Applications of the Kalman Filter Algorithm to Robot Localisation and World Modelling

Leonie Freeston

Supervisor: Professor Rick Middleton

Electrical Engineering Final Year Project 2002
University of Newcastle, NSW, Australia

Background

The University of Newcastle has entered a robot soccer playing team in the Sony Legged League of the RoboCup 2002 competition. This website outlines the localisation and world modelling strategy for this team, including the use of Kalman Filters for this task. Robot localisation is the process whereby a robot determines its own position in the world in which it functions. World modelling is the process used by a robot to determine the position of other objects in its world. The Kalman Filter has been implemented for a robot to locate its own position and the ball position. Communication between the robots has enabled cooperative world modelling to be used to determine team strategy.

RoboCup Soccer

RoboCup

RoboCup is an international organisation whose aim is to promote robotics and artificial intelligence research and education. One of RoboCup's main activities every year is an international competition which incorporates three different events: RoboCup Soccer, RoboCup Rescue and RoboCup Junior. The competition has been running since 1997 and in 2002 is being held in Fukuoka, Japan from June

19th - June 25th, coinciding with the human soccer World Cup. Currently the main focus of the RoboCup competition is RoboCup Soccer: robots playing soccer against other robots.

Why soccer?

The soccer game format provides a standard problem for teams to solve which allows opportunity for a wide range of technology and innovation to be integrated into the game. Most people are aware of the challenge of making a computer beat a human in chess. Soccer is a similar kind of problem, except that a soccer game is much more dynamic and requires a lot more use of external sensors. The ultimate goal of RoboCup is "by 2050, [to] develop a team of fully autonomous humanoid robots that can win against the human world champion team in soccer". RoboCup is not just about soccer, however. The technology and artificial intelligence research advances that are made in enabling the robots to play soccer have applications in many industrial and socially significant areas such as large scale search and rescue situations.

The University of Newcastle Team

2002 is the first year a University of Newcastle team has participated in RoboCup Soccer. The "NUBots" will be competing in the Sony Legged Robot League, sponsored by Sony. All teams in this league use Sony AIBO Entertainment Robot Dogs. Each team has access to the software of all the previous year's entrants in the league which provides a level playing field for all teams each year and allows new teams to have equal chance in the competition. It also ensures that the standard of work is improved upon each year, as the aim is to effectively utilise new technology and further develop existing AI methods. Sony retains the rights to any software or intellectual property generated as a result of the competition and in return allows the teams confidential access to any information about Sony hardware or software required to program the robots.

The University of Newcastle RoboCup 2002 team consists of representatives from the Faculty of Engineering and Built Environment (School of Electrical Engineering and Computer Science), the Faculty of Business and Law (Newcastle Business School) and the Faculty of Science and Information Technology (School of Mathematics and Physical Sciences) including nine undergraduate and postgraduate student software developers, academic staff and technical staff. The RoboCup Soccer project is being run by the Newcastle Robotics Laboratory at the University of Newcastle.

The Sony AIBO Entertainment Robots

The hardware used for this project is the Sony AIBO Entertainment Robot. The robots are programmed in a C++ software environment and the programs can be loaded onto a memory stick which is inserted into the robot's memory stick slot. The robots have an internal 64-bit processor and PC Card slot allowing communication via Wireless LAN.

The legs each have two joints (three directions of movement) to enable walking and kicking, the head has three directions of movement and the tail, mouth and ears also move. The robots have a CMOS colour image sensor (camera), microphones, speakers, a temperature sensor, an acceleration sensor, pressure sensors (on the head, back, chin and legs), LEDs on the head and tail and an infrared distance sensor. The robot is powered by a lithium ion battery pack.

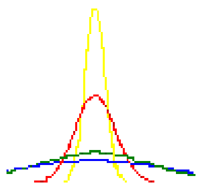
The dimensions of the robot (WxHxL) are 154x266x274mm (not including the tail and ears) and the mass is approximately 1.4 kg (including battery and memory stick).

The RoboCup Soccer Rules

Each team in the Sony Legged League of the RoboCup Soccer competition consists of four robot dogs. Both teams wear a different coloured uniform (either red or blue) and are programmed at startup via a touch sensor on their back which colour their team is so they know which goal to attack (the red team always attacks the blue goal and the yellow team always attacks the yellow goal). There is no human interaction with the robots to control them after they are turned on. The dogs are not remote controlled and are run autonomously (although a robot can be restarted if it crashes). The robots are allowed to communicate via wireless LAN.

The dimensions of the soccer field are 270 x 420cm. There are six uniquely identifiable coloured beacons placed around the field (one at each of the four corners and two at the centre of the field on either side) and blue and yellow goals at either end. The robots play with an orange ball of 8cm diameter.

Only one defending robot is allowed beyond the white defenders line (the goalie) and robots are not allowed to interfere with an opposition robot who is heading for the ball. Robots breaking this rule are penalised by being removed from the area and placed on the halfway line.



Kalman Filter Information

Why use the Kalman Filter?

To model the robot position we wish to know its x and y coordinates and its orientation. These three parameters can be combined into a vector called a state variable vector. The robot uses beacon distance and angle measurements and locomotion information about how far it has walked to calculate its position. As with any real system, these measurements include a component of error (or noise). If trigonometry is used to calculate the robot's position it can have a large error and can change significantly from frame to frame depending on the measurement at the time. This makes the robot appear as if it is "jumping" around the field. The Kalman Filter is a smarter way to integrate measurement data into an estimate by recognising that measurements are noisy and that sometimes they should ignored or have only a small effect on the state estimate. It smooths out the effects of noise in the state variable being estimated by incorporating more information from reliable data than from unreliable data. The user can tell the Kalman Filter how much noise there is in the system (including the measurements, the inputs and the model) and it calculates an estimate of the position taking the noise into account. It is less likely to incorporate one bad measurement if the current position confidence is high.

