System Wide Joint Position Sensor Fault Tolerance in Robot Systems Using Cartesian Accelerometers

Hal A. Aldridge and Jer-Nan Juang
NASA Langley Research Center
Hampton, VA 23681-0001

ABSTRACT

Joint position sensors are necessary for most robot control systems. A single position sensor failure in a normal robot system can greatly degrade performance. This paper presents a method to obtain position information from Cartesian accelerometers without integration. Depending on the number and location of the accelerometers, the proposed system can tolerate the loss of multiple position sensors. A solution technique suitable for real-time implementation is presented. Simulations were conducted using 5 triaxial accelerometers to recover from the loss of up to 4 joint position sensors on a 7 degree of freedom robot moving in general three dimensional space. The simulations show good estimation performance using non-ideal accelerometer measurements.

Keywords: robot control, fault tolerance, position sensor, accelerometer

1. INTRODUCTION

Fault tolerant control in robot systems has been an active research area in the past few years[1,2,3,4]. The research has been driven by the need for robots to work reliably in space and nuclear environments where human intervention is difficult. Robot failure can be classified into three categories: actuator failure, sensor failure, and task failure. Sensor and actuator failures involve the failure of a physical device in the robot system. Task failure occurs when a fully operational robot is not able to complete the desired task due to control errors or unexpected problems in the task. This paper will concentrate on the recovery of joint position sensor failures.

Fault tolerance to joint position sensor failures has been achieved by several techniques. The primary method used in robust robot design is the inclusion of redundant position sensors. The redundant sensor's output is compared with the primary sensor's output to detect failures and, during failure, is used instead of the primary sensor's output. The addition of the redundant sensor usually affects joint design. Another technique involves tracking the end-effector with a camera or other sensor. If the end effector pose can be determined by external means, a joint position error that affects the end-effector position can be detected and feedback signals generated to continue position control. End- effector tracking is very difficult in unstructured environments and is usually done at an update rate less than the joint servo loops require. Kinematically redundant robots are also employed, not only for their enhanced dexterity, but for their ability to freeze a failed joint and continue satisfactory operation. This solution is expensive and satisfactory operation can depend on the joint which fails, the kinematic configuration of the robot, and the trajectory desired. The problems inherent in the three position sensor failure recovery methods discussed make it difficult for an existent robot design to be made fault tolerant to position sensor failure.

Accelerometers have been used previously in robot control in three main areas: flexibility control[5], system identification[6], and runaway detection[7]. Runaway detection is the only application that is related to fault tolerant control. Possible integration based techniques to obtain Cartesian position from Cartesian acceleration are prone to bias errors. Until the recent development of small accelerometers, addition of accelerometers to a robot design was not trivial. As a result, the utility of accelerometer based control was limited in rigid robot systems.

The application of accelerometers in fault tolerance was expanded to joint position sensor fault tolerance[8]. This method used two triaxial accelerometers to determine the position of a robot joint without integration or inverse kinematics. While useful to protect a particular joint sensor, the number of accelerometers required to protect multiple joints in the robot arm can be impractical.

This paper proposes a system wide approach to position sensor fault tolerance. A combination of end effector mounted and link mounted accelerometers can be used to determine the position of multiple joints in a robot system. The accelerometers can be located in a manner to provide fault tolerance to any joint in the robot system.

2. KINEMATIC EQUATIONS

The proposed solution method is designed for an *n* revolute joint, rigid link robot system with *m* triaxial accelerometers mounted at various locations on the robot. A robot of this kinematic configuration is governed by the following kinematic equations:

$$\omega_{i+1}^{i+1} = R^{i+1}{}_{i}\omega_i^i + \dot{\theta}_{i+1}\hat{z} \tag{1}$$

$$\dot{\omega}_{i+1}^{i+1} = R_i^{i+1}\dot{\omega}_i^i + R_i^{i+1}\omega_i^i \times \dot{\theta}_i\hat{z} + \ddot{\theta}_{i+1}\hat{z} \tag{2}$$

$$a_{i+1}^{i+1} = R_i^{i+1}(\dot{\omega}_i^i \times l_i + \omega_i^i \times \omega_i^i \times l_i + a_i^i) \tag{3}$$

$$Am_{i+1}^{i+1} = \dot{\omega}_{i+1}^{i+1} \times pm_{i+1} + \omega_{i+1}^{i+1} \times \omega_{i+1}^{i+1} \times pm_{i+1} + a_{i+1}^{i+1} \tag{4}$$

where

$\omega_i^{\,i}$	angular velocity of i^{th} link in i^{th} frame
a_i^i	acceleration of i^{th} link in i^{th} frame
$R_i^{\,i+1}$	rotation matrix from i^{th} frame to $(i+1)^{th}$ frame
θ_i	position of i^{th} joint
Am_{i+1}^{i+1}	acceleration of accelerometer Am_{i+1} in the $(i+1)^{th}$ frame
pm_{i+1}	vector offset from joint $i+1$ to accelerometer Am_{i+1}^{i+1}
l_i	vector offset from joint i to joint $i+1$ in the frame i
$\hat{z}$	identity z vector $[0\ 0\ 1]^T$

