INTELLIGENT TUNING OF A KALMAN FILTER USING LOW-COST MEMS INERTIAL SENSORS

C. Goodall, N. El-Sheimy

The University of Calgary, Department of Geomatics Engineering

KEY WORDS: INS, GPS, Kalman filter, tuning, MEMS, reinforcement learning

ABSTRACT:

The demand for civil navigation systems in harsh environments has been growing over the last several years. The Global Positioning System (GPS) has been the backbone of most current navigation systems, but its usefulness in downtown urban environments or under heavily treed terrain is limited due to signal blockages. To help bridge these signal gaps inertial navigation systems (INS) have been suggested. An integrated INS/GPS system can provide a continuous navigation solution regardless of the environment.

For civil applications the use of MEMS sensors are needed due to cost, size and regulatory restrictions of higher grade inertial units. The Kalman Filter has traditionally been used to optimally weight the GPS and INS measurements, but when using MEMS grade sensors the tuned parameters are not always the optimal ones. In these cases, the position errors during loss of the GPS signals accumulate faster than the ideally tuned case. To help correct imperfect tuning, a reinforcement learning algorithm was used to tune the Kalman filter parameters as navigation data was collected.

Tuning any Kalman filter is a difficult task and is often done before navigation with the aid of the filter designer. This process often entails much iteration using the expertise of the designer, and is in no way guaranteed to result in optimal parameters. Reinforcement learning is an intelligent solution to this problem which uses a combination of dynamic programming and trial and error exploration to develop a set of optimal parameters.

In comparison to a typical iterative approach, it was found that using reinforcement learning led to slightly better estimates of the tuning parameter values; furthermore, the tuning process was performed with significantly less iteration, in comparison to an exhaustive search, due to the learning capability of the method. This benefits both static parameters as well as time varying parameters since the method is capable of constantly adapting the tuning based on collected navigation data.

1. INTRODUCTION

Integrated navigation systems are becoming increasingly popular, especially within the civil community. Applications are emerging which require better coverage and availability than is currently possible using a stand-alone satellite navigation system. Vehicle navigation in downtown/urban areas and personal navigation in heavily treed environments are typical examples of these applications.

The combination of GPS with INS is a robust solution for harsh environments where the GPS signals are frequently blocked or lost due to multipath. When available, GPS can be used to provide absolute positions, and when the GPS signals are blocked the INS can be relied upon for the navigation solution since these sensors are fully self-contained and do not rely upon external signals. Furthermore, the absolute positions from GPS can filter and minimize the cumulative drift of the dead reckoning inertial sensors. This creates a continuous navigation solution with the accuracy determined by the quality of both the GPS signals and inertial sensors used.

When integrated together, the largest errors often occur during periods of long GPS signal outages. When used as a primary navigation tool, an INS displays quadratic position error drifts with respect to time due to integration of errors from previous epochs [Barshan et al, 1995]. The magnitude of these drifts depends largely on the error characteristics of the inertial sensors. Navigation grade sensors can display errors of less than a meter over several minutes; while micro-electro mechanical sensors (MEMS) can drift several hundred meters within this same time span [Yudan et al, 2005]. Many civil applications are restricted to the use of MEMS inertial measurement units (IMUs) due to cost, size and regulatory restrictions of higher grade sensors. Furthermore, the use of differential GPS is rarely available for everyday use. Single point mode GPS (SGPS) combined with MEMS grade IMUs is currently the only practical choice for civil applications, and is the focus of this paper.

2. KALMAN FILTERING AND MEMS SENSORS

The Kalman Filter is a useful tool for combining GPS and INS measurements. Several variants of the filter exist, such as the Linearized Kalman Filter (LKF), the Extended Kalman Filter (EKF) and the Unscented Kalman Filter (UKF). The EKF was used for the analysis in this paper due to its popularity among

many designers of integrated navigation systems, but as will be shown, the general approach and conclusions can be made for any filter that requires a priori tuning parameters, regardless of what these parameters represent.

