Probabilistic Controlled Airspace Infringement Tool

Yousra S. AL-Mathami

College of Engineering, Mathematic and Physical Sciences

University of Exeter

June 2012

Abstract

A current ground based safety net called Controlled Airspace infringement Tool (CAIT) is used by Air Traffic Controllers (ATCs) in the UK. It warns them if any aircraft within uncontrolled airspace penetrates the Controlled airspace (CAS) without an advance clearance from the ATC. This penetration or 'Infringement' is considered as a major concern to ATCs where it may cause a possible conflict or mid-air collision. A conflict is an event which one aircraft loses its minimum separation to another. A current deficiency of CAIT is that it only warns ATCs if the aircraft has already infringed CAS, this gives the ATC minimum time to react and avoid any conflict.

In this research, we investigate a model which warns ATCs of possible future infringements accurately. We implement two Kalman filters (KF) as our tracking tool, one for each flight mode: constant velocity and constant acceleration each of which has its state error covariance. Where the state error covariance measures the uncertainty in the aircraft's estimated position, and is therefore important for accurately predicting the aircraft future position and since each aircraft has its own characteristic and journey type, a single parameterisation of the state covariance for all aircraft is unsuitable. Therefore, we learn these covariances in an online fashion at each time step to predict the future uncertainties more accurately. Given the two Kalman filters predictions and their error covariances, we use two methods to find the probability of infringement of CAS. The first method, proposed by Macdonald (2000), is called the shortest distance method. We extend this method to be able to find the probability of infringement when the prediction location is near a CAS corner by combining it with Monte Carlo sampling. A hybrid method is introduced to retain the efficiency of the shortest distance method with the accuracy of the Monte Carlo sampling.

We also used the switching Kalman filter (SKF) method proposed by Murphy (1997) to choose between the most appropriate Kalman filter at each time step. On testing on real tracks, the SKF was found to give superior predictions of the aircraft location, permitting better estimates of the probability of CAS infringement to be made.

Acknowledgements

I would like to thank NATS for providing us with the data which aided us throughout our research.

I specially would like to thank Richard Everson and Jonathan Fieldsend for their help and support and for providing me with the knowledge for which a student would wish to gain from his educator. Thank you mom and dad for your long distant support which made my study in Exeter feels like home.

Table of Contents

List of Common Abbreviations

ATC	Air Traffic Controller
CA	Constant Acceleration
CAS	Controlled Airspace
CV	Constant Velocity
EM	Expectation Maximisation Algorithm
HMM	Hidden Markov Model
KF	Kalman Filter
CAIT	Controlled Airspace Infringement Tool
PSR	Primary Surveillance Radar
SKF	Switching Kalman Filter
SSR	Secondary Surveillance Radar
UCAS	Uncontrolled Airspace

1 Introduction

1.1 Overview

It is the responsibility of NATS to monitor and to manage the flow of the aircraft in the United Kingdom airspace. The UK airspace is divided into seven classes; five of them correspond to controlled airspace (CAS) and two to the uncontrolled airspace (UCAS). These five classes are being controlled by the Air traffic controllers (ATCs) and therefore, for any aircraft flying within or about to fly into CAS, the communication between that aircraft and the ATC is essential to avoid any conflict. Where a conflict is an event in which one aircraft loses its minimum separation by another aircraft. ATCs use screens to monitor the aircraft locations which are being collected by radars placed all over UK. ATCs also monitor aircraft outside the controlled airspace using another tool in conjunction with radars, called Controlled Airspace Infringement detection Tool (CAIT) to detect any infringements. An infringement occurs when an unauthorised aircraft penetrates CAS without the ATC's clearance. These infringements are a major concern to ATCs, in which it could cause a possible conflict and maybe a mid-air collision.

1.2 Research Problem

Controlled airspace infringements are a major safety concern to ATCs and aircraft pilots which can cause two main issues: one is the possibility of a conflict, where an aircraft loses its minimum separation. Another is that, these infringements cause disruption to flight operations by adding more workload on the pilot and the ATC such as changing the flight routes and finding a safe maneuver to avoid a collision. As a result, CAIT was developed to detection any infringements which occur. However, it only warns the ATC if it has already infringed the CAS which gives the ATC less time to resolve any possible conflict that may arise. It would be better for the ATC to have an early warning to resolve a future conflict in advance and maintain the flow of aircraft in the CAS in the same time.

1.3 Objectives

The aim of this research is to investigate a possible model that warns ATC well in advance of any possible future infringements. In order to do that, we will examine the following:

1. Implementing an accurate tracking tools for different models
2. Review and extend available probability of infringement methods
3. Learning the models’ error covariance and test them on synthetic track
4. Examining a switching mechanism to switch between the proposed models

Given the best errors which fit the model, we will find the probability of future infringements using real tracks and CAS boundaries.

1.4 Research Structure

In chapter 2 we will present an in depth background information related to the airspace, radars, current CAIT and the tools we use to help us build the probabilistic infringement detection tool. Following that we will provide a literature review on conflict detections methods. Chapter 3, we will examine the current probabilistic CAIT and extend it by using the Monte Carlo Sampling in conjunction with it to create a hybrid model which will able to predict future infringements more accurately. In chapter 4, we will implement a learning tool proposed in (Shumway and Stoffer,1997) which finds the best error which fit the model. Given the models and their errors, we discuss in chapter 5 a switching mechanism proposed in (Murphy, 1997) to switch between the models. Chapter 6, we will provide a discussion and some recommendations for more research work.

2 Background

The research objective is to implement a probabilistic infringement detection tool which predicts future infringements accurately; therefore, it is essential to give background information related to this research such as what is airspace and who controls it, the types of radars and the current infringement detection tool and its limitation. The tracking tool we will use to predict and estimate the aircraft locations is the Kaman filter which we will review it in this chapter. Similar to the Kalman filter, is the Hidden Markov Model (HMM) which we will use it to learn the hidden states as we will see later in this chapter. This is then followed with a literature review on conflict detection and, finally, a summary.

2.1 Airspace

Historically as the number of airplanes increased it became essential to manage their flow and maintain their separation from each other in the air, resulting in the establishment of air traffic services. These services are provided by air traffic controllers (ATCs) for aircraft flying within the controlled airspace (CAS). These aircraft are supervised by the ATC and should follow their instructions as long as they are inside the CAS. Any aircraft flying in an uncontrolled airspace (UCAS) is not subject to ATC communication and the pilot should take full responsibility for his/her safety. Within controlled airspace, the ATCs monitor the air traffic flow and use separation rules to avoid any conflict or mid-air collision. A conflict is an event for which an aircraft loses its minimum separation. Typical minimum separations are 5 nautical miles horizontally and 1000 feet vertically, although these may be modified according to the particular expected behaviour of the aircraft. For example, near a terminal where aircraft are about to land, the separation may be 1 nautical mile (nm) horizontally and 500ft vertically.

Airspace is divided into 7 classes; 5 corresponding to CAS and 2 to UCAS. The UCAS classes are available for the civil users such as light aircraft, helicopters and hot air balloons …etc and they extend from the ground up to 3000 ft or to the base of CAS if there is one defined above UCAS. Pilots flying within UCAS can receive flight information from the ATC if requested. In this research, we are interested in tracking aircraft within these UCAS zones. Figure 2.1 shows the 5 controlled airspace classes over the UK; areas not coloured corresponds to UCAS.

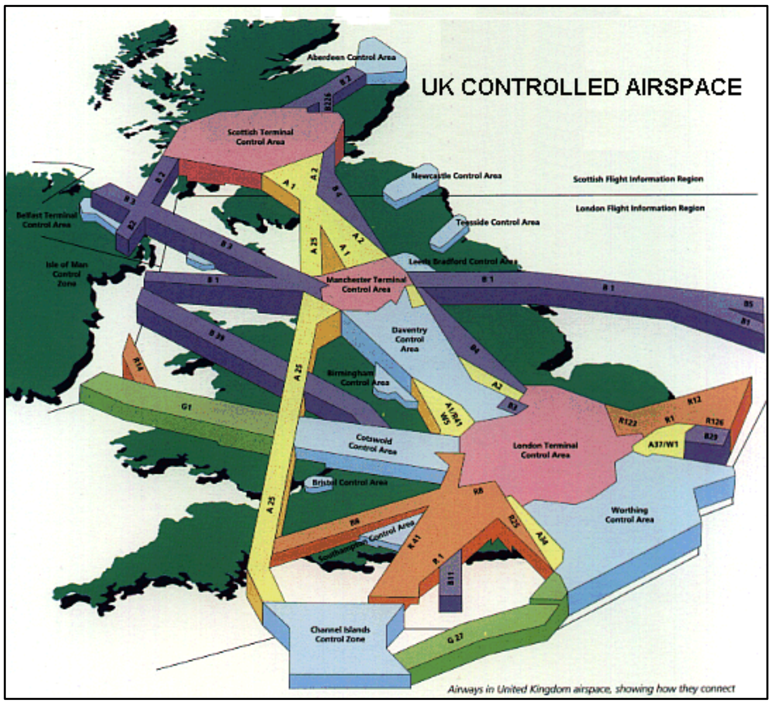

UK CONTROLLED AIRSPACE
Aberdeen Control Area
Scottish Terminal Control Area
Newcastle Control Area
Scottish Flight Information Region
London Flight Information Region
Teesside Control Area
Belfast Terminal Control Area
Isle of Man Control Zone
Leeds Bradford Control Area
Manchester Terminal Control Area
Daventry Control Area
Birmingham Control Area
Cotswold Control Area
London Terminal Control Area
Bristol Control Area
Worthing Control Area
Southampton Control Area
Channel Islands Control Zone
Airways in United Kingdom airspace, showing how they connect

Figure 2.1: UK controlled Airspace classes: Source: (Hayward & Howell, 2008)

It is very important to detect and if possible predict where an aircraft in UCAS infringes CAS; the system that ATCs use to monitor aircraft locations in UCAS is radar which will be discussed next.

2.2 Radar

Radar is an object detecting system mainly used by ATCs to detect aircraft in the air or on the ground. There are two types of radar that are used by the ATCs, the primary surveillance radar (PSR) which only detects the location of the aircraft with imprecise altitude; and the secondary surveillance radar (SSR) which provides more information to the ATC such as altitude and identity. The SSR however, only operates when the aircraft is equipped with a transponder, which makes it a dependant system that needs to communicate with that aircraft's transponder in order to provide the extra information. So aircraft with no transponders (light aircraft) can therefore only be detected by PSR but not the SSR. As a result, ATC uses the PSR in conjunction with the SSR to detect and identify the aircraft. Figure 2.2 shows one of the radars used by ATCs at London Heathrow Airport. Both radars SSR and PSR are also placed in different locations around the UK:

Figure 2.2: Heathrow airport radar. Source: wikipedia

Since most aircraft that infringe CAS are not equipped with transponders, they can only be detected by the PSR. This tells the ATC very little about that aircraft altitude, flight plan and their identity; therefore, ATCs require a safety net that warns them whenever an infringement occurs. This system is called the Controlled Airspace Infringement Tool – CAIT – and is discussed in section 2.3.

2.3 Current Infringement Detection

Controlled airspace infringements are a major safety concern to ATCs and aircraft pilots which can cause two main issues: one is the possibility of a conflict, where an aircraft loses its minimum separation. Another is that, these infringements cause disruption to flight operations by adding more workload on the pilot and the ATC such as changing the flight routes and finding a safe maneuver to avoid a collision. Therefore, a ground based safety net called the Controlled Airspace Infringement Tool (CAIT) used by NATS was created in order to warn ATCs of an infringement to the CAS by an unauthorised aircraft. This is achieved by collecting aircraft locations within UCAS by the PSR radar every four seconds, sending these locations to the ATC monitors and then highlighting the aircraft in different colour when the aircraft is inside the CAS, thus raise a warning to the ATC. Figure 2.3 shows an example of ATCs' monitor when an aircraft has infringed CAS which is highlighted in magenta.

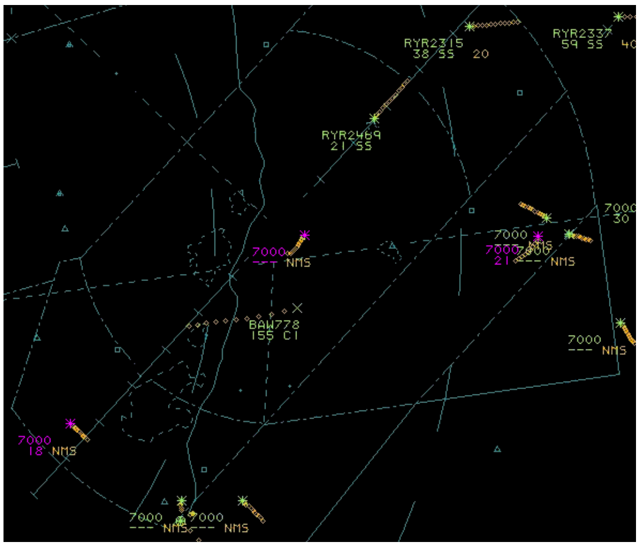


Figure 2.3: Radar monitor showing infringed aircraft in magenta instead of green

The flaw with the CAIT system is that it only warns ATCs if and only if, the aircraft has already infringed the CAS and therefore, could cause a possible conflict with another authorised aircraft flying inside it. It does not warn ATCs for possible future infringements by aircraft outside CAS. To solve this problem is our research objective, which is to investigate a possible model that warns ATC well in advance of any possible infringement in the near future. This will help the ATC to respond to it and resolve it quickly before it could cause further risk. In the next section we will give a brief introduction to two important tools used in our research.

