# Range-Only Robot Localization and SLAM with Radio

Derek Kurth

CMU-RI-TR-04-29

May 2004

Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213
dekurth@andrew.cmu.edu

© Carnegie Mellon University

**Abstract**

This paper presents three localization algorithms using radio beacons that provide the ability to measure range only. Obtaining range from radio has the advantage that line of sight between the beacons and the transponder is not required, and the data association problem can be completely avoided. An extended Kalman filter, a particle filter, and a nonlinear sliding batch method are used to fuse range data with dead reckoning data collected from a real system, and results are demonstrated. The extended Kalman filter is also applied to the SLAM problem.

**Acknowledgements**

This work would have been impossible without the help of some very talented people. Thanks to Peter Corke, Thanasis Kehagias, Evgeni Kiriy, Stephan Roth, Brad Hamner and Peter Abeles. I thoroughly enjoyed working with you!

Special thanks to Sanjiv Singh and George Kantor, for much expertise and robo-wisdom.

Thanks also to my parents, and to my wonderful wife, Karianne, who encouraged me all along. *SDG.*

# 1 Introduction

The ability to solve the problem of robot localization efficiently will have a tremendous impact on the ways in which robots can be integrated with daily living. Many tasks for which robots are seemingly well-suited require a high level of precision in localization before such application can occur in the field. For example, a robot delivering mail in an office building, mowing a golf course, or mapping an underground mine needs to maintain an accurate estimate of its location to successfully complete its task.

Many methods exist for estimating a mobile robot's pose (position and attitude). One solution is to obtain absolute position coordinates via the Global Positioning System (GPS), which uses the location of satellites orbiting Earth to triangulate longitude and latitude. This approach is limited, however, to environments in which a clear line of sight to the satellites is available. Robots navigating inside buildings or underground cannot receive GPS data, and even in outdoor environments structures and foliage can affect the ability to communicate with GPS satellites.

Deduced reckoning (dead reckoning) integrates proprioceptive measurements from wheel encoders, gyros, and accelerometers to localize. This can be a good localization method over a short distance, but it provides no means of recovering from error that inherently accumulates. So, to maintain a good pose estimate the robot must correct the accumulated error based on data collected from exteroceptive sensors -- for example, landmarks can be visually identified with a camera or detected using sonar or laser scanning. A problem that frequently arises in these forms of landmark identification is that of data association: sensed data must be associated with the correct landmark, even though multiple landmarks may have similar features. Additionally, in many settings it is not possible to guarantee line of sight to the landmarks.

In this paper, we examine localization algorithms which estimate the robot's pose by fusing dead reckoning measurements and range measurements to beacons in the robot's environment. An infrastructure of low-cost, low-power, radio frequency tags (Fig. 1) is introduced to the workspace. This system was originally intended as a means to track assets and people in an environment equipped with special RF transponders [23], but we invert the paradigm by fixing the tags in the environment and moving a transponder with a robot. In this paradigm, as the robot moves, it periodically sends out a query, and any tags within range

respond by sending a reply. The robot can then estimate the distance to each responding tag by determining the time lapsed between sending the query and receiving the response. The advantage of such a method is that it does not require line of sight between tags and the mobile robot, making it useful in many environmental conditions where optical methods fail. Note that, since each tag transmits a unique ID number, distance readings are automatically associated with the appropriate tags, so the data association problem is solved trivially.



**Figure 1** **A radio tag, approximately 12 x 9*cm* in size, with which our robot can communicate to obtain range data. Such tags can be placed throughout the robot's environment.**

In this paper, we examine two data sets collected using an autonomous lawnmower equipped with high-end GPS and inertial sensors, providing groundtruth accurate to within 2*cm*. For each data set, the robot traversed a repeating path for approximately 30 minutes at an average speed of about 1*m/s*. Range measurements to 6 beacons in the environment were collected throughout the experiment. We perform a detailed analysis of the noise inherent in these ranges, then use this noise model to process the data with three algorithms: an extended Kalman filter (EKF), a particle filter, and a sliding batch method.

As we will see, the extended Kalman filter appears the most efficient and accurate, but it is also the most sensitive to sensor noise, and is therefore limited in its ability to recover from bad measurements. Because it must linearize non-linear measurements, it is also weak in its ability to recover from a bad initial pose estimate. The particle filter converges robustly without

any initial pose estimate, but is less accurate. Likewise, the sliding batch method recovers well from a bad initial estimate but requires offline processing time.

Related work is discussed in Section 2, followed by a Problem Statement in Section 3. Sensor noise is examined in Section 4, and a detailed look at two data sets collected with a real robotic vehicle is given in Section 5. Localization results with the Kalman filter, sliding batch method, and particle filter are discussed in Sections 6, 7, and 8, respectively. Section 9 details extending the EKF for SLAM, and conclusions are given in Section 10.

## 2 Related Work

Most approaches to landmark-based localization use sensors that measure relative bearing or both range and bearing to distinct features in the environment. In the case where the location of these landmarks is unknown, the problem is more difficult and is generally called Simultaneous Localization and Mapping (SLAM) [26, 10, 20]. Interestingly, the complementary case, in which a moving sensor detects only range to landmarks, has received little attention. Some researchers have used range to estimate position, although in many cases, instead of using range, signal strength from a known transmitter is used to produce a "pseudorange" that is then used for triangulation.

An early range-based localization system, AT&T Cambridge's Active Bat [2] system, used ultrasound time-of-flight to get ranges to an infrastructure of receivers mounted in the ceiling. With a grid of 16 receivers placed 1.2*m* apart, Active Bat could estimate an object's location to within 14*cm* with probability 0.95. The Cricket system developed at MIT [3] works similarly, but employs RF signals to synchronize time measurements and filter out multipath errors. Recent results reported for this system indicate localization to within a few centimeters, as well as orientation estimation to within 3-5 degrees, when beacons are placed every 1-1.5*m*.

The RADAR system [5] localizes a laptop using 802.11b, converting the signal strength of each packet to a range estimate, and also using nearest neighbor heuristics and pattern matching to improve results. This achieves localization to within 2-3*m* with probability 0.5. Ladd, et al. [6] propose using wireless networking signal strength and employing a Bayesian formulation to localize; with this they can localize a laptop to within 1.5*m* with probability 0.83. The Spot-On system [7] performs *ad-hoc localization*, localizing wireless devices relative to one another. Distance is estimated by measuring attenuation in RF signals between special tags.

Our work involves the PinPoint system from RF Technologies [23]. PinPoint is typically installed in a building, e.g., a hospital, to track patients or other assets. Transponders are installed at fixed locations in the building, and each asset to be tracked carries a small RF tag. The transponders send a signal, and any tag that receives that signal responds -- based on the timestamp of the response, the range from transponder to tag can be estimated. For use in localization, this system has the advantage that line-of-sight is not required between the tag and the transponder, and the data association problem is removed.

Whereas the systems mentioned above all localize using only range measurements, we combine range measurements with dead reckoning measurements to localize a moving robot. To do this, we explore three methods for filtering the data. The Kalman filter, particle filter, and sliding batch methods have all been used to filter data for robot localization and SLAM under other paradigms.

Since first introduced as a method for robot localization [8], Kalman filtering has been employed by many systems to localize [18, 10, 11, 12, 13]. Originally introduced in 1960, the Kalman filter assumes a multivariate Gaussian distribution [9]. The Kalman filter has the advantage that the representation of the distribution is compact; a Gaussian distribution can be represented by a mean and a covariance matrix. The robot's pose estimation is maintained as a Gaussian distribution and sensor data from dead reckoning and landmark observations is fused to obtain a new position distribution.

Recent extensions of Kalman filtering allow for non-Gaussian, multimodal probability distributions through multiple hypothesis tracking. The result is a more versatile estimation technique that still preserves many of the computational advantages of the Kalman filter. Monte Carlo localization, or particle filtering, provides a method of representing multimodal distributions for position estimation [14, 15, 16, 17], with the advantage that the computational intensity remains the same regardless of the localization area. The main advantage here is the ability to recover robustly from a poor initial condition.

Batch techniques generally employ some form of non-linear optimization (e.g. Levenberg-Marquardt or Gauss-Newton optimization) to minimize a cost function computed over a window of data. McLauchlan [19] has developed the variable state dimension filter (VSDF), which combines Gauss-Newton nonlinear optimization with the extended Kalman filter. Originally conceived for the structure-from-motion problem in computer vision, it has also been applied to the SLAM problem [20, 21]. The VSDF motivates our sliding batch algorithm, although we do not allow the state vector to change dimension (i.e., we use a fixed window size), as we will explain in detail in Section 7.

Previously we reported results from an initial application of the EKF to robot localization and SLAM with range from radio [27, 28]. Here we extend these results, using data collected from a real vehicle to compare several localization techniques.

# 3 Problem Statement

In almost any application, an autonomous mobile robot requires the ability to accurately estimate its pose, or position and orientation. The localization process defines the way the robot's pose is estimated based on measurements received from sensors. Accurate localization is necessary for navigation and for controlling the robot in its environment, with the degree of accuracy required being dictated by the application.

For many mobile robot environments, the pose can be simplified by estimating only the Cartesian position and orientation. With this in mind, a typical mobile robot is equipped with wheel encoders, which measure the robot's incremental distance traveled, as well as gyroscopes or accelerometers, which measure the incremental change in the robot's heading. These are called proprioceptive measurements, since they estimate the robot's relative motion from onboard sensors rather than measuring from some absolute point of reference.

The process of integrating these proprioceptive measurements over time to maintain an estimate of the robot's pose is known as *dead reckoning*. Dead reckoning localization is easily achieved and can provide high accuracy for short paths. Dead reckoning alone will fail for most real-world applications, however, since it inherently provides no means of correcting error and therefore suffers from unbounded error accumulation.

To improve on this, we look for some means to sense the robot's pose absolutely, with reference to a fixed point or points. Absolute sensor measurements are called exteroceptive, and absolute localization requires the robot to sense features in its environment and estimate its pose in the world coordinate frame directly based on these (usually fixed) feature positions. In this paradigm, localization at any point in time does not require knowledge of the prior positions of the robot, and error in localization is determined only by the noise and bandwidth of the sensors. GPS is a common means of absolute localization.

Similarly, our radio frequency beacons, placed in the robot's environment and providing frequent range measurements, could be used to absolutely localize in a way analogous to GPS localization. However, noise in the signal precludes our use of the RF beacons for absolute localization (for a more detailed discussion of this noise, see Section 4), and our dead reckoning measurements are received at a higher bandwidth. Since dead reckoning alone results in unbounded error, we turn to sensor fusion methods to combine our dead reckoning and range

measurements to better localize the robot. The absolute range measurements will provide feedback to the dead reckoning, effectively bounding the error. We will look at three methods for interpreting this feedback to correct the dead reckoning error, beginning with the Kalman filter.

As mentioned in Section 2, the Kalman filter has been frequently applied to the problem of robot localization. It works recursively, and so does not require a history of the robot's previous states to be kept. This results in a streamlined algorithm that can run online in a realtime systems. Unfortunately, our absolute range measurements are non-linear, requiring the use of an *extended* Kalman Filter (EKF), which must linearize the measurements around the current state estimate. This results in a weakness common to all linear methods -- the Kalman filter will not converge when the initial state is not sufficiently accurate. Nevertheless, we will see the Kalman filter localize a robot to, at its best, within an average of 17*cm* of its true position (Cross-track error).

Given the shortcoming of the EKF, we look to some non-linear methods. First, we will examine the sliding batch method, which localizes a window of robot pose estimates and range measurements at one time. The pose estimates are optimized given all the information in that window, then the window slides forward one increment in time and the process repeats. Accuracy for this method is high (about 15*cm* average cross-track error at its best), but due to its complexity offline processing time is required.

We also consider localizing with a Particle Filter, also known as Monte Carlo Localization. This method maintains a probability distribution representing the robot's position by maintaining a set of samples (particles) from that distribution. Given an initially uniform distribution, we will see the Particle Filter localize given no initial pose estimate, thus overcoming the EKF's chief weakness. However, we will see that the overall accuracy of our estimate from the Particle Filter is lower than for the EKF (26*cm* average cross-track error at best), demonstrating a tradeoff between the algorithms.

Finally, we will consider a more difficult localization problem, in which the robot's absolute measurements come from landmarks whose positions are not precisely known. This problem is familiarly known as SLAM, or Simultaneous Localization and Mapping, and we approach it using once again the extended Kalman filter. The best result here will be on par with

the best Kalman filter localization result -- about 17*cm* average cross-track error. These and other error statistics are compared in Figures 26-29 in Section 10.

With these motivations in mind, we now begin by considering the practical issue of how to deal with noise in our range measurements.

## 4 Noise Characterization for Range Measurements

Each time the robot logs a range measurement during one of our experiments, we can determine the error in that range, since we know the true location of the robot (to within 2*cm*) and of the beacon (from surveying with GPS). Figure 2 shows some error histograms associated with a few selected measurements. Ranges received from the PinPoint system were recorded as integer range estimates in feet, but they are converted to meters in the figure (and throughout this paper) for unit consistency.

**Sample Range-Noise Histograms**

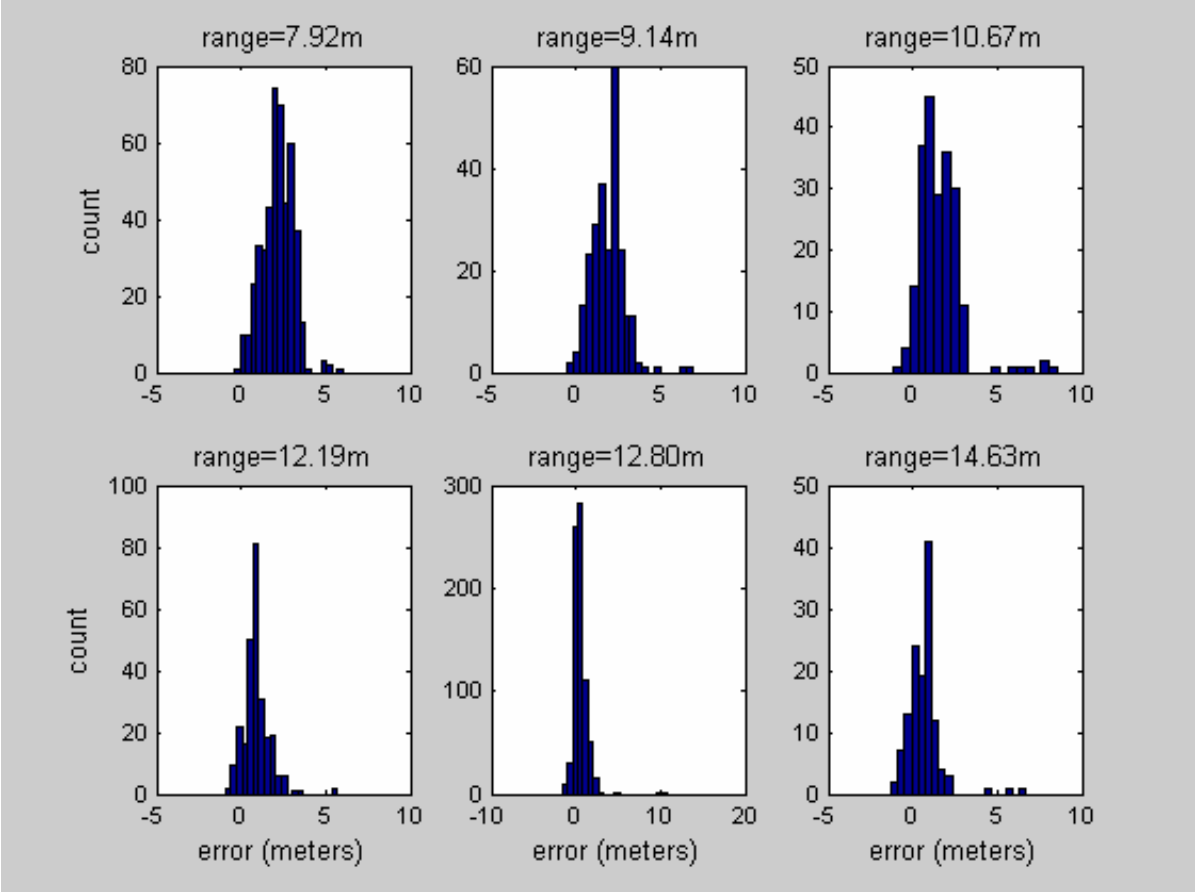



**Figure 2** **A sample of histograms showing the errors (true range - measured range) associated with several measured ranges. While not perfectly Gaussian, the shapes of these plots give us some confidence about our assumption of Gaussian noise in the measurements.**

We choose to approximate these error distributions as Gaussians. Under this assumption, it's possible to store a probability distribution for each range measurement by storing the mean and

variance of all *true* ranges associated with that measurement. To use this in localization, all incoming range measurements will be replaced with their stored means, and their variances will be used as a measure of uncertainty about the measurement.

Figures 3 and 4 plot the mean and variance for each measurement as a function of the measurement values. Figure 3 shows that the means associated with measurements below 3*m* and above about 21*m* vary erratically, when, if the readings were more accurate, they would be following an upward trend on the line y=x. Similarly, Figure 4 shows that the variance values for measurements below 3*m* and above 21*m* jump between very large and unbelievably small. These observations lead us to distrust tag measurements less than 3*m* or greater than 21*m*, so we can improve the localization results by simply ignoring such readings. The average standard deviation of measurements between 3 and 21*m* is 1.26*m*.