The Kalman Filter algorithm also makes it very easy to combine measurements from different sources (such as vision measurements and locomotion data) and different times (updates as a robot is walking). In addition to an estimate of the state variable vector, the algorithm provides an estimate of the state variable vector uncertainty ie how confident the estimate is, given the value for the amount of error in it.

That's basically all that is needed to know about the Kalman Filter in order to understand the project website, but if you wish to find out more, read on...

State Space Models

The state variable vector mentioned above (also called a plant model) is usually denoted X. To calculate this vector at any time interval we use the formula:

$$\begin{pmatrix} x \\ y \\ \theta \end{pmatrix}_{k+1} = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix}_{k} + \begin{pmatrix} \Delta & 0 & 0 \\ 0 & \Delta & 0 \\ 0 & 0 & \Delta \end{pmatrix} \begin{pmatrix} v_x \\ v_y \\ v_\theta \end{pmatrix}_{k}$$

$$X_{k+1} = AX_k + Bu_k$$

The state variable vector summarises the relevant information from the system at any time and describes how the system changes as a function of time and input. The k and k+1 subscripts represent the time of the vector. In this case, Bu is the input that we receive due to the robot walking (distance = speed x time) and A is the identity matrix. In general, u is the input vector, A is a matrix relating the state variables at the previous time step to the state variables at the current time step and B is a matrix relating the input to the state variables.

If we have a measurement of the x and y positions and orientation we could write these measurements in the form:

$$\begin{pmatrix} x \\ y \\ \theta \end{pmatrix}_{k} = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix}_{k}$$

$$Y_k = CX_k$$

This is called the measurement model and it describes how the sensor data varies as a function of the state variables. Y is the output (or measurement) variable vector and C is a matrix relating the state variables to the measurement variables. In this case C is the identity matrix.

To generalise, any system which is finite dimensional, causal (the output does not depend on future input values) and time invariant can be described by a set of variables which constitute the important aspects of the system’s internal behaviour. These variables can be combined into a vector which at any time contains the information required to characterise the system. This type of model is known as a state space model and is commonly used in control system modelling and signal processing. The equations above represent a linear, discrete time state space model.

In the robot situation, we are not able to directly measure x, y and orientation so we have to calculate them from the measurements of distance and angle to beacons. To do this, we use an observer.

What is an Observer?

We want to know x, y and orientation of the robot, but can only measure distance and angle to beacons on the field. We first make an estimate of x, y and orientation. Then we calculate what distance (or angle) we would measure if we were at this position and compare this to the actual measurement we made. If there is a difference, we know our position estimate is wrong and adjust our x, y and orientation to try to make it closer to the real position. If the difference is zero, our original position estimate must be correct and no adjustment is made. This process is repeated at each time interval. This can be represented by:

$$\hat{X}_{k+1} = A\hat{X}_k + Bu_k + J(Y_k - C\hat{X}_k)$$

J is the feedback gain. The last term in the equation represents the adjustment of the position estimate to take into account the error between the actual measurement and the value we would have got if we took that measurement at the position we estimated. The ^ sign above a variable denotes that it is an estimate (not the actual value).

So, in summary, Observer Design is a method of estimating a state variable vector using only the sensor measurement data (the vector Y) and the system model (A, B, C). It uses feedback of the difference between the actual sensor measurements and the estimated model output vector to adjust the state variables.

Ok then, how do you choose a good value for J? Use a Kalman Filter...
Mathematically, a Kalman Filter is an Observer which can be proved to have optimal properties operating under certain conditions.

Other Important Background Knowledge

An important aspect of the Kalman Filter is the Gaussian distribution. The Kalman Filter algorithm assumes that all noise is white (has equal energy at all frequencies) and has amplitude which can be modelled as a Gaussian (or Normal) distribution. The Gaussian distribution has two important parameters:

- The mean (the average size of the noise). All noise considered has zero mean.
- The variance (the square root of variance is the standard deviation which is the average size of the noise away from the mean). A probability distribution with a higher variance has a higher overall level of noise.

Gaussian distributions are shown in the graphic next to the heading at the top of this page.

Another fundamental part of the theory of the Kalman Filter is the covariance matrix. The covariance matrix of a state variable vector V with zero mean is defined as the expected value of a vector multiplied by its transpose E(VV'). The nth diagonal element of the covariance matrix is the variance of the nth element of V and the off diagonal elements are the cross covariances of the variables.

One Kalman Filter parameter we will come across is P, the error covariance matrix. When we talk about error we mean the difference between the actual state variables and the estimated state variables. P is generally used as an estimate of the uncertainty, that is, the larger P is the more uncertainty there is in the estimate of the state variables. The square root of the diagonal elements of P give the standard deviation of the error in the state variables.

Noise in the Kalman Filter model

In order to realistically assess a model, the errors, or noise must be considered. Noise in the plant model is called the model/input noise and it takes into account model uncertainty and input disturbances. Noise in the measurement model is called measurement noise and it takes into account noise from the sensors. For the Kalman Filter Q is the covariance matrix of the model/input noise and R is the covariance matrix of the measurement noise.

The Kalman Filter

There are actually several forms of the discrete Kalman Filter which all give the same results. I will attempt to give a brief explanation of the time/measurement form of the Kalman Filter. There are many good references and websites on Kalman Filters so if this explanation doesn't completely answer all your questions, try these - some of the websites I have found useful are on the Links page.

The time/measurement form of the Kalman Filter is executed in two steps. The time update projects the state variable vector estimate ahead in time taking into account the input into the system (that is, it makes a prediction of the new state). The measurement update adjusts the time update state estimate to take into account measurements made during the time interval.