2.4 Hidden Markov Models

The Hidden Markov Model (HMM) is a powerful statistical tool used to model time series data (Rabiner,1989). The model uses a set of observable states sequence Z to estimate the hidden state X where each hidden state and observation is modeled by a probability distribution. See figure 2.4

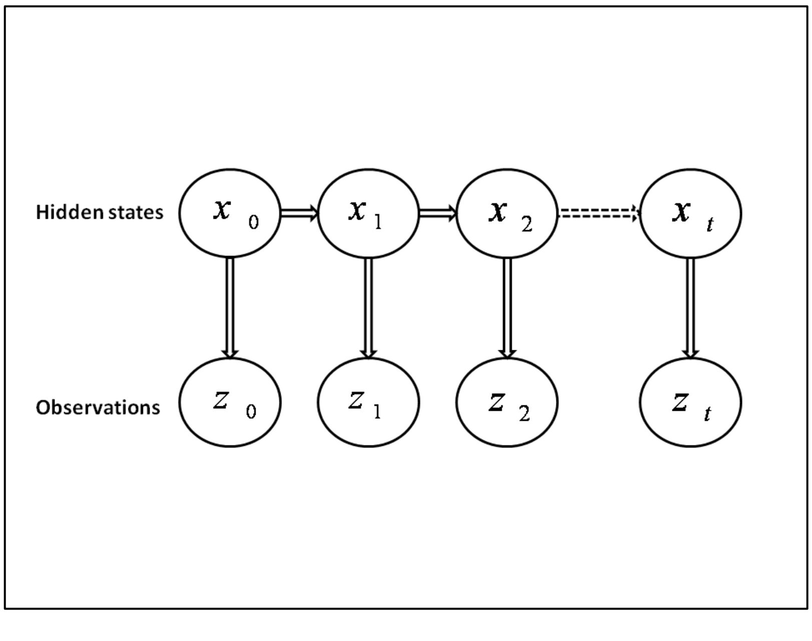


Figure 2.4: Hidden Markov Model Transition Process

Given a set of random states $x_0, x_1, x_2, \dots \dots . x_{t-1}, x_t$ from time 0 up to the current time t, from the Markov assumption which states that the probability of the next state x_{t+1}

depends only on the current state x_t, this makes it a first order HMM. Therefore, the state sequence in the HMM does not depend on the time t. Meaning, the initial state along with states up to time $t-1$ will soon be forgotten as the state sequence increases because it will not be used in the model.

To transit from one state to another, a probability is assigned to each different state; this is called transition probability which is defined below:

$$P(x_{t+1}|\, x_t)$$

Also, the probability of the current observation z_t depends only on the current state x_t and it is defined by the probability:

$$P(z_t|\, x_t)$$

The HMM can have two types of time series hidden states: discrete and continuous where each type has either discrete or continuous time observations.

1. Discrete time hidden states which generate either:
    - Discrete observations
    - Continuous observations
2. Continuous time hidden states which generate either:
    - Discrete observations
    - Continuous observations

Figure 2.5 shows the types of the HMM hidden states and observations. The HMM with discrete hidden states and discrete observations use the forward-backward algorithm to estimate the hidden state given the observation. To learn the model's parameters, the Baum-Welch algorithm is used with discrete states and observations. The HMM however, with continuous hidden states and observations use the Kalman filter for example as the best state estimator if we assume that these states and observations are linear and normally distributed. To learn this model's parameters, the Expectation-Maximization algorithm (EM) is the most suitable one to use.

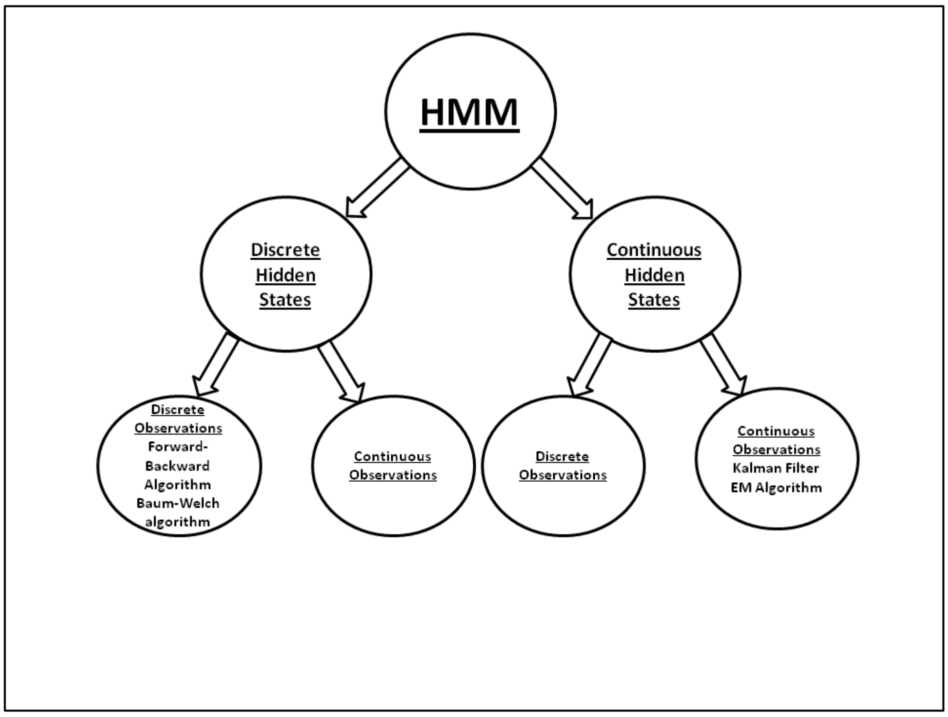


Figure 2.5: Overall breakdown for types of HMM states and observations

In this research, we will use the HMM with discrete hidden states and continuous observations. But for the purpose of clarifying how the HMM works, we will review the most widely used HMM type which is the discrete hidden state and a discrete observations.

2.4.1 Discrete hidden states with discrete observations

We will reveal the elements of a HMM and how to model the hidden state sequence given the observation symbol sequences. Then will review the basic questions concerned with learning an HMM from data and the algorithms used to solve them.

First, consider the HMM discrete hidden state sequence $x_l = \{x_1, x_2, \dots \dots., x_l\}$, where l is the number of hidden state symbols. The probability of transiting from a hidden state to another is modelled by the transition matrix $A = \{a_{ij}\}$ where $\sum_{j=1}^{l} a_{ij} = 1$. The initial hidden state probability is modelled by $\pi = P(x_1 = i)$ which means the probability of being in state i at time 1. Now consider the HMM discrete observation sequence $Z = \{z_{1,}z_{2,}, \dots \dots, z_N\}$ where N is the total length of the observation sequence. These observation sequence are represented as symbols of total m which are modelled by the observation probability matrix $B = \{b_j(k)\} = P(observation\ k\ at\ time\ t | x_t = j)$ where $1 \leq k \leq m$. These probabilities also sum to 1, $\sum_{j=1}^{m} b_j(k) = 1$. Putting the discrete observations and hidden state together into a HMM is specified by $\lambda = (A, B, \pi)$. See Figure 2.6:

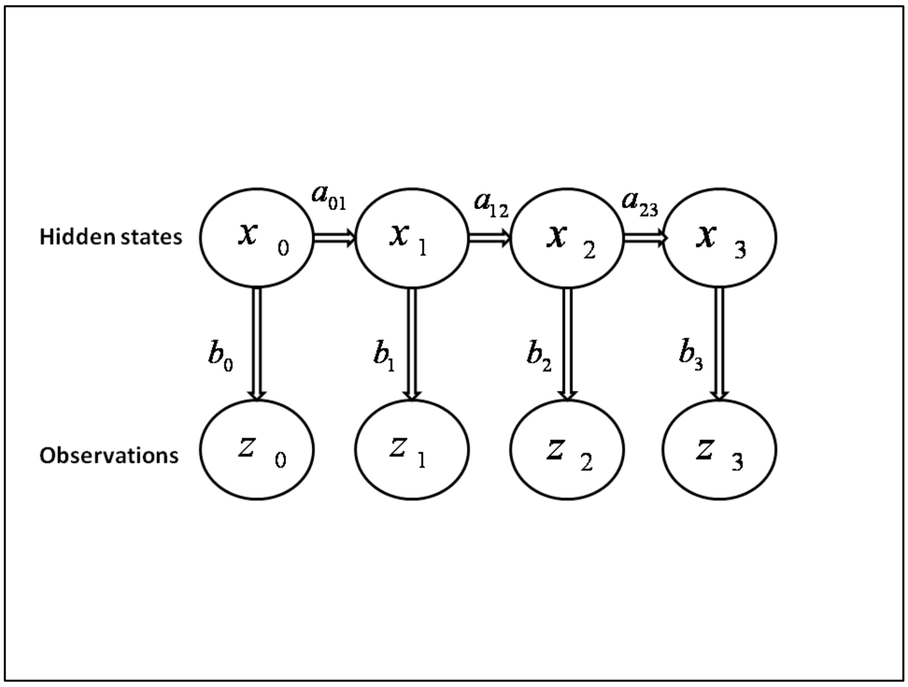


Figure 2.6: HMM which shows the observations which were generated from the hidden states using two probability matrices A and B

There are three basic problems in the HMM and they are listed below (Rabiner, 1989):

1. Given the model $\lambda = (A, B, \pi)$ and the set of observations $Z = \{z_{1,}z_{2,}, \dots \dots, z_N\}$, what is the probability that the model generated these observations $P(Z|\lambda)$?

   Solution: Forward- Backward algorithm
2. Given the model $\lambda = (A, B, \pi)$ and the set of observations $Z = \{z_{1,}z_{2,}, \dots \dots, z_N\}$, what is the most likely state sequence that generated these observations?

   Solution: Viterbi algorithm
3. Given the model $\lambda = (A, B, \pi)$ and the set of observations $Z = \{z_{1,}z_{2,}, \dots \dots, z_N\}$, how to maximize $P(Z|\lambda)$ (probability of the observation sequence) by tuning the parameters (A, B, π)?

   Solution: HMM learning with the Baum-Welch algorithm

Since our research only needs the forward-backward algorithm and the Baum-Welch algorithm we will skip the viterbi algorithm and review problems 1 and 3 in the next section:

2.4.2 Forward-Backward Algorithm

To solve the first problem, a probabilistic method with low complexity (computational time) is needed. Such a method is known as the α- pass or the forward pass. This pass computes the probability of the partial observation sequence $\{z_1, z_2, \dots, z_t\}$ where $1 \leq t \leq N$:

$$\alpha_t(i) = P(z_1, z_2, \dots, z_t, x_t = i\,|\lambda)$$

A recursive algorithm can be used to compute $\alpha_t(i)$ in terms of the $\alpha_{t-1}(i)$:

$$\alpha_t(i) = b_j(z_t) \sum_{i=1}^{l} \alpha_{t-1}(i) a_{ij}$$

If α is initialized as $\alpha_1(j) = \pi_j b_j(z_1)$, then probability of the observation sequence is:

$$P(Z|\,\lambda) = \textstyle\sum_{i=1}^{l} \alpha_N\,(i)$$

This forward pass algorithm has a complexity of l^2N whereas a naive method will take upto l^N.

Similar to the α-pass, is the backward recursion is called the β-pass. In the backward recursion, the probability of the partial observation sequence $\{z_{t+1}, z_{t+2}, \dots, z_N\}$; conditioned on the state at time t is computed as:

$$\beta_t(i) = P(z_{t+1}, z_{t+2}, \dots, z_T|x_t = i\,, \lambda)$$

The recursive form to calculate $\beta_t(i)$ is:

$$\beta_t(i) = \sum_{i=1}^{l} \beta_{t+1}(j)\ \ a_{ij}b_j(z_{t+1})$$

Having found $\alpha_t(i)$ and $\beta_t(i)$ from the forward and backward algorithms, they may be combined to calculate the probability of the observation sequence:

$$P(Z|\lambda) = \sum_{i=1}^{l} \alpha_t(i)\,\beta_t(i)$$

The *posterior* probability, defined as $\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{P(Z|\lambda)}$, is the probability of being in state i at time t given the observation sequence and the model. As a result, the most likely state i at time t is the i for which $\gamma_t(i)$ is maximum.

2.4.3 HMM learning (Baum-Welch Algorithm)

Adjusting the parameters of the model to determine the best fit to the observation sequences is part of the HMM learning process. To solve problem 3 of the HMM, which is re-estimating the parameters (A, B, π) to maximize the observation sequences, we use the forward-backward algorithms to re-estimate the α and β terms depend on the current estimates of the (A, B, π), after which the parameters are re-estimated. This recursion of the Expectation Maximization algorithm (Dempster et al.,1977) is known as the Baum-Welch Algorithm (Rabiner,1989). A good feature of this algorithm is that it will converge eventually, meaning that when we re-estimate new parameters the likelihood of the data will not decrease, however, It will continue to increase unless it reaches a local maxima.

To begin, a new variable is introduced known as the di-gamma:

$$\gamma_t(i,j) = P(x_t = i, x_{t+1} = j | Z, \lambda) \qquad (3)$$

This says that $\gamma_t(i,j)$ is the probability of being in state i at time t and in state j at time t+1. Then, via substitution in the $\alpha - \beta$ equations above we get:

$$\gamma_t(i,j) = \frac{\alpha_t(i) a_{ij} b_j(z_{t+1}) \beta_{t+1}(j)}{P(Z|\lambda)}$$

Deriving gamma from digamma we get:

$$\gamma_t(i) = \sum_{j=1}^{l} \gamma_t(i,j)$$

Now that we calculated gamma and digamma, the parameters are adjusted as follows:

- Re-estimating initial state:

$$\pi_i = \gamma_1(i)$$

- Re-estimating the state transition matrix A:

$$a_{ij} = \frac{\sum_{t=1}^{N-1} \gamma_t(i,j)}{\sum_{t=1}^{N-1} \gamma_t(i)}$$

- Re-estimating the observation transition matrix B:

$$b_j(k) = \frac{\sum_{\substack{t=1 \\ z_t=k}}^{N-1} \gamma_t(j)}{\sum_{t=1}^{N-1} \gamma_t(i)}$$

Once the forward-backward algorithm is done, the model's parameter are then re-estimated and updated. In the next section, we will explain another tool used in our research which is a special case of the HMM called the Kalman filter (continuous hidden states and continuous observations).