Some algorithms incorporate more sophisticated filtering of the data, which is explained in detail later.

**Means from PDFs for each Measurement**

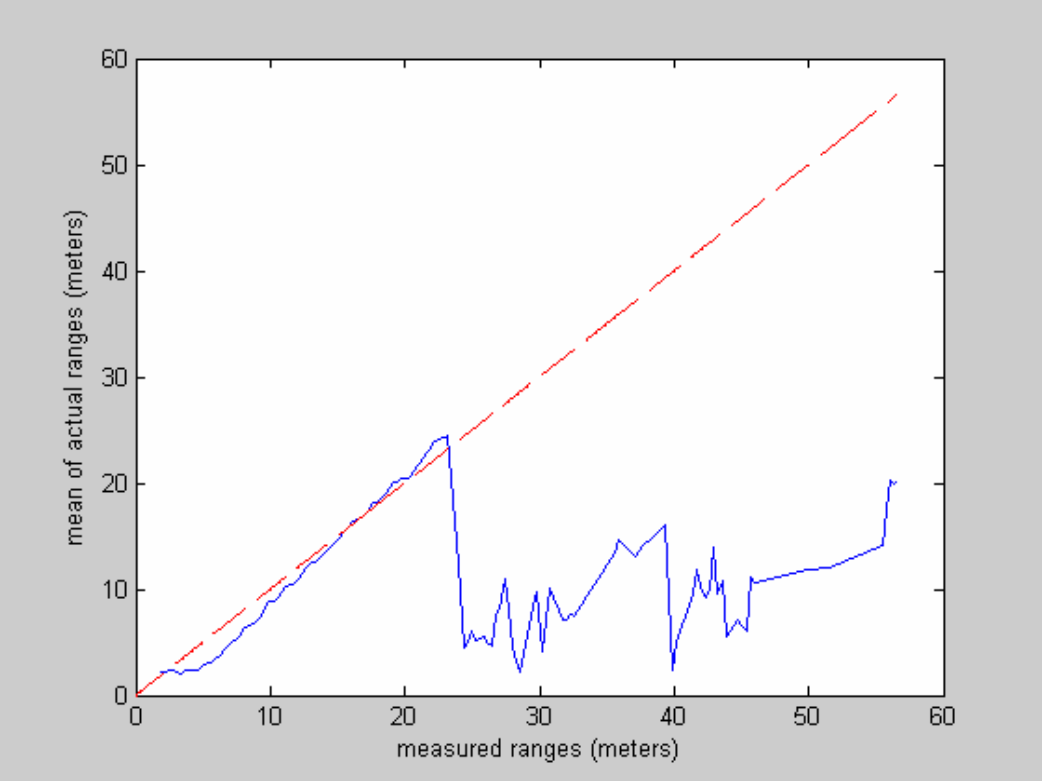



**Figure 3** **Means vs. Measurements. For each measured range (along the x-axis), a mean is computed from all of the true ranges that corresponded to that measured range throughout the experiment.**

**Variances from PDFs for each Measurement**

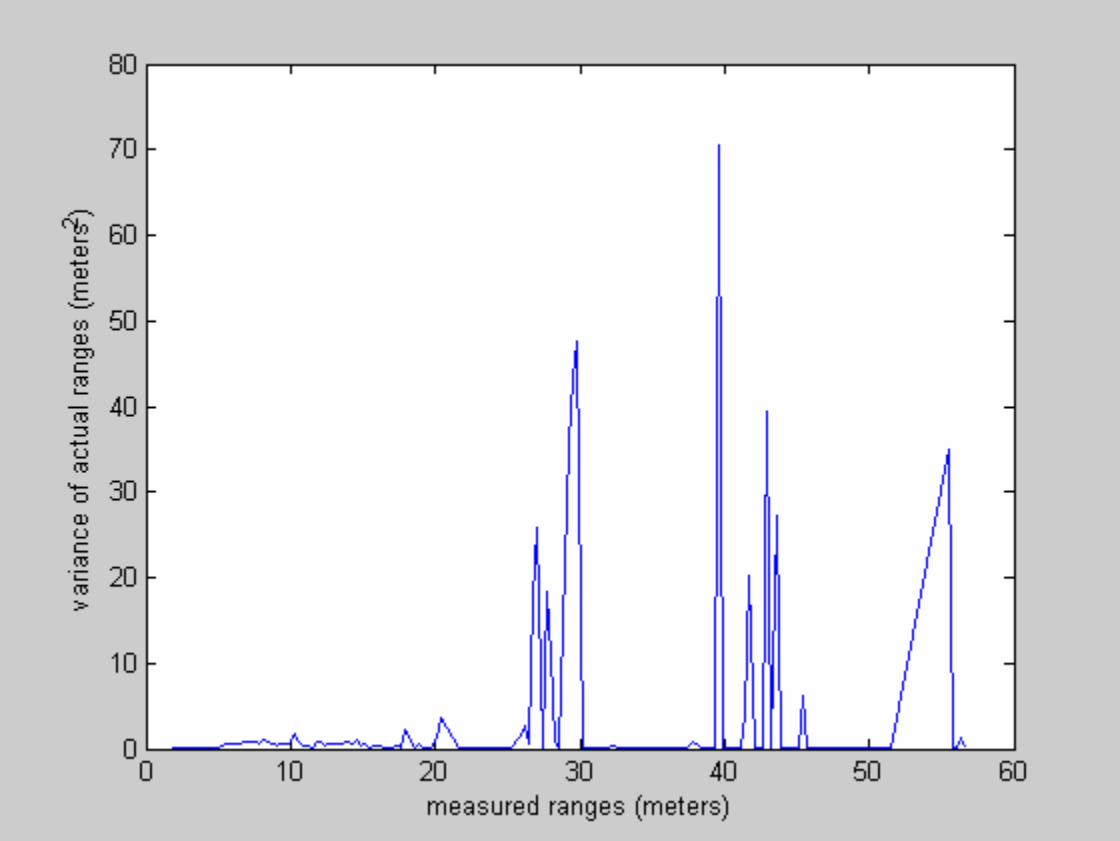



**Figure 4** **Variances vs. Measurements. By storing these variances along with the means, we are storing a probability distribution function for each range measurement, since we are assuming the noise in the measurements to be Gaussian.**

# 5 Experimental Setup

Data for this experiment was collected using an instrumented autonomous robot with highly accurate (2*cm*) positioning for groundtruth using RTK GPS receivers as well as a fiber optic gyro and wheel encoders. Groundtruth position is updated at 100 Hz.

This robot was equipped with antennae pointing in four directions and a computer to control the tag queries and process responses. For each tag response, the system produces a time-stamped distance estimate to the responding tag, along with the unique ID number for that tag and the ID of the antenna that received the response. The distance estimate is an integer estimate of the distance between the robot and the tag in feet. For experimentation, the RF tags are placed atop traffic cones 45.7*cm* above the ground.

The robot was driven on a flat, grassy area about 30 meters by 30 meters in size. We distributed 6 RF tags throughout the area, and then programmed the robot to drive in a repeating path among the tags. With this setup, we collected three kinds of data: the ground truth path of the robot from GPS and inertial sensors, the dead reckoning estimated path of the robot from inertial sensors only, and the range measurements to the RF tags. We collected two sets of data; their groundtruth and dead reckoning paths are shown in Figures 5-8.

**Figure-8 Path: Ground Truth**

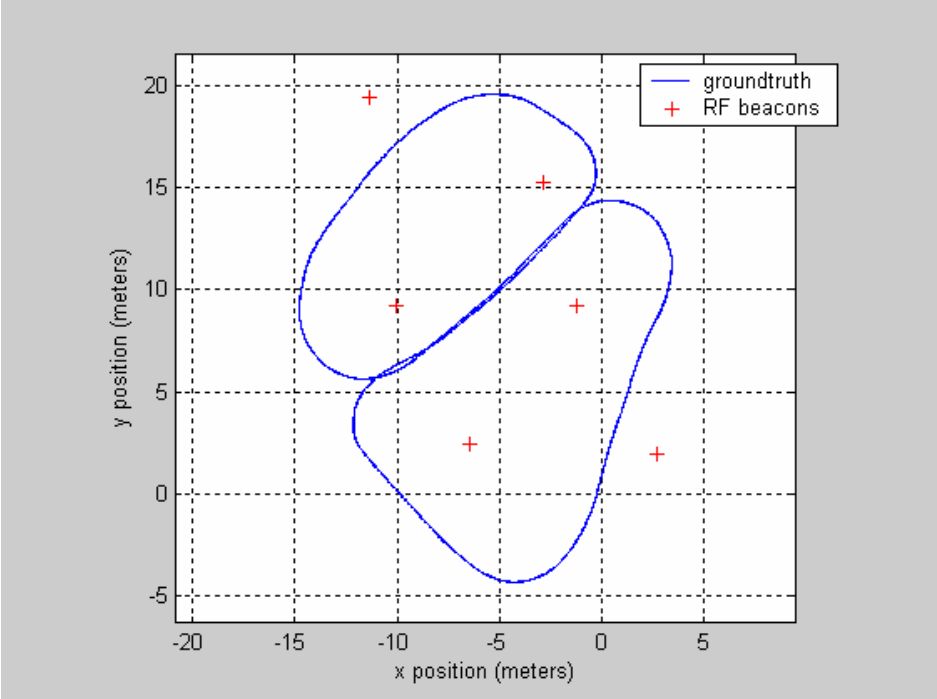



**Figure 5** **The groundtruth path and tag locations for the figure-8 dataset.**

**Figure-8 Path: Dead Reckoning Estimate**

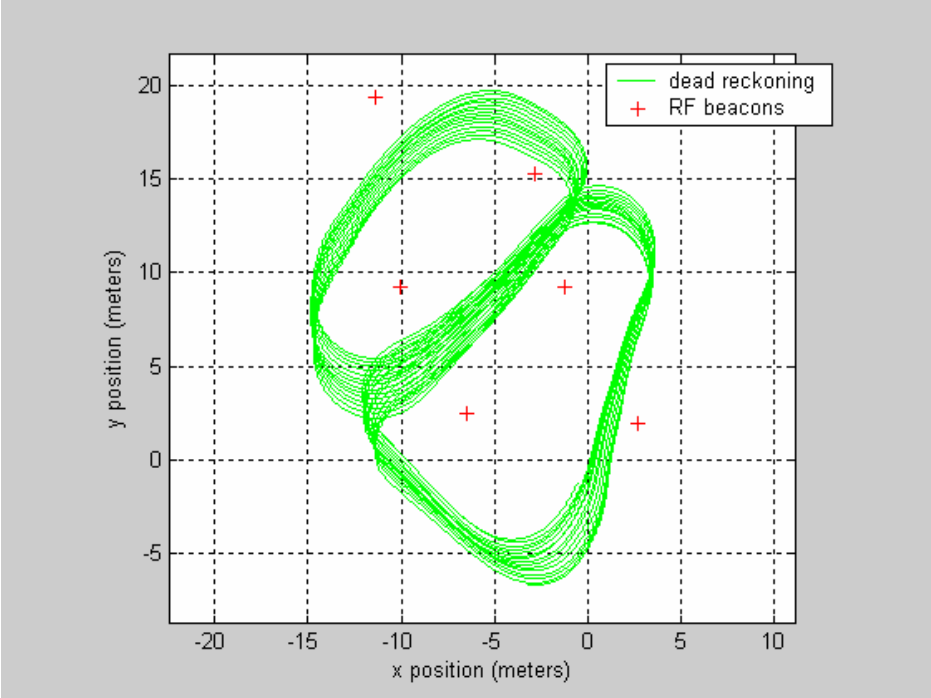



**Figure 6** **The dead reckoning path estimate and true tag locations for the figure-8 dataset. Because of the uncorrected errors in odometry, the dead reckoning path estimate tends to drift away from the true path over time.**

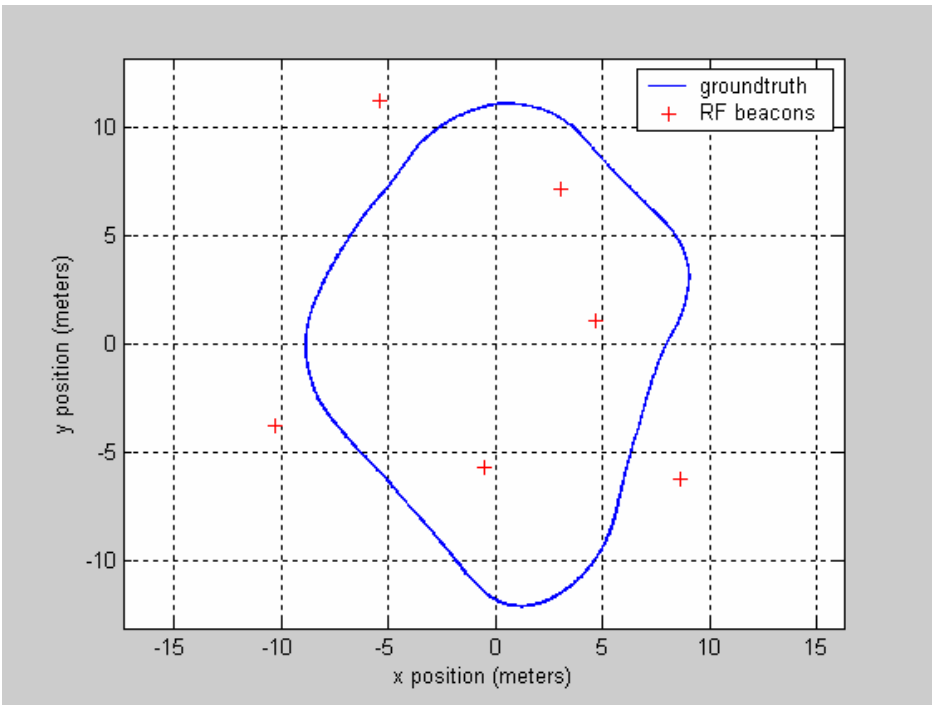



**Figure 7** **The groundtruth path and tag locations for the circular-path dataset.**

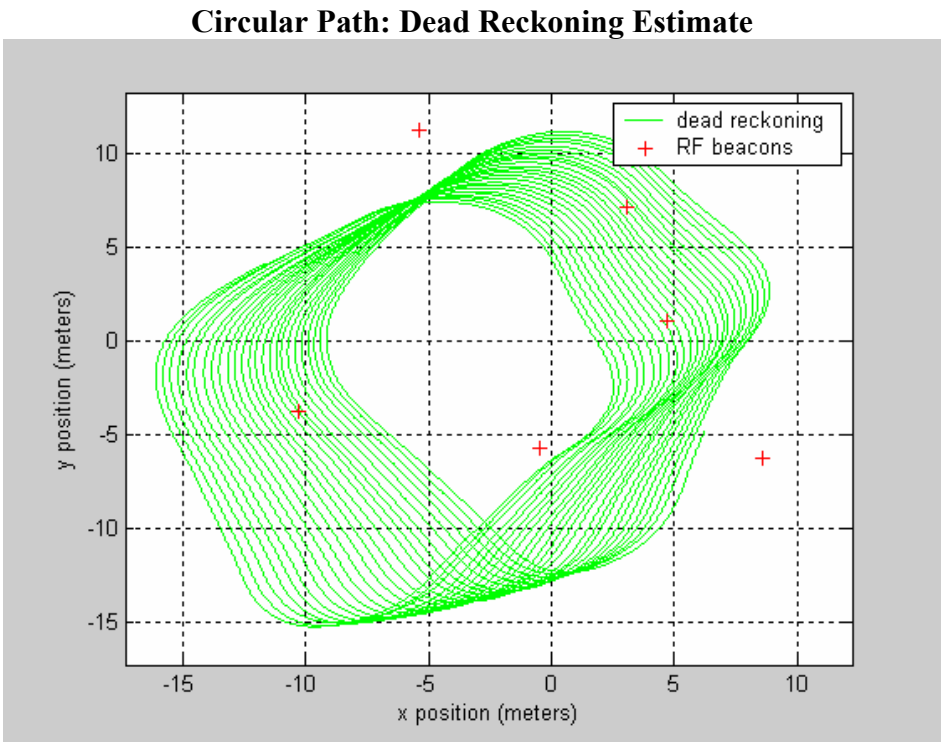



**Figure 8** **The dead reckoning path and tag locations for the circular-path dataset. Because of the uncorrected errors in odometry, the dead reckoning path estimate tends to drift away from the true path over time.**

## 5.1 Problem Geometry

To obtain the best results, the geometry of the experiment must be considered in detail. Essentially, it breaks down as follows:

1. The dead reckoning and ground truth systems obtain estimates of the robot's position based on measurements taken at the center of the front axel. We will refer to this point as the robot reference point, labeled G in Figures 9 and 10.
2. The antennae are mounted at a different location on the robot -- in our experiment, the center of the antenna mount is 71.12*cm* behind the reference point, 40.64*cm* to its left, and 180.3*cm* above it. The center of the antenna mount is labeled A in Figures 9 and 10.
3. Four antennae are mounted, each about 20*cm* from the center of the antenna mount, pointing forward, left, backward, and to the right of the robot.
4. The RF tags are placed on cones 45.7*cm* above the ground. They are not at the same height as the robot reference point, which is 17.8*cm* above the ground.