The Kalman Filter Equations are written as follows:

$$\hat{X}_{k|k-1} = A\hat{X}_{k-1|k-1} + Bu_k \quad (1)$$

$$P_{k|k-1} = AP_{k-1|k-1}A^T + Q \quad (2)$$

$$J_k = P_{k|k-1}C^T(CP_{k|k-1}C^T + R)^{-1} \quad (3)$$

$$\hat{X}_{k|k} = \hat{X}_{k|k-1} + J_k(Y_k - C\hat{X}_{k|k-1}) \quad (4)$$

$$P_{k|k} = (I - J_kC)P_{k|k-1} \quad (5)$$

Before we start the Kalman Filter we need to make an estimate of the initial value of the state variable vector and error covariance matrix.

Step 1: The Time Update.
The time update equations are equations (1) and (2). Equation 1 calculates what is called the “a priori” estimate of the state variable vector - an estimate of X at time k given measurements up to time k-1. This updates the state estimate based on the knowledge of the previous time state and the input to the system since the last time update. Equation 2 updates the “a priori” error covariance matrix P. This update represents the fact that knowledge of the system gradually decays over time due to errors in the assumed input or in the model which contribute to the uncertainty in the estimate.

Step 2: Measurement Update
The measurement update equations are equations (3), (4) and (5). The time

update state estimate in the previous step is corrected based on the feedback from the sensor measurements that have been taken during the time interval. Equation 3 calculates the Kalman gain J to minimise the a posteriori error covariance matrix P. (Remember when we were talking about state space models we said that the Kalman Filter could be used to calculate J.) Equation 4 calculates the "a posteriori" estimate of the state variable vector - an estimate of X at time k given measurement data up to time k. This updates the state variable estimate for the new measurement. It compares the actual sensor measurement (Y) with the sensor measurement predicted from the state estimate (CX) and uses the difference between these values (called the "innovation" or "residual") and the Kalman gain to determine the amount by which to adjust the time update state estimate. Equation 5 calculates the "a posteriori" error covariance matrix P. You will notice that equations (1) and (4) combined are the same as the Observer equation above.

These equations are repeated at every time interval. If the Kalman Filter does not receive any measurement information it performs the time update (Equations (1) and (2)) and no measurement update.

The Extended Kalman Filter (EKF)

The Kalman Filter equations above assume that the relationships between measurements, inputs and state variables are all linear. This is not always the case and a sometimes a different version of the Kalman Filter known as the Extended Kalman Filter (EKF) must be used. For the robots the sensor measurements are a non-linear combination of the state variable vector. The usual approach to the non-linear case is to linearise the measurement about the current estimate of the state by using Taylor's theorem and delete higher order terms (those with power greater than 1). The EKF model is:

$$X_k = AX_k + Bu_k + w_k$$
$$Y_k = h(X_k) + n_k$$

The equations for an EKF for the model above are:

$$\hat{X}_{k|k-1} = A\hat{X}_{k-1|k-1} + Bu_k$$

$$P_{k+1|k} = AP_{k-1|k-1}A^T + Q$$

$$C_k = \left.\frac{\partial h}{\partial X}\right|_{X=\hat{X}_{k|k-1}}$$

$$J_k = P_{k|k-1}C_k^T(C_kP_{k|k-1}C_k{}^T + R)^{-1}$$

$$\hat{X}_{k|k} = \hat{X}_{k|k-1} + J_k(Y_k - h(\hat{X}_{k|k-1}))$$

$$P_{k|k} = (I - J_kC)P_{k|k-1}$$

Note that in the EKF algorithm C is now time-varying whereas in the linear case it was a constant. The equations above are not the true EKF, as they still assume that the the state variable vector is a linear combination of the previous state variable vector or the inputs. If it isn't, a similar linearisation process can be followed. See a good Kalman Filter reference for a proof or extention of these results. As the model is not linear any more, the algorithm is not necessarily optimal.

What information do I need to have to run a Kalman Filter?

The Kalman Filter algorithm requires a number of parameters. These are:

- A model of the system (A, B, C matrices).
- The input at each time step.
- Sensor measurements.
- Initial state of the system X0 and initial error covariance matrix P0: Obviously the initial state of the system is the best estimate of the starting point, or at least the centre of all the possible starting points, if all are as likely as each other. It is usually good practice to make P large initially, otherwise it can take the Kalman Filter a long time to start noticing any measurement data.
- Measurement noise covariance R: R can usually be estimated offline by taking sensor measurements at a known value and using a standard variance formula.
- Model/input noise covariance Q: Q tends to be used as tuning mechanism to optimise Kalman Filter performance. Different values of Q are tested and the value that provides the best results is chosen.

As we saw from the equations the output of the Kalman Filter algorithm is an estimate of the state variable vector and the error covariance matrix (uncertainty of state variable estimate) at each time interval.

What effects do R, P, Q and J have?

R (the measurement noise covariance) determines how much information from the sample is used. If R is high we are telling the Kalman Filter that the measurement isn't very accurate. When R is smaller the Kalman Filter output will follow the measurements more closely and accept more information from them.

The effect of P (the error covariance matrix) on the Kalman Filter estimate is that when P is small the Kalman Filter incorporates a lot less of the measurement into the estimate as it is fairly certain of its position. Ideally P gets as close to zero as possible to indicate that the model is accurate. P is generally reduced by measurements received, as there is more confidence in the estimated state if there is a measurement to confirm it. However the reduction of P is limited by the model/input variable error covariance Q which is added at each time step.

Both P and R are incorporated into the Kalman Filter through the Kalman gain J. J determines how much of the innovation (the difference between the actual measurement and the model measurement) is used to correct the estimate. J varies in proportion to the error covariance matrix P and is inversely proportional to the measurement covariance matrix R. If the measurement noise covariance R is large compared to the error covariance matrix P then the Kalman gain J will be small. This means the certainty of the measurement is small relative to the certainty of the current state model and the old model is better compared to the new measurement – minimal adjustment to the estimate is required. Alternatively, if P is large compared to R, J will be large and the estimate of X is updated to look more like the measurement than the previous estimate. The innovation is weighted more heavily.