The EKF combines measurements from SGPS with those of a MEMS IMU using certain a priori information The filter requires knowledge of the system and measurement dynamics as well as a statistical description of the system noises, measurement errors and uncertainty in the dynamic models [Weston and Titterton, 1997]. This includes the noise characteristics of both the MEMS and SGPS updates. The filter then takes several assumptions, such as white noise behaviour and Gauss-Markov properties, to weight the measurements in an optimal manner.

The EKF used for this IMU/GPS integration contained 21 error states: 3 states each for the positions, the velocities, the attitudes, the accelerometer biases, the accelerometer scale factors, the gyro biases and the gyro scale factors. The biases of both the accelerometers and gyroscopes were fed back and compensated before each GPS epoch.

3. RESIDUAL ERRORS DUE TO POOR TUNING AND MEMS VARIABILITY

If the EKF estimation were perfect then the position errors would roughly follow a quadratic drift with time due to integration of time correlated stochastic sensor errors at each epoch. Since it is impossible to predict random errors at an individual epoch, this would be considered the ideal state when navigating with inertial sensors. In practical applications, several factors prevent this optimal situation when using a Kalman Filter.

Since the EKF requires a priori knowledge, in the form of statistical tuning parameters, its performance can vary. For example, poor initial estimates of the MEMS noise levels can greatly affect the drift rate experienced during GPS signal outages due to accumulation of errors from the innovation sequence. Proper tuning of the filter is best analyzed during periods of GPS signal outages [Graefe et al]. During these times, the positional errors accumulate due to the integrated inertial errors. Therefore, if not properly tuned, the filter position errors can grow more rapidly with time.

4. TYPICAL TUNING APPROACH

The actual tuning process for a Kalman filter can be very iterative and time consuming. It usually starts from a static estimate of the errors achieved through an Allan Variance analysis [Shin, 2005]. This initial estimate is typically optimistic, since many hours or even days of static data needs to be collected. It is often used as a starting point for further tuning, which is done by taking discrete steps in the tuning parameters and evaluating the position drift during GPS outages.

To determine the order of parameter tuning a priority setting can be made by evaluating the drift caused by various parameters during GPS outages. The gyro bias is usually tuned first since it contributes the largest position error. An uncompensated gyro bias in the horizontal direction introduces an angular error proportional to time, as in (1).

$$\delta\theta = \int b_g dt = b_g t \tag{1}$$

This angle error causes misalignment of the IMU and projects the acceleration vector in the wrong direction, which causes an acceleration error proportional to time. This in turn creates a positional error proportional to time cubed after double integration, as in (2).

$$\delta p = \int v dt = \int \frac{1}{2} b_g g t^2 dt = \frac{1}{6} b_g t^3 \tag{2}$$

A similar analysis can be done for uncompensated accelerometer biases to reveal that they produce errors in position proportional to time squared, as in (3). Therefore, the gyro bias parameters are often tuned first, with others, such as the accelerometer biases, angular random walks and velocity random walks, tuned after.

$$\delta p = \int v dt = \int b_a t dt = \frac{1}{2} b_a t^2 \tag{3}$$

Statistically, these parameters have to be tested on a significant number of GPS outages covering a variety of dynamics, which is hard to do without extensive field test data. Also, the size of the discrete steps taken to tune the parameters affects the optimality of the final parameters. Therefore, the optimal parameters are rarely known in reality; the parameters chosen, using this method, are often close to optimal but not necessarily ideal.

The use of MEMS IMUs further complicates the problem of accurate tuning [Basil et al, 2004]. Even similar MEMS devices can display different error characteristics and since tuning individual filters is expensive in comparison to the cost of the sensors, general tuning parameters are used for similar units. An example of different noise characteristics for three similar MEMS sensors is shown in Fig. 1. The Allan variance analysis for three Panasonic EWTS82 MEMS gyroscopes is given. The trend for the three gyros is similar but not exactly the same. Gyro 1 is quite different from gyros 2 and 3, which would result in different noise characteristics for tuning. The Allan standard deviations corresponding to cluster times of one second denote the sensors white noise (angular random walks), and the points of zero slopes indicate the sensor bias instabilities. These noise values are different for the three EWTS82 MEMS gyros and common values would not optimally represent individual noise characteristics.