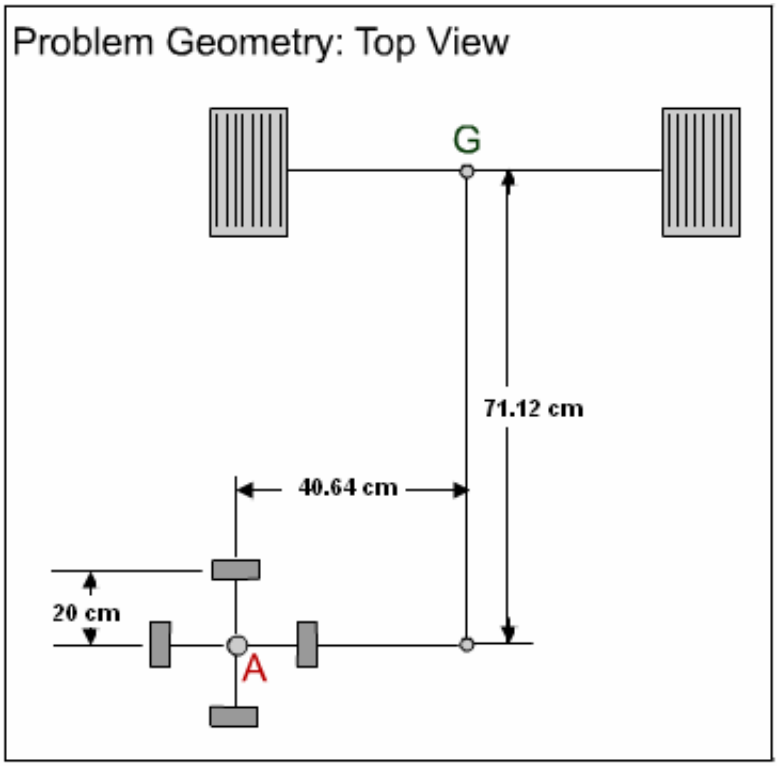



**Figure 9** **Bird's-eye view of the robot geometry.**

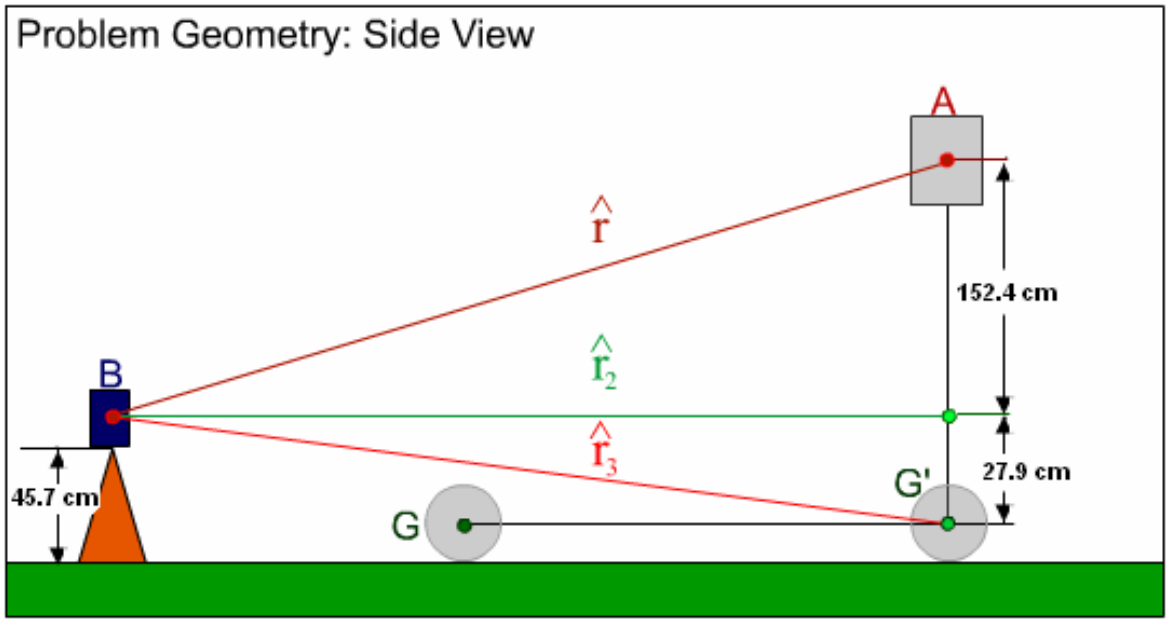



**Figure 10** **Side view of the problem geometry.**

To properly use the measurements we obtain, we must carefully transform them. Here we will look at how to project the measurements onto the same plane with the robot reference point (i.e., how to correct for the vertical offset). The planar offsets will be addressed later, as they are accounted for differently for each of the three localization algorithms.

## 5.2 Projecting onto the xy-plane

Since our dead reckoning and ground truth systems log the motion and position of the center of the front axel, we would like to localize at that point as well. However, our range measurements come from a tag that is 27.9*cm* above this reference point, and they estimate the range to an antenna that is 180.3*cm* above this reference point (for now, we ignore the antenna's offsets in x and y and only correct for the offset in z).

Fortunately, given an estimated range to an antenna, we can compute the estimated range to the point directly below the antenna which is on the same xy-plane with the robot reference point. This will reduce our offset problem to a two-dimensional shift, which we will take care of later.

Looking at Figure 10, we see that we know $\hat{r}$ and would like to compute $\hat{r}_3$. Computing $\hat{r}_2$ as an intermediary, we have these equations:

$$\hat{r}_2 = \sqrt{\hat{r}^2 - 152.4^2}$$

$$\hat{r}_3 = \sqrt{\hat{r}_2^{\,2} + 27.9^2}$$

(For clarity, we assume that $\hat{r}$ is in centimeters.)

So, we are easily able to project our range measurements onto the same plane with the reference point, effectively removing the vertical offset.

# 6 The Extended Kalman Filter (EKF) for Localization

## 6.1 Problem Formulation

**6.1.1 Process Model.** If the robot pose (position and attitude) at time $k$ is represented by the state vector $q_k = [x_k, y_k, \theta_k]^T$, then the dynamics of the wheeled robot used in this experiment are well-modeled by the following set of non-linear equations:

$$q_{k+1} = \begin{bmatrix} x_k + \Delta D_k \cos(\theta_k) \\ y_k + \Delta D_k \sin(\theta_k) \\ \theta_k + \Delta\theta_k \end{bmatrix} + v_k$$

$$= f(\hat{q}_k, u_k) + v_k$$

where $v_k$ is a noise vector. Here, $\Delta D_k$ is the distance traveled by the point at the center of the robot's front axle, obtained by averaging the distances measured by the left and right wheel encoders. The incremental orientation change $\Delta\theta_k$ is obtained by the onboard gyro. These dead reckoning measurements constitute the control input vector $u_k = [\Delta D_k, \Delta\theta_k]^T$.

The system matrix *A(k)* is represented by the Jacobian:

$$A(k+1) = \left.\frac{\partial f}{\partial q_k}\right|_{q=\hat{q}_k} = \begin{bmatrix} 1 & 0 & -\Delta D_k \sin(\theta_k) \\ 0 & 1 & \Delta D_k \cos(\theta_k) \\ 0 & 0 & 1 \end{bmatrix}$$

The input gain matrix *B(k)* is constructed similarly:

$$B(k+1) = \left.\frac{\partial f}{\partial u_k}\right|_{q=\hat{q}_k} = \begin{bmatrix} \cos(\theta_k) & 0 \\ \sin(\theta_k) & 0 \\ 0 & 1 \end{bmatrix}$$

**6.1.2 Measurement Model.** The range at time *k+1* from a beacon located at $(x_b, y_b)$ to the robot with state vector $q_{k+1}$ can be expressed as:

$$h(q_{k+1},[x_b, y_b]^T) = \sqrt{(x_{k+1} - x_b)^2 + (y_{k+1} - y_b)^2}$$

The Jacobian *H(k+1)* gives us the measurement matrix:

$$H(k+1) = \frac{\partial h}{\partial q}\Big|_{q=\hat{q}_{k+1}} = \left[ \frac{(x_{k+1} - x_b)}{\sqrt{(x_{k+1} - x_b)^2 + (y_{k+1} - y_b)^2}}, \quad \frac{(x_{k+1} - x_b)}{\sqrt{(x_{k+1} - x_b)^2 + (y_{k+1} - y_b)^2}}, \quad 0 \right]$$

**6.1.3 Process Errors.** Using the method of Chenavier and Crowley [24], referenced in [18], we assume the process errors are uncorrelated, and model them as:

$$\mathbf{Q}(k) = \begin{bmatrix} Q_{11}(k) & 0 & 0 \\ 0 & Q_{22}(k) & 0 \\ 0 & 0 & Q_{33}(k) \end{bmatrix}$$

where,

$$Q_{11}(k) = K_{ss}\left|\Delta D_k \cos(\theta_k)\right|$$
$$Q_{22}(k) = K_{ss}\left|\Delta D_k \sin(\theta_k)\right|$$
$$Q_{33}(k) = K_{s\theta}\left|\Delta D_k\right| + K_{\theta\theta}\left|\Delta\theta_k\right|$$

Here, $K_{ss}$ is a coefficient representing the odometric drift along the path traveled with respect to an incremental distance traveled, $K_{s\theta}$ represent the heading drift with respect to an incremental distance traveled, and $K_{\theta\theta}$ represents the heading drift with respect to an incremental change in heading. These coefficients were experimentally tuned to produce optimal results.

**6.1.4 Measurement Gating.** A measurement validation gate is implemented to augment the Kalman filter's weighting of the range measurements; this is meant to disallow the degradation

of the robot's state estimate by an occasional erroneous measurement. Intuitively, the greater our uncertainty about the current state estimate, the more likely we will be to incorporate an apparently erroneous measurement; conversely, if we are confident about the current state estimate then the gate should uphold a higher criteria for incoming measurements, filtering out those that are suspect. Using the method of [25], we compute the normalized innovation squared,

$$\varepsilon_\nu(k) = \nu(k)^T S(k)^{-1} \nu(k)$$

where $\nu(k)$ is the innovation (i.e., the difference between the measurement and its expected value according to the model) and $S$ is the matrix $H_k P_k^- H_k^T + R$. This $\varepsilon_\nu(k)$ had a chi-square distribution, so the gate's bounding values can be read from the chi-square table, and a measurement will be discarded if its $\varepsilon_\nu(k)$ value is outside the bounding values.

## 6.2 Applying the Kalman filter

**6.2.1 Adjusting for the antenna offset.** As described in section 5.2, we must project our range measurements onto the plane parallel to the ground (assumed flat) and at the same height as the robot reference point -- in our case, the center of the front axle. If we consider the z axis to be pointing upward, with the y axis pointing in the direction in which the robot is traveling, we can define a local coordinate frame for the robot. Using this frame, with the robot reference point at the origin, the center of the antenna mount has coordinates (-40.64, -71.12) (units in centimeters, see Fig. 9). Each of the four antennae also has an offset of 20*cm* in *x* or *y* which must be accounted for. It can be shown that given the coordinates of the robot reference point $(x_r, y_r)$, we can compute the location of the center of the antenna mount $(x_a, y_a)$ as:

$$x_a = x_r + \Delta y \cos(\pi + \theta) + \Delta x \cos(\frac{\pi}{2} + \theta)$$

$$y_a = y_r + \Delta y \sin(\pi + \theta) + \Delta x \sin(\frac{\pi}{2} + \theta)$$

where $\Delta x = 40.64$ and $\Delta y = 71.12$. The exact coordinates of a particular antenna can be found by changing these values of $\Delta x$ and $\Delta y$. For example, since antenna 0 is 20*cm* forward of the antenna mount in the positive $y$ direction, just let $\Delta y = (71.12 - 20)$ to compute the $x$ and $y$ coordinates of antenna 0.

With the problem of the reference frames appropriately addressed, we can now easily apply standard extended Kalman filtering to this problem.

**6.2.2 Time Propagation.** When a new control input vector $u(k) = [\Delta D_k, \Delta\theta_k]^T$ is received, the robot's state is updated according to the process model equation. Using the standard equations of Kalman filtering, the covariance matrix maintaining our uncertainty about the current state is propagated in time:

$$P_k^- = A(k)P_{k-1}^+ A(k)^T + B(k)\Gamma B(k)^T + Q(k)$$

So, the state maintained during the time propagation step indicates the pose of the robot at the robot reference point.

**6.2.3 Measurement Update.** When a measurement is obtained, the update step is broken up as follows:

1. Using the current state estimate, determine the location of the antenna which received the current measurement (i.e., shift the robot reference point's coordinates to get the coordinates of the current antenna, $(x_a, y_a)$).
2. Project the current measurement onto the $xy$ plane as described in Section 5.2.
3. Using $(x_a, y_a)$ and the known beacon location $(x_b, y_b)$, compute $H_k$.
4. Look up the variance $R_k$ and the mean $y_k$ associated with the current measurement from its stored PDF.
5. Using the measurement model, compute the expected range $r_k$ to the beacon. Let $\upsilon(k) = y_k - r_k$ be the innovation.
6. Compute $S_k = H_k P_k^- H_k^T + R_k$.

7. Compute the Kalman gain $K_k = P_k^- H_k^T S_k^{-1}$.
8. Compute the normalized innovation squared and test the measurement against the chi-square gating as described in Section 6.1.4.
9. If the measurement passes the gating test, update the state by letting $\hat{q}_k^+ = \hat{q}_k^- + K_k \upsilon(k)$, and update the covariance matrix by letting $P_k^+ = P_k^- - K_k S_k K_k^T$.
10. Now, using this updated estimate of the pose at the antenna which reported the current measurement, shift back in *x* and *y* to get the updated pose estimate at the robot reference point.

## 6.3 Results

**6.3.1 Running Time.** The Kalman Filter is by far the fastest localization approach we study here. The average time required per update was 0.0024 seconds.

**6.3.2 Error Metrics.** Two standard error metrics are employed in our analysis. The Cross-Track Error (abbreviated XTE) measures the component of position error that is orthogonal to the robot's path (i.e., orthogonal to the true, oriented robot pose). Similarly, the Along-Track Error (ATE) measures the tangential component of position error. In providing error analysis of our path estimates, we present the average of the absolute values of XTE and ATE for each point in the path, as well as the maximum and standard deviation of these errors.

Appling the steps from Section 6.2 to the data described in Section 5, we get the following localization results.

**6.3.3 Figure-8 Path Data Set.** In the experiment illustrated here, the true initial robot position from GPS was used as the initial estimate. The first 45 seconds of the robot's travel are clipped from this plot, giving the filter time to converge.

**Kalman Filter Localized Path Estimate (Figure-8 Path)**

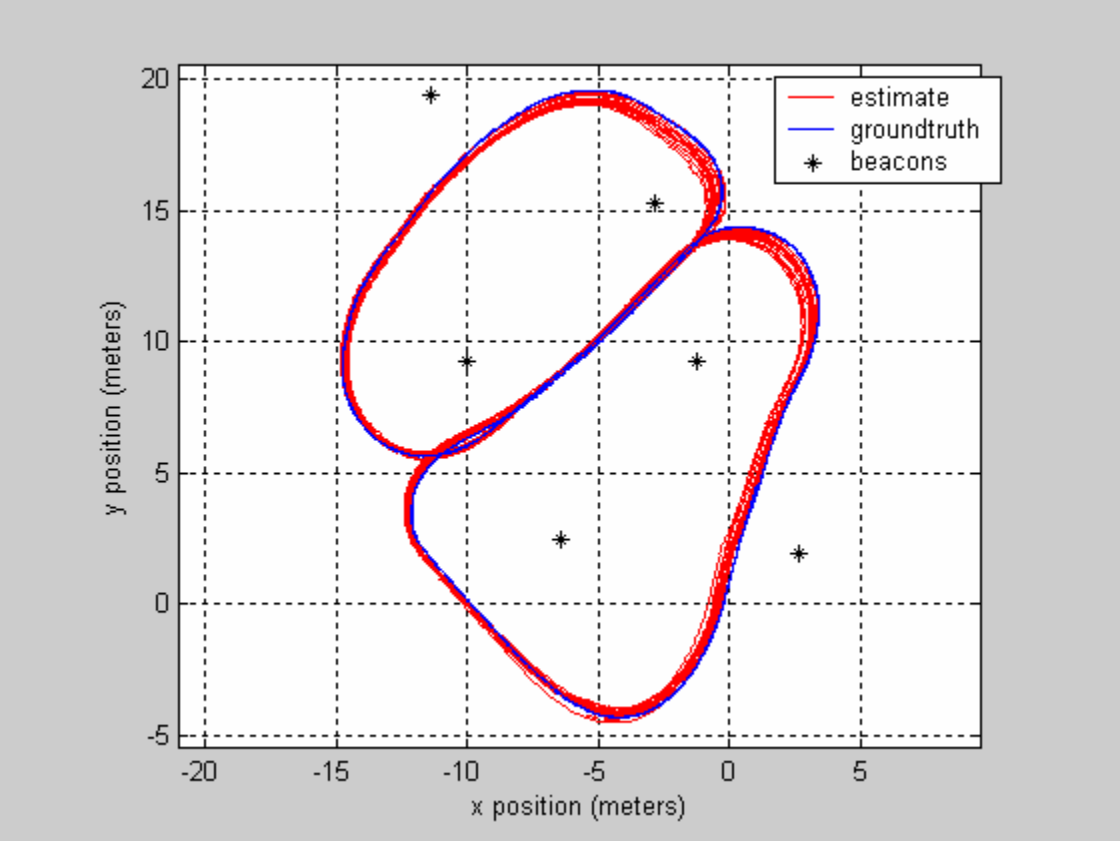



**Figure 11. Kalman filter localization performance on the figure-8 path.**

| | XTE | ATE |
|---|---|---|
| Mean Absolute | 0.1745 *m* | 0.1939 *m* |
| Maximum | 0.9139 *m* | 0.7960 *m* |
| Standard Deviation | 0.1456 *m* | 0.1346 *m* |

