Recursive Total Least Squares: An Alternative to Using the Discrete Kalman Filter in Robot Navigation*

Daniel L. Boley and *Erik S. Steinmetz*
Department of Computer Science
University of Minnesota
Minneapolis, MN 55455

Karen T. Sutherland
Department of Computer Science
University of Wisconsin – La Crosse
La Crosse, WI 54601

Abstract

In the robot navigation problem, noisy sensor data must be filtered to obtain the best estimate of the robot position. The discrete Kalman filter, commonly used for prediction and detection of signals in communication and control problems, has become a popular method to reduce the effect of uncertainty from the sensor data. However, in the domain of robot navigation, sensor readings are not only uncertain, but can also be relatively infrequent, compared to traditional signal processing applications. Hence, there is a need for a filter that is capable of converging with many fewer readings than the Kalman filter. To this end, we propose the use of a Recursive Total Least Squares Filter. This filter is easily updated to incorporate new sensor data, and in our experiments converged faster and to greater accuracy than the Kalman filter.

1 Introduction

The discrete Kalman filter, commonly used for prediction and detection of signals in communication and control problems, has become a popular method of reduc-

*This work was supported jointly by Minnesota Department of Transportation grant 71789-72996-173 and National Science Foundation grant CCR-9405380.

ing uncertainty in robot navigation. A brief summary of the Kalman filter can be found in [2] and a complete description in [9]. One of the main advantages of using the Kalman filter is that it is recursive, eliminating the necessity for storing large amounts of data. It requires a good initial estimate of the solution. It also assumes that the noise obeys a weighted white gaussian distribution. The Kalman filter is guaranteed to be optimal only in that it is guaranteed to find the best solution in the least squares sense.

Although originally designed as an estimator for dynamical systems, the filter is used in many applications as a static state estimator [13]. Also, due to the fact that functions are frequently non-linear, the extended Kalman filter (EKF) rather than the Kalman filter itself is often used [1, 11]. In this case, the function is linearized by taking a first order Taylor expansion. This linear approximation is then used as the Kalman filter equation.

There are two basic problems which can occur when using either the Kalman or extended Kalman filter in robot navigation applications:

- Due to the fact that the filter was developed for applications such as those in signal processing, it is assumed that many measurements are taken. Sensing in robot navigation, often done using camera images, is a time consuming process. To be useful, a method must succeed with relatively few readings.

- An underlying assumption in any least squares estimation is that the entries in the data matrix are error-free [7], e.g., the time intervals at which measurements are taken are exact. In many actual applications, the errors in the data matrix can be at least as great as the measurement errors. In such cases, the Kalman filter can give poor results.

Two additional problems occur when using the EKF:

- The linearization process itself has the potential to introduce significant error into the problem.

- The EKF is not guaranteed to be optimal or to even converge [14]. It can easily fall into a local minimum when an initial estimate of the solution is poor, often the type of situation faced by robot navigators.

Although limited modifications can be made to the Kalman approach to improve robustness to noise [12], our work in outdoor navigation [17], where measurements are expensive to obtain and have significant error inherent to the system, motivated

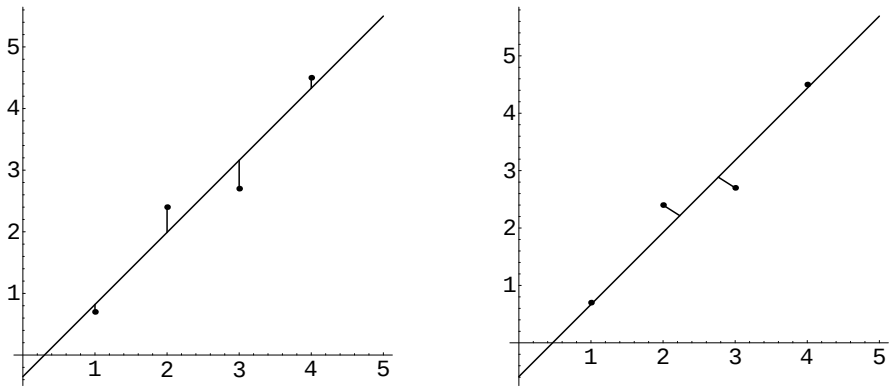

Figure 1: *In an LS solution, as shown on the left, the sum of the squared vertical distances to the line of best fit is minimized. In a TLS solution, as shown on the right, the sum of the squared perpendicular distances to the line of best fit is minimized.*

us to look for another filtering method, preferably one which would not require numerous measurements to converge and did not assume an error-free data matrix. As demonstrated by Mintz et al [8], the criterion of optimality depends critically on the specific model being used. When error exists in both the measurement and the data matrix, the best solution in the *least squares* sense is often not as good as the best solution in the *eigenvector* sense, where the sum of the squares of the perpendicular distances from the points to the lines are minimized (Fig. 1). This second method is known in the statistical literature as *orthogonal regression* and in numerical analysis as *total least squares* (TLS) [18].

