1-21-2009

Extended Kalman Filter Implementation for the Khepera II Mobile Robot

Thomas J. Otahal

Herbert G. Tanner

DEPARTMENT OF MECHANICAL ENGINEERING

An Extended Kalman Filter Implementation for the Khepera II Mobile Robot

Thomas J. Otahal
Mechanical Engineering Department
University of New Mexico
tjotaha@unm.edu

Herbert G. Tanner
Mechanical Engineering Department
University of Delaware
btanner@udel.edu

UNM Technical Report: ME-TR-08-001

Report Date: December 2008

Abstract

The accurate estimation of robot position and orientation in real-time is one of the fundamental challenges in mobile robotics. The Extended Kalman Filter is a nonlinear real-time recursive time domain filter that combines available sensor data to produce an accurate estimate of state, and has been successfully applied to the localization problem in mobile robotics and aircraft navigation. This report describes an Extended Kalman Filter implementation for the Khepera II mobile robotics platform that seeks to produce accurate localization estimates in real-time using wheel odometry data, IR sensor range data, and compass heading data.

Contents

Chapter 1

Introduction

Mobile robots are free to move in their operating environments. They are limited in mobility by available battery power and by obstacles within the environment that the robots design cannot overcome. This freedom of motion is in contrast to a rigidly fixed industrial robot arm, whose position and orientation relative to the environment is known with a high degree of accuracy from the moment the industrial robot is switched on. Both fixed and mobile robots need accurate position and orientation data in order to perform navigation and interaction tasks with the environment and with other robots. Since mobile robots frequently change their position and orientation relative to their starting locations, they require reliable and accurate methods to obtain position and orientation data in real-time as they are moving and exploring. In words, mobile robots need to constantly ask the question, "Where am I now?"

1 Context and Motivation

Mobile robots obtain position and orientation data by sensing. Two categories of position and orientation sensors are available for mobile robots. The first category is global sensors, which are external to the mobile robot, such as the Global Positioning System (GPS), navigation beacons, and other mobile robots. Global sensors have the advantage of providing the robot with an absolute position and orientation measurement at a given time. The second category is local sensors, which are located inside the robot, such as inertial navigation, odometers, magnetic compass, and laser range finders. Mobile robots using only local sensors need to rely on dead-reckoning, which is the use of sensor information to obtain the current position and orientation from a known starting point.

The use of multiple local sensors and dead-reckoning introduces several challenges for mobile robots to obtain accurate orientation and position information. The first problem is drift, where small errors in initial measurements compound with each new sensor reading and eventually grow into large errors. The second problem is integrating data from various types of local sensors that measure different physical quantities to obtain a single measurement of position and orientation. There can also be more sensor data inputs than states for position and orientation. An example situation would be integrating laser range finder data to a known object, gyroscope heading information, and odometry data from the wheel sensors on the robot. If the position state vector is only of length three (x, y, Θ), there will be more data streams than states with potentially overlapping information between each sensor. The goal is to obtain the most accurate estimate of position and orientation at a given sensor update from these different sensors.

One possible solution to the sensor integration and drift problems is the use of Kalman filtering on the mobile robots local sensor input data. The Kalman Filter is a robust, recursive time domain filter with a low computational cost [5]. Kalman filtering reduces the effects of sensor noise, drift, and also serves to integrate multiple sensor readings and produce an accurate sate update. Kalman filtering has an extensive record of highly successful

application in engineering areas such as rocket, guidance, aircraft navigation, and was used by NASA during the initial moon landings of the 1960s.

This report presents an investigation that applies an Extended Kalman Filter (a non-linear extension of the standard linear Kalman Filter) to the position and orientation estimation problem for a Khepera II mobile robot (manufactured by the KTEAM corporation).

Figure 1: Khepera II with gripper turret.

Fig. 1 shows an illustration of the Khepera II robot with a gripper turret attached. The Khepera II is two-wheeld, with odometry sensors and eight infrared range sensors. An orientation compass sensor is also available to provide heading information.

2 Literature Review

Kalman filtering began with the publication of Ronald E. Kalman's landmark paper in 1960 [11] on an improved time domain filter (over the then dominant frequency domain Wienar filter) to remove signal noise. Acceptance of Kalman's work was greatly facilitated by the work of Stanley Schmidt from NASA Ames research, which extended Kalman's strictly linear limitation of the filter to handle a narrow class of useful non-linear functions. The resulting filter was called the Schmidt-Kalman filter (now called the Extended Kalman Filter (EKF), and the application was found for the EKF during the initial Apollo moon landings. Recent work by Julier and Uhlmann [9] has produced the Unscented Kalman Filter (UKF), which can handle a much wider range of non-linear functions than the standard EKF. A definitive theoretical treatment of non-linear functions with the Kalman Filter still awaits discovery.

Kalman filtering has been applied extensively in robot navigation and sensing, both as a means of integrating

sensor data and as a means of overcoming the effects of noise on sensor data. Specifically, the EKF has been a popular choice for robotic navigation applications, due to the EKF's ability to handle the non-linear kinematic and dynamic equations that describe mobile robot motion. An investigation by Roumeliotis [15] used an EKF to integrate sensor data for a three wheeled robot along with the kinematic equations that described the robots motion. The findings of the investigation showed the EKF provides a superior state estimate to that of only using dead-reckoning, and that the addition of global sensor data from a sun-sensor stopped the EKF from drifting due to small errors in the kinematic equations. A similar study by Alessandri [1] showed the use of onboard gyros can improve EKF estimates and overcome slip effects due to terrain. Work by Ashokaraj [2] showed the EKF is effective when only using local sensor data from accelerometers and odometers. An interesting mobile robot navigation system was devised by Sasiadek and Wang [17], which fused sensor data from an Inertial Navigation System (INS) and GPS using an EKF with fuzzy logic rules. The resulting system allowed the strengths of INS to compliment the weaknesses of GPS and vice-versa. Additional work by Roumeliotis [16] broke the navigation problem into a two stage estimation process, with the orientation estimate occurring first, then followed by the position estimate, and finally use of the EKF to fuse the results. Work by Ivanjko [7] [6] has focused on using sonar range sensors with the EKF and an occupancy grid map of the environment.

