Tuning a GPS/IMU Kalman Filter for a Robot Driver

Jamie Bell, Karl A. Stol
Department of Mechanical Engineering
The University of Auckland
Private Bag 92019
Auckland 1142
jbel060@ec.auckland.ac.nz

Abstract

This research tested different ways of tuning the process noise covariance matrix of a GPS/IMU extended Kalman filter. This Kalman filter was developed for a retrofit robot driver. Because of the constraints of this application, the Kalman filter had no process model for the outputs of the robot's controller. In this situation, a significant proportion of the process noise would be caused by the robot's planned manoeuvres. Since the robot driver controls these manoeuvres in a somewhat predictable manner, the optimisation of the process noise covariances was carried out offline. Evaluating the performance of a given process noise covariance matrix was achieved by comparing the true values of states to the state estimates gained while running the Kalman filter with simulated GPS and IMU data. The process noise covariance matrix was optimised using a genetic algorithm. Three hypotheses were tested using this optimisation procedure and all were found to be true. The hypotheses were that multiobjective optimisation should be used, that the vehicle manoeuvres were the key source of process noise for the robot driver and that different manoeuvres require different process noise covariance matrices. The proposed tuning procedure for a robot driver's GPS/IMU Kalman filter was successfully field tested.

1 Introduction

Kalman filters are employed extensively for sensor fusion. These Kalman filters produce estimates of the states of a system by combining the data from a variety of different sensors. The resulting state estimates may be more accurate than those that would be produced without sensor fusion.

There is well established theory for developing Kalman filters for the integration of data from Global Positioning System (GPS) and Inertial Measurement Unit (IMU) sensors [Farrell and Barth, 1998; Grewal et al., 2001]. Nevertheless, one of the key challenges of producing a reliable Kalman filter is adequately tuning its parameters [Powell, 2002]. When educated guesses are used, trial and error can result in satisfactory Kalman filter performance in some applications; however, this approach can be time-consuming and unreliable [Bolognani et al., 2003].

More conventional techniques that produce dependable parameters include adaptive Kalman filtering and offline stochastic metaheuristic searches. Adaptive Kalman filters tune parameters online. Using this method, parameters are altered according to how accurately the internal equations of the filter can predict the measurements of the system [Loebis et al., 2004]. Fuzzy adaptive Kalman filters have been found to perform well in experiments [Hu et al., 2003; Xiong et al., 2005].

Offline optimisation of the parameters may be carried out using metaheuristic techniques. For example, the genetic algorithm may be used to optimise parameters by searching the feasible solution space for a combination of values that gives the best performance of the Kalman filter [Chan et al., 2001; Gueye et al., 2005]. Metaheuristic methods are the subject of this paper because they have been found to suit the unique application for which the Kalman filter is being created.

The application is producing localisation and heading data for a robot driver. What is unique about this application is that the robot driver will only perform a limited number of set manoeuvres. Hence, given some sensor data for the robot driver performing these manoeuvres, the GPS/IMU Kalman filter can be tuned offline using metaheuristics.

2 The Extended Kalman Filter

The extended Kalman filter, which is used in this research, consists of five recursive equations [Welch and Bishop, 2004]. The first of these is given here:

$$\mathbf{x}_k^- = \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}), \tag{1}$$

where $\mathbf{x}_k^-$ holds the predictions of the system states,
$\mathbf{x}_{k-1}$ holds state estimates from the last time step,
$\mathbf{u}_{k-1}$ holds the inputs to the system and
$\mathbf{f}(\mathbf{x}_{k-1}, \mathbf{u}_{k-1})$ is a nonlinear function.

This first equation is a process model for the system. It gives a prediction for the states at the current time step based on the previous estimate of the states and the inputs at the last time step. The Jacobian of this nonlinear function, with respect to the states is given by:

$$\mathbf{A}_{[i,j]} = \frac{\partial \mathbf{f}_{[i]}}{\partial \mathbf{x}_{[j]}}(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}). \tag{2}$$

The second equation of the Kalman filter is then:

$$\mathbf{P}_k^- = \mathbf{A}_k \mathbf{P}_{k-1} \mathbf{A}_k^T + \mathbf{W}_k \mathbf{Q}_{k-1} \mathbf{W}_k^T \, , \tag{3}$$

where $\mathbf{P}_k^-$ is the error covariance matrix for $\mathbf{x}_k^-$,
$\mathbf{P}_{k-1}$ is the error covariance for $\mathbf{x}_{k-1}$,
$\mathbf{W}_k$ is the vector of system disturbance,
$\mathbf{Q}_{k-1}$ is the process noise covariance matrix.

The next three equations of the Kalman filter correct the predictions made by the first two equations. The third equation calculates the Kalman filter gain:

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}_k^T \left(\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{V}_k \mathbf{R}_k \mathbf{V}_k^T \right)^{-1} , \tag{4}$$

where $\mathbf{K}_k$ is the Kalman gain,
$\mathbf{H}_k$ is the linearised measurement model,
$\mathbf{V}_k$ is the vector of measurement disturbance,
$\mathbf{R}_k$ is the measurement noise covariance matrix.

The fourth equation of the Kalman filter calculates the state estimates. These estimates are the key outputs of the Kalman filter. The equation is:

$$\mathbf{x}_k = \mathbf{x}_k^- + \mathbf{K}_k \left(\mathbf{z}_k - \mathbf{h}\left(\mathbf{x}_k^- \right) \right), \tag{5}$$

where $\mathbf{z}_k$ is the vector of measurements,
$\mathbf{h}\left(\mathbf{x}_k^-\right)$ is the measurement model for the system.

Finally, the corrected error covariance matrix is found by:

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_k^- \, . \tag{6}$$

Once initial estimates for the corrected state estimates and error covariance have been established, the Kalman filter equations form a continuous loop. The three corrector equations follow the two predictor equations, with state estimates being output every time step.

3 The Robot Driver Application

In this research an extended Kalman filter has been developed to combine data from GPS and IMU sensors. This Kalman filter is a part of the navigation module of a retrofit robot driver, which has been developed for the DARPA Grand Challenge race by the Grand Challenge NZ team. This robot is strapped into the driver's seat of a vehicle. It has actuators for a steering wheel, accelerator pedal, brake pedal and an automatic gear shift. The robot driver is designed to be transferable from one vehicle to another.