Given the Devanit-Hartenberg parameter[9], α_i, the 3x3 rotation matrix, $R_i^{\,i+1}$, is shown below.

$$R_i^{i+1} = \begin{bmatrix} \cos\theta_{i+1} & \sin\theta_{i+1}\cos\alpha_i & \sin\theta_{i+1}\sin\alpha_i \\ -\sin\theta_{i+1} & \cos\theta_{i+1}\cos\alpha_i & \cos\theta_{i+1}\sin\alpha_i \\ 0 & -\sin\alpha_i & \cos\alpha_i \end{bmatrix} \tag{5}$$

3. SOLUTION METHOD

Each accelerometer contains information on the position, velocity, and acceleration of joints preceding the accelerometer in the kinematic chain. End-effector mounted accelerometers contain information about all joints. However, this information is mixed together in a nonlinear fashion. Except for accelerometers attached to the first joint, there is no single acceleration that is, in general, dependent on solely the position, velocity, or acceleration of the i^{th} joint. In addition, components of the acceleration due to certain joints can dominate due to kinematic configuration or trajectory. For certain configurations, the end-effector pose does not uniquely determine the joint trajectory of the robot. As a result, only utilizing end-effector mounted accelerometers is not sufficient for determining joint trajectories. Distributing accelerometers along the arm in addition to end-effector mounted accelerometers can help alleviate these problems by obtaining information at points closer to joints of interest.

The proposed solution method utilizes the distributed Cartesian accelerometers in conjunction with any working joint position sensors to recover a lost position sensor. This is done by calculating a knot point, a point in the trajectory having position, velocity, and acceleration, that will make the measured accelerations match the accelerations determined by Eq. (4). In Eq. (4), the data from the operational joint sensors is used to the extent possible, leaving only terms involving the failed joints,

$$Q(\theta_j, \dot{\theta}_j, \ddot{\theta}_j) = A, \quad j \in b \tag{6}$$

where,

b is the set of failed joints

A is the vector of accelerometer measurements $[\,Am_1^1 \ldots Am_m^m\,]^T$

Q is a set of equations of type (4) corresponding to *A*

If *r* is the number of failed joints, then the resulting system has *3m* nonlinear equations (one equation per axis, three axes per accelerometer) in *3r* (position, velocity, and acceleration of failed joints) unknowns. If this system is solvable, then the knot points for all *r* joints can be determined and the joints controlled.

A minimum criteria for solution is that there be at least as many equations as unknowns, i.e. $m \geq r$. If this condition is not satisfied, then the minimum error solution will be one using least squares. Due to the complexity of the equations, this condition is optimistic. As *r* approaches *m*, the system becomes more ill-conditioned. A better measure of predicted system performance will be discussed later.

The method discussed also relies on the existence of a known acceleration field, usually gravity. This field provides a known, constant excitation to the system accelerometers. The solution method can be applied to systems without such a field, but positions will be relative, not absolute, and drift due to small errors is likely. The accelerometers must be of the instrument type that can detect constant accelerations.

The solution method requires a computational technique for solving a system of nonlinear equations. To be practical, the solution must be calculable in real-time.

Real time in this sense implies updates at a rate fast enough for stable joint control. This requirement limits the available solution techniques. The solution technique must also be robust to sensor noise and bias. The convergence of the technique to a good solution must be predictable. Traditional nonlinear solution techniques can be applied to this problem. Some, such as steepest descent[10], were found to be too sensitive to sensor error. Others, such as nonlinear least squares[11], gave excellent results but were too time consuming for real-time implementation. A new method, continuation minimax, is developed as a compromise between solution accuracy and suitability for real-time implementation.

4. CONTINUATION MINIMAX

This method is a combination of the continuation method[12,13] and the minimax method[14,15]. The classical continuation method is based on using sequential linear programming to optimize a nonlinear function based on nonlinear criteria. In this formulation, F is the function to be optimized, P is the vector of optimization parameters, P_a is the current approximation of the optimal solution, J_c is the gradient of the constraint function, c, and J_F is the gradient of the optimization function. Placed into a linear programming problem:

$$\text{Minimize } \Delta F = J_F^T \Delta P \tag{7}$$

subject to

$$J_c \Delta P \leq -c(P_a)\Delta\alpha \tag{8}$$

The quantity α ranges from 0 to 1 in increments of $\Delta\alpha$. The step size controls the accuracy of the approximation of the optimum P, i.e. P^*. The next approximation of P^*, which is $P_a+\Delta P$, leads to a better approximation of the optimal F, i.e. F^*.