More recent work has focused on the more general navigation problem known as Simultaneous Localization and Mapping (SLAM). SLAM research addresses the challenge of robots navigating in unknown environments, where the robot needs to navigate using a map constructed in real-time. The size of the state matrices in the full SLAM problem is proportional to the number of navigation landmarks in the map. Work by Durrant-Whyte [4] showed that the number of landmarks can be reduced to those closest to the robot with affecting the statistical significance of the calculated SLAM solution. Dellaert [10] presents the ISAM system (Incremental Smoothing and Mapping) that solves the SLAM problem by means of an incrementally constructed sparse QR matrix factorization of the pose and landmark information matrix. Work by Delahoche [3] used an EKF to accurately position the robot and to locate landmarks in a map of the environment. Related work by Asensio [8] used an EKF to associate readings from a laser range finder and a trinocular vision system. The trinocular vision system was used to map larger goals for navigation, and the laser compensated for possible drifts in the dead-reckoning navigation. An outdoor navigation system was used by Ohno [13], withe the navigation map pre-computed, and the data from GPS and odometry merged by an EKF. Recent work by Rodriguez [14] has proposed solutions to inconsistencies in non-linear system linearization when applying EKF to the SLAM problem. A different approach to SLAM is called C-SLAM or cooperative SLAM, where multiple robots aid each other in determining location, orientation, and environment mappings. Recent work by Roumeliotis [12] presents a theoretical bound on the absolute performance of cooperative localization, using the EKF as a state estimator.

3 Kalman Filtering and Khepera II Kinematics

A brief summary of the basic discrete time linear Kalman Filter follows. Useful references for the Kalman Filter and the Extended Kalman Filter can be found in Welch and Bishop [18] and Gelb [5]. The Kalman Filter produces a state estimate of a discrete time linear difference equation of the form given in equation (3.1), where $x(k)$ is the current state of the system, A is the linear process dynamics matrix, B is the matrix that relates the control input $u(k-1)$, and $w(k-1)$ is the Gaussian process noise.

$$x(k) = Ax(k-1) + Bu(k-1) + w(k-1) \tag{3.1}$$

Equation (3.2) describes how the sensors produce a measurement $z(k)$ of the process given in equation (3.1).

$$z(k) = Hx(k) + v(k) \tag{3.2}$$

H is the measurement matrix with a number of rows equal to the number of sensor inputs and $v(k)$ is the Gaussian measurement noise for the sensors. The two fundamental assumptions in the linear Kalman Filter are that

the estimated discrete time process is linear, and the measurement and process noise are Gaussian distributions with zero means.

The Kalman Filter algorithm is recursive, and runs in two stages. The first stage is called the 'predictor' in which a new state estimate is produced from the previous estimate. The second stage is called the 'corrector' in which the estimate produced by the predictor stage is adjusted based on the new sensor measurements produced by equation (3.2). The algorithm only uses data from the previous state estimate, and thus requires minimal data storage and computation to produce the new state update.

The Khepera II mobile robot is two wheeled, with kinematics similar to a tank. The available sensor data from the Khepera II includes odometry (position and velocity), a compass sensor that gives heading in degrees, and eight short range infrared sensors that cover 360 degrees around the robot. The surface that the Khepera II operates on is assumed to be smooth and planar. The discrete time kinematics for the Khepera II are given in equations (3.3), (3.4), and (3.5). These equations give the Khepera robot (x, y) position and orientation Θ.

$$x(k+1) = x(k) + \frac{(v_r + v_l)}{2} T \cos(\Theta(k+1)) \tag{3.3}$$

$$y(k+1) = y(k) + \frac{(v_r + v_l)}{2} T \sin(\Theta(k+1)) \tag{3.4}$$

$$\Theta(k+1) = \Theta(k) + \frac{(v_r - v_l)}{d} T \tag{3.5}$$

Where v_r and v_l are the right and left wheel velocities, d is the distance between the wheels, and T is the sampling time increment.

The Khepera kinematic equations are non-linear, and thus do not fit the model given in equation (3.1). The linear Kalman Filter cannot be applied. The EKF can be used with the Khepera kinematic equations to handle the non-linear equations. The EKF follows the same basic algorithm as the linear Kalman Filter, but the nonlinear equations governing the process and possibly the measurement dynamics are linearized by means of a Jacobian derivative matrix computed at each time increment update. Equation (3.6) and (3.7) show the non-linear process and measurement equations, which are analogous to the linear equations (3.1) and (3.2).

$$x(k) = f(x(k-1), u((k-1)) + w(k-1) \tag{3.6}$$

$$z(k) = h(x(k)) + v(k) \tag{3.7}$$

Equations (3.6) and (3.7) can be linearized in the EKF at each time increment by computing the Jacobian derivative matrix with respect the three state variables (x, y, Θ).

Chapter 2

An Extended Kalman Filter for the Khepera II

The Extended Kalman Filter implementation shown in this chapter is based on the work of Ivanjko in [6] and [7], where the authors describe an Extended Kalman Filter for a mobile robot with sonar sensors. The authors use an occupancy grid to describe the environment around the robot, and the straight line distance function for each sonar sensor to the nearest occupied grid cell is used as the measurement function h. The EKF implementation for the Khepera II in this report makes a modification to the measurement function h to admit Khepera II IR sensors and the compass sensor, but otherwise uses the same EKF implementation described in the work by Ivanjko.

1 EKF Equations for the Khepera II

The state estimate vector for the EKF during time step k is given in equation (1.1). The horizontal and vertical position of the robot are $x(k)$ and $y(k)$, respectively. The robot orientation is given by $\Theta(k)$.

$$\hat{x}(k) = \begin{pmatrix} x(k) \\ y(k) \\ \Theta(k) \end{pmatrix} \tag{1.1}$$

The control inputs for the Khepera robot are the left and right wheel velocities specified by v_l and v_r, respectively. The control vector $u(k)$ in equation (1.4) is comprised of the translational displacement (1.2) and rotational displacement (1.3) during one time step, with d the Khepera II axle length between the wheels, and T the amount of time in one discrete time increment.

$$\Delta D(k) = \frac{(v_l + v_r)}{2} T \tag{1.2}$$

$$\Delta\Theta(k) = \frac{(v_r - v_l)}{d} T \tag{1.3}$$

$$u(k) = \begin{pmatrix} \Delta D(k) \\ \Delta\Theta(k) \end{pmatrix} \tag{1.4}$$

The non-linear state transition function (1.5) is a function of the current state $\hat{x}(k)$ and the control input $u(k)$. The state transition function moves the current state $\hat{x}(k)$ to state $\hat{x}(k+1|k)$ according to the Khepera II kinematic

equations. The 'predict' stage of the EKF advances the state from $\hat{x}(k)$ to $\hat{x}(k+1|k)$ according to equation (1.5) before the 'correction' stage of the filter that adjusts the estimate based on new sensor input. After the 'correction' stage of the filter, the state estimate is written as $\hat{x}(k+1|k+1)$.

$$f(x(k),u(k)) = \begin{pmatrix} x(k) + \Delta D(k)\cos(\Theta(k) + \Delta\Theta(k)) \\ y(k) + \Delta D(k)\sin(\Theta(k) + \Delta\Theta(k)) \\ \Theta(k) + \Delta\Theta(k) \end{pmatrix} \tag{1.5}$$

It is also possible to express the state update equations by integration the continuous time forms of the robot kinematic equations given in equations (1.6), (1.7), and (1.8). The integration is performed under the assumption that the control inputs $\Delta D(k)$ and $\Delta\Theta(k)$ are constant for step k over a time interval of length T.