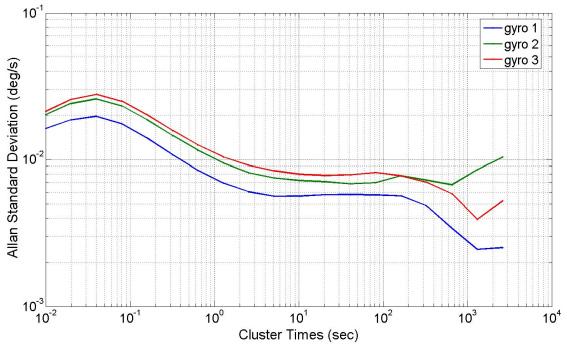

Figure 1. Allan standard deviation versus cluster times for three Panasonic EWTS82 MEMS gyros

There are also other residual errors which can be different for each unit. Axes misalignment involves non-orthogonality in the body frame when the sensors are mounted in manufacturing. This can be within the individual accelerometer or gyroscope triads (non-orthogonality), and can also appear in the relative orientations of the two triads with respect to one another. The misalignment between axes within an individual triad can be calibrated using special techniques, but for MEMS sensors this is usually a time consuming and expensive process that is not performed. Temperature also affects the tuning parameters, especially the scale factors and bias estimates. These errors are often left as residual errors for the EKF to estimate. Unfortunately, estimates are prone to change with dynamics and temperature, so a stable estimate has to be taken using a long correlation time and cannot reflect quickly changing errors [Shin, 2005].

5. REINFORCEMENT LEARNING

5.1 The Problem

Reinforcement learning (RL) involves learning what to do in certain situations, i.e. mapping correct actions to situations. An analogy to RL can be drawn when analyzing how babies learn to walk; through a combination of watching someone else walk and through their own trial and error interaction with their environment. They receive reinforcement through a number of signals such as falling (failure) or not falling (success). If successful, this reinforces to them that their previous actions were correct for walking.

RL has typically been used for intelligent game theory, such as backgammon [Tesauro, 1995], but lately it has begun to emerge in fields such as system control, dynamic prediction and even quality control. Its use in Kalman filter tuning is beneficial in helping the system learn how to properly tune the filter. Even with the traditional approach there are many assumptions that are taken such as the order of tuning and the size of discrete steps taken. Furthermore, the curse of dimensionality prevents optimal use of an exhaustive search method [Bellman, 1957].

As an example, consider the case of tuning 4 parameters, each having 20 discrete steps. The number of iterations to fully explore all combinations would be 204 or 160,000 iterations. If we wanted to generate simulated GPS outages using a forward KF on real navigation data that took 1 minute to process then this iterative tuning would take over 44 days to complete. Of course, in real applications this tuning is significantly reduced due to the intelligent input by the designer. In the case of the RL, it is this intelligent tuning that is trying to be replicated in an automatic and more optimal fashion.

5.2 RL Nomenclature

In any RL problem there is always an interaction between an agent and the environment, similar to a baby and its physical surroundings. Within these two, the RL problem contains a policy, a reward function, a value function and sometimes a model (full/partial) of the environment.

A policy explains how the agent behaves in a given scenario, or in other words, what actions should be taken in a given state. Policies are the most important element of the problem, and once learned they are all that is needed to determine appropriate behavior. In the case of tuning an EKF, the policy would roughly explain how to change the tuning parameters in order to achieve better estimates of the true value during the tuning process.

A reward function associates individual state/action pairs of the environment to a number that describes the positive or negative reward. The reward function is typically used to alter the optimal policy depending on what is perceived to be good or bad by the agent. For EKF tuning, if we are maximizing the reward function, then an immediate reward could be the negative average of several simulated GPS outage drifts over a fixed outage length.

