  cVect[0] = a;
  cVect[6] =
(a+b*Lvect[6]+c*Lvect[6]*Lvect[6]+d*Lvect[6]*Lvect[6]*Lvect[6]+e*Lvect[6
]*Lvect[6]*Lvect[6]*Lvect[6]);

  L += dL/2;
  while (L<Lvect[6])
  {
    curvPrev = curv;
    curv = a+b*L+c*L*L+d*L*L*L+e*L*L*L*L;  // curve = curvature of beam
at x
    if (fabs(curv)>1e-13)
    {
      dPsi = dL*curv;      //dPsi = dL/rho;
      dx = sin(dPsi)/curv;  //dx = rho*sin(dPsi);
      dy = (1-cos(dPsi))/curv; //dy = rho-rho*cos(dPsi);
    }
    else
    {
      dPsi = 0;
      dx = dL;
      dy = 0;
    }

    psiP = psi;
    xx  += dx*cos(-psiP)+dy*sin(-psiP);
    yy  += -dx*sin(-psiP)+dy*cos(-psiP);
    psi += dPsi;

    // calc next dL
    if ((dL+dLprev)/2>1e-15) Dcl = fabs(curv-curvPrev)/((dL+dLprev)/2);
    dLprev = dL;
    dL = (dLmin-dLmax)/DclMax*Dcl+dLmax;
    if (dL<dLmin) dL = dLmin;

    if (L+dLprev/2+dL>Lvect[6]) dL = Lvect[6]-L-dLprev/2;
    L += dLprev/2+dL/2;

    n+=1;
    if (Dcl>DclMM) DclMM = Dcl;

  }

  y[0] = Lrigid0+xx+LrigidF*cos(psi);
```

```
  y[1] = yy+LrigidF*sin(psi);
  y[2] = psi; //*180./3.141592653589793;
  //y[3] = n;
  //y[4] = DclMM;

}



#define MDL_DERIVATIVES
/* Function: mdlDerivatives
=================================================
 * Abstract:
 *      xdot = Ax + Bu
 */
static void mdlDerivatives(SimStruct *S)
{

}



/* Function: mdlTerminate
=====================================================
 * Abstract:
 *    No termination needed, but we are required to have this routine.
 */
static void mdlTerminate(SimStruct *S)
{
    UNUSED_ARG(S); /* unused input argument */
}