Q (the model/input noise covariance) contributes to the overall uncertainty of the estimate as it is added to P in each time step. The Kalman Filter output when Q is large tracks large changes in the actual output more closely than the Kalman Filter output with smaller Q. However, Q large is much noisier as the estimate is allowed to "move" more in one time step. This means there is a performance trade-off between tracking and noise in the output in the choice of Q for the Kalman Filter.

When and how exactly is the Kalman Filter optimal?

The way in which the Kalman Filter provides the optimal estimate is that it minimises the "a posteriori" error covariance matrix P. P contains the expected value of error in the estimate squared and hence the Kalman Filter is a solution to the least squares problem. The conditions under which the Kalman Filter is optimal are:

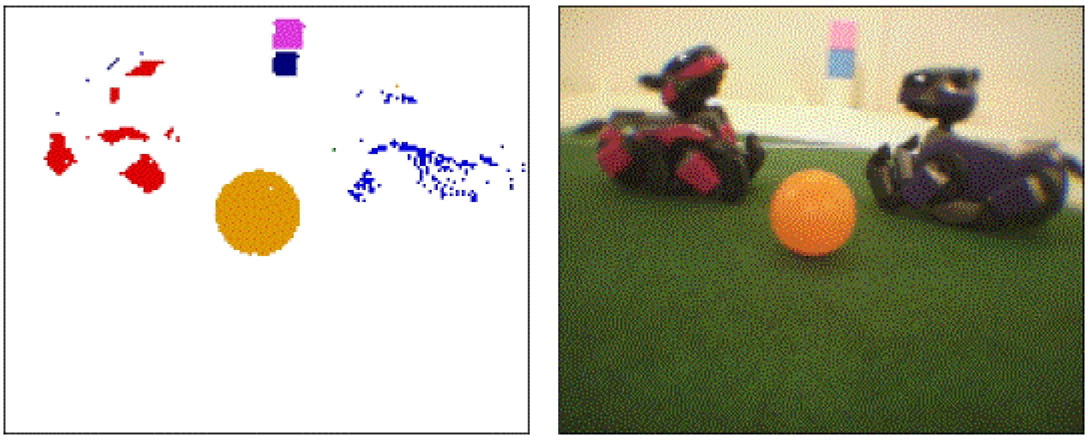
- The model/input noise and measurement noise are both independent (uncorrelated in time and with each other).
- The model/input noise and measurement noise are both white noise with a zero mean Gaussian distribution.
- The system is known.
- The system is linear.

A Kalman Filter can still be used in a system in which these assumptions do not hold, but there is no certainty that it is the optimal filter.

Project

How do the robots play soccer?

The robots receive information through the camera in their nose. Each pixel is processed to see what colour it is, and classified according to whether it is the same colour as any of the objects on the field that the robots are looking for (such as orange for the ball, or the colours of the beacons or goals). Groups of like colours are put together, and from this the robots determines if it can see anything on the field and where it is relative to the robot.


The classified camera image and the actual camera image.

This information is sent to the localisation and world modelling module, to work out where the robot is, and where the ball is. The robot position and ball position is sent to the behaviour module, where the robot determines what action to do, and sends information to the locomotion module about what to do (walk, kick, move head etc). The locomotion module sends information to world modelling about how far the robot has moved which is used to update the position. All the robots have localisation and world modelling information from all the other robots (wireless LAN link), so team strategy decisions can be made.

How can a robot localise?

There are a number of approaches that can be used to localise a robot. Odometry information (information from motors and other sensors about robot movement) can be used to determine robot position. If the initial starting position and the speed and direction of movement is known, the robot's position in the coordinate system can be uniquely determined. The disadvantage of this method is that precise knowledge of where the robot starts and very accurate odometry information is needed. In practise this is not feasible as the robot slips as it is moving and can collide with other objects. Using vision information to localise is another alternative, but if the robot can't see any distinguishing landmarks at any particular point in time it cannot localise. Therefore it is desirable to combine information from difference sources to improve the accuracy of positioning. Consequently both vision and odometry information are used in the robot localisation software.

The Kalman Filter in RoboCup

Obviously the Kalman Filter has been used in some way to help with robot localisation and world modelling (otherwise, there wouldn't be a page devoted to it). The robot position is modelled by a Kalman Filter and the ball position is also modelled by a Kalman Filter.

The robot position is determined by the x and y coordinates on the field and the orientation. There are six uniquely identifiable beacons and four goal edges on the RoboCup soccer field for which the coordinates are known before runtime, allowing them to be used as a reference for position on the field. To localise, the robot uses information from vision about the location of beacons and goals with respect to the robot and information about the amount of movement. The robot localisation routine is called every time a vision update is received which is approximately twenty five times per second. As the measurements of distance and angle are not a linear function of x, y and orientation, the Extended Kalman Filter is used.

Initially the ball’s position was computed using a simple trigonometric calculation on the information from vision. The errors in calculating the position of the ball using this method are rather high, so a Kalman Filter was implemented on the ball to eliminate some of these errors. The ball position is determined by x and y coordinates, but the Kalman Filter also includes velocity estimation in the x and y directions. The velocity states allow prediction of the ball’s position to be made when the robot cannot see the ball. Negative acceleration of the ball due to friction on the field is taken into account. For the ball there is no knowledge of the ball’s input driving force (robots kicking the ball, ball bouncing of wall edges) so the only information that is used is the vision information of distance and angle to the ball.