The minimax solution technique also uses linear programming. The object of the minimax solution is to find the maximum of the minimum error, γ between the calculated constraint equations and the desired value of the constraints:

$$\text{Minimize } \gamma \tag{9}$$

subject to

$$J_c \Delta P - \gamma I_{3m} \leq c(P_a) \tag{10}$$

$$-\Lambda_a \leq \Delta P_i \leq \Lambda_a \tag{11}$$

Where I_{3m} is the $3m$x$3m$ identity matrix. The parameter Λ_a controls the amount of change allowed in ΔP. The bound is heuristically changed depending on a comparison of the decrease in the function c and the linearized version. If the ratio of the two is small:

$$c(P_a + \Delta P) - c(P_a) \leq 0.25 J_c \Delta P \tag{12}$$

then $\Lambda_{a+1}=\Lambda_a/4$. If the ratio is large:

$$c(P_a + \Delta P) - c(P_a) \geq 0.75 J_c \Delta P \tag{13}$$

then $\Lambda_{a+1}=2\Lambda_a$, otherwise $\Lambda_{a+1}=\Lambda_a$. The minimax solution is iterated until a desired value of γ is achieved or only a minimal change in γ is produced.

The proposed computational procedure, continuation minimax, is a combination of both techniques. The rationale behind the combination is to capitalize on the predictable solution time, determined by $\Delta\alpha$, of the continuation method and the ability of the minimax algorithm to find the minimal error between the true $c(P^*)$ and the approximated $c(P_a)$.

In linear programming form, the continuation minimax formulation is:

$$\text{Minimize } F=\sum_{i=1}^{m} \gamma_i \tag{14}$$

subject to:

$$J_c \Delta P - diag(\gamma_1 .. \gamma_m) \leq \alpha c(P_a) \tag{15}$$

$$-J_c \Delta P - diag(\gamma_1 .. \gamma_m) \geq -\alpha c(P_a) \tag{16}$$

$$-\Lambda_a \leq \Delta P_i \leq \Lambda_a \tag{17}$$

$$\gamma_i \geq 0 \tag{18}$$

The bound Λ_a is adjusted as in the minimax formulation.

Reasons for some of the differences between existing methods and the new continuation minimax formulation are:

- The function F was linear, so it was used instead of a linearized version
- The quantity α ranges from 0 to 1 in increments $\Delta\alpha$ as before, but is used explicitly in the iteration to allow for the combination of methods
- The summation of several γ_i was chosen over only γ for better performance during sensor error. With

low sensor error, a formulation using only γ is also possible and would further decrease computational requirements.

- An upper and lower bound, Eqs. (15) and (16), are required because ideally $\gamma_i=0\ \forall i$. An upper or lower bound alone would not reduce to this condition at the ideal case.

For specific application to the position sensor fault tolerance problem, $c=Q(P_a)-A$, P_a is $[\theta_a\ \ \dot{\theta}_a\ \ \ddot{\theta}_a]$ (the joint position, velocity, and acceleration), and J_c is the gradient of c with respect P.

The continuation minimax method has an accuracy and an execution time that is controlled by the choice of $\Delta\alpha$. Other solution techniques, under certain conditions, may obtain a more optimal solution quicker. The primary advantage of the continuation minimax solution technique is its ability to give a solution to the problem in a predictable execution time. This predictability is a requirement for real-time implementation.

5. GRADIENT CALCULATION

Most nonlinear solution methods require a function gradient for rapid convergence. In this application, the continuation minimax solution method requires J_c, the gradient of c. The gradient of c is equal to the gradient of Q. The closed-form solution for J_c for an n link manipulator becomes more complex quickly as n grows. The gradient could be computed numerically using finite differences[16]. Finite differences are not as accurate as a closed-form solution. The approach taken here is to calculate the gradient recursively. The system equations are designed to be calculated recursively, so the proper application of partial derivatives and the chain rule will lead to the desired outcome.

The desired gradient has the form:

$$\frac{\partial Q}{\partial P_a}=\begin{bmatrix} \frac{\partial Am_1}{\partial\theta_1} & \cdots & \frac{\partial Am_1}{\partial\theta_n} & \frac{\partial Am_1}{\partial\dot{\theta}_1} & \cdots & \frac{\partial Am_1}{\partial\dot{\theta}_n} & \frac{\partial Am_1}{\partial\ddot{\theta}_1} & \cdots & \frac{\partial Am_1}{\partial\ddot{\theta}_n} \\ \vdots & & \vdots & \vdots & & \vdots & \vdots & & \vdots \\ \frac{\partial Am_m}{\partial\theta_1} & \cdots & \frac{\partial Am_m}{\partial\theta_n} & \frac{\partial Am_m}{\partial\dot{\theta}_1} & \cdots & \frac{\partial Am_m}{\partial\dot{\theta}_n} & \frac{\partial Am_m}{\partial\ddot{\theta}_1} & \cdots & \frac{\partial Am_m}{\partial\ddot{\theta}_n} \end{bmatrix} \tag{19}$$