**Table 1. Localization performance statistics on the figure-8 path.**

### 6.3.4 Circular Path Data Set.

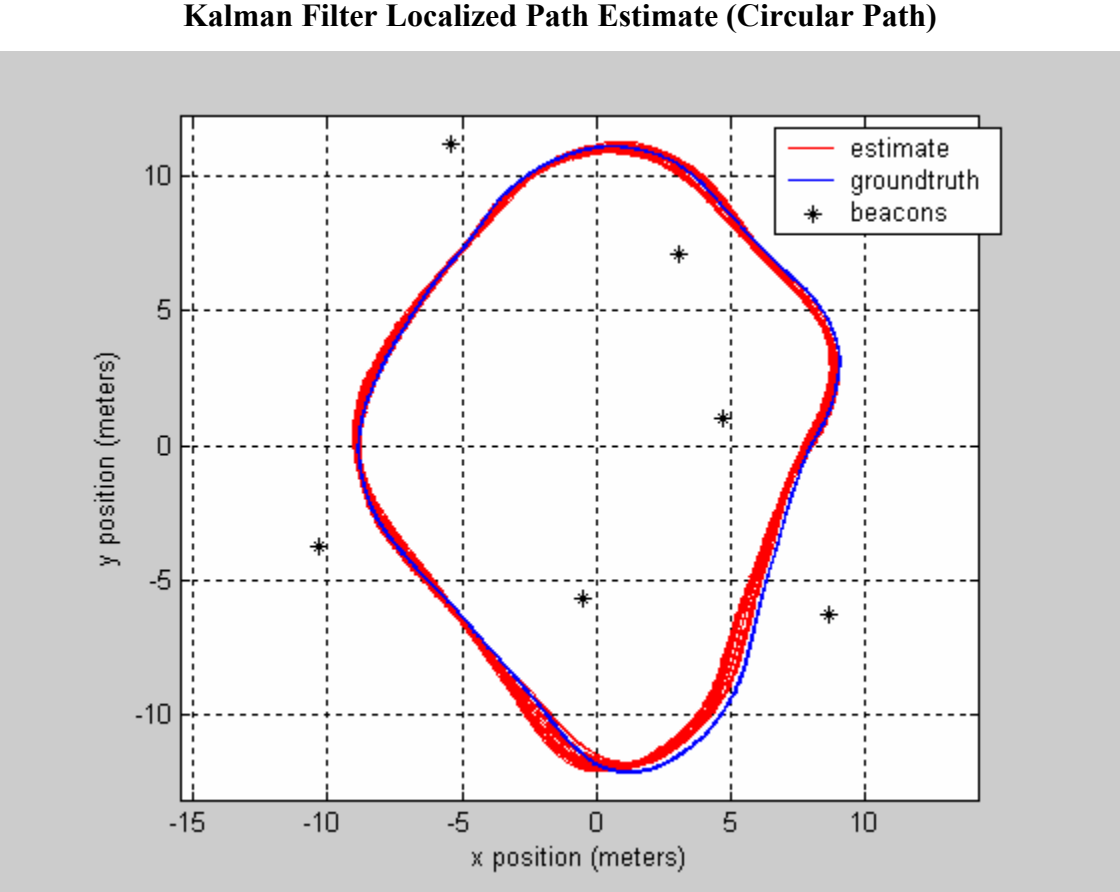



**Figure 12.** **Kalman filter localization performance on the circular path.**

| | XTE | ATE |
|---|---|---|
| Mean Absolute | 0.2065 *m* | 0.3960 *m* |
| Maximum | 0.8800 *m* | 1.2205 *m* |
| Standard Deviation | 0.1722 *m* | 0.2688 *m* |

**Table 2.** **Localization performance statistics on the circular path.**

**6.3.5 Starting with a Bad Initial Estimate.** The results in Section 6.3.3 and 6.3.4 involve the case where a good initial estimate is given for the robot's state. However, for real applications we may have greater uncertainty about the robot's initial pose. Testing the EKF with bad initial estimates yields the results in Figures 13 and 14.

**Kalman Filter Converging from a Bad Initial Estimate**

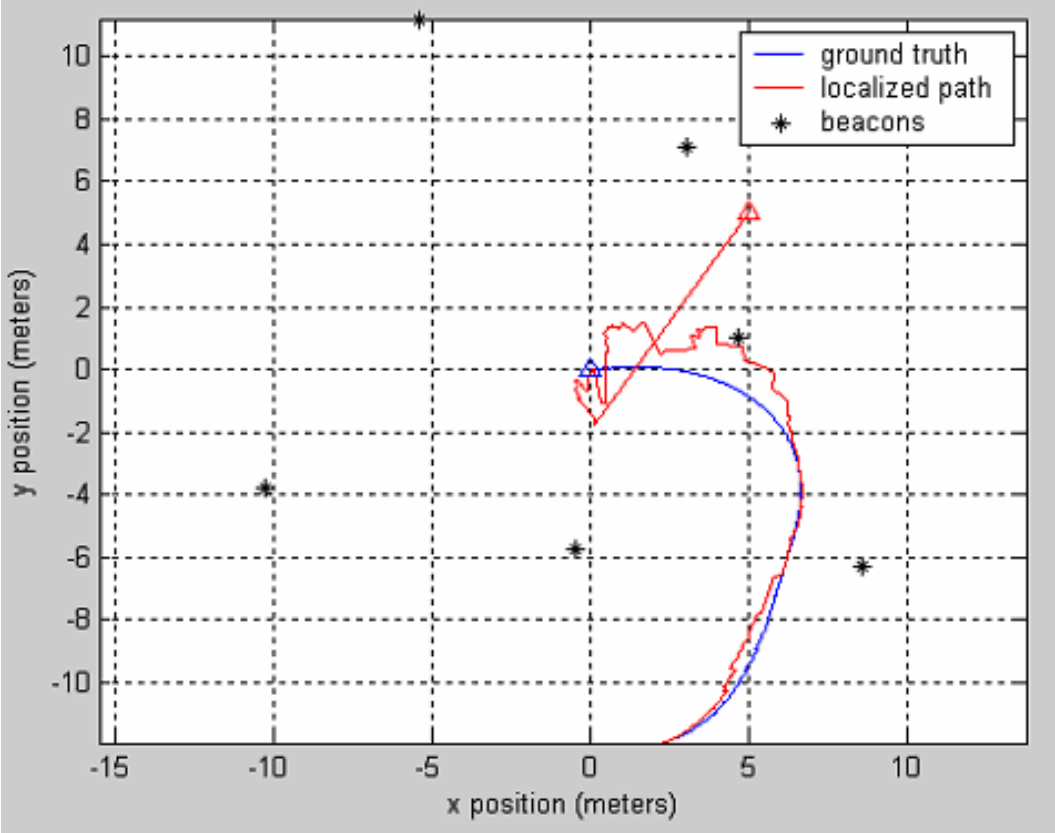



**Figure 13.** **This shows the robot's pose estimate converging from an initial estimate with an error of over 7 meters in x and y and of 1 radian in heading. Convergence occurred in under 1 minute.**

**Kalman Filter Diverging from a Bad Initial Estimate**

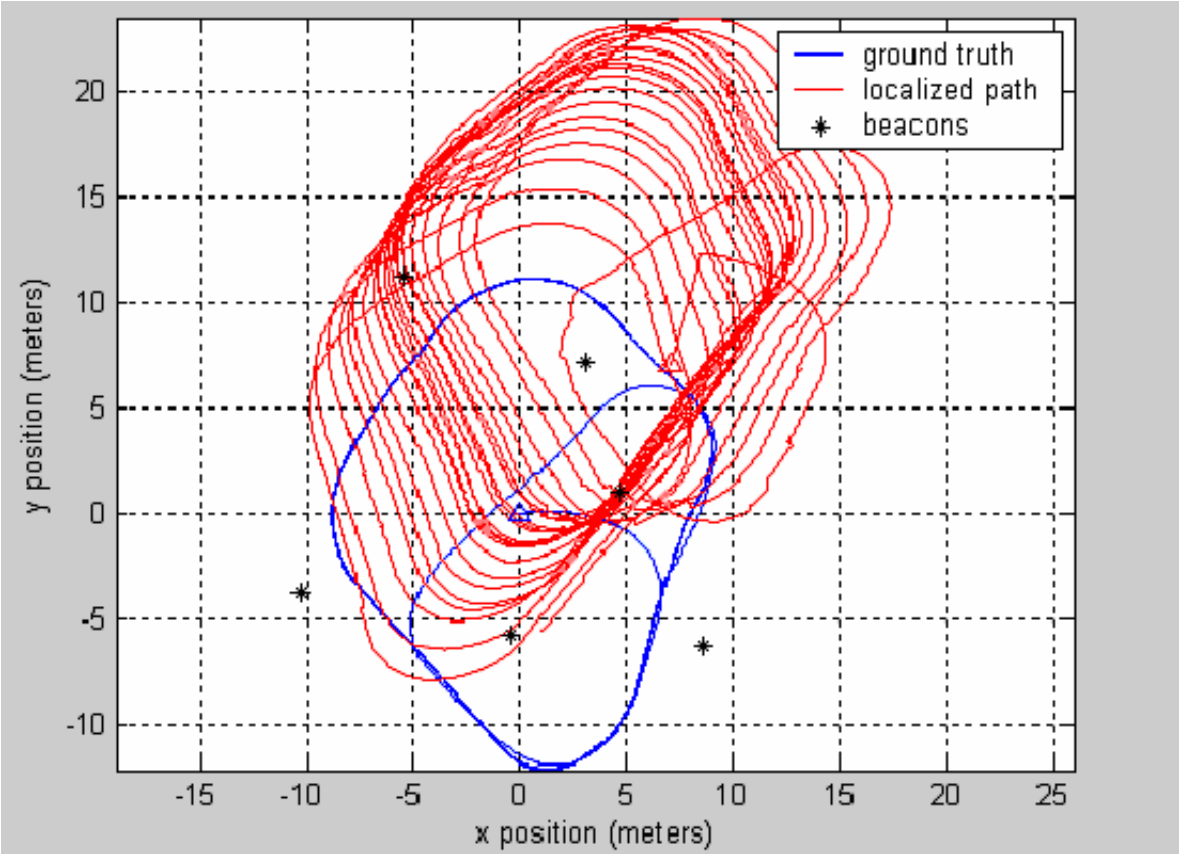



**Figure 14.** **This shows the robot's pose estimate diverging from an initial estimate with an error of just under 10 meters in x and y and no heading error.**

**6.3.6 Summary.** Because our process and measurements are non-linear, the Extended Kalman Filter linearizes about the current state and covariance matrix. This leads to the problem shown in Figure 14. It is the problem of all linear methods, i.e., you must be able to initialize the state sufficiently close to the true state to achieve convergence. Figure 13 shows that the EKF can recover when the robot's initial position estimate has an error of 7*m*; however, an error of 10*m* is too great. More robust solutions are discussed in Sections 7 and 8.

The EKF performs well for a good initial estimate, and it can easily run in real time, requiring an average of only 0.0024 seconds per measurement update.

# 7 Sliding Batch

As we saw in Section 6.3.5, the Extended Kalman Filter performs well only when seeded with a good initial estimate of the robot's pose. This is because the EKF must linearize a non-linear measurement about the current state estimate. We will now see how well non-linear optimization lets us localize from a very bad initial state estimate.

## 7.1 Problem Formulation

The EKF and particle filter are recursive algorithms, only considering the current robot state without a record of the states that led to it. We now consider a form of Gauss-Newton nonlinear batch optimization that finds the optimal robot path based on all the data collected over a window of time.

Our approach is motivated by the Variable State Dimension Filter [19], which combines elements of the EKF with Gauss-Newton nonlinear optimization. However, in this presentation, we do not allow the dimension of the state vector to vary.

To employ this algorithm, we must receive a range measurement each time a dead reckoning measurement is obtained. Since the dead reckoning measurements are received at 100 Hz, and range measurements are received at around 8 Hz (average), we modify the data by integrating the dead reckoning measurements received between range measurements, then using the initial and final positions from that integrated path to compute the distance traveled and heading change of the robot between range measurements. In other words, we artificially synchronize the dead reckoning and range measurements.

**7.1.1 The State Vector.** In a moment, we will construct a cost function which will incorporate the dead reckoning and range measurements; we will then use Gauss-Newton optimization (from the MATLAB optimization toolbox) to find the best path based on that cost function. But first we must decide what the state vector (i.e., the input to the cost function) will look like. One obvious choice, for a window of size $k$, might be:

$$q_t = \begin{bmatrix} x_t \\ y_t \\ \theta_t \\ \dots \\ x_{t+k} \\ y_{t+k} \\ \theta_{t+k} \end{bmatrix}$$

However, the size of this vector will be 3**k*. Since reducing the state size will reduce the running time, we could instead use our dead reckoning measurements:

$$q_t = \begin{bmatrix} \Delta D_t \\ \Delta\theta_t \\ \dots \\ \Delta D_{t+k} \\ \Delta\theta_{t+k} \end{bmatrix}$$

Now the window of robot states has size 2**k*. In this paradigm, our cost function would optimize the distance and heading change measurements.

We can improve further on this and still obtain good results if we are willing to make an assumption. We assume that most of the error in the robot's dead reckoning positioning comes from problems in the heading measurements, not in the wheel encoders; therefore, we will use:

$$q_t = \begin{bmatrix} \Delta\theta_t \\ \dots \\ \Delta\theta_{t+k} \end{bmatrix}$$

Of course, our initial estimate for $q_t$ comes from the gyro measurements. We will use the $\Delta D$ measurements from the wheel encoders, so our cost function does not need to optimize them. Although this ignores the encoders as a source of error, we will show that this approach improves dramatically on the EKF's ability to converge.

So, for a window of size $k$, the first $k$ ranges and corresponding dead reckoning measurements will be employed to find the optimal set of heading changes. The window will then shift forward in time, and measurements 2 through $k$+1 will be examined. This will continue until the entire dataset has been processed (see Fig. 15).

**7.1.2 The Cost Function.** The cost function is fairly simple, and works as follows:

1. Construct a vector containing all the ranges received in the current window:

$$D_t = \begin{bmatrix} r_t \\ r_{t+1} \\ \cdots \\ r_{t+k} \end{bmatrix}$$

2. Using the PDFs (see Section 4), replace each of these measured ranges with its associated mean actual range, giving us:

$$\overline{D}_t = \begin{bmatrix} \overline{r}_t \\ \overline{r}_{t+1} \\ \cdots \\ \overline{r}_{t+k} \end{bmatrix}$$

3. Now, using the $\Delta\theta$ estimates from $q_t$ and the $\Delta D$ measurements from the wheel encoders, integrate the path within the current window. Note that this gives us $[x, y, \theta]^T$ at each point in the current window without requiring us to optimize the coordinates directly.

4. Using the method from Section 6.2.1, correct for antenna offset at each point on the integrated path, using the known IDs of the antennae that received each range measurement.

5. Now, using the known locations of the tags, compute the estimated range to each tag at each point in the current window:

$$\hat{D}_t = \begin{bmatrix} \sqrt{(x_{b,t} - x_{r,t})^2 + (y_{b,t} - y_{r,t})^2} \\ \dots \\ \sqrt{(x_{b,t+k} - x_{r,t+k})^2 + (y_{b,t+k} - y_{r,t+k})^2} \end{bmatrix}$$

Here, $(x_{b,t}, y_{b,t})$ are the coordinates of the beacon reporting at time *t*, and $(x_{r,t}, y_{r,t})$ are the estimated coordinates of the robot at time *t*, based on the current state estimates.

6. Compute two cost terms. First,

$$J_D = \Sigma(\overline{D}_t - \hat{D}_t)^2$$

( i.e., the sum of the squared differences between the means of the range measurements and their estimates.)

This is the sum of *k* squared terms, so we take $J_D = \sqrt{J_D / k}$ .

The second cost term is simply:

$$J_\theta = \Sigma(q_t - \hat{q}_t)^2$$

$$J_\theta = \sqrt{J_\theta / k}$$

Experimentally, we determine that good results are obtained by scaling and combining these terms to form the following cost function:

$$C = \sqrt{J_D / k} + J_\theta$$

Clearly, more complex cost functions can be considered in the future. This simply combines a measure of the estimate's deviation from the measured ranges and a measure of our state's deviation from the measured heading changes. Although this view is simplistic, in terms of recovering from a bad initial estimate we will see a great improvement over the Kalman filter approach.

**7.1.3 The Window.** We must choose the window size carefully -- a larger window size means that more data points will be considered at each optimization step, producing better results. However, increasing the window size also increases the state dimension, which will slow down the optimization step. It was experimentally determined that the algorithm performs well with a window of size $k$=100.

Figure 15 illustrates how the algorithm optimizes over a window of robot poses at a time.