If communication between the robots is possible via the wireless LAN link it is advantageous for them to share information, especially about where the ball is. This is particularly useful if one robot cannot see the ball. In order to share information a Kalman Filter is implemented on ball position measurements from all robots. For this Kalman Filter, each robot calculates the (x, y) position of the ball using trigonometry and sends this information to all the other robots. These x and y coordinates are treated as the 'measurements' and since the measurements are a linear function of the variables we wish to know, a linear Kalman Filter is used. All the measurements from all the robots are then used in a Kalman Filter. As there were some problems with the ball position "jumping" as measurements from different robots were put into it, two ball position Kalman Filters were implemented - one with measurements from all the robots, and one with measurements from one robot. The one robot ball position was used in order to position for a kick, and the multiple robot ball position was used when the robot could not see the ball.

Constraining Position in the Field

The Kalman Filter is not always going to return values of (x, y) that are valid in the robot soccer application, that is, within the soccer field. The simplest way of solving this is to clip the individual coordinate that is out of range back into the field. A more insightful solution is to use the fact that there is an estimate of the certainty of the measurements and adjust the position back onto the field in the direction of most uncertainty. Adjustments should be made to uncertain measurements rather than measurements of reasonable accuracy. The field imposes eight constraints on position (four edges and four corners).

Localisation and World Modelling

We just talked about the Kalman Filters that are being used to implement localisation and world modelling, but how do they fit into the overall picture?

The purpose of the localisation and world modelling software module is to provide the behaviour control module with an absolute position on the field of as many objects as possible, including the robot’s own position, the ball position, other team member positions and opposition team member positions. In order to perform this function, information is received from the vision module about the identification, distance and angle of various objects on the field, including the ball, beacons, goals (the edges of which can be used as beacons) and other robots. The locomotion module provides information on the movement of the robot and the localisation and world modelling module also receives the localisation and world modelling information from the three other robots on the team (via the wireless communication network). This information is processed to determine the current position of the robot, ball and other robots and communicated to the behaviour control module.

There are a number of objects that are stored in the robot world model. These are the robot's own position, the ball position, other team member's positions, the opposition team member's positions, the goal position and the beacon positions. Each of these objects is stored with the following information: object type, x coordinate, y coordinate, orientation, standard deviation of x coordinate estimate error, standard deviation of y coordinate estimate error, standard deviation of heading estimate error. Not all of these parameters are relevant for each object (for example, orientation of the ball is not needed).

There are two different world models stored by the robots: the individual world model and the cooperative world model. The individual world model is only based on information available to the robot from its own vision and locomotion, and this world model is sent to all the other robots on the team. The cooperative world model is compiled from the individual world models of all the robots and this is the information sent to the behaviour module.

The Individual World Model

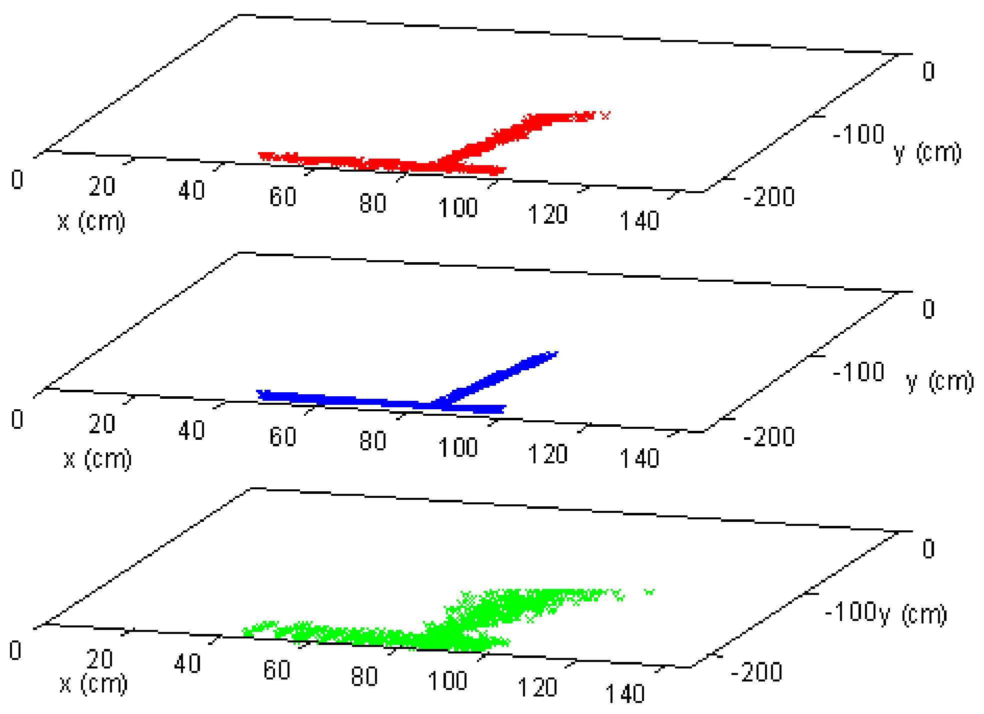
The individual world model is compiled in the following way. Each robot calculates its own position based on beacon measurements from vision and locomotion data using a Kalman Filter. If the robot can see the ball, the ball position is calculated as the robot's current position plus the ball's relative position using simple trigonometry. If any other opposition robots can be seen these are added to the world model in the same way as the ball, and are clipped inside the field. The individual world model is sent to all the other robots once every ten image frames due to the bandwidth limitation imposed by the competition rules. This allows each robot to have more information by copying data received from other robots into the cooperative world model (particularly the position of the other team members). If a robot cannot see the ball it can use information from the other robots to locate the ball.