$$\dot{x} = \frac{\Delta D(k)}{T}\cos(\Theta(t)) \tag{1.6}$$

$$\dot{y} = \frac{\Delta D(k)}{T}\sin(\Theta(t)) \tag{1.7}$$

$$\dot{\Theta} = \frac{\Delta\Theta(k)}{T} \tag{1.8}$$

The continuous time integrated state update equations (zero initial conditions) are given in equations (1.9), (1.10), and (1.11). These equations are exact if the constant control input assumption is valid. The EKF implementation used in this report made use of the discrete state update equation given in equation (1.5).

$$x(t) = \frac{\Delta D(k)}{\Delta\Theta(k)}\sin(\Delta\Theta(k)t) \tag{1.9}$$

$$x(t) = \frac{\Delta D(k)}{\Delta\Theta(k)}(1 - \cos(\Delta\Theta(k)t)) \tag{1.10}$$

$$\Theta(t) = \Delta\Theta(k)t \tag{1.11}$$

Equation (1.12) is the Jacobian derivative of the state transition function (1.5) with respect to the two control variables $\Delta D(k)$ and $\Delta\Theta(k)$.

$$\nabla f(k) = \begin{pmatrix} -\Delta D(k)\sin(\Theta(k) + \Delta\Theta(k)) & \cos(\Theta(k) + \Theta(k)) \\ \Delta D(k)\cos(\Theta(k) + \Delta\Theta(k)) & \sin(\Theta(k) + \Theta(k)) \\ 1 & 0 \end{pmatrix} \tag{1.12}$$

The process noise covariance matrix (1.13) is formed with the state transition Jacobian (1.12) and the variances for the assumed sources of noise in the robot motion kinematics. The variance σ_D^2 models the noise in the robots translational motion, and the variance $\sigma_{\Delta\Theta}^2$ models the noise in the robots rotational motion.

$$Q(k) = \nabla f(k) \begin{pmatrix} \Delta\Theta(k)^2\sigma_{\Delta\Theta}^2 & 0 \\ 0 & \sigma_D^2 \end{pmatrix} \nabla f(k)^T \tag{1.13}$$

Equation (1.14) is the Jacobian of (1.12) with respect to the state variables $x(k)$, $y(k)$, and $\Theta(k)$.

$$\nabla f^{'}(k) = \begin{pmatrix} 1 & 0 & -\Delta D(k)\sin(\Theta(k)+\Delta\Theta(k)) \\ 0 & 1 & \Delta D(k)\cos(\Theta(k)+\Delta\Theta(k)) \\ 0 & 0 & 1 \end{pmatrix} \tag{1.14}$$

The process error covariance matrix (1.15) is updated from the previous time step via equations (1.14) and (1.13). The matrix $P(k+1|k)$ is the error covariance estimate (the estimated amount of error in the estimate of each state variable) produced during the 'predict' stage of the EKF, before any measurement data has been applied during the 'correct' stage.

$$P(k+1|k) = \nabla f^{'}(k) P(k|k) \nabla f^{'}(k)^T + Q(k) \tag{1.15}$$

Equation (1.16) creates the innovation matrix $S(k+1)$ from the expected measurement function Jacobian $\nabla h(k)$. Please see section 2 for a description of the measurement function $h(k)$. The matrix $R(k+1)$ is the measurement noise covariance matrix, which is a diagonal matrix with entries along the diagonal for the error variances in the IR sensors and compass sensor. For the Khepera II with eight IR sensors and one compass sensor, $S(k+1)$ will be a nine by nine matrix.

$$S(k+1) = \nabla h \, P(k+1|k) \nabla h^T + R(k+1) \tag{1.16}$$

Equation (1.17) creates the Kalman gain matrix $K(k+1)$. Computing the Kalman gain is the most computationally expensive stage of the EKF, as the inverse of the innovation matrix $S(k+1)$ must be computed. The size of the innovation matrix is proportional to the number of sensor measurements used in the EKF. An intuitive way to think of the Kalman gain magnitude based on equation (1.17), is that the Kalman gain magnitude is directly proportional to the uncertainty in the process dynamics $P(k+1|k)$, and inversely proportional to the uncertainty in the sensor measurement dynamics $S^{-1}(k+1)$.

$$K(k+1) = P(k+1|k) \nabla h^T S^{-1}(k+1) \tag{1.17}$$

The 'correct' stage of the EKF updates the process error covariance matrix (1.18) with the Kalman gain and the expected innovation from sensor data.

$$P(k+1|k+1) = P(k+1|k) - K(k+1) S(k+1) K^T(k+1) \tag{1.18}$$

The final estimate $\hat{x}(k+1|k+1)$ is produced by equation (1.19). The Kalman gain is used to magnify or diminish the difference between the actual sensor readings in $z(k+1)$ and the expected sensor readings in $h(k+1)$, with the resulting being added to the 'predict' estimate $\hat{x}(k+1|k)$ formed at the beginning of the EKF iteration.

$$\hat{x}(k+1|k+1) = \hat{x}(k+1|k) + K(k+1)(z(k+1) - h(k+1)) \tag{1.19}$$

2 Map and Measurement Model

The non-linear measurement function $h(k)$ in the EKF gives the expected sensor measurement values for the current state estimate at time step k. Equation (2.1) gives the measurement function used for the Khepera II in this report, based on work by Ivanjko in [6] and [7].

$$h_i(\hat{x}(k+1|k), p_i) = \begin{cases} \sqrt{(x_i - x(k+1|k))^2 + (y_i - y(k+1|k))^2} & \text{if } i = 1..8, \text{ IR sensors,} \\ \Theta(k+1|k) & \text{if } i = 9, \text{ compass sensor} \end{cases} \tag{2.1}$$

The compass sensor gives a direct reading of heading in radians, thus the expected sensor value is simply the value of $\Theta(k+1|k)$ from the EKF 'predict' stage. The eight IR sensors produce eight distinct distance measures to obstacles in a straight line originating from the IR sensor in question. The measurement function is the distance from the current Khepera position to the nearest map obstacle for each IR sensor. The function input parameter $p_i = (x_i, y_i)$ provides the location of the nearest obstacle for each IR sensor from a consultation of the environment map. In the work by Ivanjko, the environment map was specified by an occupancy grid, in this report the environment map is specified by a list of line segments defining the map walls and the map boundaries. Each IR sensor is modeled as a line segment originating from the current position of the Khepera, with a length that extends outside of the map boundary, and a direction based on the current orientation of the Khepera. A line segment intersection algorithm is run against the environment map line segment list for each IR sensor line segment to obtain the value of $p_i = (x_i, y_i)$ for the IR sensor in question. The resulting eight values are then used to find the value of the expected measurement function distance for the IR sensors. The environment map and the measurement function are decoupled in way that makes the mathematics tractable and the map description quite flexible, i.e. line segment lists can describe a vast array of possible environment map configurations.