In the next section, we discuss the Recursive TLS algorithm, in section 3 we present our experimental results, and in section 4 we offer some concluding remarks.

2 Recursive Total Least Squares Algorithm

Given an overdetermined system of equations $A\mathbf{x} = \mathbf{b}$, the TLS problem, in its simplest form, is to find the smallest perturbation to A and $\mathbf{b}$ to make the system of equations compatible. Specifically, we seek a matrix E and vector $\mathbf{f}$ that minimizes $\|(E, \mathbf{f})\|_2$ such that $(A+E)\mathbf{x} = \mathbf{b}+\mathbf{f}$ for some vector $\mathbf{x}$. The vector $\mathbf{x}$ corresponding to the optimal $(E, \mathbf{f})$ is called the *TLS solution*. Recently, some recursive TLS filters have been developed for applications in signal processing [4, 5, 20]. Davila

[4] used a Kalman filter to obtain a fast update for the eigenvector corresponding to the smallest eigenvalue of the covariance matrix. This eigenvector was then used to solve a symmetric TLS problem for the filter. It was not explained how the algorithm might be modified for the case where the smallest eigenvalue is multiple (i.e., corresponding to a noise subspace of dimension higher than one), or variable (i.e., of unknown multiplicity). In [20], Yu described a method for the fast update of an approximate eigendecomposition of a covariance matrix. He replaced all the eigenvalues in the noise subspace with their "average", and did the same for the eigenvalues in the signal subspace, obtaining an approximation which would be accurate if the exact eigenvalues could be grouped into two clusters of known dimensions. In [5], DeGroat used this approach combined with the averaging technique used in [20], again assuming that the singular values could be grouped into two clusters. Recently, Bose et al.[3] applied Davila's algorithm to reconstruct images from noisy, undersampled frames after converting complex-valued image data into equivalent real data. All of these methods made some assumptions that the singular values or eigenvalues could be well approximated by two tight clusters, one big and one small. In this paper, we present a recursive algorithm that makes very few assumptions about the distribution of the singular values.

The most common algorithms to compute the TLS solution are based on the Singular Value Decomposition (SVD), a non-recursive matrix decomposition which is computationally expensive to update. The TLS problem can be solved by the SVD using Algorithm 3.1 of [18]. The main computation cost of that algorithm occurs in the computation of the SVD. That cost is $O(p^3)$ for each update. The basic solution method is sketched as follows. If $\mathbf{v} = (v_1, \ldots, v_p)^T$ is a right singular vector corresponding to the smallest singular value of $(A, \mathbf{b})$, then it is well known that the TLS solution can be obtained by setting $\mathbf{x} = -(v_1, \ldots, v_{p-1})^T / v_p$. If the smallest singular value is multiple, then there are multiple TLS solutions, in which case one usually seeks the solution of smallest norm. If v_p is too small or zero, then the TLS solution may be too big or nonexistent, in which case an approximate solution of reasonable size can be obtained by using the next smallest singular values(s) [18].

In cases such as the applications considered in this paper where the exact TLS solution is still corrupted by external effects such as noise, it suffices to obtain an approximate TLS solution at much less cost. We seek a method that can obtain a good approximation to the TLS solution, but which admits rapid updating as new data samples arrive. One such method is the so-called ULV Decomposition, first introduced by Stewart [15] as a means to obtain an approximate SVD which can be

easily updated as new data arrives, without making any a priori assumptions about the overall distribution of the singular values. The ULV Decomposition of a real $n \times p$ matrix A (where $n \geq p$) is a triple of 3 matrices U, L, V plus a rank index r, where $A = ULV^T$, V is $p \times p$ and orthogonal, L is $p \times p$ and lower triangular, U has the same shape as A with orthonormal columns, and where L has the form

$$L = \begin{pmatrix} C & 0 \\ E & F \end{pmatrix}$$

where C ($r \times r$) encapsulates the "large" singular values of A, (E, F) ($(p-r) \times p$) approximately encapsulate the $p - r$ smallest singular values of A, and the last $p - r$ columns of V encapsulate the corresponding trailing right singular vectors. To solve the TLS problem, the U matrix is not required, hence we need to carry only L, V, and the effective rank r. Therefore, a given ULV Decomposition can be represented just by the triple $[L, V, r]$.