A value function defines what is good for long-term rewards. Typically, an RL problem attempts to maximize the value function over individual rewards so that the long-term success of the policy is satisfied. When tuning the EKF, some rewards when changing an individual tuning parameter might make the GPS outage errors larger, but once other parameters are tuned the end results might actually be better. In this case the long-term rewards are most important once future states are taken into account along with the rewards of those states.

The discounted returns are often used to relate newer and older rewards. Equation (4) gives the discounted reward Rt using all future rewards discounted by γ, which is always between 0 and 1.

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots + \gamma^n r_{t+n+1} \quad (4)$$

Finally, some RL systems use models of the environment. These models can be provided ahead of time, if sufficient information is available, or they can be learned through trial and error, and then incorporated into the policy. When tuning a KF the model is first developed through trial and error learning, and then applied to help guide future tuning decisions.

In general, the interaction between the agent and environment can be visualized as in Fig. 2 [Sutton, 1998]. The agent acts on the environment, which in turn provides a reward back to the agent based on the current state. This continues for the next state/action pair in a recursive manner.

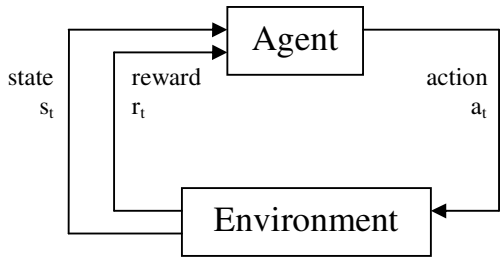

Figure 2: Agent-environment interaction [Sutton, 1998]

5.3 The RL Approach

Since the RL algorithm estimates value functions and attempts to maximize these over future iterations, we define the value of taking action a in state s under policy p as:

$$Q^p(s,a) = E_p\{R_t \mid s_t = s, a_t = a\} \tag{5}$$

And from (4) we can insert the discounted reward as:

$$Q^p(s,a) = E_p\{\sum_{n=0}^{\infty} \gamma^n r_{t+n+1} \mid s_t = s, a_t = a\} \tag{6}$$

This last equation says that as the number of states encountered reaches infinity, the average reward will approach the actual state value function. This form of learning is a type of Monte Carlo method. The most commonly used RL algorithms use a combination of modeling (dynamic programming) and trial and error learning (Monte Carlo methods).

In terms of dynamic programming, RL learning follows particular recursive relationships, such as the Bellman optimality equation for Q*(s,a) shown here, starting from (6):

$$\begin{aligned} Q^p(s,a) &= E_p\{r_{t+1} + \gamma \sum_{n=0}^{\infty} \gamma^n r_{t+n+2} \mid s_t = s, a_t = a\} \\ Q^*(s,a) &= E\{r_{t+1} + \gamma \max_{a'}[Q^*(s_{t+1}, a')] \mid s_t = s, a_t = a\} \\ Q^*(s,a) &= \sum_{s'} P^a_{ss'}[R^a_{ss'} + \gamma \max_{a'}[Q^*(s_{t+1}, a')]] \end{aligned} \tag{7}$$

The * denotes the optimal estimate, while P represents the probability of the next state s' from the current state s and action a, and finally, R gives the expected value of the reward.

The importance of (7) is that if we have the optimal Q then the agent does not even have to look one step ahead, all it has to do is pick the action that maximizes the state/action pair of Q. Unfortunately, explicitly solving the Bellman equation is similar to performing an exhaustive search. Thus, RL theory attempts to approximate solutions to this optimality equation by 'experiencing' the transitions instead of actually knowing them ahead of time.

To obtain an updated policy iteratively, generalized policy iteration (GPI) is used. Iteration is needed since there are two simultaneous processes: one makes the value function consistent with the current policy, and the other makes the policy greedy with respect to the current value function [Sutton, 1998]. The term greedy means that it usually chooses the policy with the maximum value function (if the value function is being maximized). There is always some chance that the maximum value function policy is not chosen, and this enables further trial and error learning known as exploration.