Equation (2.2) describes the sensor dynamics with Gaussian noise for each sensor modeled in $w_i(k)$.

$$z_i(k) = h_i(\hat{x}(k+1|k), p_i) + w_i(k) \tag{2.2}$$

Equation (2.3) gives the Jacobian derivative of the measurement function (2.1) with respect to the state variables ($x(k+1|k), y(k+1|k), \Theta(k+1|k)$). The measurement Jacobian matrix $\nabla h(k)$ is constructed by stacking the derivative values for each sensor into a nine by three matrix.

$$\nabla h_i(\hat{x}(k+1|k), p_i) = \begin{cases} \begin{pmatrix} \frac{x(k+1|k)-x_i}{\sqrt{(x_i-x(k+1|k))^2+(y_i-y(k+1|k))^2}} \\ \frac{y(k+1|k)-y_i}{\sqrt{(x_i-x(k+1|k))^2+(y_i-y(k+1|k))^2}} \\ 0 \end{pmatrix}^T & \text{if } i = 1..8, \text{ IR sensors,} \\ \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}^T & \text{if } i = 9, \text{ compass sensor} \end{cases} \tag{2.3}$$

Chapter 3

Results and Conclusion

The EKF equations for the Khepera II described in Chapter 2 were implemented as a combination of Matlab scripts and C program functions for both pure computer simulation, and for experiments on the Khepera II robot hardware. The following two sections describe results from simulation and experiments.

1 Simulation Results

This section presents simulation results of Khepera II straight line motion using realistic error variances for the IR sensors and the compass sensor. Please see Appendex A for a description of the calibration procedure used for the Khepera II IR sensors and compass sensor to determine the error variances in these sensors.

Simulated straight line horizontal motion of the Khepera II over a distance of one meter is shown on the following pages in Figures 2, 3, 4, and 5. All units are in millimeters. The Khepera II desired velocity over the motion was 10 millimeters per second. Zero mean Gaussian noise with a variance of 0.0001 was injected into the the control vector (equation (1.4)) to simulate friction and disturbances in the wheel-surface contact. This choice of variances was based on observation of actual Khepera II straight motion deviation. Figure 2 shows the straight line motion simulation using only odometry without the EKF turned on. The Khepera II deviates from the the straight line path by about 1 centimeter at the end of the motion, in agreement with actual observation. The RMS error between the actual and estimated position is approximately $2.5e6$.

Figure 3 shows a simulation of straight line motion under the same conditions as in Figure 2, but in this case the EKF is turned on with only the compass sensor active. The simulated compass sensor measurements are injected with zero mean gaussian noise with a variance of 0.0049. This choice of variance comes from the observed error in the compass sensor, which is approximately +/- 4 degrees. The EKF process error covariance matrix is initialized to a three by three identity matrix. The EKF compass simulation shows less deviation than the pure odometry simulation, approximately half a centimeter, and a reduced RMS error of approximately $6e5$.

Figure 4 again shows a simulation of straight line motion, but with the EKF and the IR sensors turned on. The compass sensor is turned off. A wall extending along the X axis is located 50 millimeters in the positive Y direction. The IR sensor measurements are injected with zero mean Gaussian noise with variances of 100.0. This choice of variance is based on the observed error in the IR sensors, approximately +/- 1 centimeter. The EKF IR sensors simulation shows the same deviation as the compass sensor simulation, approximately half a centimeter, and a comparable RMS error of approximately $4e5$. The IR sensors are in general very sensitive, and difficult to tune properly in the EKF, especially in comparison with the compass sensor. In particular, the threshold parameter which governs how large the difference can be between the expected and the actual IR sensor readings must be carefully selected for stable performance. In this case, the threshold parameter was selected as 20 millimeters.

Anything greater than this value was thresholded. Significantly smaller or larger threshold values caused the EKF to become unstable. The threshold problem is discussed briefly by Ivanjko in [7].

The final straight line motion simulation result shown in Figure 5. Both the compass and IR sensors are turned on, with the same conditions as the previous simulations. The EKF fuses the available sensor data to produce a trajectory with almost no deviation and a very small RMS error of approximately 500.

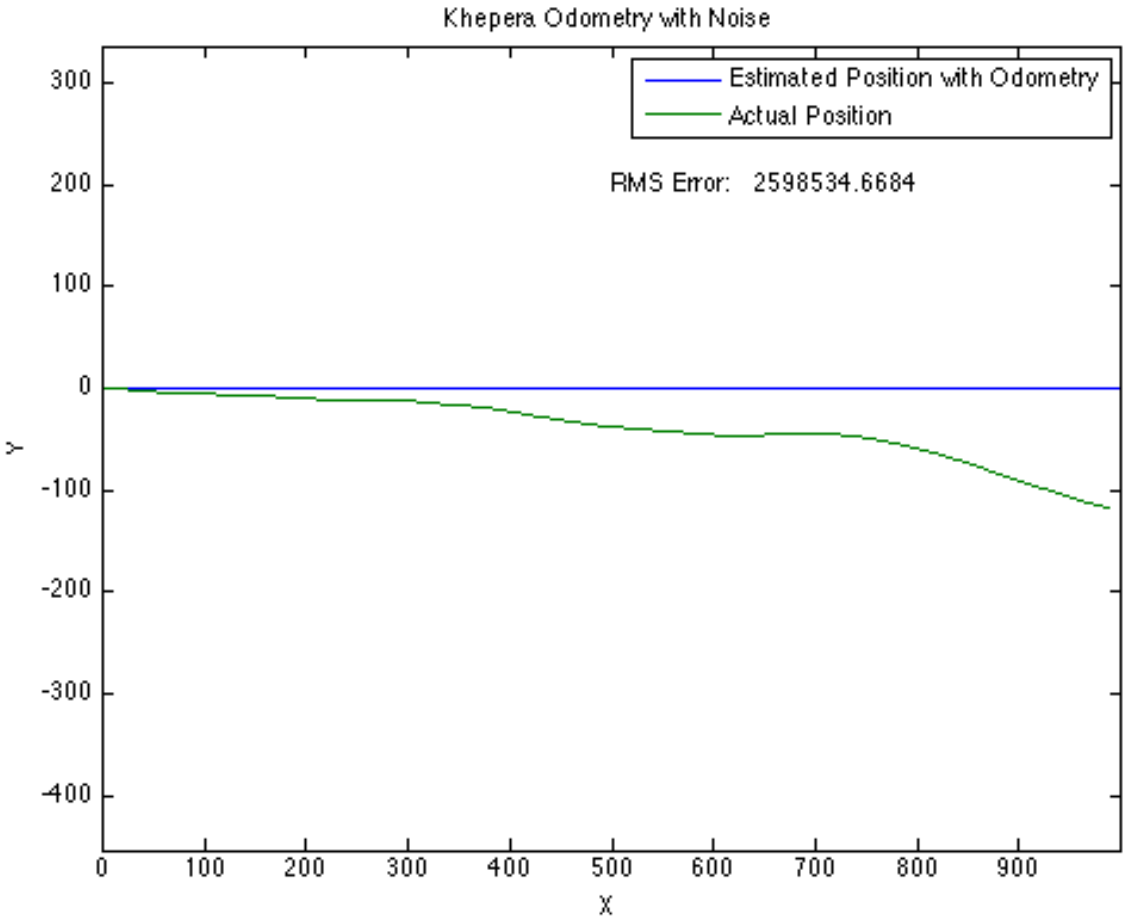

Figure 2: Simulation of Khepera motion with odometry and added noise.