The feature that makes this decomposition of interest is the fact that, when a new row of coefficients is appended to the A matrix, the L, V and r can be updated in only $O(p^2)$ operations, with L restored to the standard form above, as opposed to the $O(p^3)$ cost for an SVD. In this way, it is possible to track the leading r-dimensional "signal subspace" or the remaining "noise subspace" relatively cheaply. Details on the updating process can be found in [15, 10].

We can adapt the ULV Decomposition to solve the Total Least Squares (TLS) problem $A\mathbf{x} \approx \mathbf{b}$, where new measurements b are continually being added, as originally proposed in [2]. The adaptation of the ULV to the TLS problem has also been analyzed independently in great detail in [19], though the recursive updating process was not discussed at length. For our specific purposes, let A be an $n \times (p-1)$ matrix and $\mathbf{b}$ be an n-vector, where p is fixed and n is growing as new measurements arrive. Then given a ULV Decomposition of the matrix $(A, \mathbf{b})$ and an approximate TLS solution to $A\mathbf{x} \approx \mathbf{b}$, our task is to find a TLS solution $\hat{\mathbf{x}}$ to the augmented system $\widehat{A}\hat{\mathbf{x}} \approx \widehat{\mathbf{b}}$, where

$$\widehat{A} = \begin{pmatrix} \lambda A \\ \mathbf{a}^T \end{pmatrix} \text{ and } \widehat{\mathbf{b}} = \begin{pmatrix} \lambda \mathbf{b} \\ \beta \end{pmatrix},$$

and λ is an optional exponential forgetting factor [9].

The RTLS Algorithm:

- Start with $[L, V, r]$, the ULV Decomposition of $(A, \mathbf{b})$, and the coefficients $\mathbf{a}^T, \beta$ for the new incoming equation $\mathbf{a}^T\mathbf{x} = \beta$.

- Compute the updated ULV Decomposition for the system augmented with the new incoming equation. Denote the new decomposition by $[\widehat{L}, \widehat{V}, \widehat{r}]$.
- Partition
$$\widehat{V} = \begin{pmatrix} \widehat{V}_{11} & \widehat{V}_{12} \\ \widehat{V}_{21} & \widehat{V}_{22} \end{pmatrix},$$
where $\widehat{V}_{22}$ is $1 \times (p - \widehat{r})$.

  If $\|\widehat{V}_{22}\|$ is too close to zero (according to a user supplied tolerance), then we can adjust the rank boundary $\widehat{r}$ down to obtain a more robust, but approximate solution [2].
- Find an orthogonal matrix Q such that $\widehat{V}_{22}Q = (0, \ldots, 0, \alpha)$, and let $\mathbf{v}$ be the last column of $\widehat{V}_{12}Q$. Then compute the new approximate TLS solution according to the formula $\hat{\mathbf{x}} = -\mathbf{v}/\alpha$.

This RTLS Algorithm makes very few assumptions about the underlying system, though the user must supply a zero tolerance and a gap tolerance for $\|\widehat{V}_{22}\|$. For the application here, it sufficed to set the zero tolerance to zero and depend on just the gap tolerance of 1.5.

3 Experimental Results

To compare the performance of the Kalman filter and RTLS in practice, we ran three sets of experiments, including one with a physical mobile robot and camera, and two in simulation. In the first set of experiments, we simulated a simple robot navigation problem typical of that faced by an actual mobile robot [1, 6, 11]. The robot has identified a single landmark in a two-dimensional environment and knows the landmark location on a map. It does not know its own position. It moves in a straight line and with a known uniform velocity. Its goal is to estimate its own start position relative to the landmark by measuring the visual angle α between its direction of heading and the landmark. Measurements are taken periodically as it moves. Figure 2 shows a diagram of the problem. For simplification, it is assumed that the landmark is located at (0,0), that the y coordinate of the robot's start position does not change as the robot moves (i.e. the robot heading defines the x axis), and that the robot knows what side of the landmark it is on. To map this robot-based coordinate system to the ground coordinate system, it suffices to know only the robot's compass heading from, say, an internal compass. Below we discuss a

simple way to dynamically incorporate readings from two landmarks, avoiding the need to know the compass heading independently.