2.5 Kalman Filter and Smoother

2.5.1 Background

The Kalman filter is a recursive mathematical state-estimator tool introduced by Rudolf Kalman in 1960 (Welch and Bishop,2006). The purpose of the filter is to observe noisy outputs from an observation system such as a sensor or radar and estimate the real values for them (hidden states from the state system) and their associated uncertainties or "error covariance". The Kalman filter first predicts these hidden states and their uncertainties for time $t + 1$ given the estimated state at time t . When the time increments to $t + 1$, the filter will receive a new observation from the radar or sensor, this new observation will correct the prediction and minimize uncertainties in the estimate of the hidden state. The Kalman filter's output will the optimal state estimate with a smaller zone of uncertainty around it. see figure 2.7:

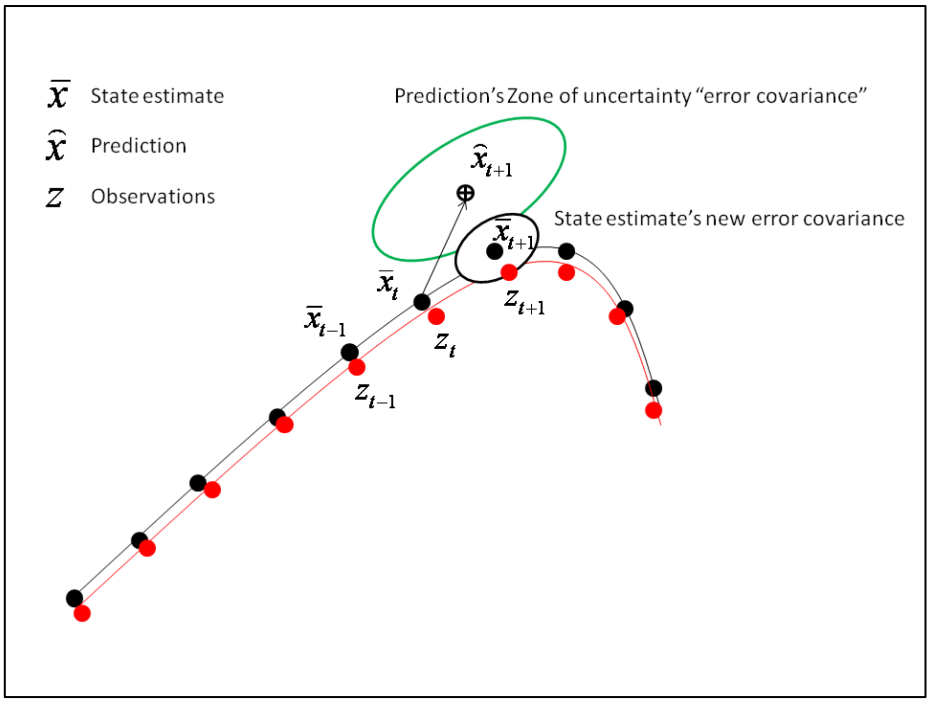


Figure 2.7: The Kalman Filter tracking predicting and correcting. The next location $t+1$ is predicted along with its zone of uncertainty (green ellipse) which were generated from the state estimate at time t and corrected given the next observation.

Figure 2.7, shows how the Kalman filter tracks a moving object such as an aircraft and finds the optimal state estimate by correcting the prediction given the new observation z_{t+1}. The observation will pull the prediction $\hat{x}_{t+1}$ closer to its real value to have a new state estimate $\bar{x}_{t+1}$.

To use standard Kalman filter the observations and the hidden state should be assumed to be linear and normally distributed. There are several ways to represent the transition process from one state to another, however, for the Kalman filter we need a linear dynamical system where the state model is:

$$x_t = Ax_{t-1} + w_t$$

The probability form of the state transition is:

$$P(x_t|x_{t-1}) \equiv P(x_t|\, Ax_{t-1}, Q)$$

where x_t is $1{\times}d$ real valued state estimate vector at time t, d describes the number of parameters in the state, A is $d{\times}d$ state transition matrix which relates the previous state to the current state and w_t is a random white noise with mean 0 and error covariance Q, $w_t \sim \mathcal{N}(0, Q)$.

The observation model for the linear dynamical system is defined as:

$$z_t = Hx_t + v_t$$

and its probability form is:

$$P(z_t | x_t) \cong P(z_t | Hx_t, R)$$

where z_t is $1\times n$ real valued observation vector at time t, n is the number of parameters in the observation, H is the $n\times d$ observation matrix, it relates the hidden state to the observation and v_t is a random white noise with mean zero and error covariance R, $v_t \sim \mathcal{N}(0, R)$.

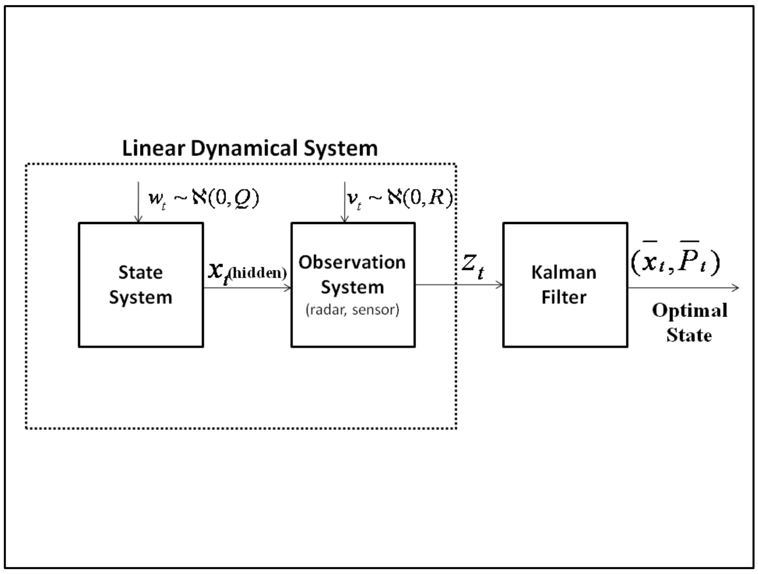


Figure 2.8: Kalman Filter using and the linear dynamical system. The Kalman filter gets the observations z with noise v from a Radar or Sensor. These observations were generated from the state system and perturbed by white noise w

Figure 2.8 shows the overall process for generating the observations and the hidden states at time t using a linear dynamical system perturbed by Gaussian noise. These observations are then used in the Kalman filter to correct the prediction calculated at time $t - 1$.

2.5.2 Kalman Filter Equations

Given the two models for our linear dynamical system:

$$x_t = Ax_{t-1} + w_t$$

$$z_t = Hx_t + v_t$$

x_t, w_t, v_t are unknown to the Kalman filter because they were generated from a LDS such as in figure 2.8 where the dotted box means that these systems (state and observation) are running in the background and are unknown to the Kalman filter. Therefore, only the generated observations are visible to the filter. In the filtering process we would like to compute the conditional hidden state probability distribution:

$$P(x_t|z_{1:t}) \cong \mathcal{N}(Ax_{t-1}|z_t, R)$$

The Kalman filter has two main processes: state prediction and state correction. The state prediction is the process when the Kalman filter uses the previous state estimate x_{t-1} and its error covariance or "uncertainty" $\overline{\mathrm{P}}_{\mathrm{t-1}}$, to predict the next location and its error covariance at time t:

$$\hat{x}_t = A\bar{x}_{t-1}$$

$$\hat{P} = \mathrm{A}\,\overline{\mathrm{P}}_{\mathrm{t-1}}\,\mathrm{A}^{\mathrm{T}} + \mathrm{Q}$$

Next, when the Kalman filter receives a new observation from the observation system at time t, the Kalman filter uses this observation z_t to correct the prediction and its error covariance J. This is achieved by calculating the residual r and its error covariance between the observation and the prediction:

$$r = z_t - H\hat{x}_t$$

$$J = (\,H\,\hat{P}_t\,H^T + R)^{-1}$$

Given the residual error covariance J, the Kalman filter then calculates Kalman gain K which is the average error covariance between the observation and the prediction used to minimize the state estimate error covariance:

$$K = \hat{P}_t H' J^{-1}$$

Finally, the Kalman filter adds the prediction location and the averaged prediction to correct the state and minimises its error covariance:

$$\bar{x}_t = \hat{x}_t + K(z_t - H\hat{x}_t)$$
$$\bar{P} = (I - KH)\hat{P}_t$$

The state estimate $\bar{x}_t$ lies now somewhere between the observation location and the prediction and its error covariance will be smaller than the state prediction's error covariance. Detailed information about the Kalman filter equations can be found in (Kalman, 1960). Algorithm 2.1 shows the two Kalman filter processes and the way they alternate:

Algorithm 2.1: Calculates $P(\bar{x}_t|z_1^t)$

```
Require: Z                          Observations matrix
Require: Q                          State error covariance
Require: R                          Observation error covariance
Require: A                          State transition matrix
Require: H                          Observation transition matrix
Require: N                          Total length of the observations
$\bar{x}_1 = z_1$
$\bar{P}_1 = HRH'$
$t = 2$
While $t \leq N$ do
                                    Beginning of Prediction Phase
    $\hat{x}_t = A\bar{x}_{t-1}$    The prediction at time t given the
                                    previous state estimate
    $\hat{P}_t = A\bar{P}_{t-1}A' + Q$   The error covariance for the prediction
                                    (the uncertainty)

                                    Beginning of Corrections Phase
    $e = z_t - H\hat{x}_t$          The residual or the innovation between
                                    the observation and the prediction

    $J = H\hat{P}_t H' + R$         The innovation's error covariance matrix
                                    The Kalman gain is the weighted average
    $K = \hat{P}_t H'/J$            error for the prediction and observation
    $\bar{x}_t = \hat{x}_t + Ke$                    The new state estimate weighted by the gain
    $\bar{P}_t = (I - KH)\hat{P}_t$                 The state estimate error covariance matrix weighted by the gain
    $t = t + 1$
end while
```

Algorithm 2.1 shows Kalman filter as a recursive estimator which only need the previous state estimate and the current observation to estimate the state.

During the prediction phase, it projects the location of the next state estimate using the state transition matrix A. It also estimates the prediction uncertainty by transiting the previous state estimate error covariance in addition to the current system error to the next location. In the correction phase, the Kalman filter has new observation which helps the filter to correct the state prediction and its covariance. The residual e_t and its error covariance J will correct the prediction and minimise the prediction error covariance. The smaller the residual the more confident we are about the prediction, when the residual equals zero that mean the new state estimate will be the prediction. Next step, the prediction will use this state estimate to predict in the future and so on.

2.5.3 Derivation of Kalman filter errors

We define $\hat{x}_t$ to be our prior that is the prediction and its error $\hat{e} = x_t - \hat{x}_t$, the difference between the original hidden state and the prediction and the observation sequence $z_{1:t} = \{z_{1,}z_{2,}, \dots \dots, z_t\}$.The error covariance matrix for the prediction is defined as:

$$\hat{P}_t = E[\hat{e}\hat{e}'] = E[(x_t - \hat{x}_t)(x_t - \hat{x}_t)|z_{1:t}]$$

$$= E[(Ax_{t-1} + w_t - A\bar{x}_{t-1})(Ax_{t-1} + w_t - A\bar{x}_{t-1})']$$

$$= A\bar{P}_{t-1}A' + Q$$

Same process is used for the posteriori error "state estimate error" $\bar{e} = x_t - \bar{x}_t$, the difference between the true state and the estimated state. The error covariance matrix for the state estimate is defined:

$$\bar{P}_t = E[\bar{e}\bar{e}^T|z_1^t] = E[(x_t - \bar{x}_t)(x_t - \bar{x}_t)'|z_1^t]$$

$$= E[(Ax_{t-1} + w_t - \hat{x}_t + K(z_t - H\hat{x}_t))(Ax_{t-1} + w_t - \hat{x}_t - K(z_t - H\hat{x}_t))']$$

$$= (I - KH)\,\hat{P}_t$$

2.5.4 Kalman Smoother

In the filtering process the objective is to find the optimal state estimate given the observations up to time t that is: $P(x_t|z_{1:t})$ assuming that both observations and the hidden state are normally distributed. In the Kalman smoother, the objective is the same except to find the optimal state estimate at time t given the observation from 1 to N, where N is the total length of observations. It starts to estimate from the end walking its way back to the state estimate at time t and when combining the filtering and smoothing it ensures better estimation for that state. This inference combination is called the forward-backward passes and they are analogous to the alpha beta-passes defined in the HMM except that the hidden states are continuous in Kalman filter.

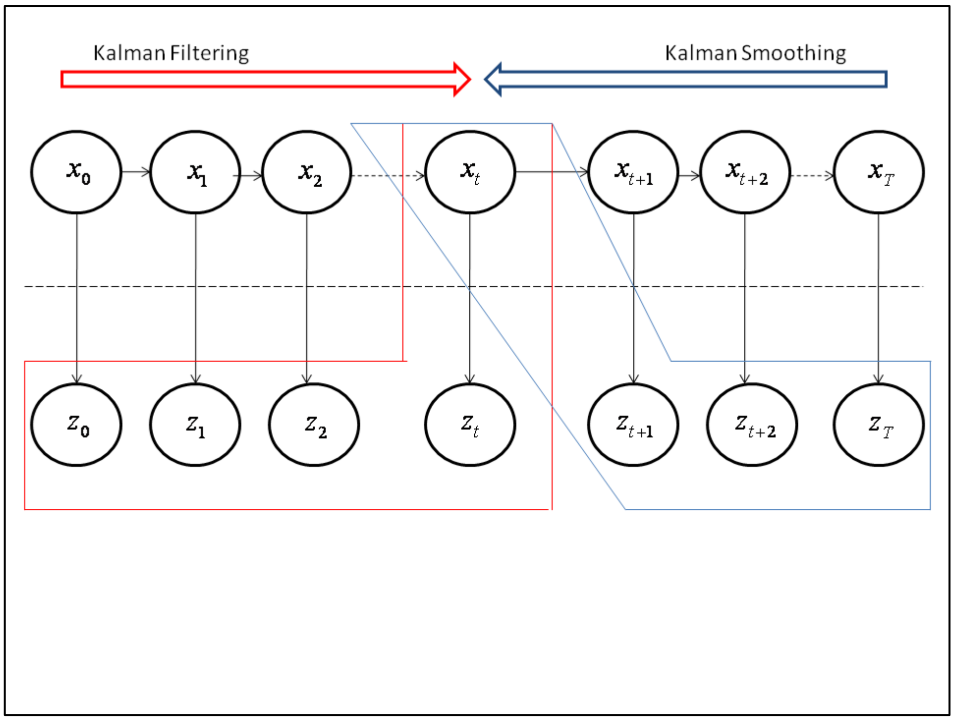


Figure 2.9: Kalman Filtering and Smoothing at time t the kalman filter estimate the statee $\bar{x}_t$ given the observation time=0 to time=t. The Kalman smoothing estimates the state $\bar{x}_t$ given all the observation from time=T to time=0

Figure 2.9 illustrates the idea of the forward backward inference process, the forward pass estimates $P(\bar{x}_t|z_1^t)$ and the backward pass estimates $P(\bar{x}_t|z_{t+1|N})$ which are combined to form $P(\bar{x}_t|Z)$ where $Z = \{z_1, z_2, \dots, z_N\}$.