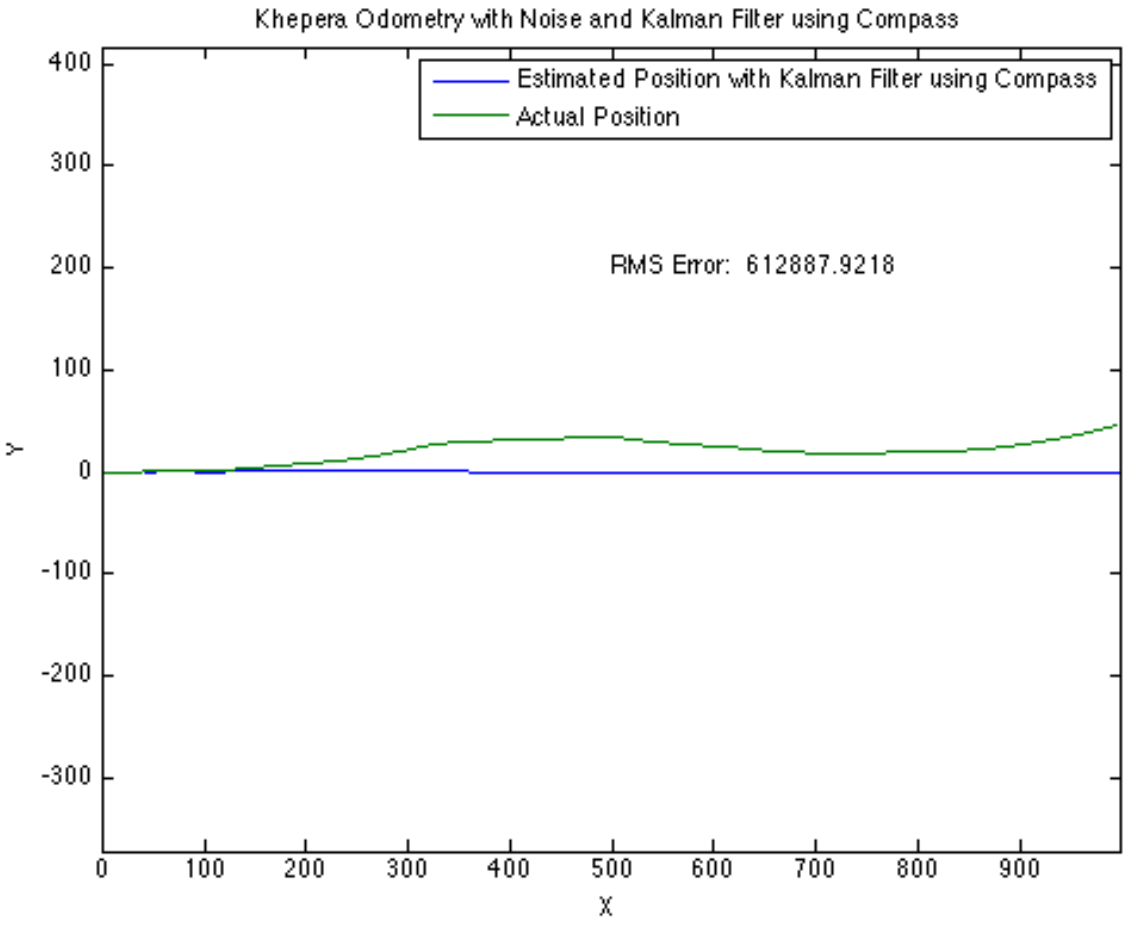

Figure 3: Simulation of Khepera motion with EKF using compass sensor only.

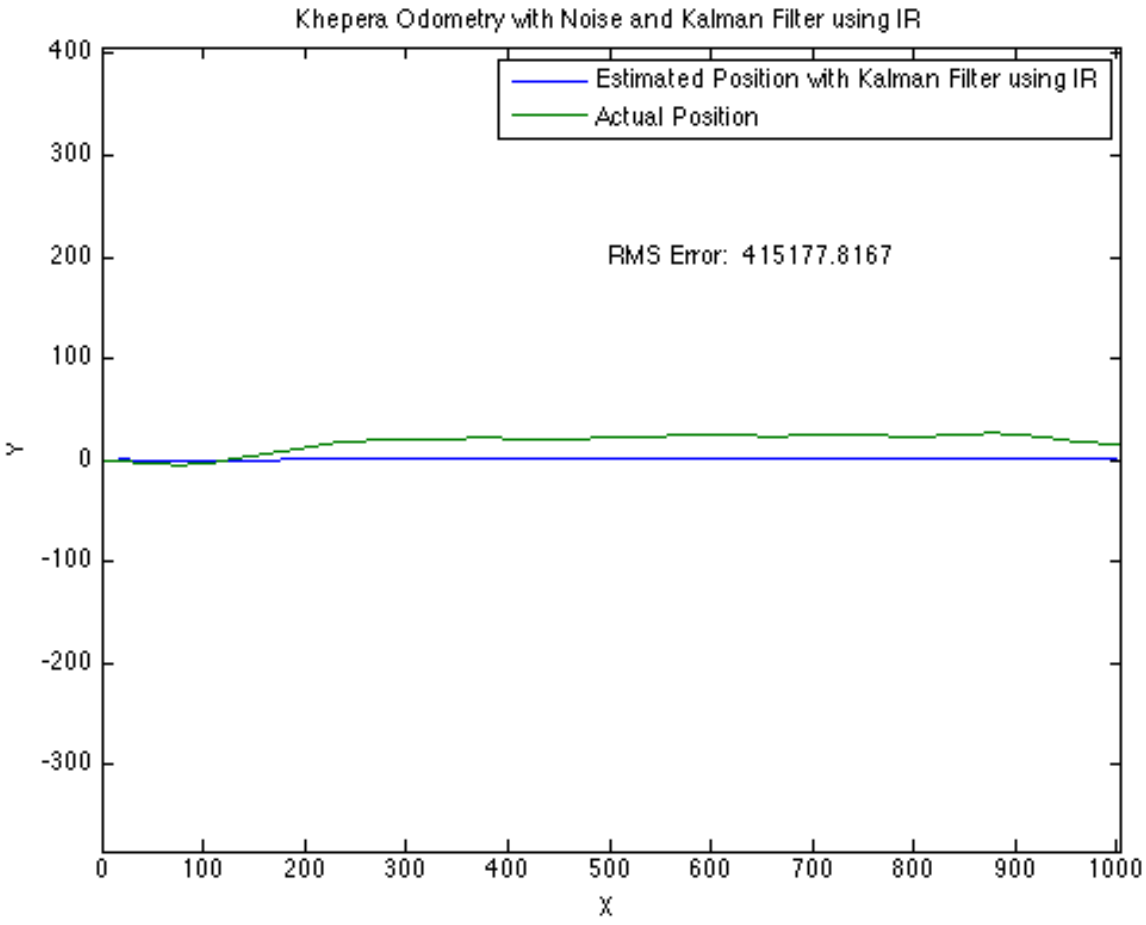

Figure 4: Simulation of Khepera motion with EKF using IR sensors only.