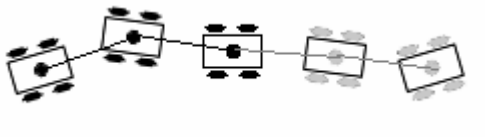
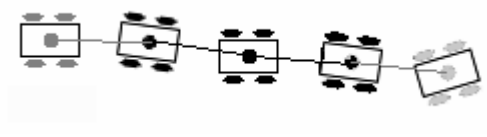
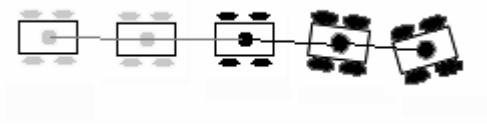
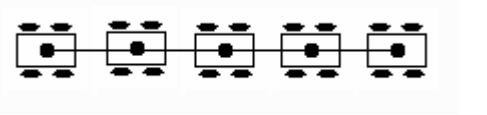
| | |
|---|---|
| Imagine we have the five pose estimates at right from dead reckoning, when in truth the robot drove straight forward. If our window size $k$ is 3, we begin by optimizing the first three estimates. |  |
| After this, the window moves forward, leaving the first pose fixed -- it cannot be improved by future measurements. We optimize the middle three poses. |  |
| Now the first two poses are fixed. The window shifts forward once more, and we optimize the final three poses. |  |
| The final result looks much more like the true, straight path. |  |

**Figure 15.** **Here the sliding batch method optimizes a window of three robot poses at a time based on range measurements received and the cost function. First poses 1, 2, and 3 are optimized. Then the window slides forward, and poses 2, 3, and 4 are optimized -- the results of the first optimization on poses 2 and 3 become the initial conditions for this second optimization. Then the window slides forward again, optimizing poses 3, 4, and 5.**

## 7.2 Results

**7.2.1 Running Time.** It is no surprise that the Sliding Batch method is the slowest of the three algorithms considered here. The average time required per update was 2.8 seconds, so offline processing is required to perform this method in the current configuration.

**7.2.2 Figure-8 Path Data Set.** The following localization results depict the performance of this method when the robot pose estimate is initialized to the true pose (from ground truth), with a window size $k = 100$.

**Sliding Batch Localized Path Estimate (Figure-8 Path)**

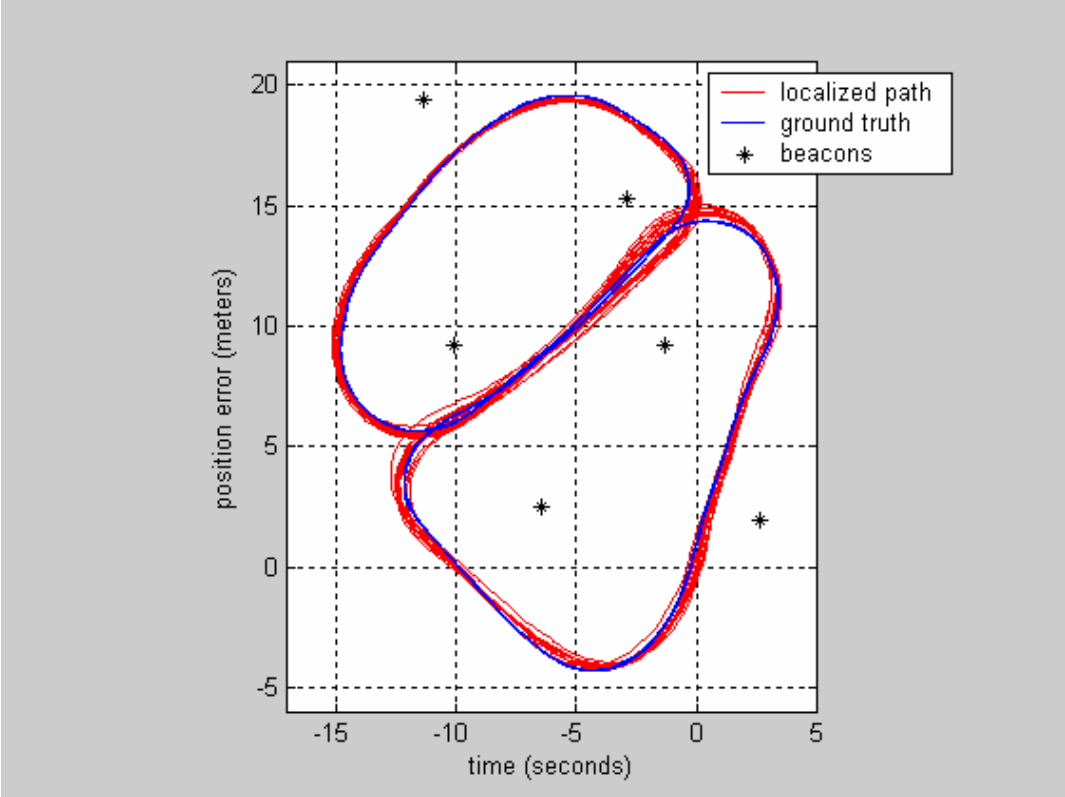



**Figure 16. Gauss-Newton non-linear optimization localization performance on the figure-8 path.**

| | XTE | ATE |
|---|---|---|
| Mean Absolute | 0.1697 *m* | 0.3954 *m* |
| Maximum | 0.7000 *m* | 1.7325 *m* |
| Standard Deviation | 0.1299 *m* | 0.3055 *m* |

**Table 3. Localization performance statistics on the figure-8 path.**

### 7.2.3 Circular Path Data Set.

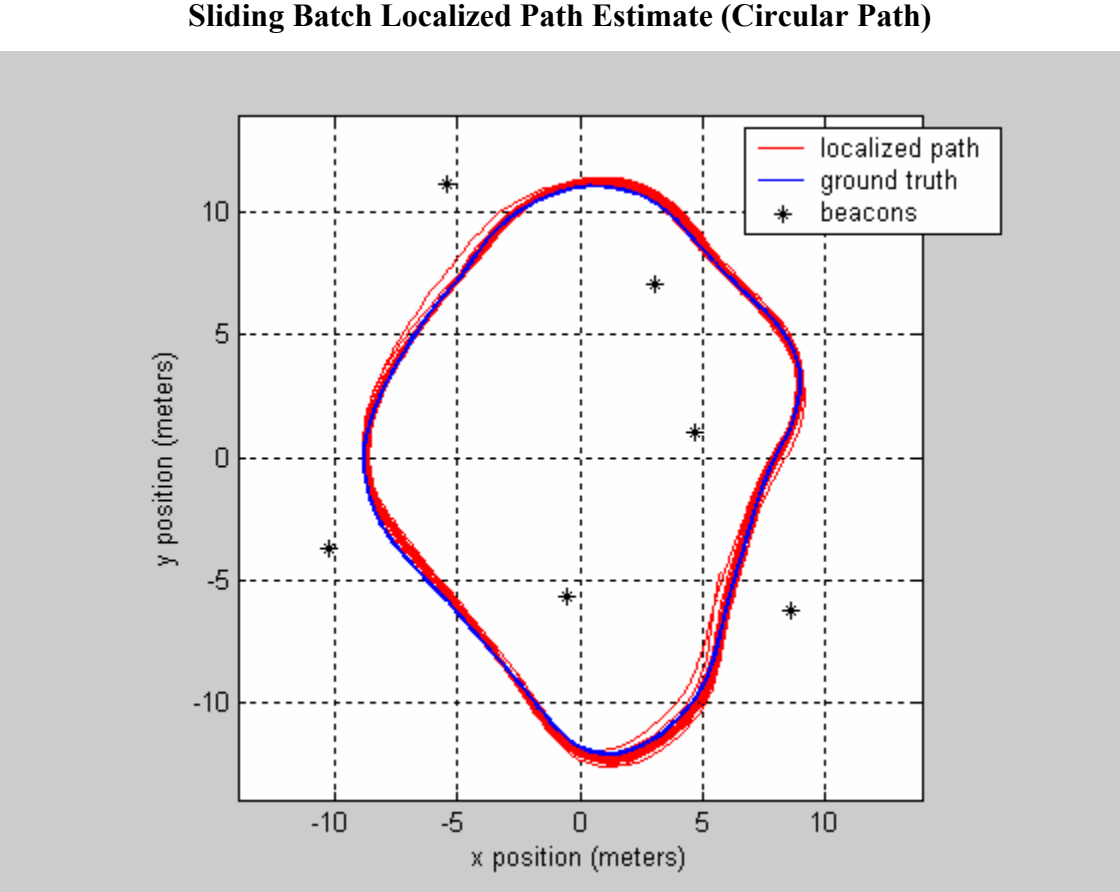



**Figure 17.** **Gauss-Newton non-linear optimization localization performance on the circular path.**

| | XTE | ATE |
|---|---|---|
| Mean Absolute | 0.1488 *m* | 0.3681 *m* |
| Maximum | 0.8256 *m* | 1.5826 *m* |
| Standard Deviation | 0.1354 *m* | 0.2876 *m* |

**Table 4.** **Localization performance statistics on the circular path.**

**7.2.4 Starting with a Bad Initial Estimate.** The real payoff of the non-linear method is in its ability to converge from a bad initial estimate. Figure 20 shows the algorithm recovering from an initial position error of 100 meters in x, 100 meters in y, and 180 degrees in heading.

**Sliding Batch Converging from a *Very* Bad Initial Estimate**

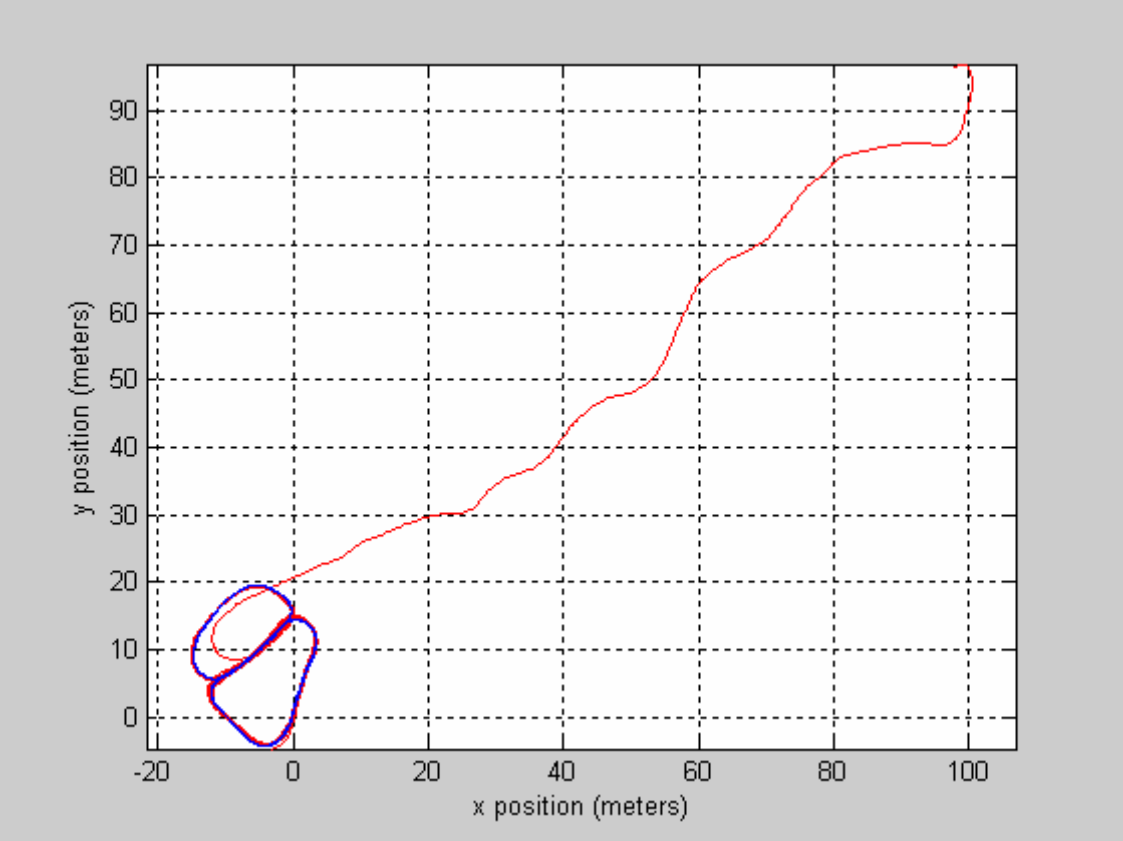



**Figure 18.** **This shows the robot's pose estimate converging from an initial estimate with an error of over 140 meters in x and y and of 180 degrees in heading. Convergence occurred in around 3 minutes.**

**Sliding Batch After Converging from a *Very* Bad Initial Estimate**

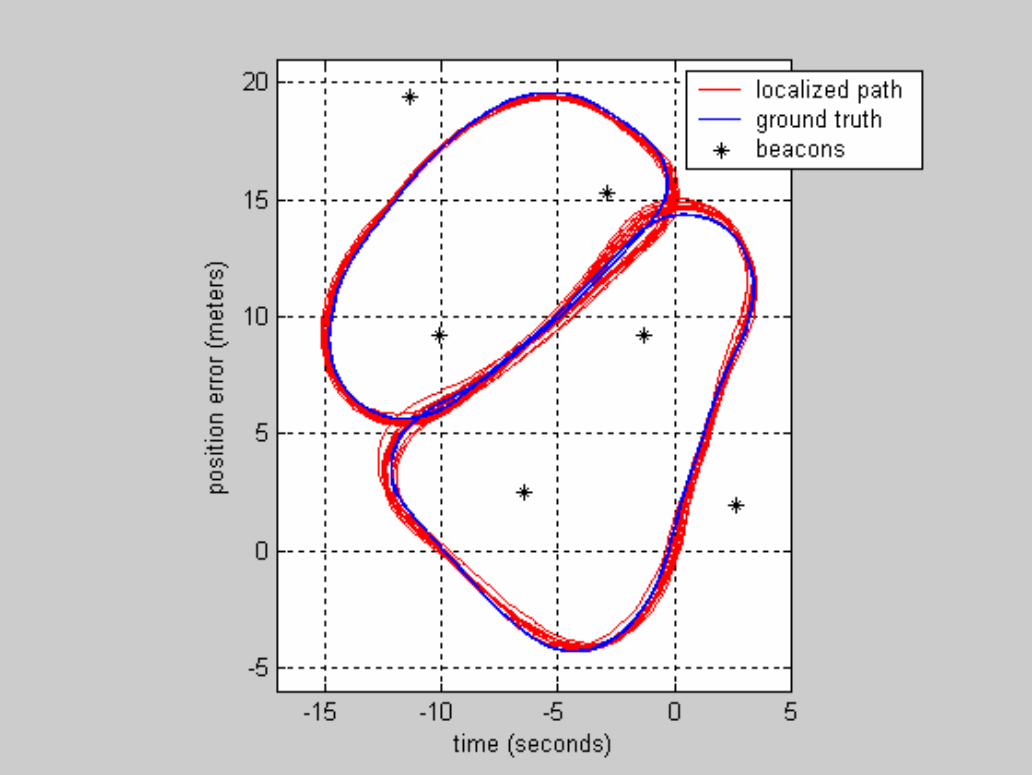



**Figure 19.** **After convergence, the path looks like this.**

| | XTE | ATE |
|---|---|---|
| Mean Absolute | 0.1719 *m* | 0.4265 *m* |
| Maximum | 0.6932 *m* | 1.7325 *m* |
| Standard Deviation | 0.1310 *m* | 0.3108 *m* |

**Table 5.** **Localization performance statistics after converging from a bad initial estimate.**

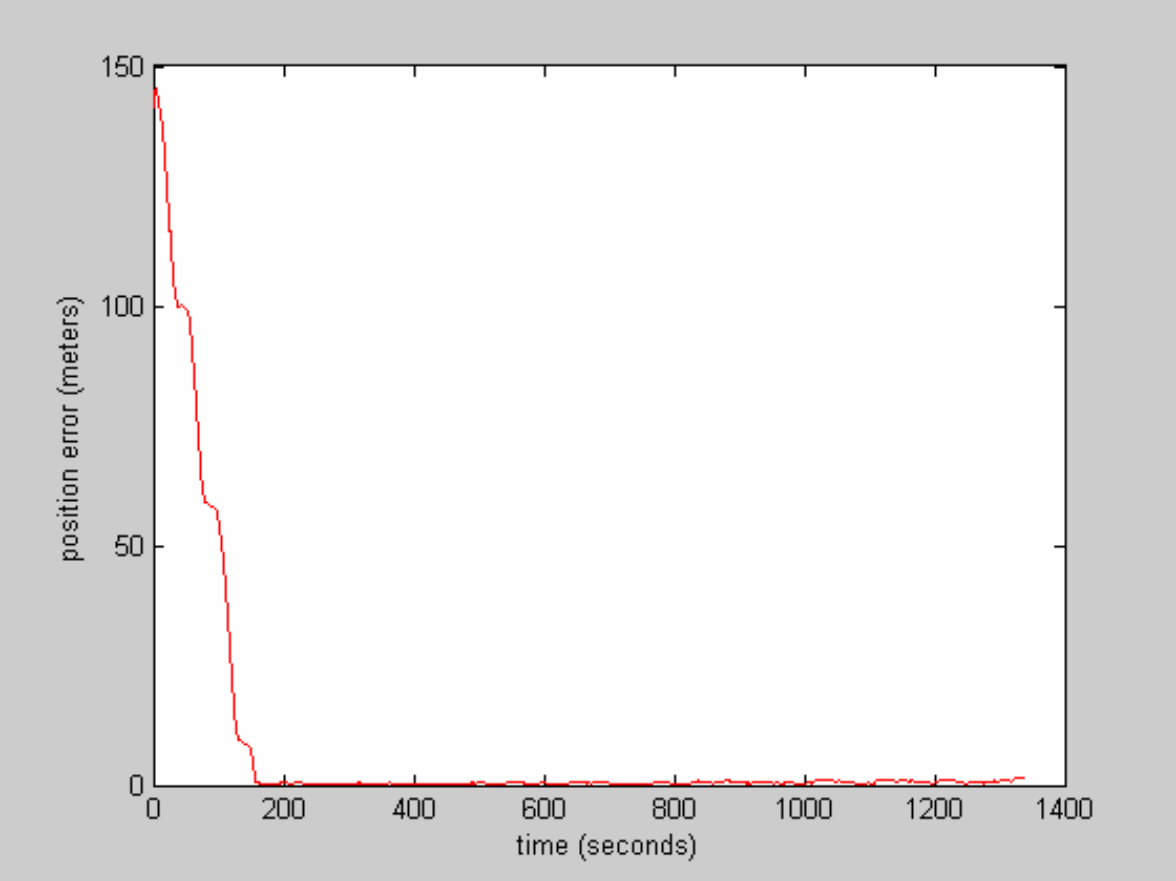



**Figure 20.** **Although the initial position estimate of the robot was off by over 140 meters and by 180 degrees, we see convergence in under 200 seconds. This required around 900 data range measurements.**

**7.2.5 Summary.** Cross-track error for the batch method is comparable to the error when the EKF is used. Along-track error is worse for this batch formulation, most likely because this formulation does not allow the estimate to recover from error in the odometric measurements. Further development of the algorithm and cost function would be necessary to improve on this. The running time required (avg. of 2.8 seconds/update) makes this method unsuitable for online use.