For simplicity, the measurement Am_i is always in the i^{th} frame. Now,

$$\frac{\partial Am_{i+1}}{\partial\theta_k}=\frac{\partial\dot{\omega}^{i+1}_{i+1}}{\partial\theta_k}\times pm_{i+1}+\frac{\partial\omega^{i+1}_{i+1}}{\partial\theta_k}\times\omega^{i+1}_{i+1}\times pm_{i+1} \\ \omega^{i+1}_{i+1}\times\frac{\partial\omega^{i+1}_{i+1}}{\partial\theta_k}\times pm_{i+1}+\frac{\partial a^{i+1}_{i+1}}{\partial\theta_k} \tag{20}$$

$$\frac{\partial Am_{i+1}}{\partial\dot{\theta}_k}=\frac{\partial\dot{\omega}^{i+1}_{i+1}}{\partial\dot{\theta}_k}\times pm_{i+1}+\frac{\partial\omega^{i+1}_{i+1}}{\partial\dot{\theta}_k}\times\omega^{i+1}_{i+1}\times pm_{i+1} \\ \omega^{i+1}_{i+1}\times\frac{\partial\omega^{i+1}_{i+1}}{\partial\dot{\theta}_k}\times pm_{i+1}+\frac{\partial a^{i+1}_{i+1}}{\partial\dot{\theta}_k} \tag{21}$$

$$\frac{\partial Am_{i+1}}{\partial\ddot{\theta}_k}=\frac{\partial\dot{\omega}^{i+1}_{i+1}}{\partial\ddot{\theta}_k}\times pm_{i+1}+\frac{\partial a^{i+1}_{i+1}}{\partial\ddot{\theta}_k} \tag{22}$$

The partial derivative will be zero for all quantities of joint k where $k>i+1$. This a consequence of the later joints not affecting the previous links.

Equations (20),(21),and (22) are not recursive themselves, but the equations for the partials they depend on are recursive. The equations require that Eqs. (1) through (3) be calculated for all n links to determine the angular velocities, angular accelerations, and Cartesian accelerations. The partial derivatives can then be calculated in the following manner:

$$\frac{\partial\omega^{i+1}_{i+1}}{\partial\theta_k}=\begin{cases} R^{i+1}_i\frac{\partial\omega^i_i}{\partial\theta_k} & k<i+1 \\ \dot{R}^{k+1}_k\omega^k_k & k=i+1 \\ \vec{0} & k>i+1 \end{cases} \tag{23}$$

$$\frac{\partial\omega^{i+1}_{i+1}}{\partial\dot{\theta}_k}=\begin{cases} R^{i+1}_i\frac{\partial\omega^i_i}{\partial\dot{\theta}_k} & k<i+1 \\ \hat{z} & k=i+1 \\ \vec{0} & k>i+1 \end{cases} \tag{24}$$

$$\frac{\partial\omega^{i+1}_{i+1}}{\partial\ddot{\theta}_k}=\vec{0} \qquad \forall k \tag{25}$$

$$\frac{\partial\dot{\omega}^{i+1}_{i+1}}{\partial\theta_k}=\begin{cases} R^{i+1}_i\frac{\partial\dot{\omega}^i_i}{\partial\theta_k}+R^{i+1}_i\frac{\partial\omega^i_i}{\partial\theta_k}\times\dot{\theta}_k\hat{z} & k<i+1 \\ \dot{R}^{i+1}_i\dot{\omega}^i_i+\dot{R}^{i+1}_i\omega^i_i\times\dot{\theta}_k\hat{z} & k=i+1 \\ \vec{0} & k>i+1 \end{cases} \tag{26}$$

$$\frac{\partial \dot{\omega}_{i+1}^{i+1}}{\partial \dot{\theta}_k} = \begin{cases} R_i^{i+1} \frac{\partial \dot{\omega}_i^i}{\partial \theta_k} + R_i^{i+1} \frac{\partial \omega_i^i}{\partial \theta_k} \times \dot{\theta}_k \hat{z} & k < i+1 \\ R_i^{i+1} \omega_i^i \times \hat{z} & k = i+1 \\ \vec{0} & k > i+1 \end{cases} \tag{27}$$

$$\frac{\partial \dot{\omega}_{i+1}^{i+1}}{\partial \ddot{\theta}_k} = \begin{cases} R_i^{i+1} \frac{\partial \dot{\omega}_i^i}{\partial \ddot{\theta}_k} & k < i+1 \\ \hat{z} & k = i+1 \\ \vec{0} & k > i+1 \end{cases} \tag{28}$$