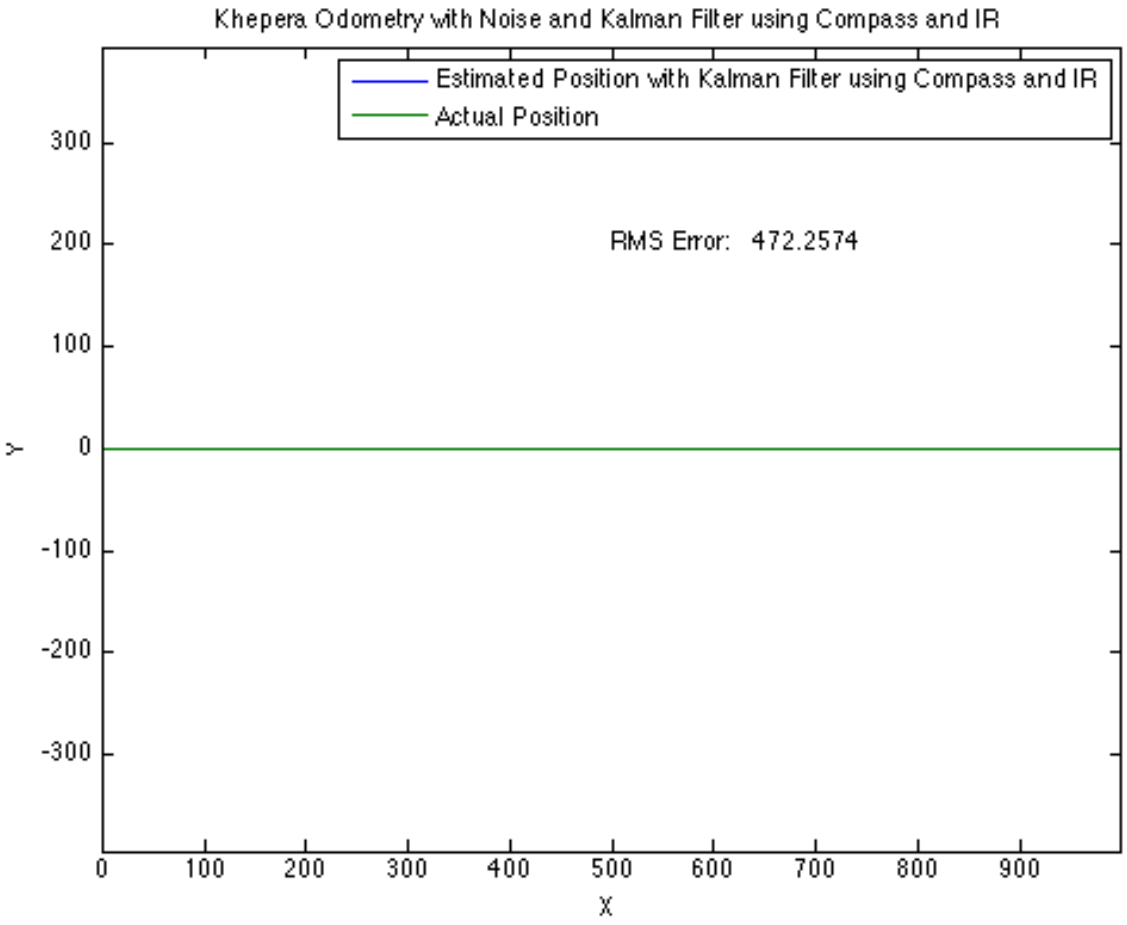

Figure 5: Simulation of Khepera motion with EKF using IR and compass sensors.

2 Experimental Results

Appendix A describes the calibration procedure used for the compass sensor and the IR sensors. The sensors were calibrated in anticipation of conducting experiments with the Khepera II robot hardware to match the results shown in the simulation section of this report.

The first attempt at getting the EKF to run on the Khepera II was to compile all of the EKF code to run directly on the Khepera II hardware. This approach did not work for two reasons. First, the amount of time to complete one iteration of the EKF on the Khepera II hardware averaged around 2 seconds, which is well outside of typical real-time update rates of about 0.1 or 0.2 seconds. Second, the numerical routine used to invert the innovation matrix would fail periodically due to numerical stability problems, most likely due to the lack of hardware floating point processing on the Khepera II.

The second attempt at implementing the EKF for the Khepera II involved running the EKF on a laptop computer with the Khepera II acting as a receiver of velocity command input and a transmitter of odometry, IR sensor, and compass data back to the laptop. The communication was done over a serial port connection between the laptop and the Khepera II.

The second implementation worked well by overcoming the problems with update rate and numerical stability encountered in the first implementation, but the available Khepera II hardware exhibited electrical problems with the serial port connection over sustained motion experiments, and also while downloading programs to the robot. This made further studies impossible. If new Khepera II hardware were available, the work could be continued to show agreement with the simulation section of this report.

3 Conclusion

This report presented an implementation of the EKF for a two-wheeled mobile robot with IR distance sensors and a compass sensor that gives heading information. The simulation section of this report showed that the EKF implementation works when properly tuned for sensor error variances and robot motion error variances. The EKF presented could be used on any two-wheeled mobile robot with similar distance measurement and orientation sensors, not just the Khepera II with the hardware presented here. It is unfortunate that the Khepera II hardware available did not function well enough to confirm the results of the simulations. If newer robot hardware is available, it may be possible to run the EKF entirely on the robot, and do away with the communication link to a laptop computer.

References

[1] A. Alessandri, G. Bartolini, P. Pavanati, E. Punta, and A. Vinci. An application of the extended kalman filter for integrated navigation in mobile robotics. *American Control Conference, 1997. Proceedings of the 1997*, 1:527–531 vol.1, Jun 1997.

[2] I. Ashokaraj, P. Silson, and A. Tsourdos. Application of an extended kalman filter to multiple low cost navigation sensors in wheeled mobile robots. *Sensors, 2002. Proceedings of IEEE*, 2:1660–1664 vol.2, 2002.