There are some unique features about driving with this robot that influence the creation of the Kalman filter. Firstly, there is no reliable model for how the outputs of the robot's high level control affect the states of the system. Ideally, such a model would be included in the Kalman filter, with the outputs of the robot's controller being fed into the first equation of the filter (equation 1) as the inputs. However, since there is no set vehicle for which the robot has been designed and because the actuators on the robot are continuously evolving, the machinery is always changing. Hence, the relationship between the controller and the system states is unpredictable. The absence of such a relationship affects the Kalman filter because it introduces significant process noise. This process noise would have otherwise been accounted for as inputs to the system in the first equation of the Kalman filter.

Another unique feature of the retrofit robot driver is that some additional information about the process noise of the system is known. Because there is no relationship between the controller outputs and the system states, the process noise is to some degree dependant on the dynamic behaviour of the vehicle. Because the dynamic behaviour of the vehicle is planned by the higher level control of the robot driver, the process noise is somewhat predictable and can be anticipated. There are two important results of such predictability: the outputs of the higher level control may be used to influence the process noise parameters in the Kalman filter online and offline optimisation of these same parameters can be employed.

The final key unique feature of the robot driver application is the controllability of the driving. The robot driver has two key types of paths that it has to follow: bends and straights. Each time the robot follows one of these types of paths, the behaviour of the robot is not accurately repeatable but is, nevertheless, to some extent predictable. The fact that the robot only follows two types of paths means that the components of the process noise that are created by these manoeuvres can be anticipated. Limiting the number of types of manoeuvres also decreases the variation in process noise.

When the robot drives along a straight, it controls the vehicle to maintain a set constant speed and a set angle. Therefore there is very little process noise in terms of planned manoeuvres when the vehicle travels along a straight path. In contrast, when the robot drives around a corner, it controls the vehicle to decelerate at a set rate, to maintain a constant speed through a constant radius turning circle and to turn through 90 degrees before accelerating again at a set rate. Therefore, there is considerable process noise due to the planned manoeuvres when the robot driver turns a corner. However, this noise is quite predictable because the manoeuvre is planned by the robot.

An adaptive Kalman filter in this context would make use of historic data, which would be measured online. Offline optimisation of the process noise covariance matrix is appropriate in this application because some prior knowledge about the process noise is known. It may also be possible to use different process noise covariance matrices for different planned manoeuvres.

4 Research Hypotheses

This research investigates three hypotheses. Firstly, that multiobjective optimisation is needed for offline optimisation of Kalman filter parameters. Secondly, that the planned paths of the vehicle are the dominant sources of process noise and hence using these is sufficient for tuning the Kalman filter. Thirdly, that different process noise covariances are beneficial for the different planned manoeuvres of the robot driver. These hypotheses are tested individually in simulation and the resulting principles are demonstrated in field tests.

5 Research Tools

In this research a genetic algorithm was used to find an optimal process noise covariance matrix for a GPS/IMU Kalman filter. To do this, simple paths, made up of the robot driver's two manoeuvres, were created. Then, depending on the experiment, these paths may have been followed by a vehicle model, which was simulated to behave as though the robot driver were in control. Simulated sensor measurements for the paths travelled were created. These measurements were repeatedly run in a Kalman filter, while the genetic algorithm altered the process noise covariance matrix from repetition to repetition. By comparing the states output by the Kalman filter with the true values of the states, the performance of each process noise covariance matrix in the Kalman filter was evaluated. Using these evaluations, the genetic algorithm directed the search for the optimal set of parameters. The way in which these research tools fitted together in the optimisation process is shown in Figure 1.

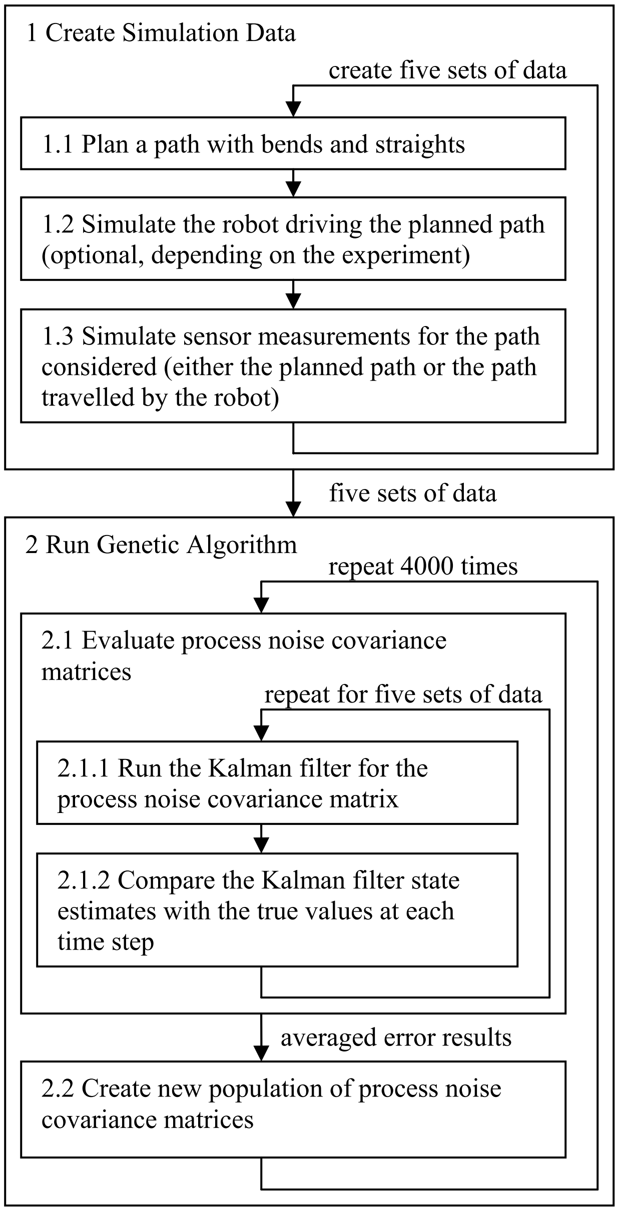

1 Create Simulation Data
create five sets of data
1.1 Plan a path with bends and straights
1.2 Simulate the robot driving the planned path (optional, depending on the experiment)
1.3 Simulate sensor measurements for the path considered (either the planned path or the path travelled by the robot)
five sets of data
2 Run Genetic Algorithm
repeat 4000 times
2.1 Evaluate process noise covariance matrices
repeat for five sets of data
2.1.1 Run the Kalman filter for the process noise covariance matrix
2.1.2 Compare the Kalman filter state estimates with the true values at each time step
averaged error results
2.2 Create new population of process noise covariance matrices

Figure 1: Research tools in the optimisation process.