$$\frac{\partial a_{i+1}^{i+1}}{\partial \theta_k} = \begin{cases} R_i^{i+1} \begin{pmatrix} \frac{\partial \dot{\omega}_i^i}{\partial \theta_k} \times l_i + \frac{\partial \omega_i^i}{\partial \theta_k} \times \omega_i^i \times l_i + \\ \omega_i^i \times \frac{\partial \omega_i^i}{\partial \theta_k} \times l_i + \frac{\partial a_i^i}{\partial \theta_k} \end{pmatrix} & \mathrm{k < i+1} \\ R_i^{i+1} \left(\dot{\omega}_i^i \times l_i + \omega_i^i \times \omega_i^i \times l_i + a_i^i \right) & \mathrm{k < i+1} \\ \vec{0} & \mathrm{k > i+1} \end{cases} \tag{29}$$

$$\frac{\partial a_{i+1}^{i+1}}{\partial \dot{\theta}_k} = \begin{cases} R_i^{i+1} \begin{pmatrix} \frac{\partial \dot{\omega}_i^i}{\partial \dot{\theta}_k} \times l_i + \frac{\partial \omega_i^i}{\partial \dot{\theta}_k} \times \omega_i^i \times l_i + \\ \omega_i^i \times \frac{\partial \omega_i^i}{\partial \dot{\theta}_k} \times l_i + \frac{\partial a_i^i}{\partial \dot{\theta}_k} \end{pmatrix} & \mathrm{k < i+1} \\ \vec{0} & \mathrm{k \geq 0} \end{cases} \tag{30}$$

$$\frac{\partial a_{i+1}^{i+1}}{\partial \ddot{\theta}_k} = \begin{cases} R_i^{i+1} \left(\frac{\partial \dot{\omega}_i^i}{\partial \dot{\theta}_k} \times l_i + \frac{\partial a_i^i}{\partial \dot{\theta}_k} \right) & \mathrm{k < i+1} \\ \vec{0} & \mathrm{k \geq 0} \end{cases} \tag{31}$$

where,

$$\dot{R}_i^{i+1} = \begin{bmatrix} -\sin\theta_{i+1} & \cos\theta_{i+1}\cos\alpha_i & \cos\theta_{i+1}\sin\alpha_i \\ -\cos\theta_{i+1} & -\sin\theta_{i+1}\cos\alpha_i & -\sin\theta_{i+1}\sin\alpha_i \\ 0 & 0 & 0 \end{bmatrix} \tag{32}$$

is the derivative of R_i^{i+1} with respect to θ_{i+1}.

Using Eqs. (20) through (32), the elements of the gradient can be calculated. The previous equations have been verified against numerically calculated gradients of Q.

6. SIMULATION

Simulations were conducted to verify the solution method presented in the previous sections. A Robotics Research 807i manipulator was simulated using Matlab and Simulink numerical simulation software. The 807i is a 0.8m long robot with seven revolute joints. The full 7 DOF configuration, as shown in Fig. 1, was implemented and tested in general 6 DOF space. Two triaxial accelerometers were attached to the end-effector and one triaxial accelerometer was attached to joints 2, 4, and 6.

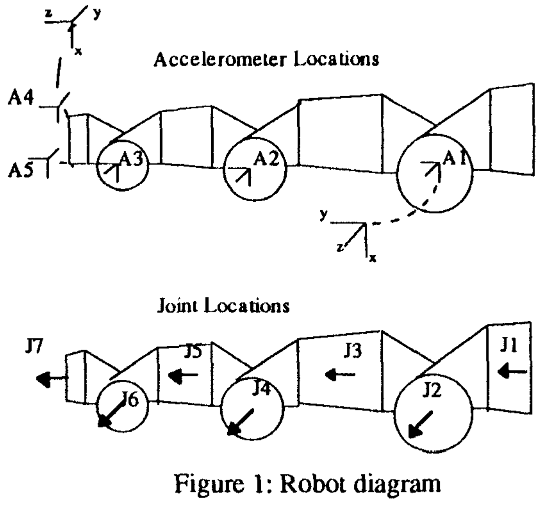

Figure 1: Robot diagram

Trajectories were generated using a standard 3-5-3 trajectory generation algorithm[9]. The desired trajectory starting and ending positions were [1.1 0.2 -0.3 0.4 -1.5 0.6 0.7] radians and [0.5 -0.3 0.0 0.2 -0.2 0.0 1.4] radians respectively. The trajectory was to be completed in 1.0 second followed by 0.7 second of zero velocity and acceleration command. This rather fast move would show the resilience of the method to dynamic effects. The calculated positions were not used as feedback to the position controller. A very low tracking accuracy controller was used in the system to show that precise, smooth trajectory following is not required by the solution method. This low precision controller accounted for the need for a 0.7 second settling period after the command was completed.

6.1 Performance Tests

To verify the method, a nonlinear least squares algorithm was used with ideal sensor data. Joints 2, 3, 5, and 6 were assumed to have failed joint sensors. The maximum position error produced was 8×10^{-7} radians. Assuming ideal data, this maximum error could be further reduced by changing the tolerance of the algorithm. To maintain the tolerances that this algorithm produced, a large number of function and gradient calls were required in addition to the overhead of the solution method. The subroutine calls required are plotted in Fig. 2. Although the number of subroutine calls is lower if the desired solution tolerance is lowered, the number of calls is unpredictable. During error spikes, the tolerance may not be achievable, causing difficulty for the solution algorithm unless properly handled.