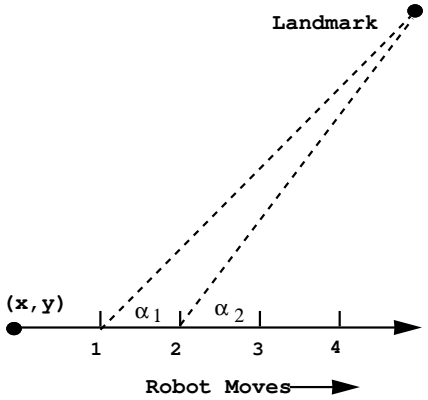

Figure 2: *Diagram of a simulated robot navigation problem. The robot moves along the horizontal line. Landmark location and velocity are known. Angle α_i is the angle from robot heading to the landmark at time t_i. The goal is to estimate the initial robot location (x,y).*

In our experiments, it was assumed that the y coordinate of the robot path was negative (i.e., the path, as shown in Figure 2, was on the side below the landmark), that robot $velocity$ was 20 per unit of time and that measurements of α were taken at unit time intervals. At any time t_i:

$$cot(\alpha_i) = \frac{x + t_i * velocity}{y}$$

where (x, y) is the robot start position and α_i is the angle from the robot heading to the landmark. Random error with a uniform distribution was added to the angle measures and a normally distributed random error was added to the time measurement. We formulated the problem so that the data matrix, as well as the measurement vector contained error:

$$A_i = \begin{bmatrix} 1 & -cot(\alpha_i) \end{bmatrix}; \quad \mathbf{x}_i = \begin{bmatrix} x \\ y \end{bmatrix}; \quad \mathbf{b}_i = -t_i * velocity$$

where, at time t_i, A_i is the data matrix, $\mathbf{b}_i$ is the measurement vector, and $\mathbf{x}_i$ is the estimated state vector consisting of the coordinates (x, y) of the robot start position. The Kalman filter was given an estimated start of (0,0). The RTLS algorithm

had no estimated start position provided. The leading column of the data matrix was scaled by $\eta = 100$ to reduce the allowed errors. Results are summarized in Figure 3.

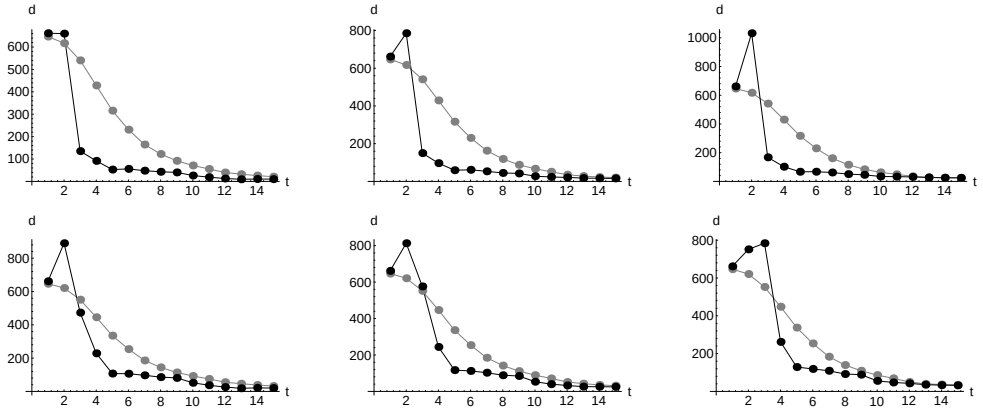

Figure 3: *Comparison of mean deviations from estimated to actual start position. Measurements were taken at unit time intervals (horizontal axis). The vertical axis gives the mean deviation d. The top three graphs have uniformly distributed error in α of $\pm 2°$ and normally distributed error in t with sd = 0, .05 and .1. The bottom three graphs have uniformly distributed error in α of $\pm 4°$ and normally distributed error in t with sd = 0, .05 and .1. Results using the RTLS algorithm are shown in black. Results using the Kalman filter are shown in grey.*