2.5.5 Kalman Smoother Equations

Starting from the last time step N, we initialize the smoothed state $\tilde{\boldsymbol{x}}$ and its covariance $\tilde{\boldsymbol{P}}$ with the state estimate and its covariance at that time:

$$\begin{aligned}\tilde{x}_N &= \bar{x}_N \\ \tilde{P}_N &= \bar{P}_N\end{aligned}$$

and given the Kalman gain at time K_N, we initialize the averaged smoothed error covariance between time step $N-1$ and $N-2$ to become:

$$\widetilde{PP}_N = (I - K_N H)A\bar{P}_t$$

The smoothed state and its error covariance are then calculated using the Kalman filter results with the following recursive equations, where $t = N-1, N-2, \dots, 1$:

$$J_t = (\bar{P}_t A')/\hat{P}_{t+1}$$

$$\tilde{x}_t = \bar{x}_t + (J_t(\tilde{x}'_{t+1} - A\tilde{x}'_t))'''$$

$$\tilde{P}_t = \bar{P}_t + J_t(\tilde{P}_{t+1} - \hat{P}_{t+1})J'_t$$

Algorithm 2.2 shows how to recursively smooth out the state estimates and its error covariance:

Algorithm 2.2: Calculates $P(\bar{x}_t|z_{t+1|N})$

```
Require: $\bar{P}$                 State estimate error covariance matrix
Require: $\bar{x}$                 State estimate
Require: $\hat{P}$                 State Prediction error covariance matrix
Require: A                         State transition matrix
Require: H                         Observation matrix
Require: N
$t = N$-1                          Initialize $t$ with total number of
                                   observation N-1 (starting from the back)
$\tilde{x}_N = \bar{x}_N$          Initialize the last element of the
                                   smoothed estimate with the last state
                                   estimate
$\tilde{P}_N = \bar{P}_N$          Initialize the last element of the
                                   smoothed covariance with the last state
                                   covariance
$K = \hat{P}_t H^T/(H\hat{P}_N H^T + R)$
$\widetilde{PP}_N = (I - KH)A\bar{P}_t$

while  $t \neq 0$ do
    $J_t = (\bar{P}_t A')/\hat{P}_{t+1}$
    $\tilde{x}_t = \bar{x}_t + (J_t(\tilde{x}'_{t+1} - A\tilde{x}'_t))'$
    $\tilde{P}_t = \bar{P}_t + J_t(\tilde{P}_{t+1} - \hat{P}_{t+1})J'_t$
    $t = t - 1$
end while
$t = N - 2$                        Estimates $\widetilde{PP}^{t-1}_{t-2}$ that is the best
while $t \neq 0$                   estimate of the error covariance when at
                                   time $t$-1 and $t$-2
    $\widetilde{PP}_t = \bar{P}_t J'_{t-1} + J_t(\widetilde{PP}_{t+1} - A\bar{P}_t)J'_{t-1}$
    $t = t - 1$
end while
```

In summary, we reviewed both algorithms for the Kalman filter and smoother. They are an efficient tool to track and estimate the location of a moving object that contains Gaussian noise; where it filters out this noise in the observations which were taken from the radar and finds the optimal state estimate. Because our research needs a reliable tracking tool with minimum errors we use the Kalman filter/smoother as our tracking tool.

2.6 Literature Review on Conflict Detection

In controlled airspace (CAS), each aircraft has a zone of separation that no other aircraft should penetrate. Should this zone be penetrated, the event is termed a “conflict”. The size of the protected zone defined for aircraft varies but the standard amount of separation each aircraft should have is 5 nautical miles horizontally and no less than 1000ft vertically. These conflicts are a major concern for all air traffic controllers for which they have to constantly monitor the airspace and avoid any possible conflict, see figure 2.10 below:

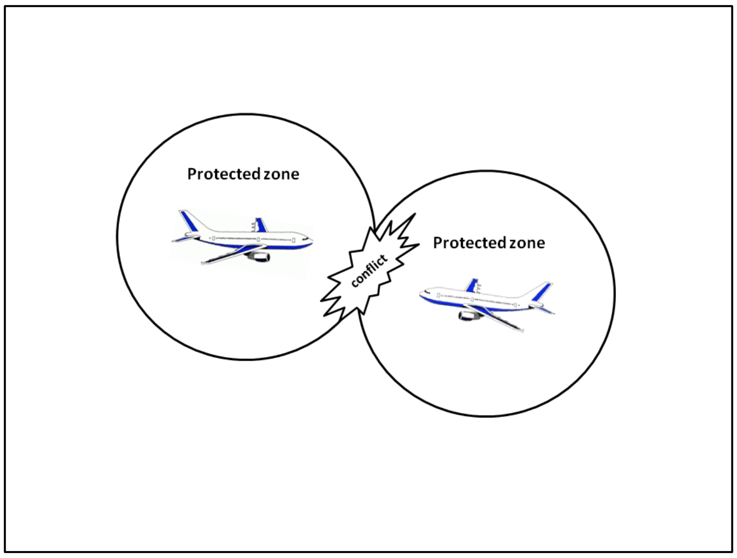


Figure 2.10: An example of a conflict where two aircraft lose their minimum separation

Current conflict detection systems cannot warn ATC of a conflict more than 2 minutes away which gives ATCs a small amount of time to resolve it. This issue of conflict detection has been an attractive field of study by researchers, some have developed new models with their own conflict detection algorithms and others have optimised current models used by ATCs.

Different approaches to determine the probability of a conflict and conflict resolution have been introduced: one approach adopted was by Yang and Kuchar (Yang and Kuchar, 1997) who created an alerting system for free flight that uses Monte Carlo simulations (MC) to estimate the probability of a conflict of traffic encounters over time. Because it is a free flight alerting system, they assumed that there is a data-link between aircraft to communicate with each other in the airspace. The idea of the data-link is to collect other aircraft's information in the airspace, such as current state and future trajectory. The current state information for both aircraft contains speed, heading and altitude which are then fed into the MC engine as the initial state. Each MC run projects a path for both aircraft and predicts if a conflict is ahead. The prediction process issues an alert when the host aircraft's protected zone is violated by an intruder aircraft. The probability of a conflict is calculated as follows:

$$\mathrm{P(C)} = \frac{\text{\# of protected zone intrusions}}{\text{\# of MC runs}}$$

The protected zone was divided into four stages (where 1 means a remote intruder whereas 4 means nearby intruder and Air Traffic Controllers should take control from here). The size of the protected zone is a trade off between the successful alerts (SA) and unnecessary alerts (UA) and it was examined by using a System Operative Characteristic (SOC). The level of alert in each of the first 3 stages varies and has a number of manoeuvres available (N). As the intruder aircraft gets closer to the host aircraft the value of N decreases.

Using Monte Carlo simulations with on-line applications are computationally expensive because prediction models are limited within time constraints. To reduce the amount of computation, Yang and Kuchar (1998) proposed incorporating intent information into the Monte Carlo simulation engine. Their method was that by knowing the waypoints of two aircraft, they can create a series of straight segment lines between these waypoints, where each endpoint represents a change of heading or speed. Then check if the host's segment line intersects the intruder's trajectory line.

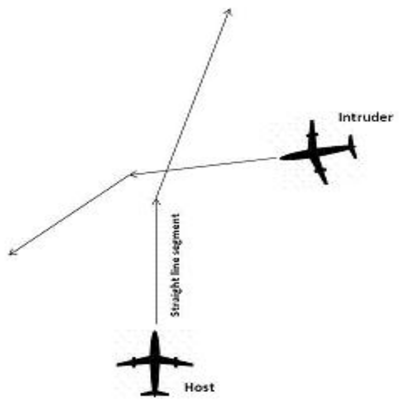


Figure 2.11: Monte Carlo's new output (straight line segment) approximation, based on (Yang and Kuchar, 1998)

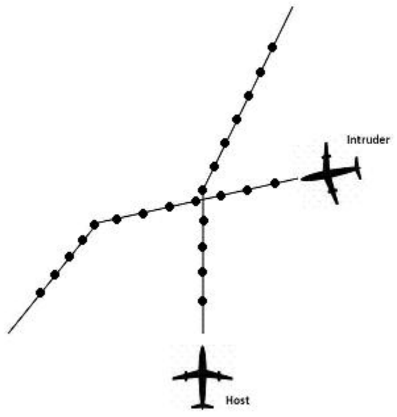


Figure 2.12: Monte Carlo's old output (increments of time), based on (Yang and Kuchar, 1998)

The Monte Carlo simulation engine is fed with intent information, current state, protected zone size and uncertainties such as tracking errors, manoeuvring characteristics then outputs a probability of a conflict $P(C)$. The performance was calculated using the System Operating Characteristic (SOC) by comparing the P(C) both cases :when the intent information is assumed to be supplied and when its not (nominal case) the results are showing in figure 2.13:

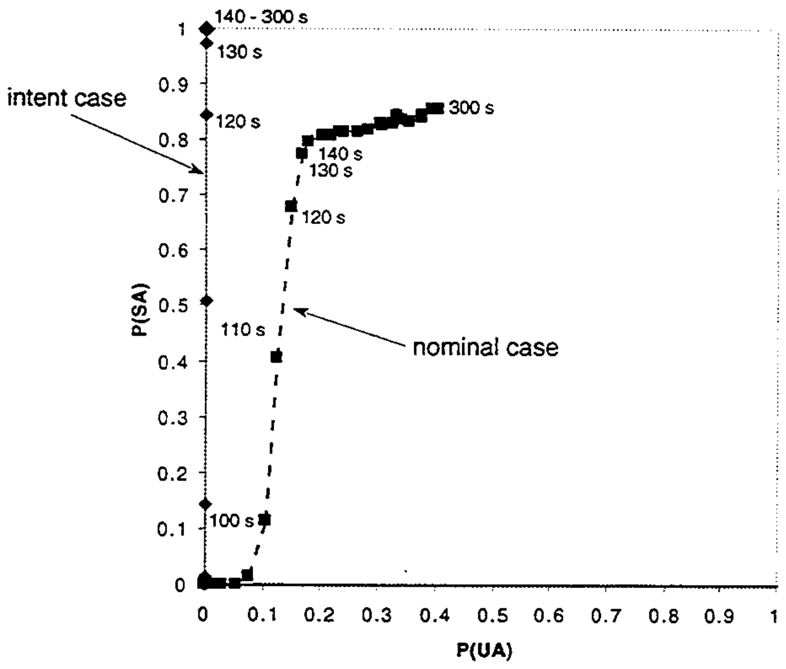


Figure 2.13: SOC curve taken from (Yang and Kuchar ,1998)) where it shows the P(C) for both cases

As the time increase, the aircraft is approaching another aircraft and the probability of a conflict is increased for both cases. However, when the intent information is available, the alert is issued earlier to the air traffic controller as appose to the nominal case. Meaning when it is 120 seconds the P(SA)=0.85 where as in the nominal case the P(SA)=0.69.

Another study was conducted in (Paielli et al., 1997) to predict a conflict in free flight; their method is applied to two aircraft travelling along a straight line with constant errors. They modelled the trajectory prediction errors as randomly distributed based on the live air traffic data and combined covariance error pairs into a single covariance error relative to the position, this was done because common errors cancel each other. The authors defined the two aircraft “Stochastic” and “Reference” as S and R respectively, then to combine both aircraft covariance errors,

$$\Delta \mathrm{x} \equiv \mathrm{x}_S - \mathrm{x}_R \rightarrow \Delta \hat{\mathrm{x}} \equiv \hat{\mathrm{x}}_S - \hat{\mathrm{x}}_R \rightarrow \Delta \hat{\mathrm{e}} \equiv \hat{\mathrm{e}}_S - \hat{\mathrm{e}}_R$$

$$\mathrm{M} \equiv \mathrm{Cov}(\Delta \hat{\mathrm{e}})$$

Where x is the aircraft position, $\hat{\mathrm{x}}$ position prediction, $\hat{\mathrm{e}}$ prediction error and M is the combined prediction error covariance. Figure 2.14 shows that given the combined error covariance, they transform it to an error ellipse centred on the stochastic aircraft and the conflict zone is centred on the reference aircraft using the coordinate transformation to present an analytic solution.

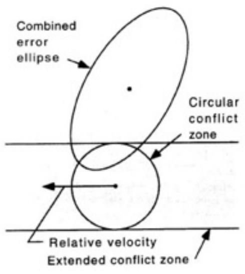


Figure 2.14: Encounter geometry, taken from (Erzberger et al,1997)

The conflict probability prediction is the area under the combined error ellipse within the extended conflict zone. The way to resolve this conflict, is to move the extended conflict zone away from the centre of the error ellipse by increasing velocity of the aircraft (Erzberger et al.,1997). They compared their output with a set of a conflict encounters and found that they are similar, moreover, the maximum difference between them was 1.8% only 5 out of 72 were over 1.