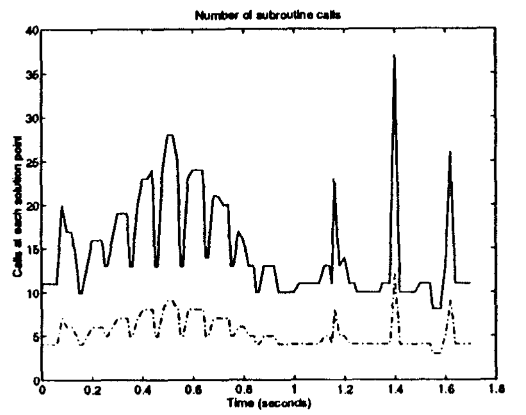

Figure 2: Number for function and gradient evaluations for Nonlinear Least Squares simulation. (Number of function evaluations is solid line)

Now, The response of the continuation minimax is analyzed with realistic sensor conditions. Accelerometers are susceptible to several error types: bias, noise, and placement error. The bias and noise can be caused by temperature, off axis accelerations, amplifier noise, vibration, or many other causes. Since the accelerometers must be placed at different points on the arm, small errors between their specified and actual positions are possible.

The primary usage of this technique is to continue operation after sensor failure without external calibration. Between the time a sensor failure occurs and when it is detected, an error will accumulate. All nonlinear optimization methods can converge to an "optimal" solution that is not the desired solution. While it is maintained that given a position reasonably close to the actual position the optimal solution will be the desired solution, higher numbers of failures will lead to a smaller convergence range. In [12] and [13], the basic continuation method has been shown not to require a very good seed value to converge to the desired optimum.

Figures 3 through 5 show the error caused individually by sensor bias, noise, and placement error. Three joints, 2,5, and 6, were assumed failed. The following plots include an error of [-0.1 -0.08 0.1] radian in the initial joint position of the failed joints. The accelerometer bias varied from axis to axis with magnitudes of -0.2973m/s^2 to 0.2949m/s^2. The accelerometer error due to noise ranged in magnitude from -0.4039m/s^2 to 0.4285 m/s^2. Accelerometer placement error ranged from 50mm to -50mm. All simulations used $\Delta\alpha$=0.333333. This choice of $\Delta\alpha$ implies 3 function and gradient calls per solution.

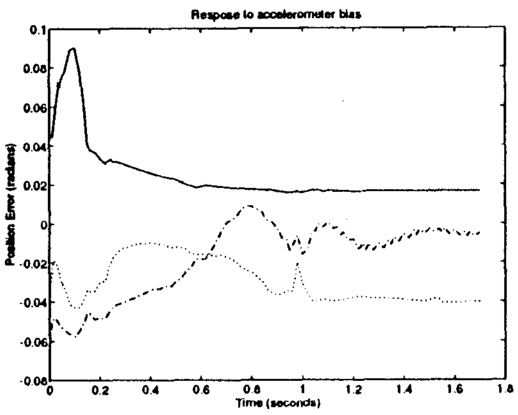

Figure 3: Response to accelerometer bias (Joint 2 error is solid line, Joint 5 is dash-dot, Joint 6 is dotted)

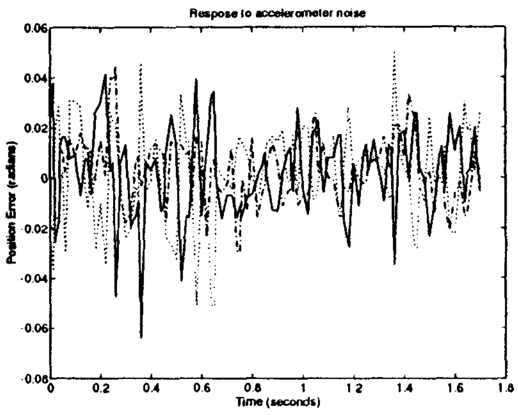

Figure 4: Response to accelerometer noise (Joint 2 error is solid line, Joint 5 is dash-dot, Joint 6 is dotted)

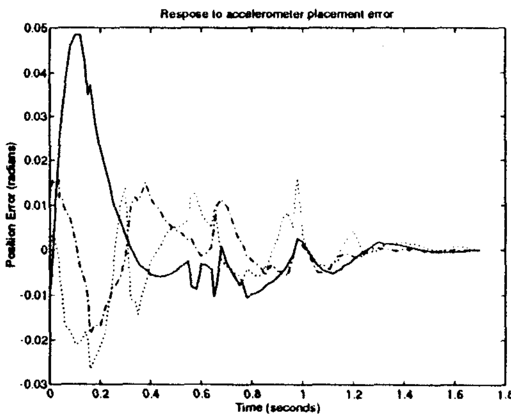

Figure 5: Response to accelerometer placement error (Joint 2 error is solid line, Joint 5 is dash-dot, Joint 6 is dotted)

To show the effect of a combination of sensor errors, Fig. 6 shows the response to sensor bias, noise, and placement errors. For variety, the same trajectory was used but joints 1,4, and 7 were assumed failed. An initial position error of [0.1 0.12 -0.05] radian was used for the failed joints. The same accelerometer bias and placement errors were used as in earlier simulations along with similar noise amplitude. It should be noted that the accelerometer configuration need not be altered to provide fault tolerance for the different set of failed joints.