Error in α	Error in t	0	.05	.1
$\pm 2°$	Kalman	32.47	20.27	24.54
	RTLS	20.24	15.90	24.81
$\pm 4°$	Kalman	21.01	31.80	34.63
	RTLS	10.11	24.97	32.13

Table 1: *Mean deviation of estimate from actual location after 15 measurements.*

The mean deviations d (of 10 trials) of the estimates from the actual start location of (-460, -455) are compared for six different error amounts. The top three graphs have uniformly distributed error in α of $\pm 2°$ and normally distributed error in t with standard deviation sd = 0, .05, and .1. The bottom three graphs have

uniformly distributed error in α of $\pm 4°$ and normally distributed error in t with sd = 0, .05 and .1. The jump in the RTLS distance at the second measure is due to the fact that the RTLS filter does not require, and is not given, an initial estimate of location. The velocity/time interval used, combined with the error distribution used, produced error on some runs that gave readings of $\alpha_2 < \alpha_1$ (see Figure 2). Since there were only two measurements taken at this point, the system was not yet overdetermined, and the erroneous measures were given significant weight. This demonstrates how quickly the RTLS filter can recover from such errors. Table 1 gives the mean deviation from the actual location after 15 measurements. For all six groups of experiments, the RTLS filter converged more quickly than did the Kalman filter. After 15 measurements, the RTLS estimate was closer to the actual location than was the Kalman in five of the six groups.

The second set of experiments consisted of a sequence of indoor robot runs. As in the first set of experiments, the robot did not know its own position on the map, but did know the location of a single landmark. Its task was to take an image, find the landmark in the image, and use the result to determine its start position relative to the landmark.

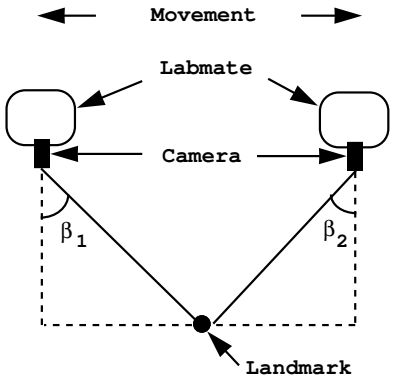

Figure 4: *TRC Labmate with camera mounted at* $90°$. *Angle measure is bound by* $\pm 25°22'$ *for the given field of view.*

A Panasonic WV-BL202 camera was mounted on a TRC Labmate at an angle of $90°$ to robot bearing. Horizontal field of view was $50°44'$. "Landmarks" were mini Maglite high intensity flashlight candles. The angular position of the landmark was measured in a sequence of images taken while the robot moved across the room at a constant velocity. In addition to the error in angle measure, error also occurred

in velocity, robot bearing and in the times at which the images were taken. It is not possible to predict and model these errors. For example, velocity was set at 20mm/second, but average true velocity across runs ranged from 21.4mm/second to 22.5mm/second. In addition, the supposed constant velocity was not constant throughout a single run, varying in an unpredictable manner. It would be unrealistic to assume any of these errors or their combined result to have a gaussian distribution. Figure 4 shows a diagram of how the angles are measured. When the landmark is in the left of the camera image, the angle (β_1 in the diagram) is negative. When the landmark is in the right of the camera image, the angle (β_2 in the diagram) is positive. Angle measure is thus bound by $\pm 25°22'$ for the given field of view.