There were five key tools used in this research: a simple vehicle path planner, a robot driver simulator, a simulated sensor data generator, an extended Kalman filter and a genetic algorithm. The three simulators were used to create data for the Kalman filter. The genetic algorithm was used to find optimal parameters for the Kalman filter. Then a copy of the Kalman filter and sets of separately created test data were used to finally trial the optimised parameter values. A diagram of how the research tools were used in the testing stage is given in Figure 2. This procedure was very similar to the optimisation process, though without the genetic algorithm. Note that all testing data was kept the same throughout all of the experiments. All five tools used in this research are described in more detail in the subsections that follow.

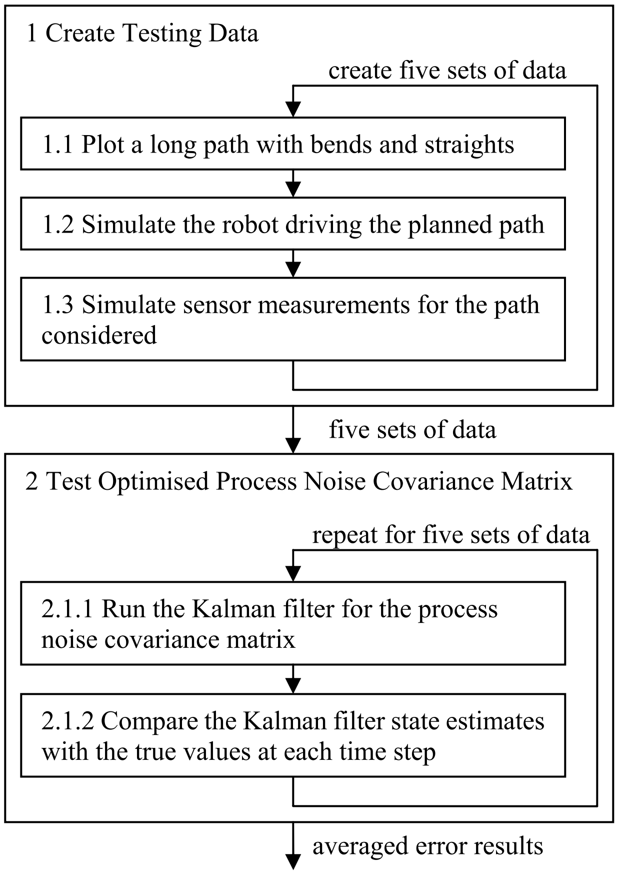

1 Create Testing Data
create five sets of data
1.1 Plot a long path with bends and straights
1.2 Simulate the robot driving the planned path
1.3 Simulate sensor measurements for the path considered
five sets of data
2 Test Optimised Process Noise Covariance Matrix
repeat for five sets of data
2.1.1 Run the Kalman filter for the process noise covariance matrix
2.1.2 Compare the Kalman filter state estimates with the true values at each time step
averaged error results

Figure 2: Research tools in the testing stage.

5.1 The Path Planner

The vehicle path planner program plotted points for ideal movements of a vehicle. Three planned paths were defined at a sampling frequency of 1 Hz (Figure 3). Two shorter paths and one longer path were created. The paths were defined not just in terms of the positions in a Northing Easting coordinate system; the accelerations in the body coordinate system, the angular velocity at every time step and the heading of the vehicle were also calculated. The paths generated began at a constant speed in a straight line, which was followed by a period of constant deceleration, then turning around a corner at a constant radius and speed, which was followed by a period of constant acceleration. The paths produced by the path planner were used to create data for the optimisation and testing stages of the experiments in this research.