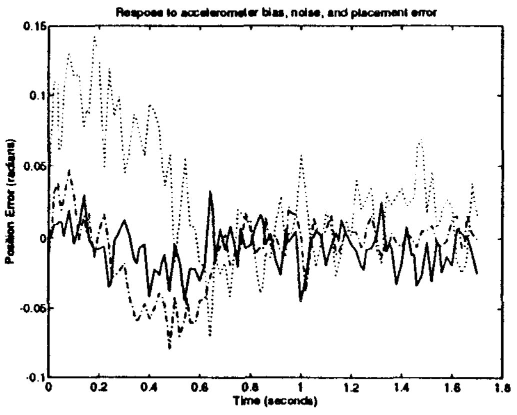

Figure 6: Response to accelerometer bias, noise, and placement error (Joint 1 error is solid line, Joint 4 is dash-dot, Joint 7 is dotted)

As with all sensor based methods, better data will produce better results.

6.2 Solution existence

Given a specific set of operational joint sensors, accelerometers, and a joint trajectory, the joint positions may or may not be recoverable. As discussed earlier, simple comparison of the number of equations and unknowns is not sufficient. Along certain trajectories, equations may not be unique due to accelerometer alignment or other numerical problems. Even when a certain situation is solvable in the ideal case, it may be too sensitive to sensor error to yield a reasonable solution. It may also be too sensitive to small parameter changes for the chosen solution method to converge quickly, if ever.

The proposed method to determine whether a sensor configuration and joint trajectory are solvable is inspection of the condition number of the gradient used by the solution. This usage is based on the application of condition numbers in linear algebra[17].

Ideally, a proposed trajectory and configuration is evaluated off-line. If acceptable condition numbers are produced along the trajectory, the trajectory will be given to the robot to execute. The main problem is determining what constitutes an acceptable condition number. Figure 7 shows what occurs to the gradient's condition number as failures occur. All plots assumed noise free accelerometers in ideal locations but did include the sensor bias used in previous simulations.

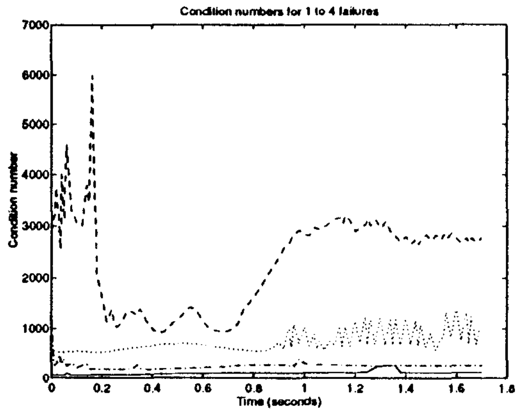

Figure 7: Condition number of gradient matrix with 1,2,3, and 4 failures. (1 failure is solid, 2 failures is dash-dot, 3 failures is dotted, 4 failures is dashed)

The error of joint 2 associated with each condition number is show in Fig. 8.

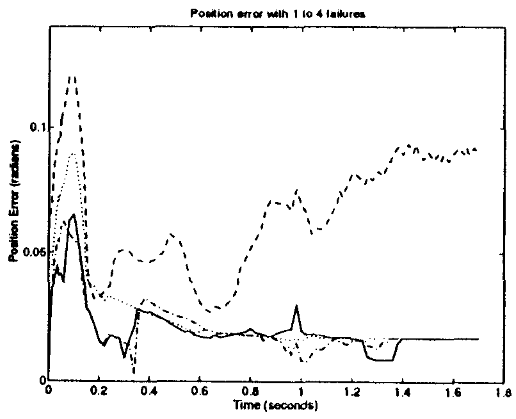

Figure 8: Error in joint 2 during 1,2,3, and 4 failures. (1 failure is solid, 2 failures is dash-dot, 3 failures is dotted, 4 failures is dashed)

Examining Figs. 7 and 8, the position error of joint 2 is relatively small for 1,2, and 3 failures. These failures have gradient condition numbers of <1500. With 4 failures, the error is still acceptable, but is becoming significantly worse. While the condition number between 0.2 and 0.7 seconds is low, it maintains an average close to 3000. Although not shown in the figures, five failures leads to an unacceptable position error which peaks at -1.4 radians. The condition number during 5 failures has values above 10^6 and an average above 10^5. It is suggested that an average condition number of below 2000 be used as a cutoff for trajectory solvability. This number can be relaxed if more robust solution techniques (at the cost of computational complexity) and better accelerometers are utilized. Further familiarization with a specific group of trajectories can also allow the designer to be confident with higher condition numbers in the area of known solvable trajectories.