One approach was to predict both mid-range (10-20 minutes) and short-range conflicts (2 minutes) using different critical measures of two aircraft encounters, which is the maximum instantaneous probability of a conflict (Prandini et al.,2000)

That is,

$$C(\gamma) = \max_{t \in [0,T]} P(C_t)$$

where γγ, is the flight plans for all aircraft, C is the threshold and $C(\gamma\gamma)$ is the critical measure. The critical measure is equal to the maximum probability of a conflict $P(C)$ over time horizon T (20 minutes ahead). Every time γ changes, the algorithm computes $C(\gamma)$ then checks if it is below the threshold C, if not, a conflict is issued. This method would give the exact solution, however, the major concern is the computation amount of the $C(\gamma)$ which makes it unsuitable for online applications, thus forcing the researcher to look for an approximated solution which is a randomised estimate of $C(\gamma)$. The approximated solution is generate a random integer M and interval of time t ∈ [0,T] $i = 0,1,\dots,M$. If $PC(t_i) > C(\gamma)$ then $C(\gamma) = PC(t_i)$.

The performance of each algorithm was compared after creating the trajectories of aircraft using stochastic differential equations, computing the critical measure $C(\gamma)$ then computed and plot the probability of false alerts and successful alerts using the SOC (System Operative Characteristic) curve using Monte Carlo simulations. The algorithm proposed by the authors increased the probability of successful alerts and decreased false alerts by 16.4% and this is the case when the flight path is in a zig-zag pattern. The results P(SA)=0.778 and P(FA)= 0.164 as appose to the results in (Erzberger et al.,1997) where P(SA)=0.727 and P(FA)=0.264, the SOC curve below shows the comparison.

Over all, the authors were able to present better results in terms of early conflict detection and resolution. They however, assume the availability of intent information and the amount of uncertainties are known. The relationship between conflict detection and infringement detection systems is that they are both a ground based safety nets which warns ATCs for a possible conflict whether it is between two or several aircraft (conflict detection system) or between an unauthorised aircraft and a controlled airspace boundary (infringement detection system).

Because the current infringement detection system used by NATS has a limitation which is the lack of early warning time or predicting future infringements more accurately, our main focus will be to investigating this system to possibly predicting future infringements more accurately.

2.1 Summary

In this chapter we introduced definitions and tools which will be used throughout our research. We first defined what an airspace is and how it is monitored by ATCs using radars. Another tool used to aid ATCs, called the Controlled Airspace Infringement Tool, was introduced which warns ATCs of any infringements; however this tool has a current limitation to it which is the warning time. From this problem, we drew our research question which is: can we build a probabilistic infringement detection model that can warn ATCs for future infringements accurately? To answer this question we reviewed two important tools from the literature which will be used mainly for our tracking and learning process ; the Kalman filter and HMM. Research has been done on this type of model which predicts future infringement and assigns a probability of infringement to it and it will be presented in the next chapter.

3 Probability of Infringement

As we have seen in the previous chapter, a system was developed called Controlled Airspace Infringement Tool (CAIT) to warn air traffic Controllers (ATC) about any current infringements. However, by using this system ATCs can only react to resolve a possible conflict caused by an infringing aircraft (intruder) when the aircraft is already inside the controlled airspace, which gives them minimum time to seek a quick maneuver to avoid mid-air collision. Therefore, it would better to improve this safety feature by predicting the future locations of the aircraft in UCAS accurately and thereby uncovering the probability of infringement. In Chapter 3 we will discuss and extend a Probabilistic Infringement Detection Tool proposed in (Macdonald, 2009) and show how it can benefit this research. We will also more closely consider the Monte Carlo sampling and the possibility of developing a hybrid method to estimate the probability of infringement.

3.1 Current Probabilistic Infringement Detection Tool

Macdonald in (2009) proposed probabilistic prediction model which is used to find the probability of an infringement at a given instant. His method uses the aircraft prediction $\hat{x}$, its error covariance $\hat{P}$ (zone of uncertainty around the prediction) and calculates the shortest distance d from this prediction to the CAS boundary C to estimate the probability of infringement. See figure 3.1:

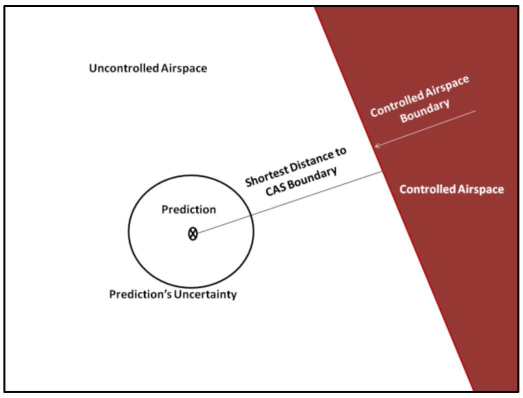


Figure 3.1: Shortest Distance Method

Macdonald first assumes that the CAS boundary is a straight line with no bends or corners, and then he calculates the shortest distance from that prediction to the boundary. Next he uses the error function to find the probability of infringement given the shortest distance from prediction to the CAS boundary and the prediction's location uncertainty or "error covariance". This covariance is consisted of two variances showing in figure 3.2:

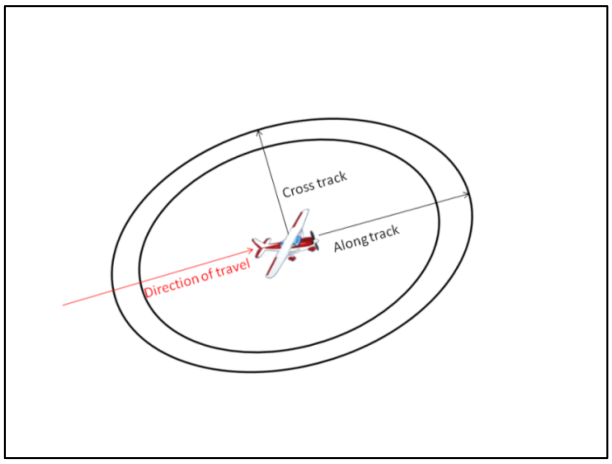


Figure 3.2: Aircraft cross track and along track errors (σ_x,σ_y)

The probability that the prediction is inside CAS can be defined below:

$$P(I) = \int_{x \in CAS} \mathcal{N}(x|\hat{x}, \hat{P})\, dx$$

where $\hat{x}, \hat{P}$ are the prediction and its error covariance respectively. An example is shown in figure 3.3 for more illustration:

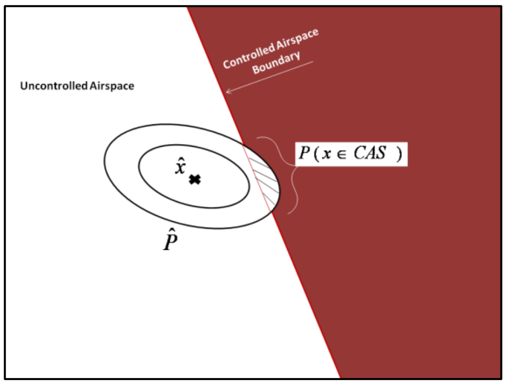


Figure 3.3: The probability that the aircraft prediction is inside or after CAS

To find the probability of infringement Macdonald first proposed three cases, they are:

1. Simple 2D multivariate model where the errors are assumed to be equal ($\sigma_{cross} = \sigma_{along}$).
2. Advanced 2D multivariate model, ignoring the heights and the errors are not equal ($\sigma_{cross} \neq \sigma_{along}$), same as in figure 3.3.
3. 3D model which includes the altitude and the errors are not equal

The simple 2D model, where the errors are equal the probability of infringement $P(I)$ is calculated as:

$$P(I) = \frac{1}{2} - \frac{1}{2} \operatorname{erf}\left(\frac{d}{\sigma\sqrt{2}}\right)$$

where d is the perpendicular shortest distance from the prediction to the CAS boundary, σ is the error $\hat{P} = \sigma^2 I$ where I is the identity matrix.

The distance is calculated using the prediction location $\hat{x} = (x, y)$, and a CAS boundary which is defined by a line with end points (x_1, y_1),(x_2, y_2) the shortest distance is:

$$d = \frac{|[(x - x_1)(y_2 - y_1)] - [(x_2 - x_1)(y - y_1)]|}{\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}}$$

In the case of advanced 2D Multivariate model, in order to integrate $P(x \in CAS)$, the problem transformed so that the errors covariance is isotopic ($\sigma_{cross} = \sigma_{along}$). Figure 3.4 shows this transformation:

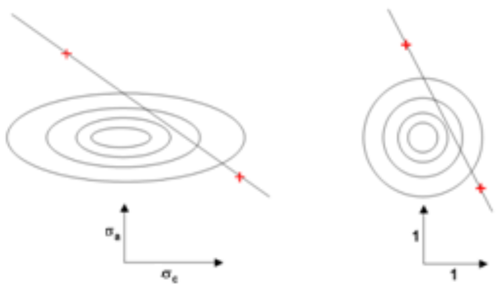


Figure 3.4: Standardizing the Advanced 2D problem to become the Simple 2D problem, taken from (Mcdonald,2009)

The new probability of infringement now where d is the distance after the standardisation becomes:

$$\mathrm{P(I)} = \frac{1}{2} - \frac{1}{2}\,\mathrm{erf}\left(\frac{d^*}{\sqrt{2}}\right)$$

where d^* is the distance from the transformed mean prediction to the transformed boundary. The last case the author proposed is the 3D model, the equation is similar to the advanced 2D model, but this time he also standardised the height (the base level) in addition to the location (x,y) to become (0,0,0) and applied the equation:

$$\mathrm{P(I)} = \widetilde{\mathrm{P}}(\mathrm{I}) \left(1 - \Phi(\frac{\mathrm{BL}}{\sigma_\mathrm{h}})\right)$$

where $\tilde{P}I) = \mathrm{P(I)}$, BL is the new standardised base level and σ_h is the height error.

The only disadvantage of these methods is that this model assumes that the boundary of the CAS is a straight line without bends or corners where in reality they do. In this research, the method we will use is the advanced 2D model to find the probability of infringement in conjunction with another method which we will present next.

3.2 Probabilistic Infringement Detection Tool

Our method assumes that the CAS boundary can take any shape which is composed of straight line segments. Figure 3.5 shows the different shapes of CAS boundaries projected into 2-D from different heights.

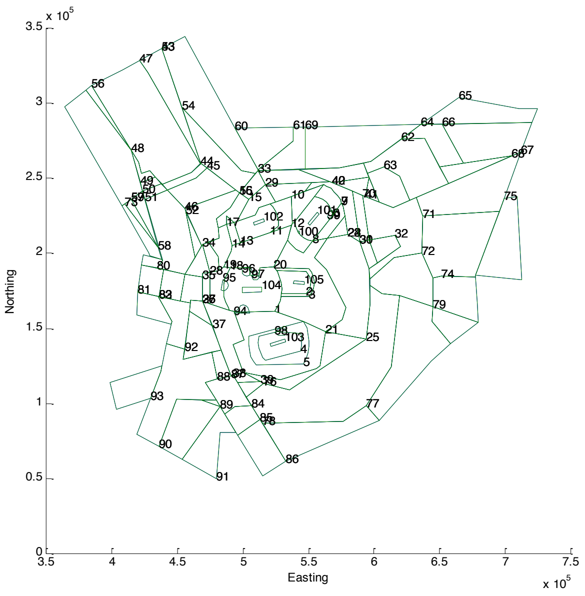


Figure 3.5: 104 different shapes of CAS boundaries ignoring the heights over the UK. Source: NATS

We can track the aircraft in three dimension space with a location point (x, y, z) which will be a good future work recommendation, this can be done by adding a third parameter to both the prediction location and its error covariance, in the mean time, we will assume that our model tracks the aircraft in two dimension with location point (x, y) and its errors are not necessary equal $(\sigma_{cross} \neq \sigma_{along})$. We will use a combination of methods to calculate the probability of infringement; the advanced shortest distance 2D model and Monte Carlo (MC) sampling.

3.2.1 Monte Carlo Sampling

MC sampling method uses the prediction and its error covariance in the same way as the shortest distance method to find the $P(I)$ but here the MC sampling draws a random number of samples N from the prediction error covariance and then calculates the fraction of samples which fall inside the CAS. To estimate $P(I)$:

$$P(I) \approx \frac{\#\ of\ samples\ that\ fall\ inside\ CAS}{Total\ number\ of\ samples}$$

The MC sampling performs well in both cases when the aircraft prediction is near a CAS boundary which is either a straight line or has a sharp corner. Figure 3.6 show an example of a prediction $\hat{\mathbf{x}}$ and its error covariance $\hat{P}$ as it is about to approach a complicated controlled airspace boundary.