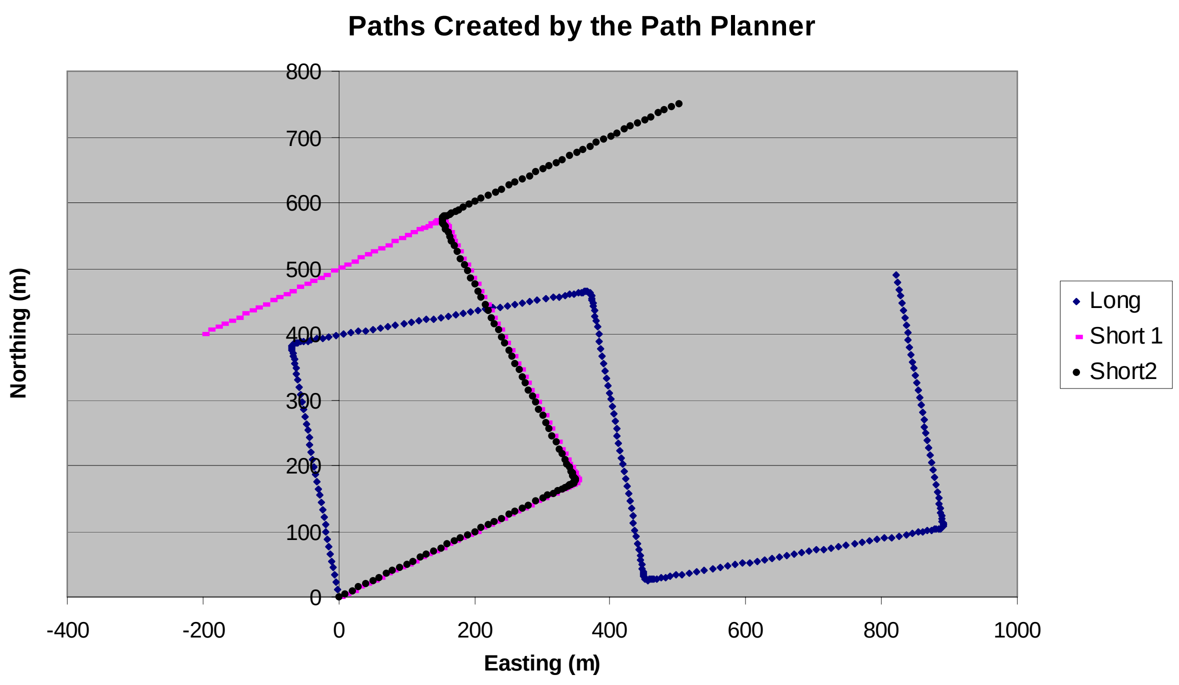


Figure 3: The three paths generated by the path planner.

The paths were defined so as to mimic real paths that the robot might have to traverse. It is thought that, since all possible paths could be constructed from paths "Short 1" and "Short 2" or their mirror images, these should be representative of the robot driver paths.

The values of the speeds, accelerations and the turning circle radius were chosen to be realistic. For the paths plotted by the path planner, the speed along the straights was made to be 40 km/h. The rate of deceleration into a corner was taken to be 1 ms^{-2} and the same rate of acceleration was used out of a corner. The turning circle at the corners had a radius of 6.4 m. The speed around the corners was set at 7.2 km/h.

The two shorter paths were only used to create data for optimisation. The test data and the optimisation data for experiment 2 was created from the longer path.

5.2 The Robot Driver Simulator

The robot driver simulator generated an output of the states of the system as though the robot was driving a vehicle. The inputs to the robot driver simulator came from the path planner. The simulator produced state outputs that would be expected if the robot were following the path from the path planner. The states output were position, velocity, accelerations, heading and angular velocity.

The purpose of the robot driver simulator is to generate realistic driving data for each planned path. The components of the simulator created in MATLAB/ Simulink are: a vehicle dynamics model, a trajectory tracking controller, and first-order actuator dynamics models. The vehicle is modelled as a planar rigid body with non-holonomic kinematic constraints imposed by standard Ackermann steering. An engine torque map, aerodynamic drag and rolling resistance are included. The trajectory tracking controller uses PID control for speed tracking and pure pursuit for path tracking [Shin et al., 1992].

5.3 The Sensor Simulator

The sensor simulation program created realistic GPS and IMU data from the outputs of either the path planner or the robot driver simulator, depending on the experiment. These outputs had to include positions, accelerations and angular velocities. GPS data (*GPS*) was made by adding noise (*n*) to vehicle positions (*p*) in a Northing (denoted by subscript *N*) and Easting (denoted by subscript *E*) coordinate system:

$$GPS_N = p_N + n_N \tag{7}$$
$$GPS_E = p_E + n_E \tag{8}$$

The noise was modelled based on GPS measurement noise, collected in field tests. The IMU data (*IMU*) was generated by first adding noise and a linearly time varying bias to vehicle accelerations (*a*) and angular velocities (*ω*). The result was then multiplied by a linearly time varying scaling factor. The noise (*n*), biases (*b*) and scaling factors (*f*) were modelled based on lab tests of the sensors. The equations for the IMU measurements in the roll (*r*), pitch (*p*) and yaw (*y*) directions were:

$$IMU_r = f_r(a_r + n_r + b_r) \tag{9}$$
$$IMU_p = f_p(a_p + n_p + b_p) \tag{10}$$
$$IMU_Y = f_Y(\omega + n_y + b_y) \tag{11}$$

The output of the sensor simulator was sets of optimisation and testing data. In addition to the sensor data described here, the true values of states were also kept at this stage, so that they could be compared with the state estimates of the Kalman filter. Within a set of optimisation paths, the noise created for each path was different. However, the same optimisation paths were recycled from experiment to experiment to keep the results comparable. Only one set of test paths was created and the noise in this did not change.

5.4 The Kalman Filter

The extended Kalman filter was designed for a vehicle travelling in two dimensions. The sensor measurements taken as inputs were GPS positions, in terms of Northings and Eastings relative to the starting point, and IMU measurements. The IMU sensor included an accelerometer (which was aligned with the roll direction of the vehicle), another accelerometer (which was aligned with the pitch direction of the vehicle) and a gyroscope, measuring the yaw rate. The key states of the vehicle in the Kalman filter were the positions, velocities and accelerations in the North and East coordinate system, the heading of the vehicle and the angular velocity. The other states were the biases and scaling factors in the IMU measurements.

5.4.1 Coordinate System Definition

The definition of the coordinate systems for the Kalman filter and the data going into the Kalman filter is given in Figure 4. The dynamics of the vehicle were defined in both the global North East coordinate system and the local body coordinate system. The global coordinate system is the domain of the GPS data and the robot controller. The IMU sensors give data for the local body coordinate system. Only three degrees of freedom were considered so for the local body coordinate system the motion is described in the roll (*r*) direction, the pitch (*p*) direction and by the angle theta (θ), which is defined as clockwise rotation of the local body axes relative to the global axes.

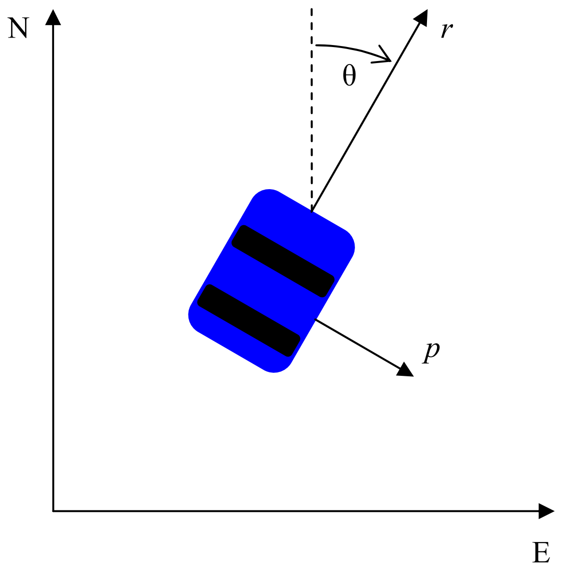


Figure 4: The Coordinate System Definitions.