As seen in the previous plots, for systems with reasonable condition numbers, the proposed solution method shows bounded error response to bounded sensor errors. The same cannot be stated for integration based techniques. A classical technique for end effector tracking uses double integration of accelerometer readings to obtain Cartesian position. Any sensor error will accumulate quickly in this technique. As a result, the method can only be applied for short periods. The proposed technique can be applied indefinitely.

There are techniques that can be used to enhance the solvability of the equations. These include:

- Engage joint brakes to eliminate joint velocity and acceleration components. This can be used to improve the initial position estimate before executing a controlled move.
- Lower desired velocities and accelerations of the system. Low energies imply lower contribution of non-acceleration field components on the system
- Take into account the condition number of the gradient as a penalty function for designing the trajectory thereby producing a more robust trajectory for the solution method.
- Check for noise and bias in the sensor data during periods where all sensors are operational. Filter coefficients identified during this period could be used to improve the accelerometer data during sensor failure.

7. CONCLUSIONS

This paper has discussed a method using Cartesian accelerometers to recover joint position sensor information lost during failure. A solution technique, continuation minimax, was proposed to solve the required nonlinear equations in real-time. A recursive technique to calculate the function gradient required for solving the nonlinear equations was detailed. Simulations were presented showing the response of the technique to non-ideal sensor data. It was shown that a set of accelerometers can protect against different sets of sensor failures throughout the robot system. By not utilizing integration in the solution method, this method neutralizes the concern of accumulating error in using accelerometers to produce position related information.

8. REFERENCES

1. Ting, Y. et. al., "Control Algorithms for Fault-Tolerant Robots", 1994 IEEE Conference on Robotics and Automation, vol. 1., pp. 910-915, 1994.
2. Visinsky, M.L. et. al., "Layered Dynamic Fault Detection and Tolerance for Robots", 1993 IEEE Conference on Robotics and Automation, vol. 1., pp. 180-187, 1993.
3. Chladek, J.T., "Fault Tolerance for Space Based Manipulator Mechanisms and Control System", First International Symposium On Measurement and Control in Robotics, 1990.
4. Paredis, C.J.J. et. al., "Kinematic Design for Fault Tolerant Manipulators", Computers and Electrical Engineering, vol. 20, no. 3, pp.211-220, 1994.
5. Kotnik, P.T. et. al., "Acceleration Feedback Control for a Flexible Manipulator Arm", 1988 IEEE Conference on Robotics and Automation, vol. 1., pp. 322-23, 1988.
6. Milford, R.I. et. al., "Experimental On-line Identification of an Elastic Robot Manipulator", Journal of Electrical and Electronics Engineering, vol. 13, no. 4, pp. 321-7, 1993.
7. Bennekers, B., and Teal, H., "Robot Runaway Protection System", 1988 IEEE Conference on Robotics and Automation, vol. 3, pp. 1474-76, 1988.
8. Aldridge, H.A., and Juang, J-N., "Joint Position Sensor Fault Tolerance in Robot Systems Using Cartesian Accelerometers", 1996 AIAA Guidance, Navigation, and Control Conference, AIAA paper number 96-3898.
9. Craig, J.J., Introduction to Robotics: Mechanics and Control, 2nd edition, Addison-Wesley, 1989.
10. Press, W.H. et. al., Numerical Recipes in C, Cambridge University Press, 1988.
11. Seber, G.A.F. and Wild, C.J., Nonlinear Regression, John Wiley and Sons, 1989.
12. Richter, S.L. and DeCarlo, R.A., "Continuation Methods: Theory and Applications", IEEE Transactions on Automatic Control, vol. AC-28, no. 6, pp. 660-665, 1983.

13. Horta, L.G. et. al., "A Sequential Linear Optimization Approach for Controller Design", Journal of Guidance, Control, and Dynamics, vol. 9, no. 8, pp. 699-703, 1986.
14. Madsen, K., "An Algorithm for Minimax Solution of Overdetermined Systems of Non-linear Equations", Journal of the Institute of Mathematics and Applications, vol. 16, pp. 321-328, 1975.
15. Bandler, J.W. et. al., "A Superlinearly Convergent Minimax Algorithm for Microwave Circuit Design", IEEE Transactions on Microwave Theory and Techniques, vol. MTT-33, no. 12, pp. 1519-1530, 1985.
16. Gerald, C.F. and Wheatly, P.O., Applied Numerical Analysis, 5th edition, Addison-Wesley, 1994.
17. Strang, G., Linear Algebra and its Applications, 3rd edition, Harcourt Brace Jovanovich, 1988.