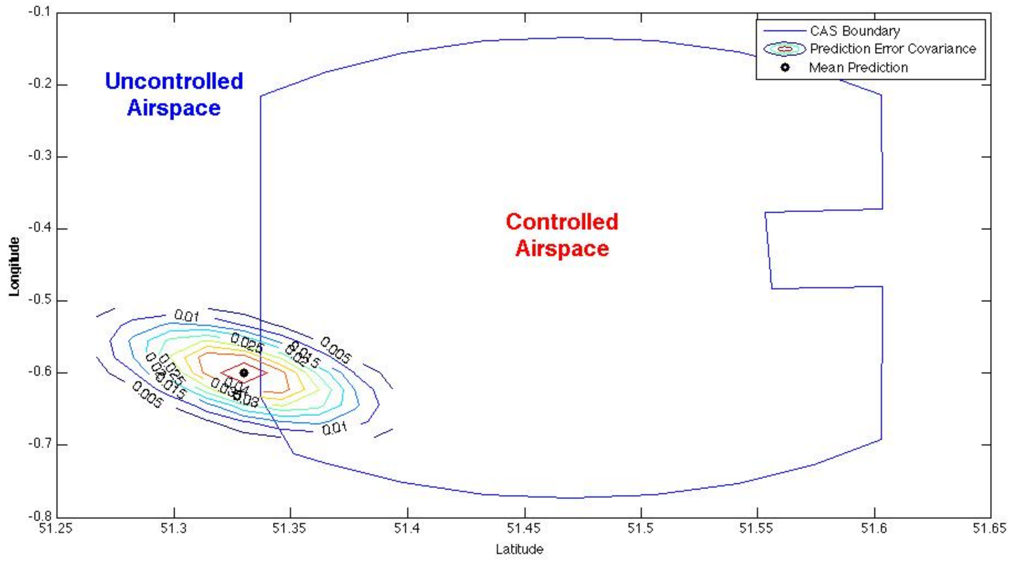


Figure 3.6: Mean prediction and its error covariance approaching a complicated CAS boundary from UCAS

Given the prediction and its error covariance in the figure above we will estimate the probability of infringement P(I) using the MC sampling method by drawing N number of samples from the prediction error covariance shown in figure 3.7 below:

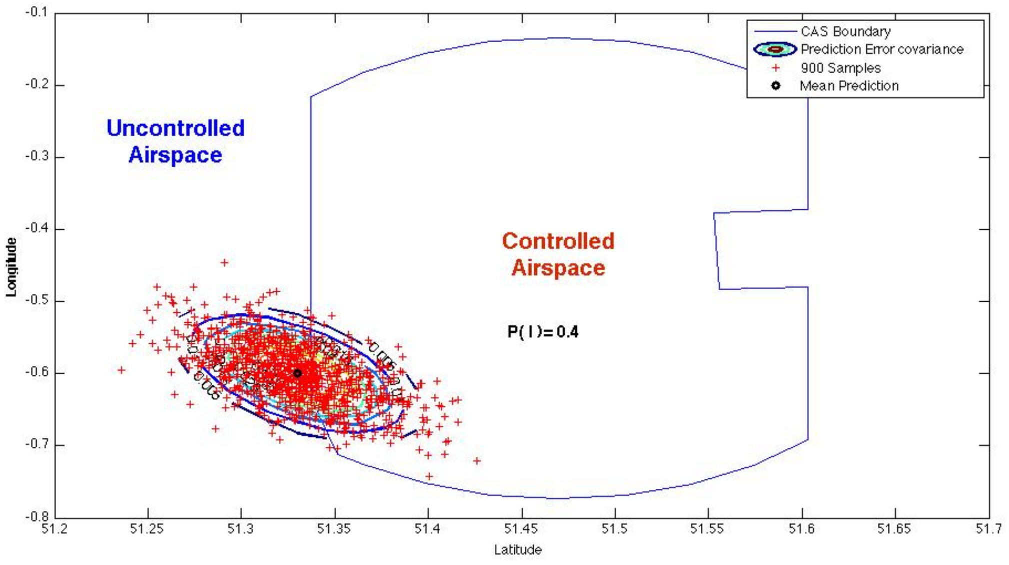


Figure 3.7: 1000 Samples drawn from the prediction error covariance

To find the optimal number of samples N and calculation time, we estimated the P(I) and calculation time using Mac machine with OSX 2.6 GHz intel Core i5 where 50<N<2000 and showed that 900 samples can be optimal as in figures 3.8 and 3.9 below:

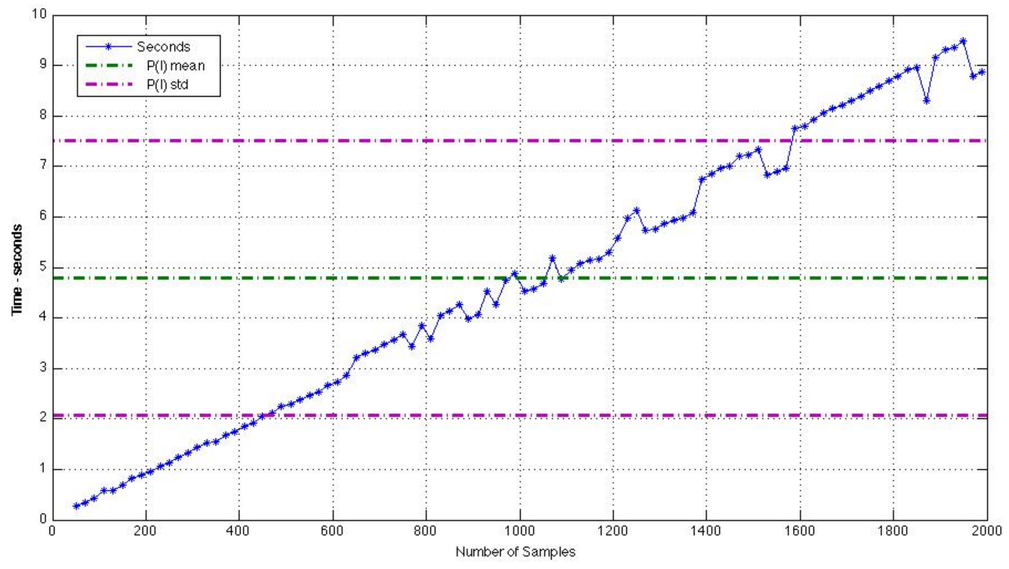


Figure 3.8: Calculation time for a different N sample sizes where 50<N<2000, where the mean calculation time is 4.7 seconds with 2.7 standard deviation

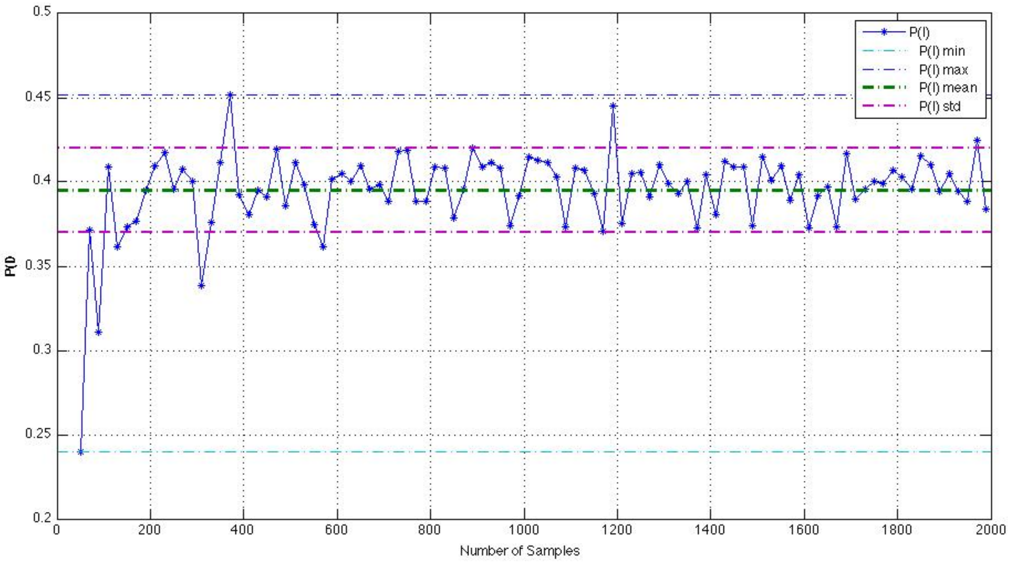


Figure 3.9: The probability of infringement given different number of N size samples. The standard deviation for all N size is 0.02 and mean=0.39

As we increase the number of samples by 100 the amount of time to calculate the P(I) will increase by approximately 0.5 seconds, the P(I) however, changes in very small percentage. As a result, we chose the number of samples N=900 where it gave the closest estimate to the mean P(I) for 100 different sizes of N samples ranges between 50 and 2000. Also, the amount of time required to estimate P(I) using 900 samples is the same or less than the amount of time a radar takes to observe the next location which is T=4 seconds.

3.2.2 Hybrid Method

The shortest distance method works as well as the MC sampling in calculating the probability of infringement as long as the aircraft is not close to a CAS boundary corner. Given the general Kalman filter's equations in chapter 2, we use it on a real track to predict and estimate the locations of an aircraft flying towards the edge of the CAS, before turning away from it shown in figure 3.10.

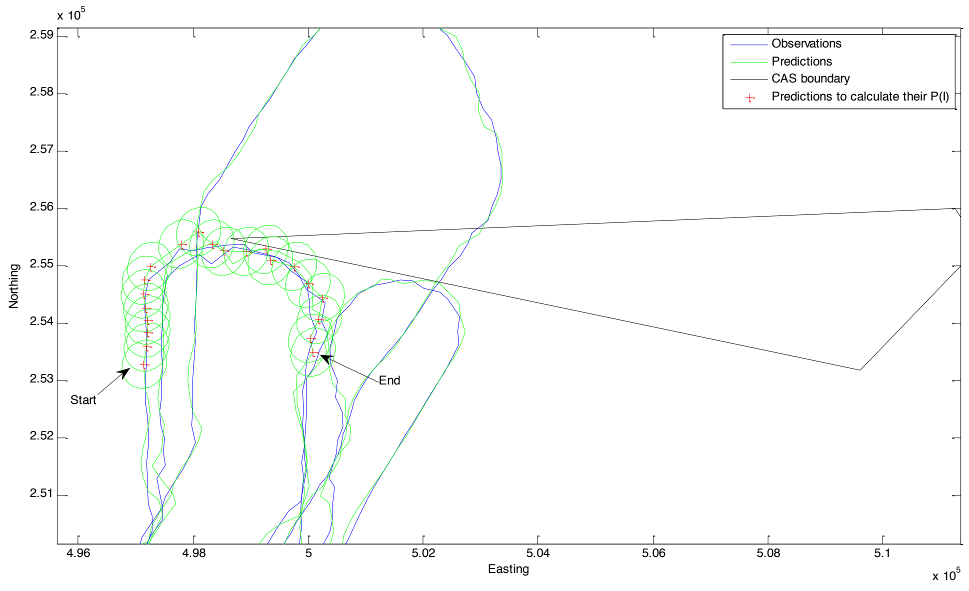


Figure 3.10: Aircraft approaching CAS boundary which has a sharp corner

where the observation matrix was defined as $H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$ because we are only observing the locations of the aircraft. The state transition matrix $A = \begin{bmatrix} 1 & T & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T \\ 0 & 0 & 0 & 1 \end{bmatrix}$, where $T = 4$ seconds (the time interval to observe the next location from the radar). The green ellipses in figure 3.10 are the predictions error covariance $\hat{P}$, where $\hat{P} = A\bar{P}A + Q$. We fixed the state error covariance Q through out this particular example with a random number since we have not learned the amount of error in the model yet, which we will in the next chapter. The observations were plotted in blue, 1 step ahead predictions in green, and highlighted in red crosses, are the predictions we are interested in finding their probability of infringement which are plotted in figure 3.11.

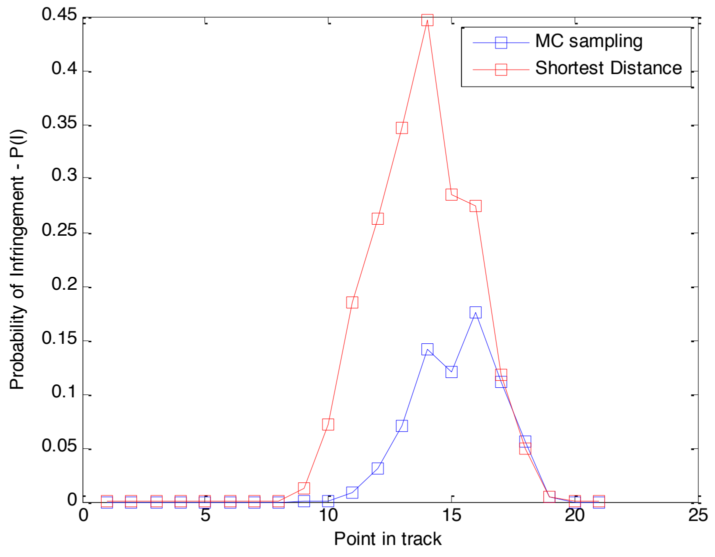


Figure 3.11: Probability of Infringement of the predictions plotted in red crosses in figure 3.10

Figure 3.11 shows the probability of infringements using the shortest distance and MC sampling methods. We used 1000 samples for the MC sampling. As the aircraft approaches the CAS boundary corner the shortest distance, assumes that the corner of CAS is a straight line, therefore, the probability of infringement is close to $\frac{1}{2}$ here the $P(I) = 0.45$. The MC sampling, however, estimates the probability of infringement less than 0.2, since the error covariances shown in figure 3.10 represent almost $\frac{1}{4}$ here, the $P(I) = 0.2$. Another example in figure 3.12 and 3.13 shows how the shortest distance assumes CAS boundary as a straight line.