# 8 Particle Filter/Monte Carlo Localization

Localization with a particle filter works by maintaining a set of samples (particles) representing the probability distribution of robot states. In this paradigm, arbitrary distributions can be represented, with each particle being weighted based on its probability given incoming measurements. When new measurements are received, the particles are resampled and reweighted accordingly.

Dr. Peter Corke developed an implementation of a particle filter for the range-only localization problem during his visit to CMU from July to October 2003. It was intended as a first step, so it is purposely simplistic, and the result is presented here to demonstrate the concept.

## 8.1 Problem Formulation

This particle filter estimates the robot's planar position only, not orientation. So, each particle is a point in the state space (in this case the *xy* plane) and represents a particular solution. The primary reference for this filter is the paper on Condensation by Blake [22].

**8.1.1 Initialization.** We choose the number of particles to be $N$=1000[1], with each particle initialized to a random state $\begin{bmatrix} x \\ y \end{bmatrix}$, so the particles are initially evenly spread over the *xy* plane within set boundaries. The probability of the $i^{\text{th}}$ particle is $\frac{1}{N}$, and a cumulative probability distribution is maintained for the particles as well. The cumulative probability for the $i^{\text{th}}$ particle is $\frac{i}{N}$.

[1] Errors for $N$=100 were substantially worse, while errors for $N$=5000 and 10000 were almost identical to those for $N$=1000.

**8.1.2 Drift.** The drift is simply computed as the translation in *x* and *y* from the integrated dead reckoning path. The dead reckoning measurements are treated as a simple (*x, y*) translation since the particles are not oriented. So, at each timestep the particles all drift by the same amount.

**8.1.3 Diffusion.** After drifting, a random number is added to each particle's coordinates. A diffusion rate of B=0.03 m/s is chosen, so the random diffusion amount is scaled by B*(Δt), where Δt is the amount of time since the last measurement.

**8.1.4 Resampling.** The particles are resampled to represent their current probability distribution. This is done by the standard method of inverting the cumulative probability distribution.

**8.1.5 Sampling.** A probability is assigned to each of the particles according to the standard Gaussian formula:

$$P(q \mid r_m) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{\hat{r}-r_m}{\sigma})^2}$$

where $r_m$ is the range measurement, $\hat{r}$ is the range estimate from the particle to the beacon, and σ is the standard deviation from the PDF for that range measurement.

**8.1.6 Antenna Offset.** Since the robot state estimate is not oriented, we cannot account for the antenna offset in the same way done for the EKF. Instead, we choose to modify the dataset; in this way we will get an idea of how this method would perform if the antenna and the robot reference point were at the same position on the robot. The dataset is modified as follows:

1. For each range measurement, use groundtruth to compute the true range from the beacon to the antenna.
2. Let *n* = *(measured range)* - *(true range)*, i.e., compute the noise in the measurement.
3. Compute *t* = the true range from the beacon to the robot reference point.
4. Let *m* = *t* + *n* be the adjusted range measurement.

In other words, we compute the noise in the range to the antenna, then add that noise to the true range to the robot reference point.

## 8.2 Results

**8.2.1 Running Time.** The Particle Filter fits between the EKF and the sliding batch method, requiring an average of .29 seconds/update. Given that measurements are received at about 8Hz, this method would require some streamlining before it could be used in an online system.

**8.2.2 Figure-8 Path Data Set.** As illustrated in Figure 23, the Particle Filter requires no initial estimate of the robot's position. In this experiment, the first 2.5 minutes of the robot's travel are clipped from this plot, giving the filter time to converge.

**Particle Filter Localized Path Estimate (Figure-8 Path)**

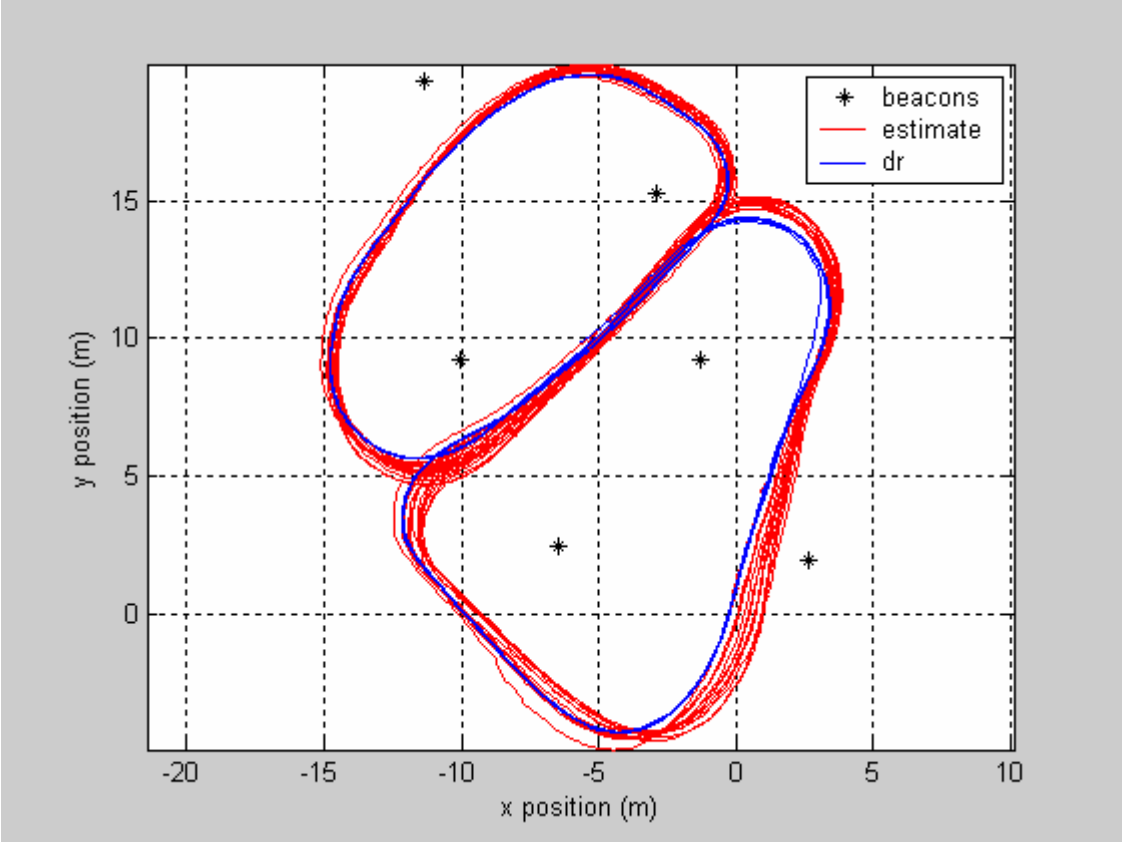



**Figure 21. Particle filter localization performance on the figure-8 path.**

| | XTE | ATE |
|---|---|---|
| Mean Absolute | 0.2631 *m* | 0.4018 *m* |
| Maximum | 1.2278 *m* | 1.7012 *m* |
| Standard Deviation | 0.2270 *m* | 0.3145 *m* |

**Table 6. Localization performance statistics on the figure-8 path.**

### 8.2.3 Circular Path Data Set.

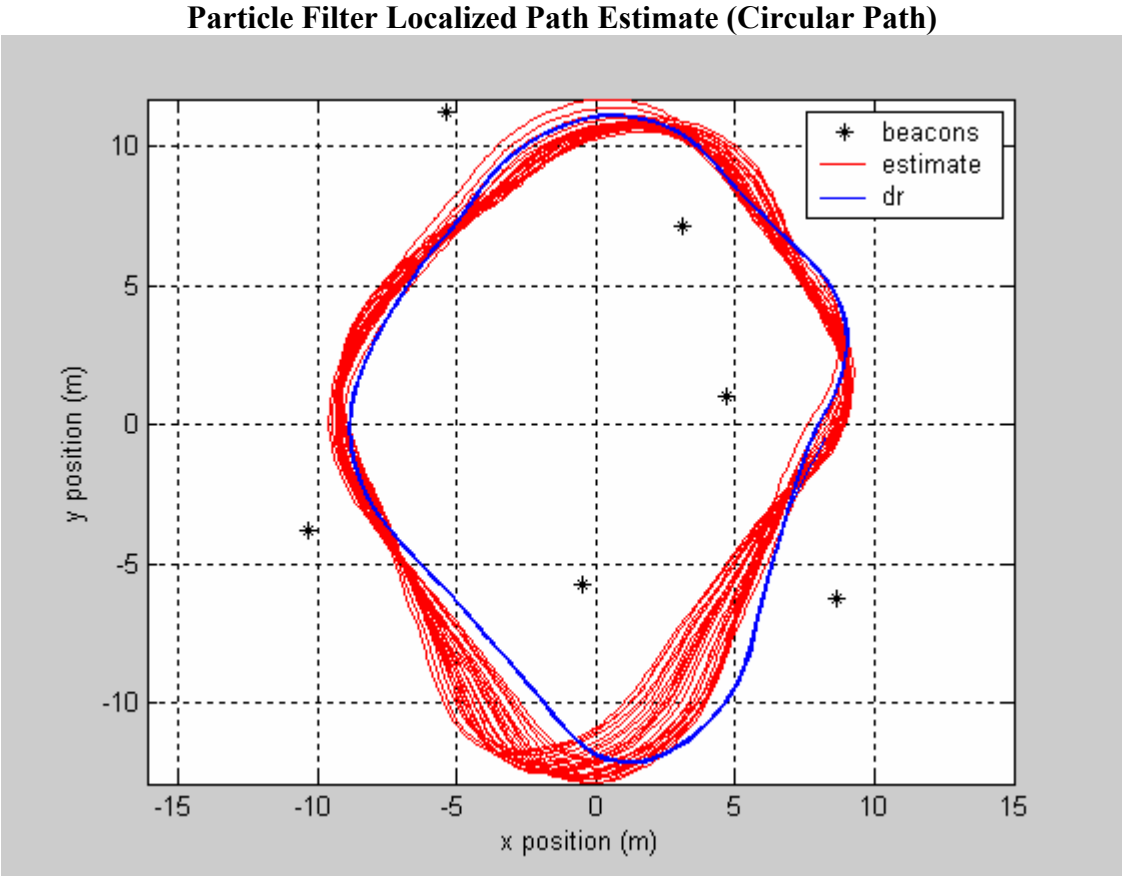



**Figure 22.** **Particle filter localization performance on the circular path.**

| | XTE | ATE |
|---|---|---|
| Mean Absolute | 0.6350 *m* | 1.8839 *m* |
| Maximum | 3.5212 *m* | 5.1972 *m* |
| Standard Deviation | 0.5800 *m* | 0.9427 *m* |

**Table 7.** **Localization performance statistics on the circular path.**

**8.2.4 Summary.** Clearly these results are less accurate than our EKF or sliding batch results. However, Figure 17 illustrates a key benefit of the particle filter: in under 5 seconds of data collection and with fewer than 30 range measurements, the particle filter converges to a single, fairly compact blob estimating the distribution of the robot's position estimate. In the future, it may be desirable to use a method like this to obtain an initial state estimate with which to seed the EKF. With some running time improvements, it may be possible to use this method to bootstrap a realtime system; it currently requires an average of 0.29 seconds/update. Alternatively, it could be used online as-is, simply ignoring ranges that it couldn't process in time.

## Particle Filter Converging from no Initial Estimate

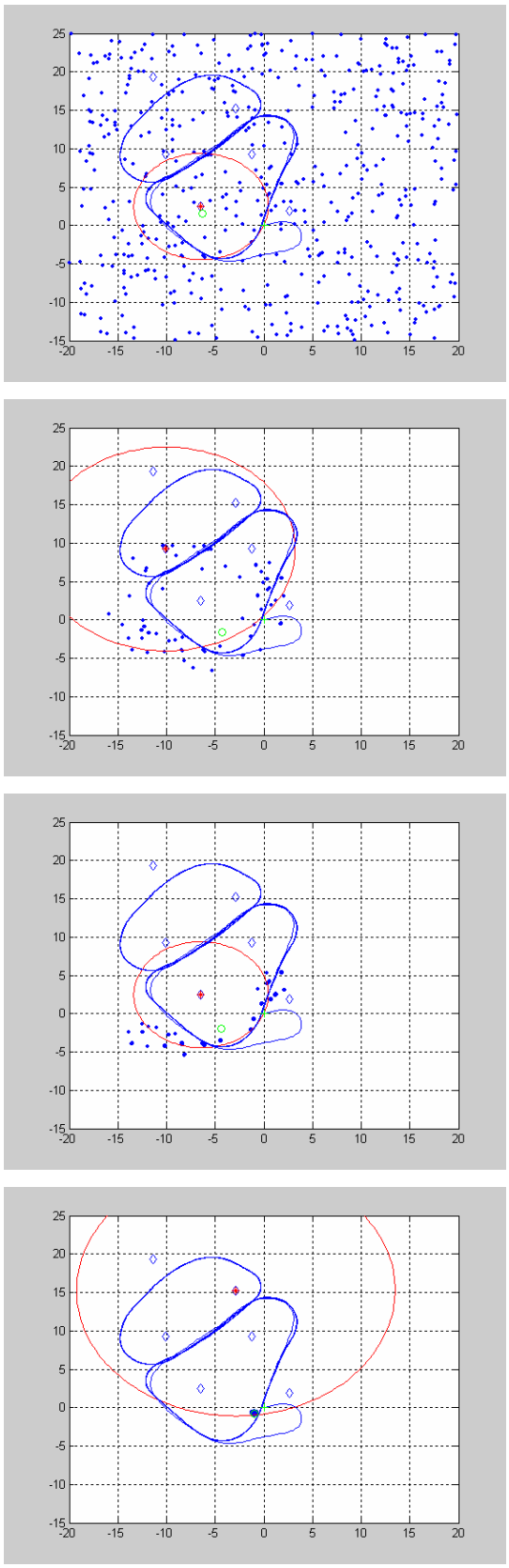



**Figure 23**

**Initially, the 1000 particles are evenly distributed, i.e., the initial state of the robot is completely unknown. It converges to a single clump of particles after receiving 28 range measurements, which takes about 4.25 seconds. This illustrates one of the main advantages of the particle filter: its ability to bootstrap without a good initial condition.**

**Gridlines on these plots are at 5 meter intervals.**

# 9 Extending the Kalman Filter for SLAM

So far we have considered the case where we have complete knowledge of the location of the RF beacons in the robot's environment. It will be more interesting, however, to see if we can localize the beacons from some noisy initial estimates while we also estimate the robot's pose. This is commonly called the problem of SLAM, or Simultaneous Localization and Mapping.

To test our approach to SLAM, we artificially create noisy initial beacon position estimates by adding noise to their known true positions. However, we could obtain these noisy estimates by other methods in a real application. For example, we could easily trust dead reckoning for some initial portion of the robot's travel and use triangulation of range circles to estimate the beacon positions. Later, once a beacon is known with some certainty, we would begin incorporating it's measurements into our robot pose estimates.