The Cooperative World Model

The cooperative world model is sent to the behaviour module. This world model consists of the robot's own position, team member positions (copied from the data received via the wireless link), the cooperative ball position (which is a Kalman Filtered version of the data received from all the robots), the individual ball position (which is a Kalman Filtered version of data from only one robot) and any opposition robots that any robot can see. The reason for using two different ball positions is that the cooperative ball position estimate tends to jump around as it receives information from different robots who locate the ball at different positions due to natural inconsistencies in their own estimated position and vision ball measurements. This makes it hard for the robot to kick the ball as it cannot get a good ball position estimate. The cooperative ball position is used if the robot cannot see the ball, and the individual ball position is used if the robot can see the ball.

Matlab and C++

The robot localisation Kalman Filter was first coded in Matlab and run using simulated vision data. The diagram below shows a comparison of Kalman Filter position estimate (top) and trigonometry position calculation (bottom) to the actual path (centre) for a simulated robot. It shows that the Kalman Filter estimate is a much smoother estimate than the trigonometry calculated path and the Kalman Filter path follows the actual path fairly closely.



The localisation and world modelling software was also coded into C++ for implementation on the robots.

How does the robot use localisation and world modelling?

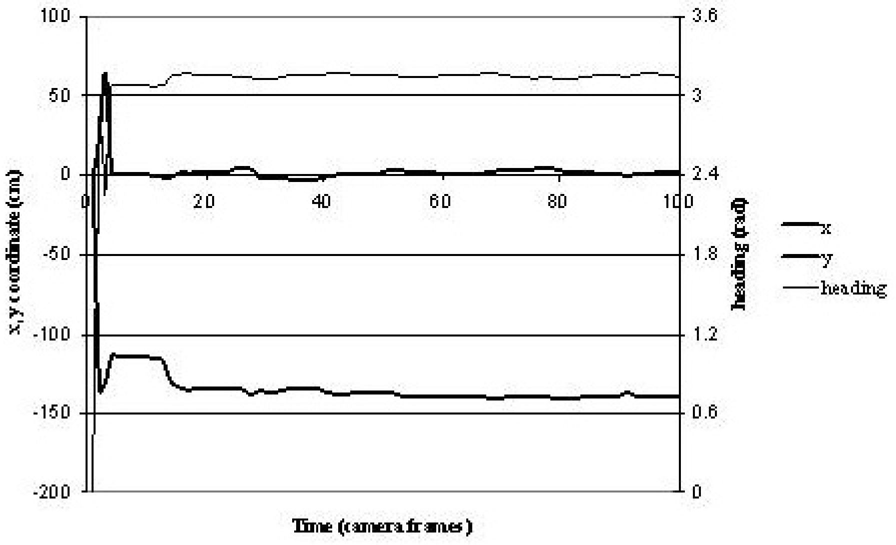
When the wireless communications network is operating the robots decide during the startup sequence which robot will be the team captain and make decisions about the overall team strategy. The team captain uses the cooperative world model information to make decisions about the mode of play (attack or defence) depending on the position of the robots and the ball, and will make decisions about which robot plays which position. The robot localises in the ready state by panning for beacons, and also pans while it walks during a game. When the wireless communications are not operating, each robot just looks for the ball and tries to score (much like 6-year old soccer!).

Results

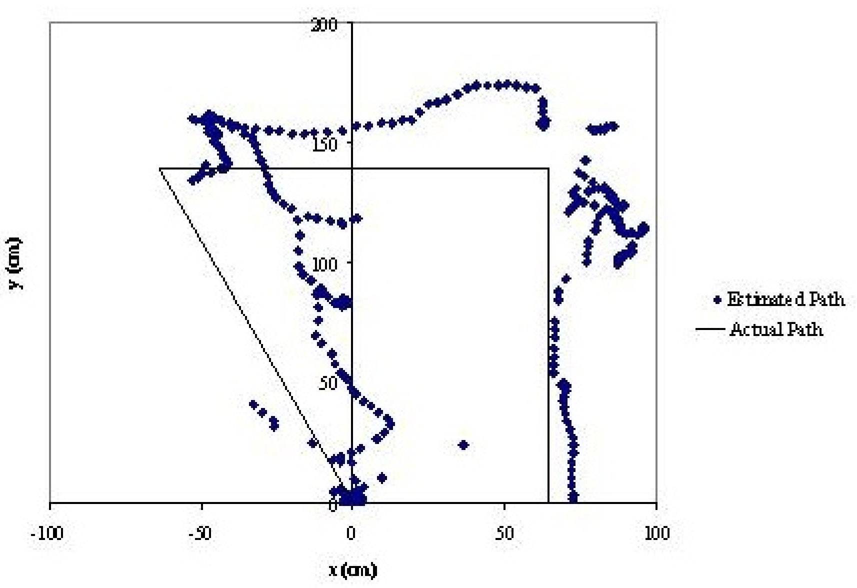
How well does the Kalman Filter work for robot localisation?

The accuracy of localisation was tested for two situations: when the robot is standing still, and when the robot is following a set path around the field. These are not realistic tests of how the robot will perform in a soccer game as is very difficult to quantify the actual performance of localisation and world modelling for a robot in play, but they do give some indication of how accurately the Kalman Filter can perform.

The robot was switched on in different positions on the soccer field and allowed to localise by panning for beacons. The estimated position of the robot was logged to a file on every vision update and then examined. In some instances the robot could localise to within 1cm of its actual position. In other cases there was over 10cm of error. There was a large variation in localisation depending on position on the field which could probably be attributed to inconsistent lighting conditions. The graph below shows the data logged when the robot position was x=0cm, y=-140cm and heading=pi radians. Observe on this diagram that the robot can localise in under twenty camera frames (0.8 seconds).