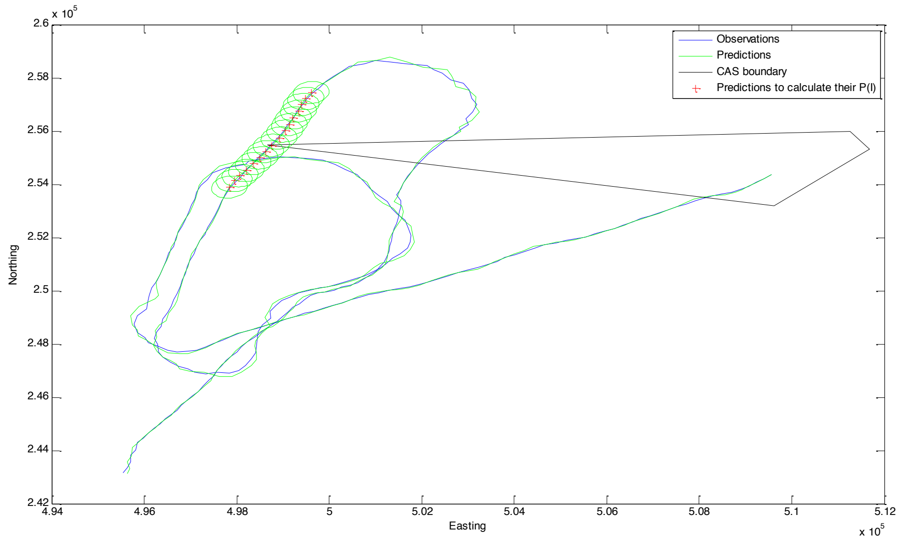


Figure 3.12: Real Track showing an aircraft cutting the corner of CAS boundary. The red cross (+) corresponds to the prediction

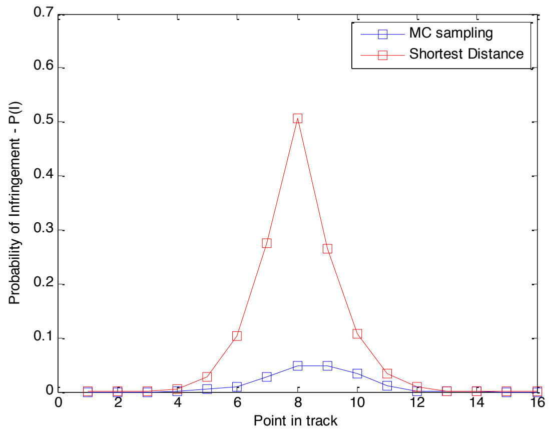


Figure 3.13: Probability of infringement using MC sampling (500 Samples) and the shortest distance for the red crosses in figure 3.12

Figure 3.12 shows real track where the aircraft only cuts the corner of the CAS boundary. We calculated the $P(I)$ using both methods on the red crosses and found out the SD method results in a $P(I) = 0.5$ whereas the MC sampling results in $P(I) = 0.05$, see figure 3.13.

If the aircraft is not approaching CAS corner, both methods perform well. For example, in figure 3.14 shows a partial real track which we used to predict the locations of the aircraft where the red crosses marks the predictions. We are interested in finding their probability of infringement since the rest of the track is not close to CAS, their probability of infringement is equal to zero.

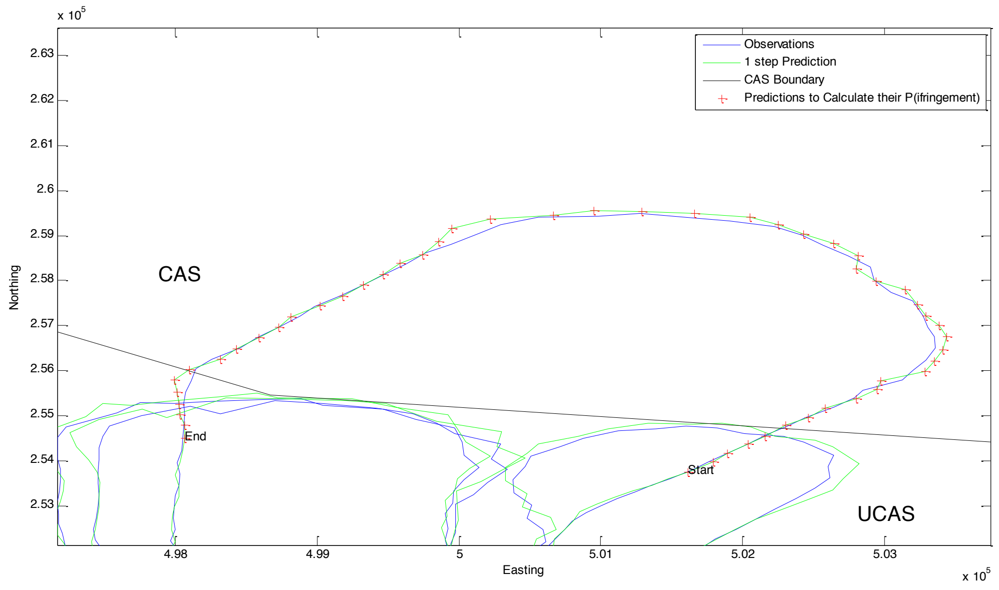


Figure 3.14: Predictions we used to find their probability of infringement using the shortest distance and the MC sampling methods at the same time as shown in figure 3.15

Figure 3.15 shows the probability of infringement using both methods for the prediction locations in figure 3.14. When the aircraft approaches CAS boundary the probability infringement using both methods starts to increase and when the aircraft gets inside it gives a high probability of infringement close to to 1. When it comes back to UCAS, however, the probability of infringement then starts to decrease for both methods.

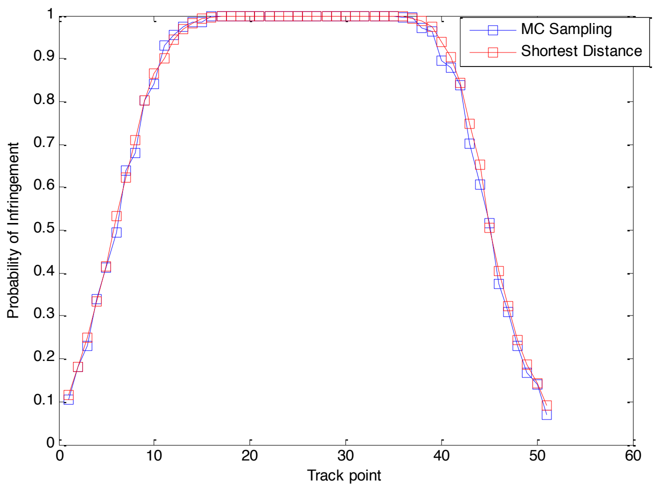


Figure 3.15: Probability of infringement using both methods for the red crosses plotted in figure 3.14

The previous two plots shows an example of when an aircraft indeed infringes the CAS and both models performed well. The predictions marked in red crosses are 1 step ahead predictions; therefore, their covariances are quiet small. As a result, there is usually not too much differences between the methods, however, we will see in the next chapter how to estimate the $P(I)$ for the 5 steps ahead predictions instead of 1 step. In figures 3.16 and 3.17 show different example of estimating the probability of infringement on another partial real track, but this time the aircraft does not infringe the CAS instead it turns away from it.

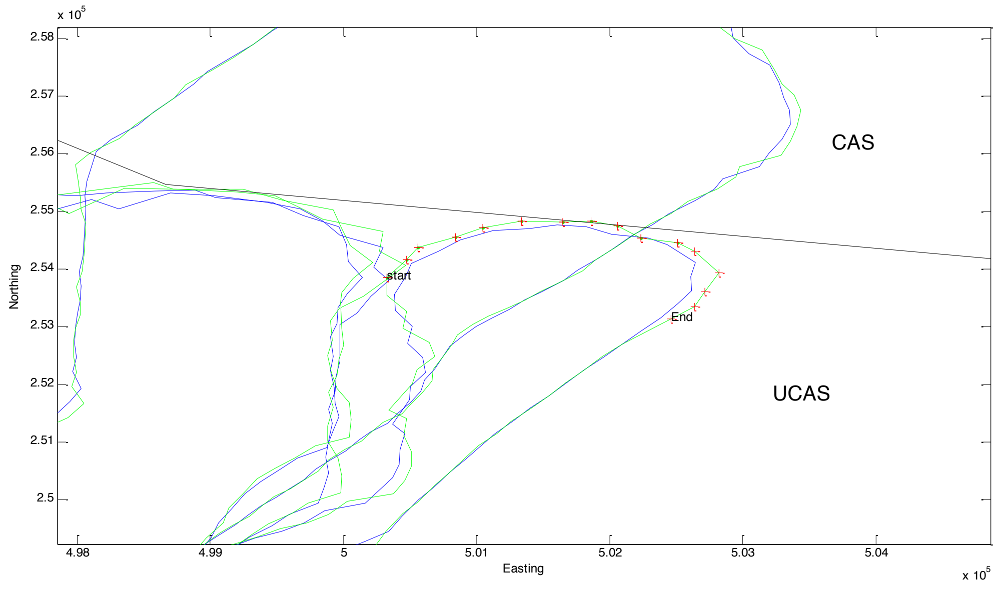


Figure 3.16: Predictions marked in red cross (+) used to find their probability of infringement using the shortest distance method and MC sampling as shown in figure 3.17

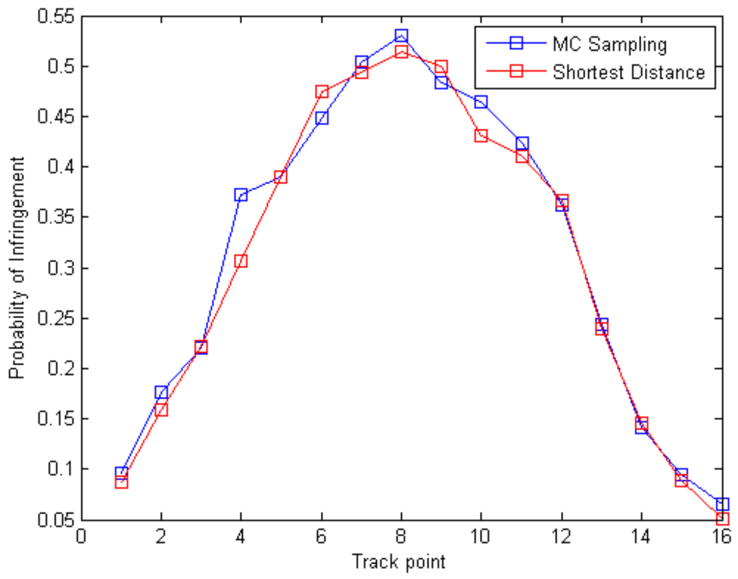


Figure 3.17: Probability of infringements for the red crosses in figure 3.16

MC sampling method works for arbitrary shaped boundaries, however, it is computationally expensive. Since both methods show the same results when the aircraft is distant from corners in the CAS boundary, it would be better to use the shortest distance whenever we can as long as the prediction is away from CAS boundary corners. Therefore, in order to effectively use both of them in the right time, we propose flexible conditions which can be changed to determine when to use one method or the other in order to find the probability of infringement accurately, using the following conditions:

1. If the shortest distance is greater than $3\sqrt{\sigma_{max}}$, then $P(I) = 0$. This means the prediction is very far away from the CAS therefore, the probability of infringement will be effectively zero.
2. If the shortest distance from the prediction to any vertex of the CAS boundary is greater than $2\sqrt{\sigma_{max}}$, where σ_{max} is the largest variance from σ_{cross} and σ_{along}. This condition means that we are not very close to corner of a CAS boundary therefore, the shortest distance can estimate $P(I)$ accurately.
3. If none of the previous conditions stands, then we use the MC sampling method.

How to calculate the distance from the prediction to nearest point on the CAS boundary and to find out the nearest CAS vertex is explained as follows: First, we assume we have a prediction in a vector $\hat{x} = (\hat{x}, \hat{y})$ and the two points on the CAS boundary $v_1 = (x_1, y_1)$, $v_2 = (x_2, y_2)$ and a point x between them where it is the shortest distant point on the CAS boundary to the prediction vector $\hat{x}$. The parametric equation for the line is defined by:

$$x = v_1 + \lambda\,(v_2 - v_1) \text{ where } 0 \leq \lambda \leq 1$$

The condition for the closest approach of x to x' is that $(x - x')$ is perpendicular to $(v_2 - v_1)$ therefore, $(x - x')^\circ(v_2 - v_1) = 0$. To calculate λ we solve the above equation for λ to become $\lambda = \frac{(x' - v_1)^\circ(v_2 - v_1)}{\|v_2 - v_1\|^2}$

To find the shortest distance from the prediction to the point x is listed below:

1. Solve for $\lambda = \frac{(x' - v_1)^{\circ}(v_2 - v_1)}{\|v_2 - v_1\|^2}$
2. If $\lambda > 1$, then it's off one end of the CAS boundary
3. If $\lambda < 0$, then it's off one end off the other end of the CAS boundary
4. Calculate $x = v_1 + \lambda\,(v_2 - v_1)$
5. Calculate the distance $d = \|x - x'\|^2$

Figure 3.18 shows the prediction near a CAS boundary where we calculated the shortest distance to it as a line from the prediction and perpendicular to CAS boundary. It also shows how we can determine the location of the CAS corners so we can use the conditions above in order to use the most accurate method at any specific time.

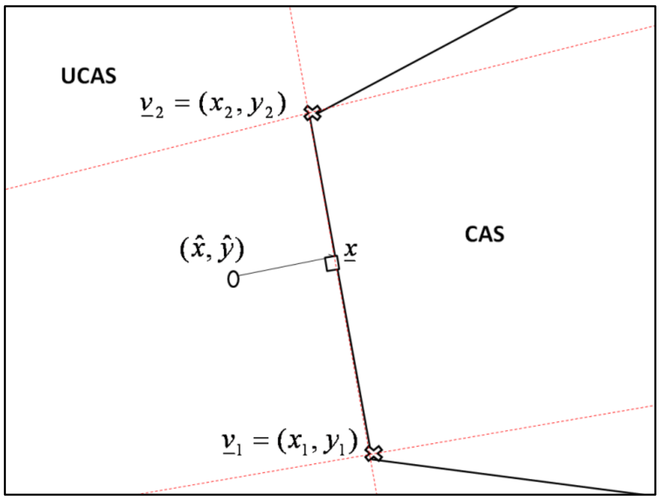

UCAS
$\underline{v}_2 = (x_2, y_2)$
$(\hat{x}, \hat{y})$
$\underline{x}$
CAS
$\underline{v}_1 = (x_1, y_1)$

Figure 3.18: Calculating the shortest distance from the prediction to a point on the CAS boundary:

Now that we know the conditions and how to find the nearest vertices of the CAS boundary, we use the same partial track in figure 3.10 and apply the conditions on it to find $P(I)$ in figure 3.19.