#ifdef  MATLAB_MEX_FILE    /* Is this file being compiled as a MEX-file?
*/
#include "simulink.c"      /* MEX-file interface mechanism */
#else
#include "cg_sfun.h"       /* Code generation registration function */
#endif
```

REFERENCES

[1] S. Kimura, S. Tsuchiya, S. Nishida, and T. Takegai, "A module type manipulator for remote inspection in space," presented at IEEE SMC'99 Conference Proceedings. 1999 IEEE International Conference on Systems, Man, and Cybernetics, Piscataway, NJ, USA, 1999.

[2] A. Casal and M. Yim, "Self-reconfiguration planning for a class of modular robots," Proceedings of the SPIE - The International Society for Optical Engineering, vol. 3839, pp. 246-57, 1999.

[3] D. Rus, "Self-reconfiguring robots," IEEE Intelligent Systems, vol. 13, pp. 2-4, 1998.

[4] A. Castano and P. Will, "Representing and discovering the configuration of Conro robots," presented at Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation, Piscataway, NJ, USA, 2001.

[5] A. Pamecha and G. Chirikjian, "A useful metric for modular robot motion planning," presented at Proceedings of IEEE International Conference on Robotics and Automation, New York, NY, USA, 1996.

[6] S. Murata, H. Kurokawa, and S. Kokaji, "Self-assembling machine," presented at Proceedings of the 1994 IEEE International Conference on Robotics and Automation, Los Alamitos, CA, USA, 1994.

[7] M. Yim, K. Roufas, D. Duff, Y. Zhang, and S. Homans, "Modular reconfigurable robots in space applications," presented at 10th IEEE International Conference on Advanced Robotics (ICAR 2001), Budapest, Hungary, 2001.

[8] R. O. Ambrose, M. P. Aalund, and D. Tesar, "Designing modular robots for a spectrum of Space applications," Proceedings of the SPIE - The International Society for Optical Engineering, vol. 1829, pp. 371-81, 1992.

[9] S. J. Julier and J. K. Uhlmann, "A non-divergent estimation algorithm in the presence of unknown correlations," presented at Proceedings of 16th American CONTROL Conference, Evanston, IL, USA, 1997.

[10] M. A. M. X. Zhu, and S. Park, "Distributed robust control of compliant framed wheeled modular mobile robots," Submitted to ASME J. of Dyn. Syst., Meas., and Contr., 2005.

[11] Y. K. a. M. A. Minor, "Unicyle kinematics based bounded smooth time invariant motion control of compliant framed wheeled modular mobile robots," Submitted to IEEE Trans. on Contr. Syst. Tech., 2004.

[12] S. Park and M. A. Minor, "Modeling and dynamic control of compliant framed wheeled modular mobile robots," presented at 2004 IEEE International Conference on Robotics and Automation, 26 April-1 May 2004, New Orleans, LA, USA, 2004.

[13] J. Borenstein, "Internal correction of dead-reckoning errors with a dual-drive compliant linkage mobile robot," Journal of Robotic Systems, vol. 12, pp. 257-273, 1995.

[14] M. G. Bekker, "Vehicle with Flexible Frame." United States, 1962.

[15] P. L. Spanski, "Flexible Frame Vehicle." United States: United States Army, 1970.

[16] J. Borenstein, "Experimental results from internal odometry error correction with the OmniMate mobile robot," Robotics and Automation, IEEE Transactions on, vol. 14, pp. 963-969, 1998.

[17] C. Altafini, "Why to use an articulated vehicle in underground mining operations?," presented at Proceedings of International Conference on Robotics and Automation, Piscataway, NJ, USA, 1999.

[18] A. Kemurdjian, V. Gromov, V. Mishkinyuk, V. Kucherenko, and P. Sologub, "Small Marsokhod configuration," Proceedings - IEEE International Conference on Robotics and Automation, pp. 165-168, 1992.

[19] K. J. Waldron and C. J. Hubert, "Control of contact forces in wheeled and legged off-road vehicles," presented at Proceedings of Sixth International Symposium on Experimental Robotics, London, UK, 2000.

[20] T. Truscott, "Inertial Measurement Unit," University of Utah, Salt Lake City, Report 12/72002 2002.

[21] J.-C. Piedboeuf and S. Miller, "Estimation of endpoint position and orientation of a flexible link using strain gauges," presented at Proceedings of IFAC Symposium on Robot Control 1994, 2-Volume Set, Oxford, UK, 1995.

[22] S. J. Julier and J. K. Uhlmann, "Simultaneous localisation and map building using split covariance intersection," presented at Proceedings of RSJ/IEEE International Conference on Intelligent Robots and Systems, Piscataway, NJ, USA, 2001.

[23] P. O. Arambel, C. Rago, and R. K. Mehra, "Covariance intersection algorithm for distributed spacecraft state estimation," presented at Proceedings of American Control Conference, Piscataway, NJ, USA, 2001.

[24] B. W. Albiston and M. A. Minor, "Curvature based point stabilization for compliant framed wheeled modular mobile robots," presented at IEEE International Conference on Robotics and Automation. IEEE ICRA 2003 Conference Proceedings, 14-19 Sept. 2003, Taipei, Taiwan, 2003.

[25] T. D. Larsen, K. L. Hansen, N. A. Andersen, and O. Ravn, "Design of Kalman filters for mobile robots; evaluation of the kinematic and odometric approach," presented at Proceedings of the 1999 IEEE International Conference on Control Applications, 22-27 Aug. 1999, Kohala Coast, HI, USA, 1999.