The other test situation was for a robot following a set path on the field. The graph below is of the data logged for this path compared to the actual path. The movement data is still relatively accurate, but there are instances where the data moves away from the actual path quite considerably. In this test situation, the robot would walk for a while, stop to obtain measurements and then continue walking. A more accurate result would be obtained if the robot was looking around while it was walking, but the remote control software used to obtain the results was unable to do this. In the actual soccer game, more measurement data would be obtained than in this test situation, and the results would be more consistently accurate.



One indication of how well localisation performs during the soccer game was given by the goalie code. The goalie is told what absolute position to move to in front of the goal in order to protect it. During testing, the goalie robot moved to the correct position on the field and regulated its position there very accurately.

How well does the Kalman Filter work for ball position?

The individual ball Kalman Filter operating on the ball measurements received from one robot has a significant smoothing effect on the ball data received from vision. When the ball is stationary the vision distance measurements can still significantly change from frame to frame, but the Kalman Filter has the desired effect of making the estimate remain stationary. The Kalman Filter can also track a moving ball with high accuracy. Outlier detection has been implemented on the ball so any measurements that are significantly different from the current estimate do not get included in the Kalman Filter update.

The cooperative ball Kalman Filter provides a fairly noisy estimate of the ball position, but is a useful estimate of the ball position for a robot that cannot see the ball.

Debugging

The robots have the capacity to output text via a serial cable, log data to files and send data over the wireless network. All of these methods were used for debugging the software. A Java application, written by RoboCup team members Michael Quinlan and Craig Murch, receives data from the robots over the wireless network and displays this information. The main debugging screens used for this project were the World Map and Shared World Map screens which showed the positions of where the robot thought objects were on the field. The World Map displays the individual world model and the Shared World Map displays the cooperative world model. Vision data is also displayed on the World Map screen which enables the information that the localisation and world modelling module is receiving from the vision module to be compared to the actual values.

What errors contribute to the results?

The errors that contribute to the robot localisation Kalman Filter include:

- Robot locomotion data errors: Inaccuracy in locomotion data can be attributed to a number of factors including time lag (the data is not sent until after motion has taken place), slip on the field surface, collision with other objects (walls and other robots) and individual robot motor differences.
- Model linearisation errors
- Misclassified camera pixels: Colour similarities (for example, yellow and orange) can cause objects to be 'seen' where there is no object, or pixels at the edge of a colour block not to be recognised making the object appear a different size to what it is.
- Granularity (limit of reading): The calculation performed to determine a beacon's distance from the robot is 1800/(height of pixels) which means the distance measurements are not a continuous function. For example, the difference in the distance measurement between seeing a beacon height of five pixels and six pixels is 60cm even though the robot may not have moved very far.
- Lighting conditions: Different lighting conditions have a significant impact on the accuracy of the vision information. Shadows and reflections in different parts of the field will make the colours change significantly and cause them to be misclassified.
- Camera velocity: Observations of the distortion of objects when the camera is moving fast indicate that each camera frame update is not very fast. Objects can appear to be 'spread' across the frame which distorts the distance and angle readings.
- Synchronisation of vision and sensor data: Vision angle measurements are adjusted to take into account the angle of the head. When the head is moving fast the head angle and camera frame aren't always synchronised which causes the angle to be inaccurate.

The ball position input errors arise from the fact that there is constant movement of the ball but no knowledge of the input driving this. In addition to the vision errors already mentioned, errors in the ball Kalman Filter can be attributed to:

- Centre finding technique: The centre of the ball is calculated by selecting three points on the circumference of the circle and using circle geometry (perpendicular bisectors) to calculate the centre of the ball. The accuracy of this technique is highly dependent which points are chosen, and bad point selection can distort the measurements.
- Glare: The shiny surface of the ball is highly susceptible to glare, and causes the ball image pixels to be misclassified and hence the distance and angle measurements can be inaccurate.

Other Factors Affecting Performance

The accuracy of the Kalman Filter is strongly related to the accuracy of the vision data which is, in turn, heavily dependent on the lighting conditions on the field. At the time of testing there were significant shadows on some corners of the field. Also, since the last vision calibration was performed the angles of the floodlights lighting the field had been slowly decreasing (due to some loose bolts). This changed the lighting conditions but wasn't noticed until shortly before testing was finalised. It is expected that in the tournament which is held in a large auditorium the light will be more evenly dispersed and the accuracy of localisation will not be dependent on the area of the field.

The accuracy of the robot localisation Kalman Filter is also related to the quantity of vision data that is received and performance is better when many different beacons can be used in the position update. The results shown above in the path following example were limited by the fact that as the robot was walking it could see a maximum of one beacon and therefore the results were influenced by one measurement only. When the robot stopped walking to look around for beacons, the position was adjusted back closer to the actual path.

Other performance inhibitors include the misclassification of objects. For a significant period of time there were cases when the vision module was misclassifying beacons as goals which had a very negative effect on the ability of the robot to determine its position. In other cases, due to the similarity in colouring, the yellow beacons or goals can be classified as the ball and vice versa. A particular phenomenon observed for localisation was that after the robot panned for beacons, it would stop moving its head and could end up with its sight fixed on one particular beacon. If the distance measurement was consistently under or overestimated for this beacon, the robot position estimate was influenced significantly by this measurement and resulted in large inaccuracies. One solution is that the robot should not stop panning for beacons at any time during the ready state (which is the state in which the robot cannot move its legs, but can localise by moving its head).

How were the parameters chosen?

Robot Position Model/Input Noise Covariance Q
The value for Q for robot localisation was chosen by simultaneously running a number of test Kalman Filters with different Q values and comparing the results. The position estimates returned by the Kalman Filters were logged to a file and compared to the actual position to determine which Q performed better. For both the situations of a robot standing still and a robot moving, a small Q performed more accurately. This prevented the robot from changing its position estimate too much when it received inaccurate data. Q was chosen to be equivalent to 1cm/sec and 1 degree/sec for optimum results.