## 9.1 Problem Formulation

**9.1.1 Process Model.** Extending the formulation given in Section 6.1 is simply a matter of expanding the state vector to contain the beacon locations. So,

$$q_k = \begin{bmatrix} x_k \\ y_k \\ \theta_k \\ x_{t1} \\ y_{t1} \\ \dots \\ x_{tn} \\ y_{tn} \end{bmatrix},$$

where $n$ is the number of RF beacons. Now the system matrix $A(k)$ is represented by the Jacobian:

$$A(k+1) = \frac{\partial f}{\partial q_k}\Big|_{q=\hat{q}_k} = \begin{bmatrix} 1 & 0 & -\Delta D_k \sin(\theta_k) & 0 & & 0 \\ 0 & 1 & \Delta D_k \cos(\theta_k) & 0 & & 0 \\ 0 & 0 & 1 & 0 & & 0 \\ 0 & 0 & 0 & 1 & & 0 \\ & & & & \ddots & 0 \\ 0 & 0 & 0 & 0 & & 1 \end{bmatrix}$$

The input gain matrix *B(k)* is constructed similarly:

$$B(k+1) = \frac{\partial f}{\partial u_k}\Big|_{q=\hat{q}_k} = \begin{bmatrix} \cos(\theta_k) & 0 \\ \sin(\theta_k) & 0 \\ 0 & 1 \\ 0 & 0 \\ \vdots & \vdots \\ 0 & 0 \end{bmatrix}$$

**9.1.2 Measurement Model.** The range at time *k* from a beacon *b*, located at $(x_b, y_b)$, to the robot with state vector $q_k$ can be expressed as:

$$h(q_k, [x_b, y_b]^T) = \sqrt{(x_k - x_b)^2 + (y_k - y_b)^2}$$

The Jacobian *H(k)* gives us the measurement matrix:

$$H(k+1) = \frac{\partial h}{\partial q}\Big|_{q=\hat{q}_{k+1}} = \left[ \frac{\partial h}{\partial x_k}, \frac{\partial h}{\partial y_k}, \frac{\partial h}{\partial \theta_k}, \frac{\partial h}{\partial x_{t1}}, \frac{\partial h}{\partial y_{t1}}, \cdots, \frac{\partial h}{\partial x_b}, \frac{\partial h}{\partial y_b} \cdots, \frac{\partial h}{\partial x_{tn}}, \frac{\partial h}{\partial y_{tn}} \right]$$

where,

$$\frac{\partial h}{\partial x_k} = \frac{(x_k - x_b)}{\sqrt{(x_k - x_b)^2 + (y_k - y_b)^2}}$$

$$\frac{\partial h}{\partial y_k} = \frac{(y_k - x_b)}{\sqrt{(x_k - x_b)^2 + (y_k - y_b)^2}}$$

$$\frac{\partial h}{\partial \theta_k} = 0$$

$$\frac{\partial h}{\partial x_{t1}} = \frac{\partial h}{\partial y_{t1}} = \frac{\partial h}{\partial x_{tn}} = \frac{\partial h}{\partial y_{tn}} = 0$$

$$\frac{\partial h}{\partial x_b} = \frac{-(x_k - x_b)}{\sqrt{(x_k - x_b)^2 + (y_k - y_b)^2}}$$

$$\frac{\partial h}{\partial y_b} = \frac{-(y_k - y_b)}{\sqrt{(x_k - x_b)^2 + (y_k - y_b)^2}}$$

Only the terms in *H(k)* directly related to the current range measurement (i.e., the partials with respect to the robot pose and the coordinates of the tag giving the current measurement) are non-zero.

To complete the new formulation, **Q** is expanded to the correct dimensionality (i.e., 2*n*+3 square), with the extra elements filled in as zeros. $\Gamma$ remains the same, and **P** is expanded to the same dimensionality as **Q**. The values of **P** will be mentioned in the Results section.

## 9.2 Results

We assume that the initial robot pose is known to within 1*m* in x and y and to within 0.1 radians. In the results given here, Gaussian noise with a variance of $0.5m^2$ is added to the true tag position coordinates to test how well the filter converges from noisy initial tag position estimates. So, the initial value of the covariance matrix **P** is:

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 & & 0 \\ 0 & 1 & 0 & 0 & & 0 \\ 0 & 0 & 0.1 & 0 & & 0 \\ 0 & 0 & 0 & 0.5 & & 0 \\ & & & & \ddots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.5 \end{bmatrix}$$

Since the Kalman filter must linearize the range measurements, we have the familiar problem that occurs in all linear methods. If the initial estimate is reasonably close to the correct value, then the filter will converge correctly. A bad initial estimate, however, leads to divergence. This SLAM formulation is more delicate than the localization formulation, so we must keep the initial estimate of the robot pose close to the true pose. In the results given below, the initial robot pose estimate is perturbed by 20*cm* in the x and y directions and by 0.1 radians. These are plots of sample results; the results differ with each experiment, since the noise added to the tag positions changes.

**9.2.1 Running Time.** When applied to SLAM, the Kalman Filter requires an average of 0.0025 seconds/update (essentially identical to the time required for localization).

**9.2.2 Figure-8 Path Data Set.** In the experiment illustrated here, the true initial robot position from GPS was used as the initial estimate. The first 45 seconds of the robot's travel are clipped from this plot, giving the filter time to converge. The tag positions are initially estimated at the upward-pointing triangles, and their final position estimates are the downward-pointing triangles. The black stars indicate the tags' true positions.

**Kalman Filter SLAM Path and Beacon Position Estimates (Figure-8 Path)**

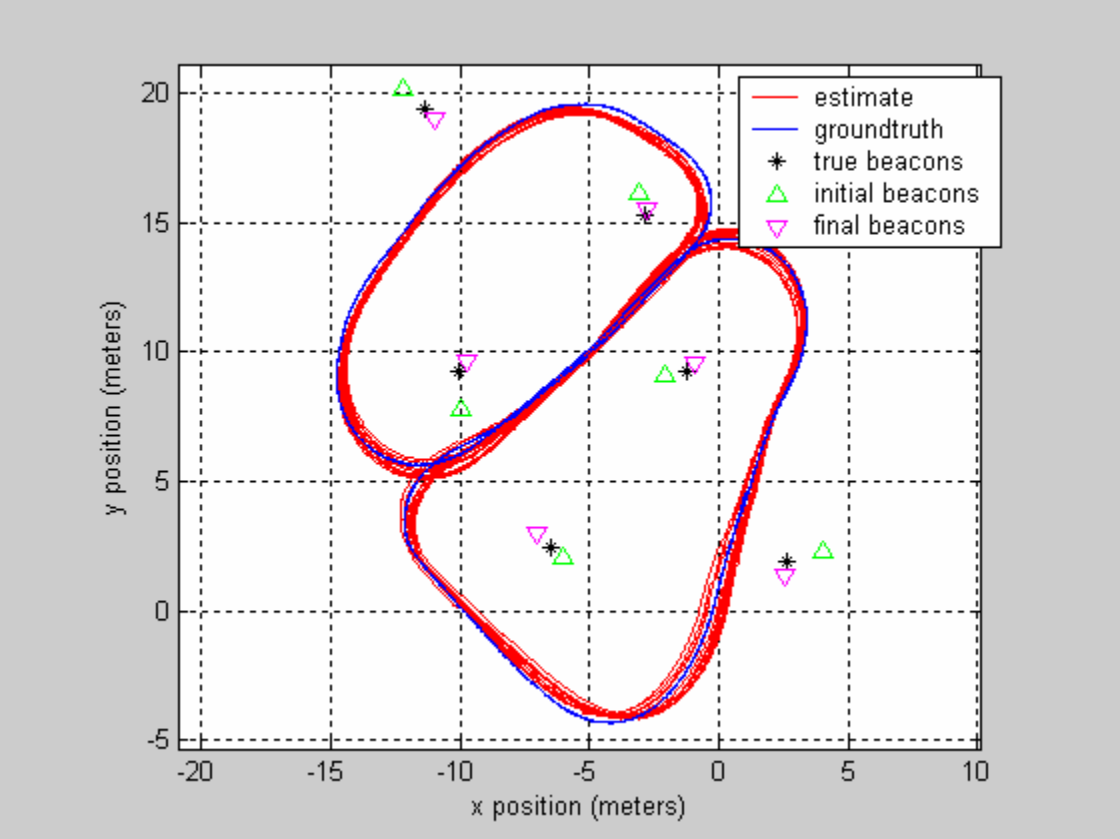



**Figure 24. Kalman filter SLAM performance on the figure-8 path.**

| | XTE | ATE |
|---|---|---|
| Mean Absolute | 0.2367 *m* | 0.4727 *m* |
| Maximum | 1.0482 *m* | 1.2961 *m* |
| Standard Deviation | 0.1407 *m* | 0.3544 *m* |

| | Initial | Final |
|---|---|---|
| Tag position error | 1.0561 *m* | 0.5169 *m* |

**Table 8. SLAM performance statistics on the figure-8 path.**

### 9.2.3 Circular Path Data Set.

**Kalman Filter SLAM Path and Beacon Position Estimates (Circular Path)**

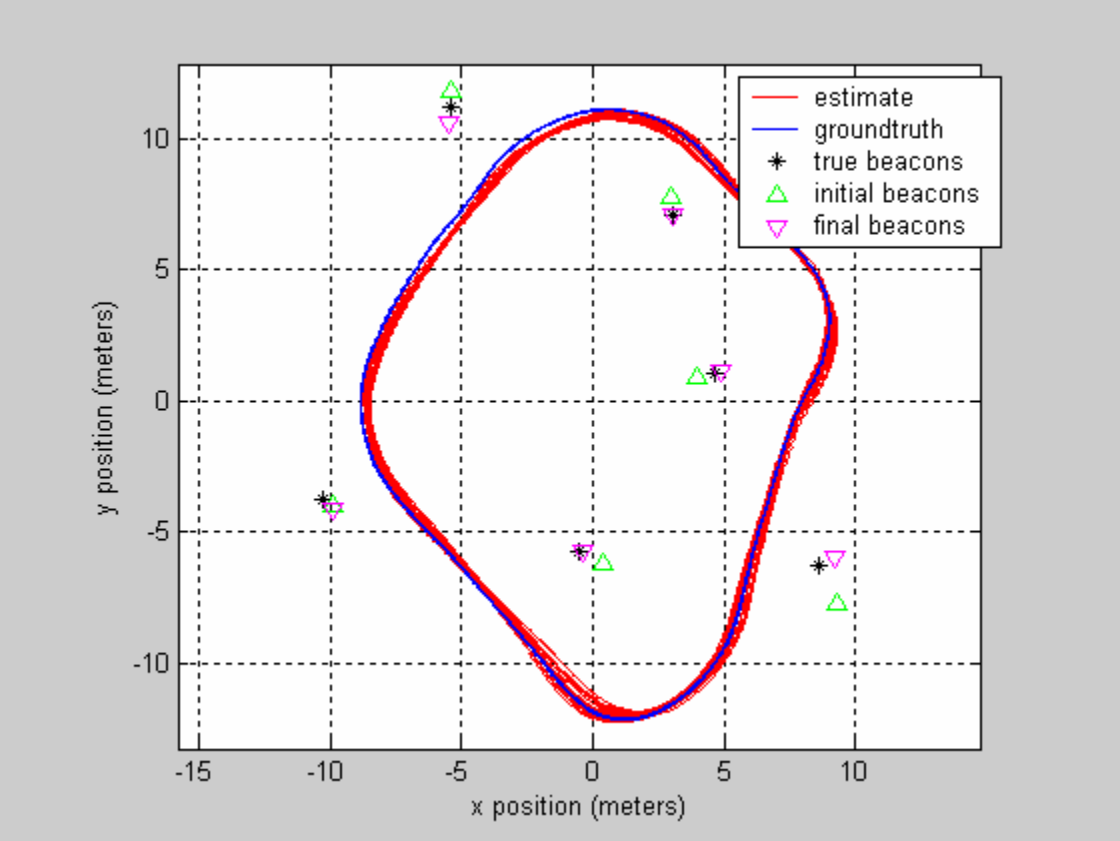



**Figure 25.** **Kalman filter SLAM performance on the circular path.**

| | XTE | ATE |
|---|---|---|
| Mean Absolute | 0.1695 *m* | 0.2744 *m* |
| Maximum | 0.6184 *m* | 0.8537 *m* |
| Standard Deviation | 0.1291 *m* | 0.1873 *m* |

| | Initial | Final |
|---|---|---|
| Tag position error | 0.8498 *m* | 0.3606 *m* |

**Table 9.** **SLAM performance statistics on the circular path.**

**9.2.4 Summary.** In this case, the error in the estimated path is comparable the errors in straight localization with an EKF. However, note that the initial tag location errors are very low -- around 1 meter or less. In a real world application, we expect that we may not know the tag locations this accurately, so we will probably need a more robust method. We note, however, that this method can easily run online, just as the EKF localization method, having the same running time.

# 10 Conclusions and Future Work

In this report we have presented two carefully-collected data sets and processed them with an extended Kalman filter, a sliding batch method, and a particle filter. The results are summarized graphically in Figures 26-29. Chiefly, we consider the cross-track error (abbreviated XTE), which gives the component of position error that is orthogonal to the robot's path. We also present the along-track error (ATE), which measures the tangential component of position error.

Table 10 summarizes the average time each algorithm requires to incorporate an incoming range measurement into the robot position estimate.

**Running Times:**

| | seconds per measurement update |
|---|---|
| EKF | 0.0024 |
| Sliding Batch | 2.8 |
| Particle Filter | 0.29 |
| EKF SLAM | 0.0025 |

**Table 10. Comparing the time required to update the robot pose estimate after a range measurement is taken.**

The EKF's compact representation allows real-time localization to within 20 cm or less (XTE), but requires an accurate *a priori* estimate of the robot's pose. The particle filter shows great promise as a means to acquire an initial pose estimate in real time; a formulation which estimates orientation as well as planar position is needed. A future system might coordinate the accuracy of the Kalman filter with the particle filter's robustness, switching between the two when appropriate. Switching could possibly be triggered by a threshhold in the state covariance matrix and/or a measure of the variance in particle positions.

The batch method can recover quite well from an erroneous initial estimate, but it requires offline computing time. Many extensions to the current batch algorithm are possible, including a SLAM formulation (which would surely overcome the limitations of our EKF SLAM formulation) and a variable state dimension filter.

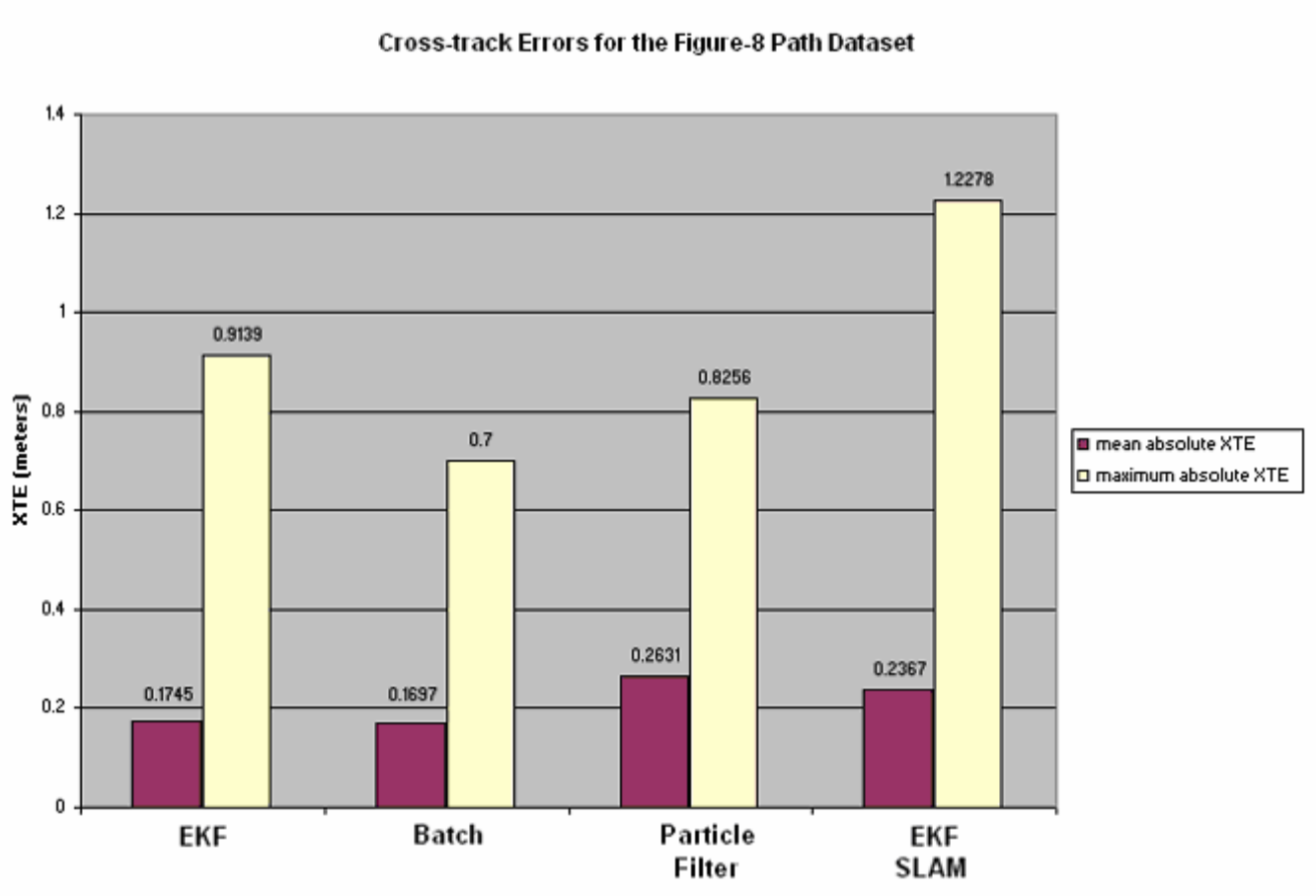



**Figure 26.** **The sliding batch method outperformed all other algorithms when measuring cross-track error (the error orthogonal to the true path). However, due to assumptions in our formulation, it did not do as well when along-track error is considered.**