[3] Laurent Delahoche, Claude Pégard, El Mustapha Mouaddib, and Pascal Vasseur. Incremental map building for mobile robot navigation in an indoor environment. In *ICRA*, pages 2560–2565, 1998.

[4] Gamini Dissanayake, Stefan B. Williams, Hugh Durrant-Whyte, and Tim Bailey. Map management for efficient simultaneous localization and mapping (slam). *Auton. Robots*, 12(3):267–286, 2002.

[5] A. Gelb. *Applied optimal estimation*. MIT Press, 1974.

[6] E. Ivanjko and I. Petrovic. Extended kalman filter based mobile robot pose tracking using occupancy grid maps. *Electrotechnical Conference, 2004. MELECON 2004. Proceedings of the 12th IEEE Mediterranean*, 1:311–314 Vol.1, May 2004.

[7] E. Ivanjko, I. Petrović, and M. Vašak. Sonar-based pose tracking of indoor mobile robots. *Automatika - Journal for control, measurement, electronics, computing and communications, Vol. 45, No. 3-4*, pages 145–154, 2004.

[8] J. M. M. Montiel J. R. Asensio and L. Montano. "goal directed reactive robot navigation with relocation using laser and vision". In *In Proc. 1999 IEEE Int. Conf. On Robotics and Automation*, pages pp. 2905 – 2910, Detroit, Michigan, 10-15 May 1999.

[9] Simon J. Julier and Jeffrey K. Uhlmann. A new extension of the kalman filter to nonlinear systems. In *In Int. Symp. Aerospace/Defense Sensing, Simul. and Controls*, pages 182–193, 1997.

[10] M. Kaess, A. Ranganathan, and F. Dellaert. iSAM: Incremental smoothing and mapping. *Accepted by IEEE Trans. on Robotics*, 2008.

[11] R. E. Kalman. A new approach to linear filtering and predication problems. *Transactions of the ASME – Journal of Basic Engineering*, 82:45–46, 1960.

[12] A. I. Mourikis and S. I. Roumeliotis. Performance analysis of multirobot cooperative localization. *IEEE Transactions on Robotics*, 22(4):666–681, Aug. 2006.

[13] Kazunori Ohno, Takashi Tsubouchi, Bunji Shigematsu, and Shin'ichi Yuta. Differential gps and odometry-based outdoor navigation of a mobile robot. *Advanced Robotics*, 18(6):611–635, 2004.

[14] Diego Rodríguez-Losada, Fernando Matía, Agustín Jiménez, and Ramón Galán. Consistency improvement for slam - ekf for indoor environments. In *ICRA*, pages 418–423. IEEE, 2006.

[15] Stergios I. Roumeliotis and George A. Bekey. An extended kalman filter for frequent local and infrequent global sensor data fusion. In *in SPIE International Symposium on Intelligent Systems and Advanced Manufacturing*, pages 11–22, 1997.

[16] Stergios I. Roumeliotis, Gaurav S. Sukhatme, and George A. Bekey. Circumventing dynamic modeling: Evaluation of the error-state kalman filter applied to mobile robot localization. In *In Proceedings of the 1999 IEEE International Conference in Robotics and Automation*, pages 1656–1663, 1999.

[17] J.Z. Sasiadek and Q. Wang. Sensor fusion based on fuzzy kalman filtering for autonomous robot vehicle. *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, 4:2970–2975 vol.4, 1999.

[18] Greg Welch and Gary Bishop. An introduction to the kalman filter. Technical report, University of North Carolina at Chapel Hill, Chapel Hill, NC, USA, 1995.

Appendix A

Khepera Compass and IR Sensor Calibration

A Compass Sensor Calibration

The compass sensor used for this report is the R1525 magnetic Hall effect sensor manufactured by the Dinsmore company. The compass is mounted on the I/O turret of the Khepera II and provides two analog output voltages as the Khepera II is turned 360 degrees. The compass senses the earths magnetic field and is not disturbed by nearby magnetic field sources. The manufacturer states the compass is accurate to within +/- 1 degree.

The calibration of the compass sensor turned the Khepera II in a complete circle, and took compass output readings every 20 degrees of rotation. Figure 6 shows a plot of the data gathered from this experiment. The maximum and minimum analog output values were recorded to scale the output data between -1 and 1. The sensor output functions are sin() and cos() curves, so the arctan() function can be used to recover the heading in radians. If s_3 is the sin() curve output scaled between -1 and 1, and s_4 is the cos() curve output scaled between -1 and 1, then the output heading h in radians can be found with equation (A.1).

$$x = \arctan(s_3, s_4) \begin{cases} h = x + 2\pi & \text{if } x < 0 \\ h = x & \text{if } x >= 0 \end{cases} \tag{A.1}$$

Repeated tests of the compass reading while pointing towards true north, showed the compass sensor error is approximately +/- 4 degrees.