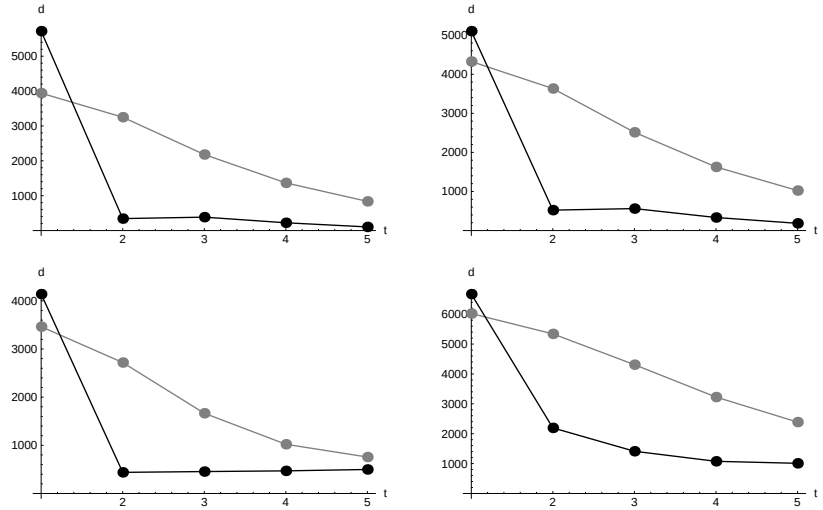

Figure 5: *Comparison of filters with actual robot runs: Images were grabbed at time intervals t (horizontal axis) 12 seconds apart. The vertical axis gives the deviation of the estimated start position from the actual start position in millimeters. The landmark was placed at a different location for each run. Results using the RTLS algorithm are shown in black. Results using the Kalman filter are shown in grey.*

It is again assumed that the landmark is located at (0,0), that the y coordinate of the robot's position does not change as the robot moves, and that the robot knows which side of the landmark it is on. At any step i:

$$tan(\beta_i) = \frac{x + (t_0 + i * interval) * velocity}{y}$$

where (x, y) is the robot start position, β_i is the measured angle, t_0 is robot start time, $interval$ is the interval at which images are grabbed and $velocity$ is the robot velocity. The problem was expressed as a linear function so that no accuracy was lost by linearizing. However, the data matrix as well as the measurement vector contained error:

$$A_i = \begin{bmatrix} 1 & -tan(\beta_i) \end{bmatrix}; \quad \mathbf{x}_i = \begin{bmatrix} x \\ y \end{bmatrix}; \quad \mathbf{b}_i = -(t_0 + i * interval) * velocity$$

where at any step i, A_i is the data matrix, $\mathbf{b}_i$ is the measurement vector and $\mathbf{x}_i$ is the estimated state vector consisting of the coordinates (x, y) of the robot start position. As in the previous set of experiments, the Kalman filter was given an estimated start position of (0,0) and the leading column of the data matrix was weighted by $\eta = 100$.

Figure 5 shows a comparison of four of the robot runs. The robot velocity was set to 20mm/sec. Five images were grabbed 12 seconds apart. Robot start position relative to the landmark used for localization was different in each run. The deviations d of the estimate of start location from actual start location at each 12 second time interval t are compared. As in the simulated runs, the RTLS filter converged faster and to more accuracy than did the Kalman.

The third set of experiments was again run in simulation, but used two landmarks without assuming any prior knowledge of the robot's heading. We assume that the robot has no instrument such as a compass which could be used to register its compass heading. Such instruments can give varying, incorrect readings in outdoor, unstructured environments [17], so that it is useful to design and evaluate methods to obtain heading information from external sources. Such heading information could be used independently or as corrections to estimates from internal sources. The robot knows the location of the two landmarks on a map (ground coordinate system). A coordinate system is arbitrarily centered at one landmark. The goal is to determine the robot start position plus the location of the second landmark. Knowing which landmark is which in the view will allow the robot to uniquely determine its position, except for certain degenerate configurations, but even if the robot does not know the order of the two landmarks in its view, it can limit its start position to only two possible locations in the ground coordinate system, symmetrically located on either side of the line joining the landmarks, without any *a priori* knowledge of direction.

The coordinate system is defined by placing landmark 1 at $(0, 0)$ and landmark 2 at coordinates (l, m) to be determined by the filter. The x-axis is defined by the

direction of the robot heading. The computed coordinates (l, m) permit mapping this coordinate system to the ground coordinate system. Generalizing figure 2, we let α_{1i}, α_{2i} be the angles from the robot heading to each of the landmarks at time t_i. We have the following relationships:

$$\begin{aligned} -sin(\alpha_{1i}) * x + cos(\alpha_{1i}) * y &= t_i * velocity * sin(\alpha_{1i}) \\ -sin(\alpha_{2i}) * x + cos(\alpha_{2i}) * y + sin(\alpha_{2i}) * l - cos(\alpha_{2i}) * m & \\ = t_i * velocity * sin(\alpha_{2i}) & \end{aligned}$$

where (x, y) is the robot start position. Random error with a uniform distribution was added to the angle measures and a normally distributed random error was added to the time measurement. As in the previous experiments, the problem was expressed as a linear function with the data matrix as well as the measurement vector containing error:

$$A_i = \begin{bmatrix} -sin(\alpha_{1i}) & cos(\alpha_{1i}) & 0 & 0 \\ -sin(\alpha_{2i}) & cos(\alpha_{2i}) & sin(\alpha_{2i}) & -cos(\alpha_{2i}) \end{bmatrix};$$

$$\mathbf{x}_i = \begin{bmatrix} x \\ y \\ l \\ m \end{bmatrix}; \quad \mathbf{b}_i = \begin{bmatrix} t_i * velocity * sin(\alpha_{1i}) \\ t_i * velocity * sin(\alpha_{2i}) \end{bmatrix}$$

where at any step i, A_i is the data matrix, $\mathbf{b}_i$ is the measurement vector and $\mathbf{x}_i$ is the estimated state vector consisting of the coordinates (x, y) of the robot start position and the coordinates (l, m) of the second landmark. Results are summarized in Figure 6. The mean deviations d (of 19 trials) of the estimates from the actual start location of (-460, -455) are compared for six different error amounts. As in the first set of simulations, the RTLS algorithm quickly recovers from the jump due to its lack of an initial estimate. Furthermore, in the regions where the RTLS error exceeds the Kalman filter error, neither filter yields any accuracy at all, since both errors are larger than the values being estimated.

4 Conclusion

In this paper, we have proposed a Recursive Total Least Squares (RTLS) filter. This filter is easily updated as new data arrives, yet makes very few assumptions about

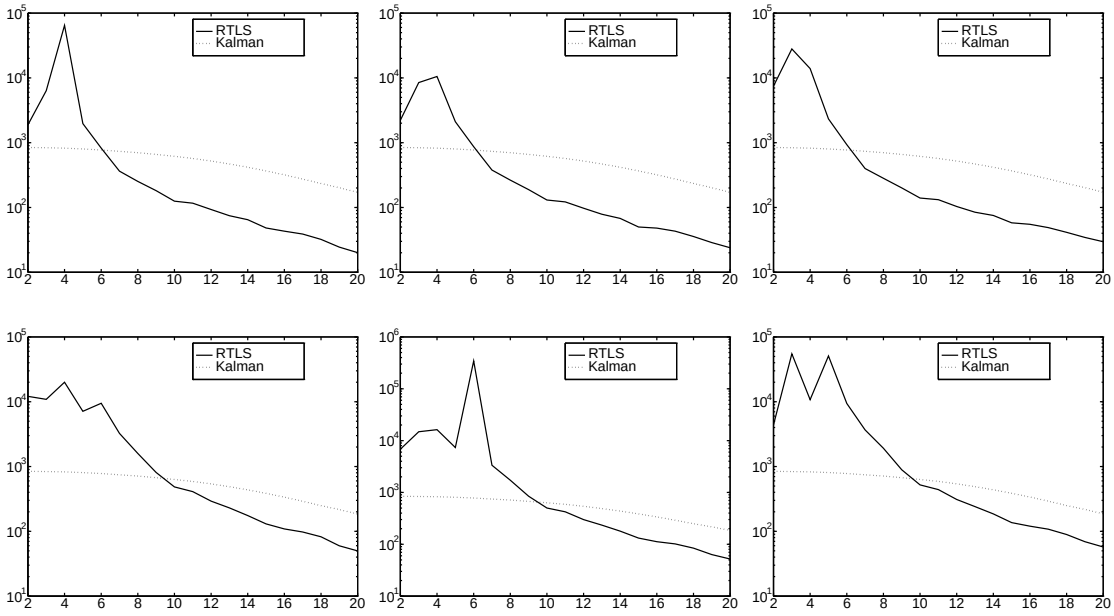

Figure 6: *Comparison of mean deviations from estimated to actual start position. Measurements were taken at unit time intervals (horizontal axis). The vertical axis gives the mean deviation d. The top three graphs have uniformly distributed error in both α_1 and α_2 of $\pm 2^\circ$ and normally distributed error in t with sd = 0, .05 and .1. The bottom three graphs have uniformly distributed error in both α_1 and α_2 of $\pm 4^\circ$ and normally distributed error in t with sd = 0, .05 and .1.*

the data or the problem being solved. The method was based on the ULV Decomposition. We suggest its use as an alternative to the Kalman filter in reducing uncertainty in robot navigation. In this context RTLS does not require an initial state estimate, avoids modeling errors introduced by the extended Kalman filter, does not suffer the traps of local minima, and converges quickly. We have illustrated the method with simulated as well as actual robot runs. It is demonstrated that in the domain of robot navigation the RTLS can often provide more accurate estimates in fewer time steps than the Kalman filter, especially when errors are present in both the measurement vector and the data matrix. Future work includes utilizing the filter in navigation problems with actual outdoor terrain data and combining its use with the higher level reasoning described in [16].