5.4.2 Kalman Filter Derivation

To formulate a Kalman filter, the most important equations are the state dynamics model and the measurement model. All of the equations of the Kalman filter were derived from these two models. The general state dynamics equation is:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x},\mathbf{u}), \qquad (12)$$

where $\mathbf{x}$ is the system states,
$\mathbf{u}$ is the system inputs,
$\mathbf{f}(\mathbf{x},\mathbf{u})$ is a nonlinear function of the states and inputs.

For the robot driver's Kalman filter, inputs are not considered. As a result, this equation becomes:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) = \begin{bmatrix} v_N \\ v_E \\ v_E\omega \\ v_N\omega \\ 0 \\ 0 \\ \omega \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} p_N \\ p_E \\ v_N \\ v_E \\ a_N \\ a_E \\ \theta \\ \omega \\ b_r \\ b_p \\ b_y \\ f_r \\ f_p \\ f_y \end{bmatrix}, \qquad (13)$$

where V is the vehicle velocity.

The measurement model of the Kalman filter relates the sensor measurements to the states of the system. It is given in equation 14.

$$\mathbf{h}(\mathbf{x}) = \begin{bmatrix} p_N \\ p_E \\ f_r\left(a_N \cos(\theta) + a_E \sin(\theta) + b_r\right) \\ f_p\left(a_E \cos(\theta) - a_N \sin(\theta) + b_p\right) \\ f_y\left(\omega + b_y\right) \end{bmatrix} \qquad (14)$$

The matrix exponential of **f**(**x**) in the state dynamics equation (equation 13) is the nonlinear function of the state estimates from the last time step in the first equation of the Kalman filter (equation 1). The Jacobian of equation 14 gives the linearised measurement model, which is used in the third equation of the Kalman filter (equation 4). For more information on how to derive the Kalman filter equations from the state dynamics equation and measurement model see [Welch and Bishop, 2004].

5.4.3 Two Dimensional Assumption

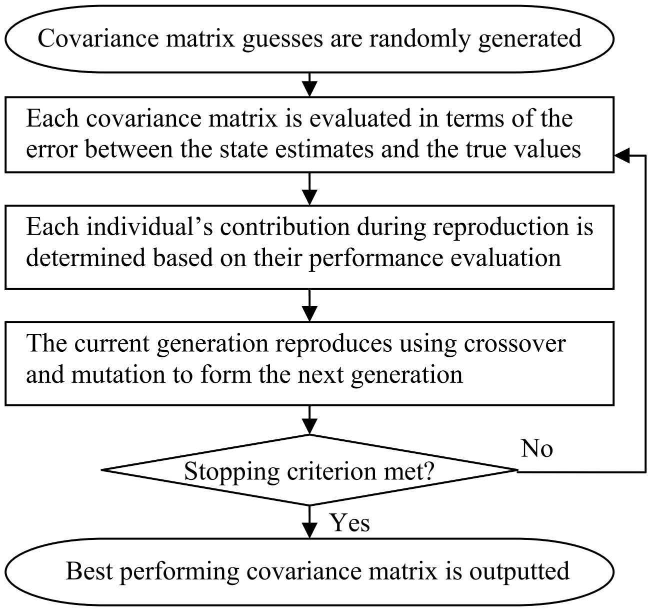
The Kalman filter and hence the sensor simulation and path planners have all been created for two dimensions. If implemented in three dimensions, this same Kalman filter would produce errors due to three-dimensional effects. For example, it is commonly recognised that biases are the major cause of inaccuracy in IMU measurements [Sukkarieh et al., 1999]. In a real life implementation, gravity would contribute to these biases and other forms of noise. However, gravity is not considered in the two dimensional model. Nevertheless, the two dimensional Kalman filter has worked well in field tests, as will be demonstrated in section 9. In any case, the two dimensional Kalman filter simply serves as a platform to test the various hypotheses, concerning process noise covariance matrix optimisation, presented in this research.

5.5 The Genetic Algorithm

A simple genetic algorithm was made to optimise the Kalman filter parameters. The genetic algorithm begins with the creation of an initial group of solution guesses [Charbonneau, 2002]. In this case, each guess was a randomly generated vector of parameter values. The next step may be called evaluation and involves evaluating the performance of each vector of parameters. In this research, the performance of each set of parameter values was evaluated by running the Kalman filter for those parameter values and comparing the true values of states with the state estimates calculated by the Kalman filter. Selection follows evaluation and consists of ranking the parameter vectors. The success of a vector of parameter values during evaluation determines its ranking and hence how much that solution will contribute during reproduction.

Reproduction is the creation of new sets of parameters from the members of the current generation. A process called crossover is used, where the parameters from existing individuals are recombined to form new population members. Parameters in one individual are swapped with equivalent parameters in the other individual. Crossover points mark the boundaries for swapping and are randomly selected. The result of crossover is two new solutions, which are new combinations of the original solution elements. Mutation also occurs during reproduction. Mutation occurs on a parameter by parameter basis and simply involves the replacement of a vector element with a randomly generated value. Mutation should occur with a low probability if the set of parameter values are to converge on the optimum.

After crossover and mutation has taken place, reproduction is complete and a new generation of vectors, containing parameter values, will have been formed. This new generation will be subjected to evaluation and selection. This new generation will also reproduce to form another generation and so on the genetic algorithm proceeds as shown in Figure 5. In this research, the genetic algorithms were run for 4000 generations.



Covariance matrix guesses are randomly generated
Each covariance matrix is evaluated in terms of the error between the state estimates and the true values
Each individual's contribution during reproduction is determined based on their performance evaluation
The current generation reproduces using crossover and mutation to form the next generation
Stopping criterion met?
No
Yes
Best performing covariance matrix is outputted

Figure 5: The genetic algorithm summarised.