This leads to the core RL algorithm used in this paper: temporal difference (TD) learning. TD learning uses a combination of DP and Monte Carlo methods. TD can learn directly from experience, similar to Monte Carlo, and TD can also provide estimates based on previously learned estimates, similar to DP.

For any RL algorithm, incremental learning is the key to updating a policy, as is the case for TD learning. The basic formula for this update is:

$$New = Old + Step[T\arg et - Old] \tag{8}$$

The bracketed expression is an error in the old estimate, and is reduced by taking a step towards the Target value. When learning Q values, the update can be expressed as:

$$Q(s_t,a_t) = Q(s_t,a_t) + \alpha[r_{t+1} + \gamma Q(s_{t+1},a_{t+1}) - Q(s_t,a_t)] \tag{9}$$

In (9), the step size is α and the target is expanded into the first two terms within the brackets, taken from (7). This method of updating the Q values is commonly referred to as SARSA since it uses the following information: st, at, rt+1, st+1, at+1. The pseudocode for the SARSA updating algorithm, combined with a greedy choice of actions is given in Fig 3.

```
Initialize Q(s,a)
For each episode:
        Initialize state s
    Choose a from s using policy from Q
    (greedy)
    For each step of episode
        Take action a
        Experience r and state s'
            Choose a' from s' using policy
            from Q(greedy)
            Update Q using (9)
            Set s=s', a=a'
```

Figure 3: SARSA pseudocode

An episode is defined as the end of the game or process, but is not critical to the implementation of SARSA. For instance, in

many games such as checkers there exist distinct episodes, such as the end of a game, and new games are played repeatedly. But for some applications there may be only a single episode, known as a continuous episode, in which case the steps of the individual episode direct the learning. For the example of checkers a step would be a single move within the game.

The advantage of this TD learning method over Monte Carlo and DP methods are numerous from an applications perspective. TD methods learn estimates based on previous estimates, but unlike Monte Carlo methods they do not need to wait until the end of an episode, they can update after each step. In comparison to DP methods they do not require a model of the environment, but instead they can build this model through interactions. These two advantages are the main reasons why RL algorithms are better suited to on-line applications, such as tuning a KF incrementally as navigation data is collected.

6. RESULTS USING A TRADITIONAL APPROACH

As a comparative baseline, a traditional tuning approach was performed. Five EKF parameters were tuned in order as follows:

1. GPS position update scaling
2. Gyro bias standard deviation
3. Accelerometer bias standard deviation
4. Gyro angular random walk (arw)
5. Accelerometer velocity random walk (vrw)

The exact values of these parameters are unknown in reality since the tested integration system consisted of SGPS from a NovAtel OEM4 receiver and inertial measurements from MEMS sensors (Analog Devices Inc.). Rough estimates of the true values were obtained through static tests, and appropriate ranges around these values were set as shown in Table 1.

Parameter (units)	Min	Max	Discrete step
GPS pos scale (unitless)	0.05	2.05	0.5
Gyro bias std (deg/h)	200	1000	200
Accel bias std(mGal)	2000	10000	2000
arw (deg/sqrt(h))	0.5	2	0.5
vrw (m/s/sqrt(h))	0.5	5.5	1

Table 1: Tuning parameter ranges

Due to the curse of dimensionality not all combinations of these tuning parameters could be tested. Instead, the parameters were all fixed to default values (lab values) then were altered in order. Once a parameter had been tuned, its future values were fixed to that value. This approach resulted in 25 tuning iterations.

This approach is clearly far from optimal since not all cases can be considered, but this is often done in practice. It is imperative that the correct tuning order be used, and this often depends on the various errors within the system. But even if an appropriate order is used, this does not guarantee that future tuning parameter state changes will not alter current optimality of states.

The tuning results led to the parameters in Table 2. They were evaluated by calculating the minimum average positional drift during eight 60 second simulated GPS outages from the L trajectory shown in Fig 4. The forward EKF was run each time with different a priori parameters, and the case with the lowest errors was kept as the optimal set of parameters. The final averaged GPS positional error drifts was found to be 135 meters which is reasonable for this grade of MEMS IMU combined with SGPS.