Robot Position Measurement Noise Covariance R
To determine R, testing was done on the beacon measurements observed by a stationary robot. The head was moved randomly so the beacon appeared in different parts of the image frame and the measurements were logged to a file. This was repeated at five different distances and each time measurements were taken of all six different beacons on the field. The distance error was calculated and it was determined that the standard deviation of distance error was equivalent to four pixels variation in the measurement. The angle standard deviation was also calculated and was roughly three degrees.

Ball Position Model/Input Noise Covariance Q
The ball model/input noise covariance is fairly high to account for the fact that the ball can move quite fast and there is no indication of any change in movement available as an input to the Kalman Filter. It is assumed that the ball can move 10cm in every second. Although this variable was not ‘tuned’ as such, observation has determined that this provides a good estimate of the ball position.

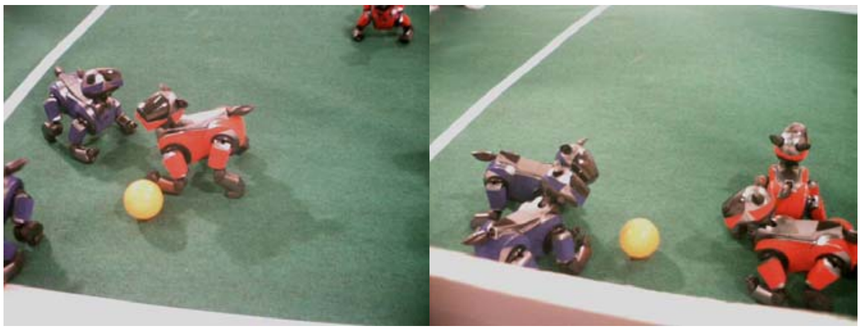
Ball Position Measurement Noise Covariance R
Tests of the measurements of a stationary ball at different distances were done in order to determine the variance of the measurements. A standard deviation of approximately a quarter of the distance was observed for the distance measurement error and the angle error standard deviation was approximately two degrees.

Conclusion

This project has demonstrated that the Kalman Filter can be effectively applied to robot localisation and world modelling and significantly enhances the operation over other methods. It is possible to obtain an estimate of a stationary robot position using a Kalman Filter accurately to within a few centimetres with reasonable consistency, and a robot soccer player has been able to maintain a set absolute position with a high degree of accuracy. The performance of localisation can decrease as the robot moves, but is highly dependent on the amount of measurement data that is received as it is walking. There is still room for improvement in the consistency of the estimate, particularly while the robot is moving, but confidence is held that in the leadup to the competition, significant improvements can be made to the accuracy. The Kalman Filter can be used for other objects on the field such as the ball and can significantly improve the estimate of the ball position over a vision ball position estimate. Cooperative world modelling can successfully be used to determine team strategy, and sharing information about where the ball is is highly advantageous as it allows robots who cannot see the ball to have ball position information, although at this stage it has been found that a cooperative ball estimate is not useful for individual robots to kick the ball. The Kalman Filter is a useful tool in localisation and world modelling of mobile robots and has been successfully applied to the University of Newcastle's Sony Legged League RoboCup 2002 robot soccer team.

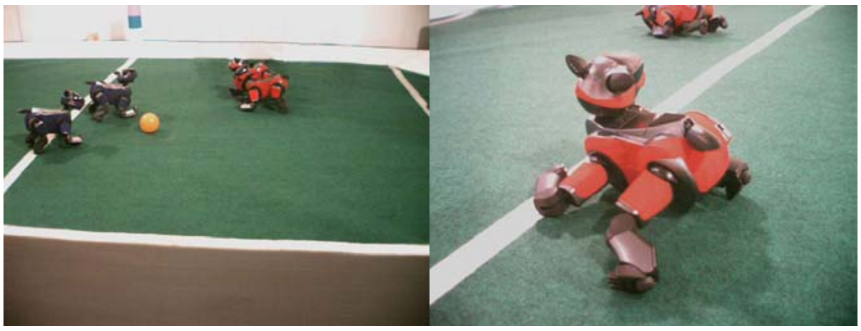
Photos

Click on the images to enlarge the photos.



The robots playing a soccer match.

The robot kick and the goalie save.


A game in progress, and my thoughts after completing the final year project! (Look closely at the top dog...)

Useful Links

Click on the graphics to link to the following sites:

The School of EE and CS at the University of Newcastle, Australia

The department under which this project was completed.

Professor Rick Middleton

Rick is a Professor with the School of Electrical Engineering and Computer Science in the Faculty of Engineering and Built Environment at the University of Newcastle, Australia. Rick supervised this project and is an extensive source of information about Kalman Filters. Rick is also the Director of CIDAC.

Centre for Integrated Dynamics and Control

CIDAC is a Special Research Centre funded by the Australian Government, linked to the School of Electrical Engineering and Computer Science at the University of Newcastle, Australia. Their focus is on advancing control techniques. They run a good short course on Kalman Filtering.

Newcastle Robotics Laboratory

The Newcastle Robotics Laboratory is a research group within the University of Newcastle with representatives from the Faculty of Engineering and Built Environment (School of Electrical Engineering and Computer Science), the Faculty of Business and Law (Newcastle Business School) and the Faculty of Science and Information Technology (School of Mathematics and Physical Sciences). They are interested in research in robotics and artificial intelligence. The RoboCup 2002 project was run by this lab.

The Kalman Filter

This site is run by Greg Welch and Gary Bishop from the Department of Computer Science at the University of North Carolina at Chapel Hill. It has a useful introductory paper, and links to other Kalman Filter sites.

RoboCup 2002 Competition Site

The official RoboCup 2002 Competition Site.

RoboCup Organisation

The RoboCup Organisation site.

The Sony AIBO Entertainment Robot

All you ever wanted to know about the Sony AIBO...