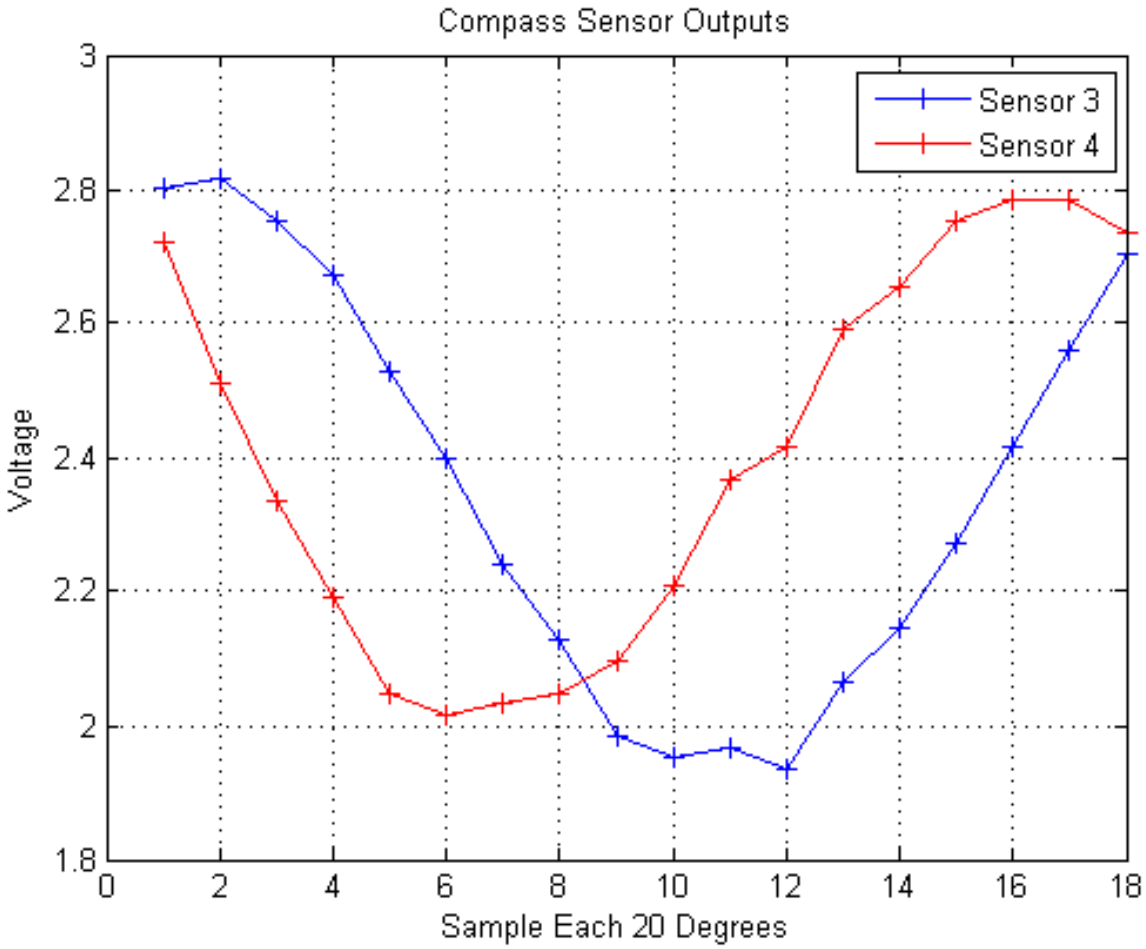

Figure 6: Compass sensor output voltages over a 360 degree rotation.

B IR Sensor Calibration

The Khepera II has eight short range IR sensors arrayed around the outside of the robot for the purpose of detecting obstacles. The IR sensors have an effective range of about 8 centimeters. With proper calibration, the IR sensors can be used to measure the distance to objects close to the robot.

The calibration procedure used in this report gathered data at one centimeter increments for each IR sensor. A white reflective surface was placed at one centimeter from an IR sensor and moved in one centimeter increments out to a distance of eight centimeters. The measurement procedure was repeated five times and an average value was calculated at each distance. The results of this experiment are shown plotted in Figure 7.

An exponential regression curve was fit to the data for each IR sensor. The regression curve fits are shown plotted in 7 along with the experimental data. The exponential regression equation has the form given in (B.1) where m is the IR sensor measurement, d is the distance to the obstacle, and a and r are parameters to be determined by regression analysis.

$$m = ar^d \tag{B.1}$$

Equation (B.1) can be expressed as a linear equation of the form $y = mx + b$, where the slope $m = \log(r)$ and the intercept $b = \log(a)$. Fitting this linear equation to the experimental data yields two parameters, a and r, for each IR sensor. Inverting equation (B.1) produces a distance value for an input IR sensor measurement. The error in the IR sensors was found to be +/- 1 centimeter after the IR sensor calibration was completed.

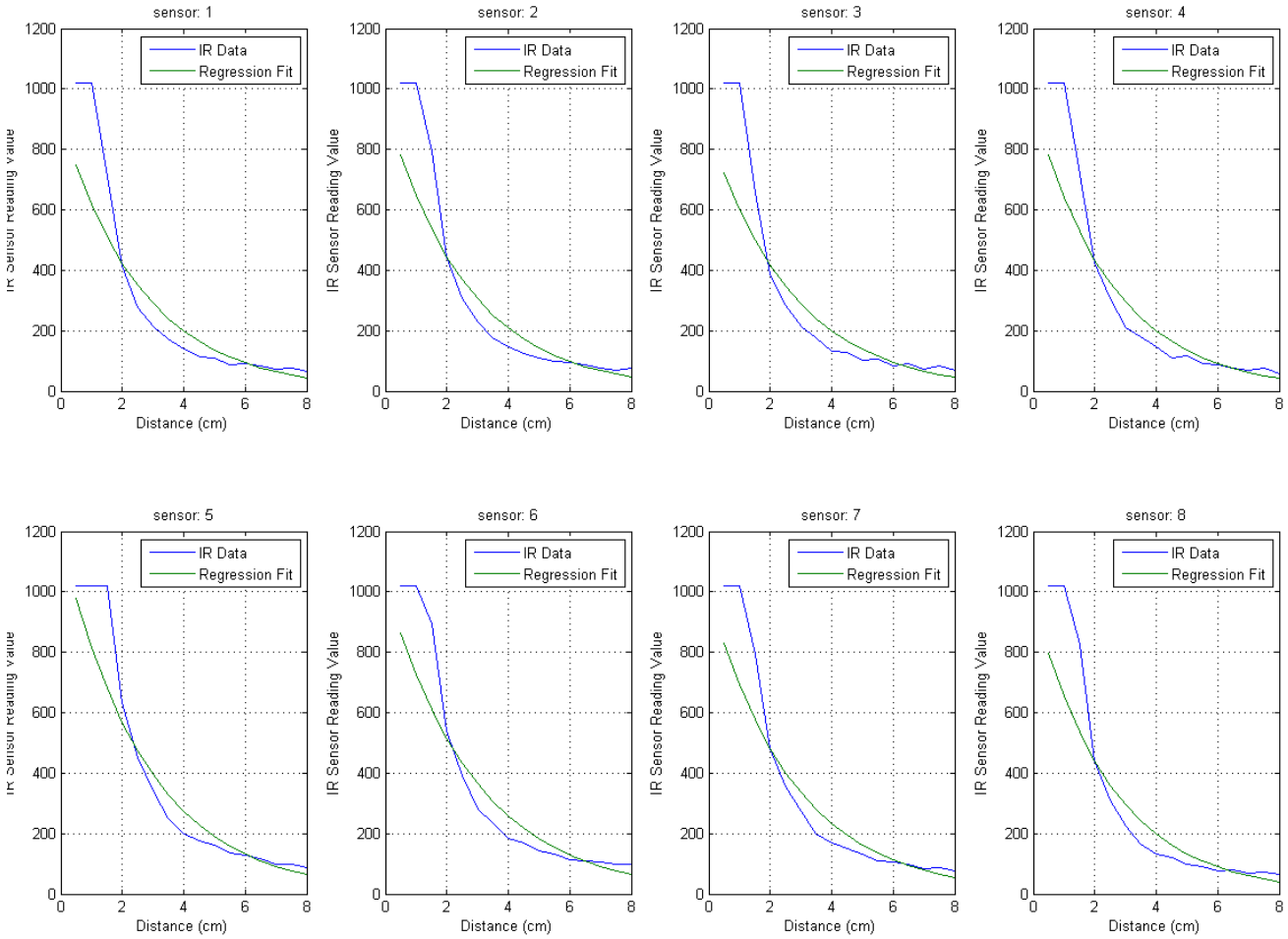

Figure 7: IR sensor value versus distance to a reflective object for the eight IR sensors.