Parameter (units)	Tuned value
GPS pos scale (unitless)	0.05
Gyro bias std (deg/h)	200
Accel bias std(mGal)	2000
arw (deg/sqrt(h))	2
vrw (m/s/sqrt(h))	5.5

Table 2: Traditional tuning method results

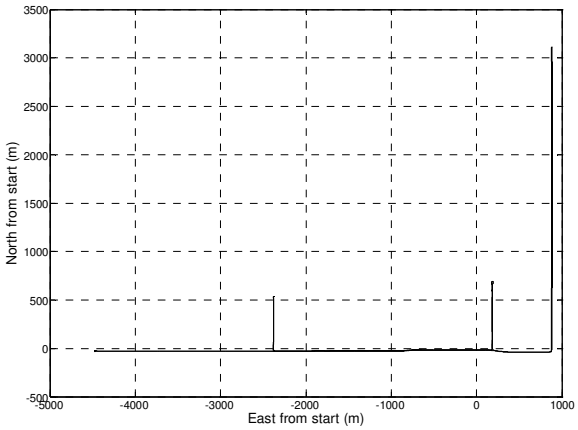

Figure 4: Tuning trajectory

7. RESULTS USING RL TUNING

7.1 EKF Tuning using SARSA

The SARSA iterative method was also applied to the same tuning problem. An episode was defined as terminated if the average of the GPS position drifts became less than 100 meters or if more than 20 tuning steps were performed.

A primary difference between the RL method of tuning using SARSA and the traditional method is that SARSA can adjust tuning parameters in parallel. If using an exhaustive search this would result in a total of 3125 states for the ranges and discrete steps given in Table 1. The goal of the RL tuning was to come up with well tuned parameters in significantly less steps than an

exhaustive approach and a better estimate than the traditional approach.

The actions at each state were delta changes to the current state tuning values. These deltas could be +1, -1, or 0 multipliers of the discrete steps. The original states (tuning parameters) were set randomly. The rewards at each step were set as the negative value of the averaged position drift in meters, so as to maximize total reward.

7.2 SARSA Results

The tuning iterations were stopped after 20 episodes, or 400 parallel tuning alterations. Fig. 5 shows the last 20 steps that occurred in episode 20. This demonstrates the convergence of the tuning to a value close to 126 meters for these specific GPS outage drifts.

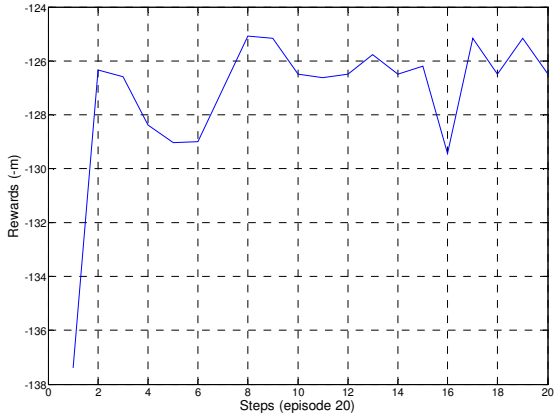

Figure 5: Convergence of tuning in episode 20

The resulting tuning parameters for this optimal set are given in Table 3. It can be noticed that the first two parameters are similar to the traditional case, but the remaining three are significantly different. This is likely a result of weighting the GPS measurements less strongly through the larger value of the GPS position scale (the EKF R matrix). This in turn would create a larger dependence on the remaining parameters governing the Q matrix, which was not the case using the traditional tuning method. Unfortunately, without knowing the true parameter values the only strong indicator of performance is the averaged position drift of many simulated GPS outages.