6 Experiment 1

The first experiment was conducted to investigate how the performance of a set of parameter values should be evaluated. It was hypothesised that the Kalman filter parameters should be optimised in terms of more than just one state of the system. This is multiobjective optimisation. It was thought that optimising the process noise covariance matrix in terms of just one objective would lead to the degradation of the performance of the Kalman filter state estimates in terms of the other objectives. So if the covariance was optimised in terms of minimum position error, the Kalman filter might produce large angular errors. It is hypothesised that multiobjective optimisation would be a good solution to this problem.

6.1 Method

To test this hypothesis, firstly, simulated data was created. Separate data was created for optimisation and testing. The optimisation data was created by putting the path planner data for the two shorter paths through the sensor simulation program. One path ("short 2") was put through the sensor simulation program two times; the other ("short 1") was put through it three times. The result was sensor measurements for five vehicles paths. These sensor measurements were kept constant throughout all of the subsequent testing. The testing data was created by putting the long path from the path planner through the robot driver simulation and then through the sensor simulation program 100 times. The result of this was realistic sensor measurements for 100 distinct trips.

The performance of a single set of parameter values was evaluated by feeding the sensor measurements for the five optimisation vehicle paths into the Kalman filter one at a time. During each time step, states generated by the Kalman filter were compared with the true values, which were the path planner data values, in this particular experiment. The errors between the true values and the Kalman filter estimates were set to positive and averaged out over all of the five optimisation paths. What combination of these mean absolute errors should be used in the evaluation step of the genetic algorithm was the question behind this experiment. The mean absolute errors chosen for evaluation included the position error in metres, angular error in degrees and a combination of the position and angular error:

$$position\,error = \overline{(position\,estimate - true\ position)} \tag{15}$$

$$angular\ error = \overline{(filter\ angle\ estimate - true\ angle)} \tag{16}$$

$$mixed\ error = \sqrt{(angular\ error)^2 + (position\ error)^2} \tag{17}$$

In three separate runs, one for each type of error, the genetic algorithm was used to optimise the Kalman filter process noise covariance matrix in terms of the position error, angular error and the combination of these two. After the optimisation was complete, the resulting process noise covariance matrices were taken as inputs in different runs of another version of the Kalman filter. In each of these runs, the 100 lots of testing data were fed into the Kalman filter one after another. For each set of testing data the position error, angular error and the combination of these two was calculated. The averages of these 100 errors for each type of error and each process noise covariance matrix, from the optimisation step, were calculated.

6.2 Results

Table 1 gives the performance of the Kalman filter for the different process noise covariance matrices, optimised in terms of position error, angular error and a combination of position and angular error. The results in terms of the averaged position, angular and combined error are given.

Table 1: The results from experiment 1.

		Test results in terms of:		
		Position error	Angular error	Mixed error
Process noise covariance optimised for:	Position error	1.9 m	150°	150
	Angular error	1.9 m	8.8°	9.3
	Mixed error	1.9 m	9.1°	9.6

It is clear from the data in Table 1 that optimising the Kalman filter parameters in terms of just the position error gives a poor performance in terms of angular error, which appears to be relatively more sensitive to the values of the parameters. The best overall performance has actually come by optimising in terms of the angular error. However, this is very close to the performance gained by considering both the angular and position error. In fact, the process noise covariance matrices for these two cases were found to be very similar.

The conclusion drawn here is that it would be best to include angular error in the optimisation of the process noise covariance matrix. In the experiments that follow, the combination of both the position and angular error will be used to assess process noise covariance matrix performance during optimisation.

7 Experiment 2

The second experiment that was conducted investigated whether it was sufficient to optimise just in terms of the data generated by the path planner program and the sensor simulation program or if the robot driver program had to be used as well. If the process noise covariance matrix can be adequately optimised in terms of the data given by the path planner and sensor simulator, the implication is that the robot driver simulator does not need to be used during parameter optimisation. This would suggest that the vehicle manoeuvres were the key source of process noise.

7.1 Method

Three sets of data were created for this experiment. The test data was the same as for the last experiment. The optimisation data was created using the five copies of the long path from the path planner. In one case, this was fed through the robot driver simulator and then the sensor simulator. To make the other set of data, only the sensor simulator was used. The same noise was used to create both sets of optimisation data.

The performance of a single set of process noise covariance matrix values was evaluated by using the Kalman filter to compute state estimates for each of the five paths of a single set of optimisation data. The combination of the angular error and position error was calculated at each time step of the Kalman filter. After the Kalman filter had finished running for the current process noise covariance matrix, the combination of the angular and position error was averaged over the entire period of time covered by the five paths. The genetic algorithm optimised the process noise covariance matrix in terms of the averaged combined angular and position error for both sets of optimisation data. The resulting pair of process noise covariance matrices was tested in the same way as in the first experiment.

7.2 Results

The resulting averaged errors in Table 2 show that there is virtually no difference between optimising the process noise covariance matrix with data that is fed through the robot simulator and performing the same optimisation with data that is not created using the robot simulator. This is a useful result because it means that one step can be eliminated in the process of creating data for the optimisation stage.

Table 2: Results for experiment 2. Case 1 is where the process noise covariance matrix had been optimised with data that had been run through the robot driver simulator. Case 2 used data that had not involved the robot driver simulator.

	Test results in terms of:		
	Position error	Angular error	Mixed error
Case 1	1.8 m	7°	7.5
Case 2	1.9 m	6.9°	7.5

8 Experiment 3

The third experiment in this research investigated if different manoeuvres should have different process noise covariances. This would require optimising two process noise covariances: one each for the bends and straights.

8.1 Method

The production of optimisation and test data was the same in this experiment as it was in the first experiment. The key changes were made in the optimisation and testing stages. Two different process noise covariance matrices were optimised by the same run of the genetic algorithm in this experiment; whereas, in previous experiments, only a single process noise covariance matrix was optimised by a single run of the genetic algorithm. The performance of a pair of process noise covariance matrices was evaluated by firstly defining one as being for straight movements and the other as being for turning movements. A turning movement included the period of deceleration into and acceleration out of the corner. The process noise covariance matrix used at any given time step was dependant on whether the vehicle was moving straight or turning. For example, when the vehicle was moving straight, the process noise covariance used in the Kalman filter was switched to the correspondingly defined matrix. Apart from this adjustment, the Kalman filter was run as before: for five optimisation paths with the average combination of angular and position error calculated at the end. The genetic algorithm then optimised the pair of process noise covariance matrices in terms of the average combined angular and position error.