References

[1] N. Ayache and O. D. Faugeras. Maintaining representations of the environment of a mobile robot. *IEEE Transactions on Robotics and Automation*, 5(6):804–819, December 1989.

[2] D. L. Boley and K. T. Sutherland. Recursive total least squares: An alternative to the discrete Kalman filter. Technical Report CS TR 93-32, University of Minnesota, April 1993.

[3] N. K. Bose, H. C. Kim, and H. M. Valenzuela. Recursive implementation of total least squares algorithm for image reconstruction from noisy, undersampled multiframes. In *Proceedings of 1993 International Conference on Acoustics, Speech and Signal Processing*, pages V–269–V–272. IEEE, May 1993.

[4] C. E. Davila. Efficient recursive total least squares algorithm for FIR adaptive filtering. *IEEE Trans. Sig. Proc.*, 42(2):268–280, 1994.

[5] R. D. DeGroat. Noniterative subspace tracking. *IEEE Transactions on Signal Processing*, 40(3):571–577, March 1992.

[6] H. Durrant-White, E. Bell, and P. Avery. The design of a radar-based navigation system for large outdoor vehicles. In *Proceedings of 1995 International Conference on Robotics and Automation*, pages 764–769. IEEE, June 1995.

[7] G. H. Golub and C. F. V. Loan. *Matrix Computations*. Johns Hopkins, 2nd edition, 1989.

[8] G. Hager and M. Mintz. Computational methods for task-directed sensor data fusion and sensor planning. *The International Journal of Robotics Research*, 10(4):285–313, August 1991.

[9] S. Haykin. *Adaptive Filter Theory*. Prentice Hall, 2nd edition, 1991.

[10] S. Hosur, A. H. Tewfik, and D. Boley. Multiple subspace ULV algorithm and LMS tracking. In M. Moonen and B. D. Moor, editors, *3rd Int'l Workshop on SVD and Signal Processing*, pages 295–302. Elsevier, August 1995. Leuven, Belgium.

[11] A. Kosaka and A. C. Kak. Fast vision-guided mobile robot navigation using model- based reasoning and prediction of uncertainties. *CVGIP: Image Understanding*, 56(3):271–329, November 1992.

[12] H. Schneiderman and M. Nashman. A discriminating feature tracker for vision-based autonomous driving. *IEEE Transactions on Robotics and Automation*, 10(6):769–775, December 1994.

[13] R. C. Smith and P. Cheeseman. On the representation and estimation of spatial uncertainty. *The International Journal of Robotics Research*, 5(4):56–68, Winter 1986.

[14] H. W. Sorenson. Least-squares estimation: from Gauss to Kalman. *IEEE Spectrum*, pages 63–68, July 1970.

[15] G. W. Stewart. Updating a rank-revealing ULV decomposition. *SIAM J. Matrix Analysis*, 14(2), 1993.

[16] K. T. Sutherland. Ordering landmarks in a view. In *Proceedings 1994 ARPA Image Understanding Workshop*, November 1994.

[17] K. T. Sutherland and W. B. Thompson. Localizing in unstructured environments: Dealing with the errors. *IEEE Transactions on Robotics and Automation*, 10(6):740–754, December 1994.

[18] S. Van Huffel and J. Vandewalle. *The Total Least Squares Problem - Computational Aspects and Analysis*. SIAM, Philadelphia, 1991.

[19] S. Van Huffel and H. Zha. An efficient total least squares algorithm based on a rank-revealing two-sided orthogonal decomposition. *Numerical Algorithms*, 4(1-2):101–133, January 1993.

[20] K.-B. Yu. Recursive updating the eigenvalue decomposition of a covariance matrix. *IEEE Transactions on Signal Processing*, 39(5):1136–1145, May 1991.