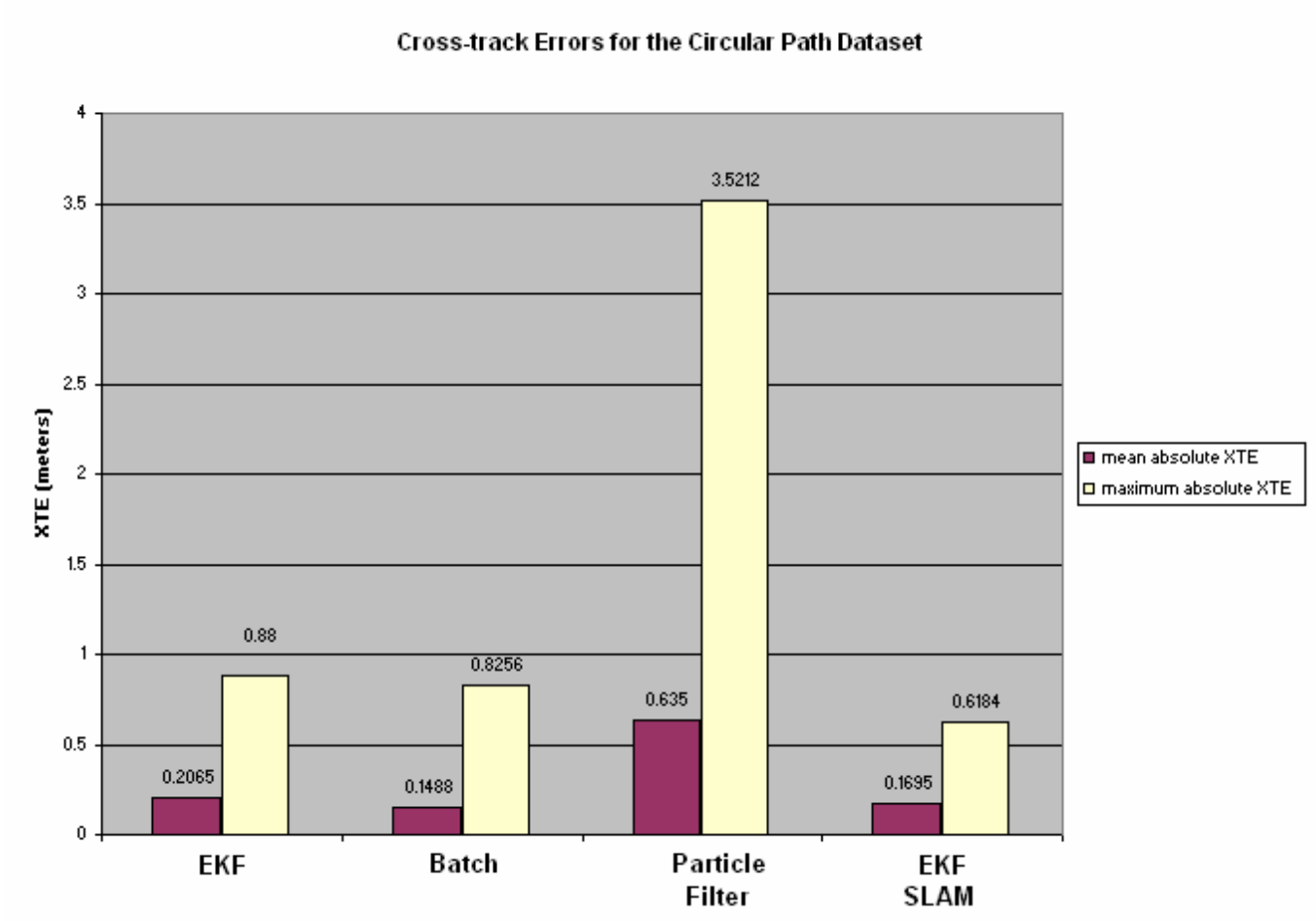

Cross-track Errors for the Circular Path Dataset
XTE (meters)
4
3.5
3
2.5
2
1.5
1
0.5
0
0.2065
0.88
0.1488
0.8256
0.635
3.5212
0.1695
0.6184
EKF
Batch
Particle Filter
EKF SLAM
mean absolute XTE
maximum absolute XTE

**Figure 27.**

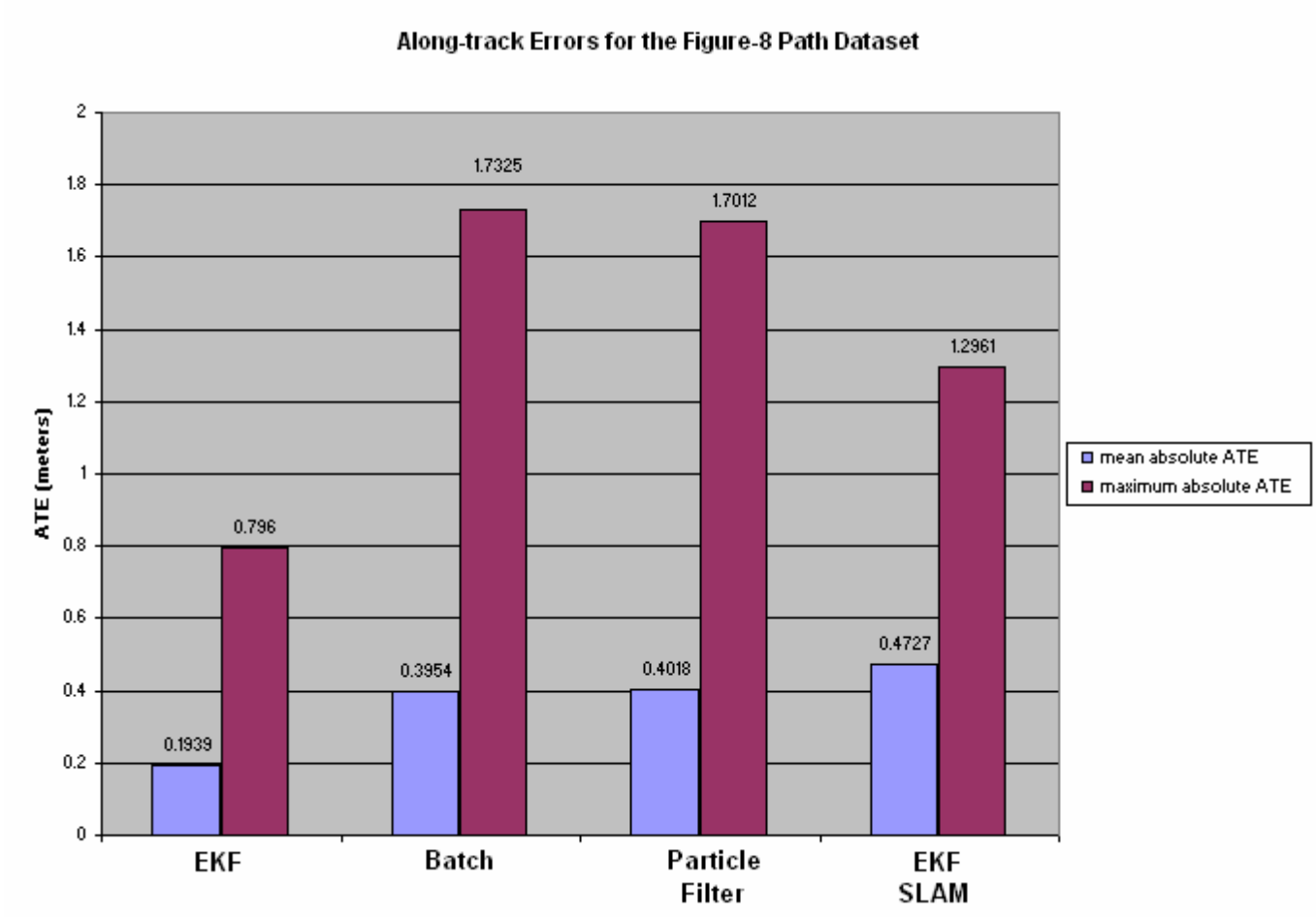



**Figure 28.** **The EKF approach performed the best when considering along-track error. Other algorithms might perform better with further development.**

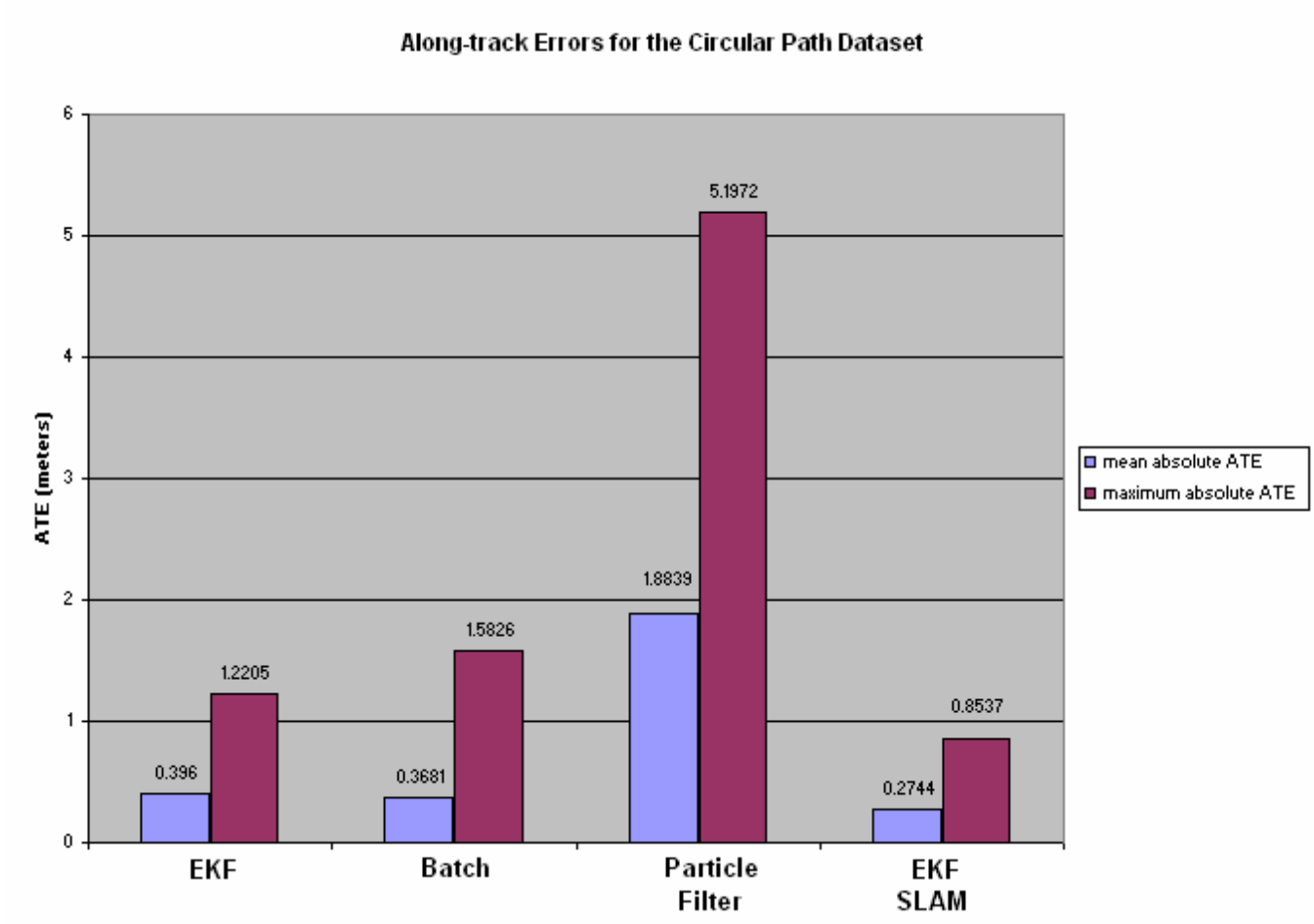

Along-track Errors for the Circular Path Dataset
ATE (meters)
6
5
4
3
2
1
0
5.1972
1.8839
1.5826
1.2205
0.8537
0.396
0.3681
0.2744
EKF
Batch
Particle Filter
EKF SLAM
mean absolute ATE
maximum absolute ATE

**Figure 29.**

# 11 References


[1] R. Want, A. Hopper, V. Falco, and J. Gibbons. The Active Badge location system. *ACM Transactions on Information Systems*, 10:91-102, Jan. 1992.

[2] A. Ward, A. Jones, and A. Hopper. A new location technique for the active office. *IEEE Personal Communications*, 4(5):42-47, Oct. 1997.

[3] N. Priyantha, A. Chakraborty, and H. Balakrishman. The Cricket location support system. In *Proc. of the 6th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM 2000)*, pages 32-43, Boston, MA, August 2000.

[4] N. Priyantha, A. Miu, H. Balakrishman, and S. Teller. The Cricket compass for context-aware mobile applications. In *Proc. of the ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM 2000)*, pages 1-14, Rome, Italy, July 2001.

[5] P. Bahl and V. N. Padmanabhan. RADAR: An in-building, RF based user location and tracking system. In *Proc. of IEEE Infocom 2000*, vol. 2, pp. 775-784. Tel-Aviv, Israel, March 2000.

[6] A. M. Ladd, K. E. Bekris, G. Marceau, A. Rudys, D. S. Wallach, and L. E. Kavraki. Robotics-based location sensing for wireless Ethernet. In *ACM MobiCom*, 2002.

[7] J. Hightower, R. Want, and G. Borriello. Spoton: An indoor 3d location sensing technology based on RF signal strength. In *Technical Report UW CSE 00-02-02, University of Washington, Dept. of Computer Science and Engineering*, Seattle, WA, February 2000.

[8] R. Smith and P. Cheeseman. On the representation of spatial uncertainty. *Journal of Robotics Research*, 5(4):56-68, 1987.

[9] R. E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME*, pages 35-45, 1960.

[10] J. F. Leonard and H. Durrant-Whyte. Mobile robot localization by tracking geometric beacons. *IEEE Transactions Robotics and Automations*, 7(3):376-382, June 1991.

[11] L. Moreno, J. M. Armingol, A. de la Escalera, and M. A. Salichs. Global integration of ultrasonic sensors information in mobile robot localization. In *Proceedings of the Ninth Annual Conference on Advanced Robotics*, pp. 283-288, October 1999.

[12] J. Z. Sasiadek and P. Hartana. Sensor data fusion using Kalman filter. In *Proceedings of the Third International Conference on Information Fusion*, vol. 2, pp. 19-25, 2000.

[13] J. S. Guttman, W. Hatzack, I. Herrmann, B. Nebel, F. Rittinger, A. Topor, and T. Weigel. Reliable self-localization, multirobot sensor integration, accurate path-planning and basic soccer skills: playing an effective game of robotic soccer. In *Proceedings of the Ninth Annual Conference on Advanced Robotics*, pp. 289-296, October 1999.

[14] S. Thrun, D. Fox, W. Burgard, and F. Dellaert. Robust Monte Carlo localization for mobile robots. *Artificial Intelligence*, pp. 99-141, 2000.

[15] J. Liu and R. Chen. Sequential Monte Carlo methods for dynamic systems. *Journal of American Statistical Association*, 93, 1998.

[16] M. Pitt and N. Shephard. Filtering via simulation: auxiliary particle filter. *Journal of American Statistical Association*, 94, 1999.

[17] A. Doucet, J. F. G. de Freitas, and N. J. Gordon, editors. *Sequential Monte Carlo Methods in Practice*. Springer, 2001.

[18] E. Kiriy. A Localization System for Autonomous Golf Course Mowers. M. Eng. Thesis, Dept. of Electrical Engineering, McGill University, November 2002.

[19] P. F. McLauchlan. A batch/recursive algorithm for 3d scene reconstruction. In *International Conference on Computer Vision and Pattern Recognition*, Hilton Head, SC, USA, 2000, vol. 2, pp. 738-743.

[20] M. Deans. Maximally informative statistics for localization and mapping. In *Proc. of the IEEE Internation Conference on Robotics and Automation*, Washington, D.C., May 2002.

[21] P. Newman and J. Leonard. Pure range-only subsea SLAM**.** *IEEE International Conference on Robotics and Automation*, Taiwan, September 2003.

[22] M. Isard and A. Blake. Condensation -- conditional density propagation for visual tracking. International Journal of Computer Vision 29(1), pp. 5--28, 1998.

[23] J. Werb. Designing a positioning system for finding things and people indoors. *IEEE Spectrum*, 35(9):71-78,1998.

[24] F. Chenavier and J. Crowley. Position estimation for a mobile robot using vision and odometry. *Proc. of the IEEE International Conference on Robotics and Automation*, Nice, France, May 1992.

[25] Y. Bar-Shalom, X. Rong Li, and T. Kirubarajan. *Estimation with Applications to Tracking and Navigation*, 2001.

[26] R. Smith, M. Self, and P. Cheeseman. Estimating uncertain spatial relationships in robotics. In *Autonomous Robot Vehicles*, I. Cox and G. Wilfong, eds., pp. 167-193, Springer-Verlang, 1990.

[27] G. Kantor and S. Singh. Preliminary results in range-only localization and mapping. In *Proc. of the IEEE Conference on Robotics and Automation*, May 2002.

[28] D. Kurth, G. Kantor and S. Singh. Experimental results in range-only localization with radio. In Proceedings, IROS, October 2003, Las Vegas, USA.