Both of the process noise covariance matrices were trialled using the test data. As with experiments one and two, the testing was carried out by running a copy of the Kalman filter 100 times with 100 different sets of data, which had been created by the path planner, robot simulation and sensor simulation. The mean absolute angular error, position error and combined error was calculated for each run of the Kalman filter. The difference in this experiment was that the matrix designated for straights was used for segments where the vehicle was travelling on straight line paths and the process noise covariance matrix that had been assigned to turns was used in the Kalman filter for sections where the vehicle was turning.

8.2 Results

The first row of Table 3 (Case 1a) gives the performance of the Kalman filter during testing when two process noise covariances were used: one for bends and one for straights. The second row (Case 2a) gives the performance of the Kalman filter when one covariance matrix was used.

Table 3: Test results for the case where different process noise covariance matrices are optimised for different manoeuvres. Note that the bottom row of data is from experiment 1.

	Test results in terms of:		
	Position error	Angular error	Mixed error
Case 1a	1.8 m	7.2°	7.8
Case 2a	1.9 m	9.1°	9.6

Firstly, consider the circumstance where the planned paths used in optimisation are different from the planned paths used to create the test paths. This situation is more likely to be the case. A comparison with the case where only one process noise covariance matrix is used reveals that using different process noise covariance matrices for different planned manoeuvres gives an improved performance.

Table 4: Test results for the case where different process noise covariance matrices are optimised for different manoeuvres. Note that the bottom row of data is from experiment 2.

	Test results in terms of:		
	Position error	Angular error	Mixed error
Case 1b	1.8 m	7.2°	7.8
Case 2b	1.8 m	7°	7.5

Now, consider Table 4. Case 1b is the same as case 1a in Table 3; it is where two process noise covariance matrices have been optimised for the two different manoeuvres of the robot driver. Case 2b is where the optimisation paths were the same as the test paths; this is Case 2 in Table 2. Comparing the results from these different cases shows that optimising for one covariance matrix by using the path that will actually be travelled gives better Kalman filter performance than optimising for two process noise covariance matrices by using paths different from the path travelled. However, the difference in performance is very small and it may not be possible to optimise for every path travelled. For long paths this would slow down the optimisation stage considerably. In this case, generally optimising process noise covariances for different manoeuvres should be the preferred option.

9 Field Tests

A field test was performed to demonstrate the performance of a GPS/IMU Kalman filter, optimised offline using the findings of the three experiments conducted previously. The offline optimisation was performed in terms of the mixed error objective function. Separate process noise covariances were used for bends and for straights.

9.1 Method

To create sensor data for the optimisation of the process noise covariances, measurements were collected from a GPS unit and an IMU, while driving over a marked course. The course was a rectangle with dimensions of approximately 50 metres by 80 metres (Figure 6). Top speeds of approximately 30 km/h were reached along the straights. Corners were taken at low speeds with a very tight turning radius.

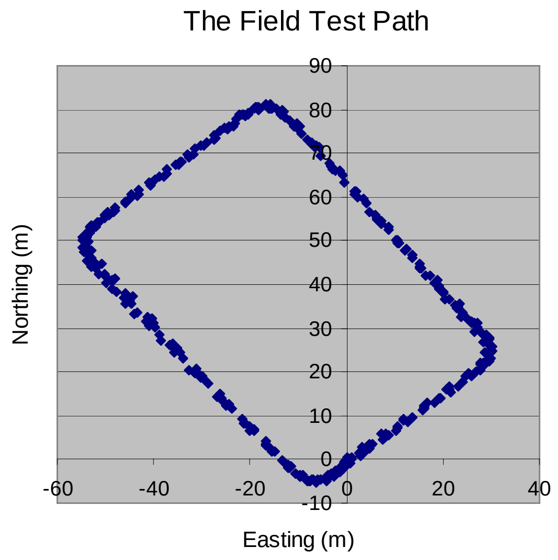


Figure 6: GPS data points showing the path followed by the vehicle during the field test. The Northings and Eastings were taken relative to the origin, which was the starting point.

This GPS and IMU data was run through the genetic optimisation algorithm as before in experiment 3, where separate process noise covariance matrices were found for bends and straights. However, a change was made to improve the measurement model of the Kalman filter with an extra measurement being added. This measurement was the angle created by the two most recent GPS readings.

During optimisation, approximations for the position and angular error were made. Before optimisation began the rectangular course was surveyed using approximately one thousand GPS measurements. Straight line equations for each edge of the rectangle were found in terms of Northings and Eastings. During optimisation, the position error was taken as the minimum distance between the Kalman filter position estimate and the straight line equations for the course. The straight lines defining the edges of the rectangular path were also used to find approximate values for the angle at every time step. The true angles at the corners were calculated as linear interpolations of the angles of both corresponding edges of the rectangular path. The difference between the true angle estimates and the Kalman filter angle estimates were used in the calculation of the mixed error as in equation 17.

The definition of a bend and a straight was set based on a threshold value of the angular velocity seen in the gyroscope measurements. Angular velocities below the threshold value indicated that the vehicle was travelling on a straight section. Gyroscope measurements above the threshold indicated a bend.

The outputs of the optimisation were two process noise covariance matrices; one for bends and the other for straights. These covariances were used in running the Kalman filter in a field test. The field test was conducted over the same course, which was used for the collection of optimisation data. However, faster driving was conducted. To measure the performance of the Kalman filter, the average mixed error, as defined in equation 17, was calculated for the Kalman filter state estimates. The position errors and angular errors were calculated as described for optimisation. In addition, the raw GPS data was logged during the field test. This data was post-processed to produce angle estimates at each time step using the two most recent GPS position measurements. The mixed error for the GPS data was also calculated.

9.2 Results

The mixed error produced by the state estimates of the Kalman filter and the mixed error calculated for the GPS measurements are compared in Table 5. Clearly, the Kalman filter produces significant improvements by combining the GPS data with the IMU measurements. This suggests that the method of tuning the Kalman filter by offline optimisation, using multiobjective optimisation and separate process noise covariances for the different manoeuvres, is a useful one.