Parameter (units)	Tuned value
GPS pos scale (unitless)	0.1
Gyro bias std (deg/h)	200
Accel bias std(mGal)	8000
arw (deg/sqrt(h))	0.5
vrw (m/s/sqrt(h))	0.5

Table 3: RL SARSA tuning results

Other interesting values to take note of are the EKF error estimates, which are given in the P matrix. These describe the predicted level of errors during the GPS outages, and if the filter is consistent they should be close to the simulated average. In the case of the traditional tuning the P matrix positional standard deviations were over 250 meters, while the RL SARSA parameters resulted in P matrix values of 105 meters. Clearly, the RL developed parameters resulted in a more consistent filter and this is also likely due to the weighting between the R and Q matrices through parallel tuning.

8. ADDITIONAL CONSIDERATIONS

This preliminary analysis considered only 5 tuning parameters for an EKF. The full set of tuning parameters for a standard EKF also involves a GPS velocity scale factor, gyro and accelerometer scale factor standard deviations, as well as time correlations for the biases and scale factors. These parameters are typically considered as lesser parameters, but should be considered in future parallel tuning to fully assess the validity of this claim.

For other filters, such as the UKF, additional tuning parameters are also present such as the sigma points. It is not the intent of this paper to tune individual filters, but to show that regardless of the parameters or filter type, the a priori information can be developed in an on-line method without human intervention/tuning.

Finally, for higher grade IMUs that contain strict manufacturer specifications, the tuned values should be within a certain threshold of these values. The initial purpose of this method was for tuning MEMS integrated systems, but if using higher grade units this consideration should be built into the algorithm.

9. CONCLUSIONS

A new technique for tuning a Kalman filter was presented using reinforcement learning. This method can be applied on-line as navigation data is collected to further update the a priori parameters needed for the filter.

In terms of performance two distinct indicators were used: time to tune the filter and accuracy of the tuning. In comparison to an exhaustive search the tuning took about 400 iterations to converge, which was significantly less than performing an exhaustive search of 3125 iterations. For accuracy, the method was compared to a traditional tuning approach. In this case the tuning converged to 126 meters with the RL method and 135 meters with a traditional method.

10. ACKNOLEDGEMENTS

This work was supported in part by NSERC and the GEOIDE NCE. Eun-Hwan Shin is also acknowledged as a co-author of the Kalman Filter toolbox used to generate the Kalman Filter results.

11. REFERENCES

Basil, H., Anathasayanam, M. and Puri, S. (2004): Adaptive Kalman Filter Tuning in Integration of Low-Cost MEMS-INS/GPS, AIAA Guidance, Navigation and Control Conference, Providence, RI, Aug. 16-19.

Bellman, R.E. (1957): Dynamic Programming. Princeton University Press, Princeton.

A. Cichocki, A., Unbehauen, R. (1996): Neural Networks for Optimization and Signal Processing, Wiley, New York.

Graefe, G., Caspary, W., Heister, H., Klemm, J. and Sever, M. The road data acquisition system MoSES, Applanix POSLV product article, available: http://www.applanix.com/products/poslv_articles.php

Hou, H. Kalman filter fine tuning report, MMSS internal document, unpublished.

Shin, E-H. (2005): Estimation Techniques for Low-Cost Inertial Navigation, PhD Thesis, University of Calgary Report 20219, May 2005. Available: http://www.geomatics.ucalgary.ca/research/publications/GradTheses.html

Sutton, R .S and Barto, A. G. (1998): Reinforcement Learning An Introduction. Cambridge, Massachusetts, The MIT Press.

Tesauro, G.J. (1995): Temporal difference learning and TD-Gammon. Communications of the ACM, 38:58-68.

Watkins, C.J.C.H. (1989): Learning from Delayed Rewards. Ph.D. thesis, Cambridge University.

Weston, J.L., Titterton, D.H. (1997): Strapdown Inertial Navigation Technology, London: Peter Peregrinus Ltd.

Yudan, Y., Grejner-Brzezinska, D.A. and Toth, C.K. (2005): Performance Analysis of a Low Cost MEMS IMU and GPS Integration, Institute of Navigation annual meeting, June 27-29, Cambridge, MA.