Table 5: Field test results comparing the mixed errors from the state estimates of the Kalman filter and the state estimates calculated from GPS data alone.

	Mixed error
Kalman filter	4.9
GPS	6.6

The results in Table 5 also suggest that the two dimensional assumption, discussed in section 5.4.3, was fair. The path used in the field tests had a gradient of approximately one percent. Based on the success of the Kalman filter in the field tests, the two dimensional assumption is justfied for relatively flat terrains.

10 Conclusions

This research has tested three hypotheses. The conclusion from the first experiment was that when optimising a Kalman filter offline using a metaheuristic, the optimisation should be multiobjective or in terms of the most sensitive state. Otherwise, there may be a serious degradation in performance in terms of states not included in the optimisation. The second experiment showed that the simulated path measurements used in optimisation did not need to include process noise for the robot driver considered. The final experiment showed that, where the optimisation paths are kept the same, using different process noise covariance matrices for different manoeuvres gives better accuracy than a single optimised covariance matrix.

The outcome of this research is a method that may be used when optimising the process noise covariance matrix of a Kalman filter in similar or analogous applications to the robot driver described. The first key factor that would lead to the adoption of this method would be the absence of a model relating the inputs and the system states. This would lead to increased process noise. If this noise was somewhat predictable, as it is in this robot driver application, then the optimisation method described in this paper may be relevant.

The important points of this method are, firstly, to consider carefully the states to be used in the objective function during optimisation. Multiobjective optimisation or optimisation in terms of the more sensitive variables may be appropriate. Secondly, simple paths may be sufficient for optimising the process noise covariance matrix. The process noise may not have to be included in simulated paths during the optimisation process, if the process noise created by not including inputs in the Kalman filter is dominant. Another key result is that different process noise covariances may be used in distinctly different sections of the journey. This should give improved performance, when the process noise changes dramatically and predictably in different segments of a planned path. The proposed method of using multi-criteria optimisation and different process noise covariances for different manoeuvres, during offline optimisation, has been successfully demonstrated with a field test.

11 Future Work

This paper has described research that has tested three hypotheses, concerning the offline optimisation of the process noise covariance matrix of a GPS/IMU Kalman filter for a robot driver application. The experiments in this research have been conducted on simple two dimensional models. To extend the findings of this research to more general cases, it would be appropriate to test the same concepts on more complicated three dimensional models. Future work for this research will involve developing and experimenting with Kalman filters in three dimensions. In addition, the performance of an adaptive Kalman filter may be compared to the offline optimisation procedure described in this paper.

Acknowledgements

Warm thanks to Greg MacDonald and Alistair Poolman, undergraduate students at The University of Auckland, for providing the robot driver simulator for this research. The authors wish to thank Grand Challenge NZ for providing the opportunity to work on the robot driver, which has formed the basis of this research. Thanks also to the Mechanical Engineering Department of The University of Auckland for providing summer studentship funding for this research.

References

[Bolognani et al., 2003] Silverio Bolognani, Luca Tubiana and Mauro Zigliotto. Extended Kalman filter tuning in sensorless PMSM drives. *IEEE Transactions on Industry Applications*, 39(6):1741-1747, November/ December 2003.

[Chan et al., 2001] Zeke S.H. Chan, H.W. Ngan, Y.F. Fung and A.B. Rad. An advanced evolutionary algorithm for parameter estimation of the discrete Kalman filter. *Computer Physics Communications*, 142(1-3):248-254, December 2001.

[Charbonneau, 2002] Paul Charbonneau. *An Introduction to Genetic Algorithms for Numerical Optimization.* NCAR Technical Note, March 2002.

[Farrel and Barth, 1998] Jay Farrell and Matthew Barth. *The Global Positioning System and Inertial Navigation*. McGraw-Hill, New York, 1998.

[Grewal et al., 2001] Mohinder S. Grewal, Lawrence Weill and Angus P. Andrews. *Global Positioning Systems, Inertial Navigation, and Integration*. John Wiley, New York, 2001.

[Gueye et al., 2005] Ousmane Gueye, Karina Lebel, Jean De Lafontaine and Charles-Antoine Brunet. Fine-tuning of a Kalman filter with a genetic algorithm with gradient based optimization methods. *Advances in the Astronautical Sciences*, 119(III):2521-2536, 2005.

[Hu et al., 2003] Congwei Hu, Wu Chen, Yongqi Chen and Dajie Liu. Adaptive Kalman filtering for vehicle navigation. *Journal of Global Positioning Systems*, 2(1): 42-47, November 2003.

[Loebis et al., 2004] D. Loebis, R. Sutton and J. Chudley. A fuzzy Kalman filter optimized using a multi-objective genetic algorithm for enhanced autonomous underwater vehicle navigation. *Proceedings of the Institution of Mechanical Engineers Part M: Journal of Engineering for the Maritime Environment*, 218(1):53-69, February 2004.

[Powell, 2002] Thomas D. Powell. Automated tuning of an extended Kalman filter using the downhill simplex algorithm. *Journal of Guidance, Control and Dynamics*, 25(5):901-908, September/ October 2002.

[Shin et al., 1992] D.H. Shin, S. Singh and J.J. Lee. Explicit path tracking by autonomous vehicles. *Robotica*, 10:539-554, 1992.

[Sukkarieh et al., 1999] Salah Sukkarieh, Eduardo M. Nebot and Hugh F. Durrant-Whyte. A high integrity IMU/GPS navigation loop for autonomous land vehicle applications. *IEEE Transactions on Robotics and Automation,* 15(3):572-578, June 1999.

[Welch and Bishop, 2004] Greg Welch and Gary Bishop. *An Introduction to the Kalman Filter*. University of North Carolina at Chapel Hill, North Carolina, 2004.

[Xiong et al., 2005] Zhilan Xiong, Yanling Hao, Jinchen Wei and Lijuan Li. Fuzzy adaptive Kalman filter for marine INS/GPS navigation. *Proceedings of ICMA 2005*, *IEEE International Conference on Mechanics and Automation*, 747-751, 2005.