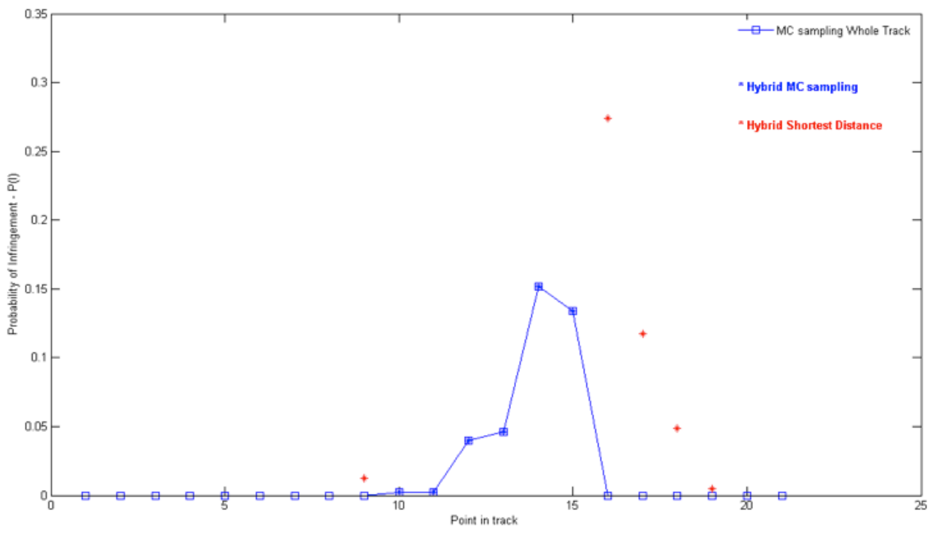


Figure 3.19: Probability of infringement using the Hybrid method, where we apply the conditions

Figure 3.19 shows the hybrid method which using the conditions. Notice the point in track from 0 to 8 we used the second condition which is the aircraft is very far from the CAS boundary. However, at point 9 this is when it gets closer to the CAS boundary and not near any of its corners, the first condition is applied which is calculating $P(I)$ using the shortest distance method. Points from 10 to 15, the aircraft is approaching a CAS boundary, therefore, we use the third condition which is MC sampling. From point 16 to 19, the aircraft heads away from the CAS boundary corner, as a result, condition 1 is used and will calculate $P(I)$ using the shortest distance. Finally, the last two points, the aircraft is far enough from the CAS boundary and therefore, it gave a probability of infringement equal to 0.

The hybrid method is an effective way to save the computation time required to calculate the $P(I)$. Because instead of using the MC sampling all the which is considered to be slow, we can use the shortest distance method whenever we can since it gives the same results as the MC sampling as long as it is away from a the CAS boundary corner.

3.3 Conclusion

To conclude this section, we reviewed the current probabilistic infringement tool which uses the shortest distance method; and extended it to be able to find the probability of infringement with the assumption that the CAS boundary is not a straight line. The downside of the shortest distance method is that it does not work whenever the prediction is near a corner of the CAS, therefore, we used the MC sampling method which can deal with this problem. The MC sampling method however, can be very slow depending on the number of samples to be drawn. Therefore, we implemented the hybrid method which switches between the two methods. When we used both methods together they were very similar, so when we use the hybrid method it eliminates each method’s disadvantage. For example, if the aircraft is away from a corner, we would use the shortest distance to calculate the probability of infringement because it’s faster than the MC sampling method. Therefore, in order for either method to work well we need to first learn the aircraft’s position and the errors in the prediction as accurately as possible. We can then predict the uncertainty better than having one error for the entire flight time. In the next chapter we will show how to track the aircraft using the Kalman filter and learn the model’s error using the EM algorithm.

4 Prediction with Online Learning

Our research objective is to investigate a model which predicts future infringements more accurately using the probability methods covered in the previous chapter. A benefit of using probability methods is that we can estimate the $P(I)$ based on the mean prediction and its covariance. It is apparent by looking at the tracks that the state error covariance varies for different aircraft and for any particular journey which leads the model to predict inaccurately. Therefore, it is necessary to learn the model's error covariance for any track, this will not only help in predicting more accurately but also find the probability of its infringement. There are two types of model learning: one is learning the errors for each track given the whole journey time step N (all the observations). Second case is learning the errors for any particular journey using observation up to time t where $t \leq N$. We will be using the Kalman filter reviewed in chapter 2 as our tracking tool and the EM algorithm as our learning tool for both types.

Because aircraft tend to fly in different fly modes such as in a constant velocity mode or in a constant acceleration mode (turning mode), it is essential to implement different kalman filters for different flight modes and then switch between them when appropriate this switching process will be discussed in the next chapter. In this chapter however, we will focus on the two types of covariance learning for all our Kalman filters separately. We will use synthetic track and show the results by comparing the true covariances and the learned ones.

4.1 Learning Using the Expectation-Maximization Algorithm

Fixing the model with any error for the entire flight is not efficient because the aircraft will tend to change its heading at some point and fly around in different patterns. Therefore, we need to learn the model's changing state covariance for any track. This allows the covariance to change during the flight. And by finding them, it will minimize the errors in the state estimation and prediction and increases the likelihood of the observation. The expectation maximization algorithm is a method used to maximize the likelihood of the observations given the new parameters of the model (Dempster, 1977). It alternates between the E-step which computes the log-likelihood of the observations given the current parameters and the M-step which maximizes the log-likelihood computed in the E-step by re-estimating the parameters $\theta = \{R, Q\}$. The EM algorithm is analogous to the Baum-Welch algorithm we have seen in the HMM section in chapter 2.

The two steps are:

- E-step: given the current set of parameters θ^t, observed data Z and hidden states X, then the auxiliary function is defined as:

  $$G(\theta|\theta^i)=E_{x|z,\theta}[\log \mathrm{P}(\theta; X, Z)]$$

  The complete data log-likelihood is calculated as:

  $$\log \mathrm{P}(\{X\}.\{Z\}) = -\sum_{t=1}^{N}[\frac{1}{2}\,(z_t - H\hat{x}_t)'R^{-1}(z_t - H\hat{x}_t)] - \frac{N}{2}\log(|R|)$$

  $$-\sum_{t=2}^{N}\left[\frac{1}{2}\,(\hat{x}_t - A\hat{x}_{t-1})'Q^{-1}(\hat{x}_t - A\hat{x}_{t-1})\right] - \frac{N-1}{2}\log(|\mathrm{Q}|)$$

- M-step: find the parameters that maximize the auxiliary function calculated in the E-step:

$$\theta^* = argmax\ G(\theta|\theta^t)$$

The M-step equation defined above is done by differentiating G with respect to R or Q and solving $\frac{\partial G}{\partial R} = 0$ or $\frac{\partial G}{\partial Q} = 0$. This results in the following equations (Shumway, 1988):

$$R = \sum_{t=1}^{N} ((\mathrm{z_t} - \mathrm{H_m}\tilde{\mathrm{x}}_\mathrm{t}^\mathrm{m})(\mathrm{z_t} - \mathrm{H_m}\tilde{\mathrm{x}}_\mathrm{t}^\mathrm{m})' + (\mathrm{H_m}\widetilde{\mathrm{P}}_\mathrm{t}^\mathrm{m}\mathrm{H_m}'))/N$$

where $\tilde{\mathrm{x}}_\mathrm{t}^\mathrm{m}$ and $\widetilde{\mathrm{P}}_\mathrm{t}^\mathrm{m}$ are the smoothed state, its error covariance for time t for the m^{th} Kalman filter. For the model error covariance we need to sum the smoothed error covariances from $t = 2$ upto N to re-estimate Q:

$$D = \sum_{t=2}^{N} (\tilde{\mathrm{x}}_\mathrm{t-1}^\mathrm{m}\tilde{\mathrm{x}}_\mathrm{t-1}^\mathrm{m} + \widetilde{\mathrm{P}}_\mathrm{t-1}^\mathrm{m})$$

$$E = \sum_{t=2}^{N} (\tilde{\mathrm{x}}_\mathrm{t}^\mathrm{m}\tilde{\mathrm{x}}_\mathrm{t-1}^\mathrm{m} + \widetilde{\mathrm{PP}}_\mathrm{t}^\mathrm{m})$$

$$F = \sum_{t=2}^{N} (\tilde{\mathrm{x}}_\mathrm{t}^\mathrm{m}\tilde{\mathrm{x}}_\mathrm{t}^\mathrm{m} + \widetilde{\mathrm{P}}_\mathrm{t}^\mathrm{m})$$

where $\widetilde{\mathrm{PP}}_\mathrm{t}^\mathrm{m}$ is the smoothed error covariance between time t-1 and t-2 for the m^{th} Kalman filter. The estimate of the state covariance is then Q:

$$Q = (F - EA' - E'A - ADA')/(N - 1)$$

The overall process of learning the Kalman filter using the EM-algorithm is listed below:

E-step:

- Run the Kalman filter, forward-pass
- Run the Kalman smoother, backward pass

M-step:

- Re-estimate Kalman filters error parameters (R, Q)

The process stops when we find the optimal error covariance at the end of the M-step, and by optimal is by finding the error coariances that is closest to the true error covariance.

We have seen how to estimate the state and smoothing them using the Kalman filter in chapter 2 section 2.5. In this section, we will focus on the M-step which re-estimates the errors of the model. This is a crucial step in which it helps us to find the best (Q^{*}, R^{*}) which gives the maximum likelihood of the observation.
There two types of error learning we implemented in this research: first, learning the error given the whole track, and second learning the error at each time step given a history window of N steps behind. Although the first learning type is not very efficient in computation time and estimating the optimal error at each time step, but it would be interesting to find out how the EM steps works for a given whole track. This type of learning is presented in algorithm 4.1:

Algorithm 4.1: Overall Process of Learning given the whole track N

```
iters = 0
MaxIterations= 100                                          Maximum EM iteration
m = 2                                                       Number of models

while iters < MaxIterations do
    for t=2 to N

        $(\hat{x}_t, \hat{P}_t, \bar{x}_t, \bar{P}_t, L_t^m) =$                          E- step:
        $Filter(\hat{x}_{t-1}, \hat{P}_{t-1}, \bar{x}_t, \bar{P}, Q_m^*, R_m^*, H_m, A_m, t)$    Kalman Filtering


        $(\tilde{x}_{t-1}, \tilde{P}_{t-1}, \widetilde{PP}_t) = Smoother(\hat{x}_t, \hat{P}_t, R_{cv}^*, H_{cv}, A_{cv}, t)$
                                                            Kalman Smoothing
        Next t
    $(Q_m^*, R_m^*) = ErrorEstimation(z, (\tilde{x}, \tilde{P}, \widetilde{PP}_t, N)^m)$      M-step:
    $iters = iters + 1$                                     Re-estimating models' and
end while                                                   observation errors
```

In order to re-estimate the errors of the model we need the Kalman smoother outputs $(\tilde{X}, \tilde{P}, \widetilde{PP})$ which were calculated during the E-step to estimate the new errors for the whole track. The error estimation too seen in the M-step in algorithm 4.1 is used to re-estimate the errors using the Kalman smoother outputs: the smoothed state $\tilde{x}_t$, the smoothed error covariance $\tilde{P}_t$ and the smoothed error covariance $\widetilde{PP}_t$ between the $t-1$ and $t-2$, the error estimation is shown in detail in algorithm 4.2:

Algorithm 4.2: $(Q_m^*, R_m^*) = \mathrm{ErrorEstimation}(Z, \tilde{X}, \tilde{P}, \widetilde{PP}, N)$

```
Require: Z                                  Observation matrix
Require: $\tilde{X}$                                  Smoothed states matrix
Require: $\tilde{P}$                                  Smoothed error covariance
                                            matrix
Require: $\widetilde{PP}$                                 Smoothed error covariance
                                            matrix between time $t$-1 and $t$-2
Require: N                                  Total Length of the journey
m= 0 or 1                                   Flight mode type
$D, E, F = 0$
$t = 1$
R=0
while $t \leq N$ do
```

```
    $R = R + (z_t - H_m \tilde{x}_t^m)(z_t - H_m \tilde{x}_t^m)' + (H_m \tilde{P}_t^m H_m')$
    $t = t + 1$
end while

$R_m^* = R/N$                                   Observation error re-estimation

$t = 2$
while $t \leq N$
    $D = D + (\tilde{x}_{t-1}\tilde{x}_{t-1}' + \tilde{P}_{t-1})$
    $E = E + (\tilde{x}_t\tilde{x}_{t-1}' + \widetilde{PP}_t)$
    $F = F + (\tilde{x}_t\tilde{x}_t' + \tilde{P}_t)$
    $t = t + 1$
end while
                                                Model error re-estimation
$Q_m^* = \frac{(F - EA' - AE' - ADA')}{N - 1}$
Return $(R_m^*, Q_m^*)$
```

We tested this type of learning on a synthetic track shown in figure 4.1; results of the learned error covariance for the whole track is shown in figure 4.2:

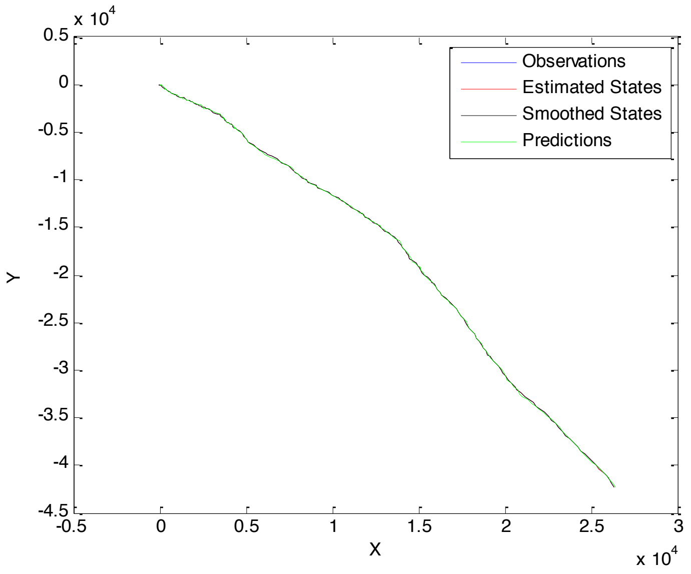


Figure 4.1: Synthetic Track used to test Learning the model's error covariance given the whole track

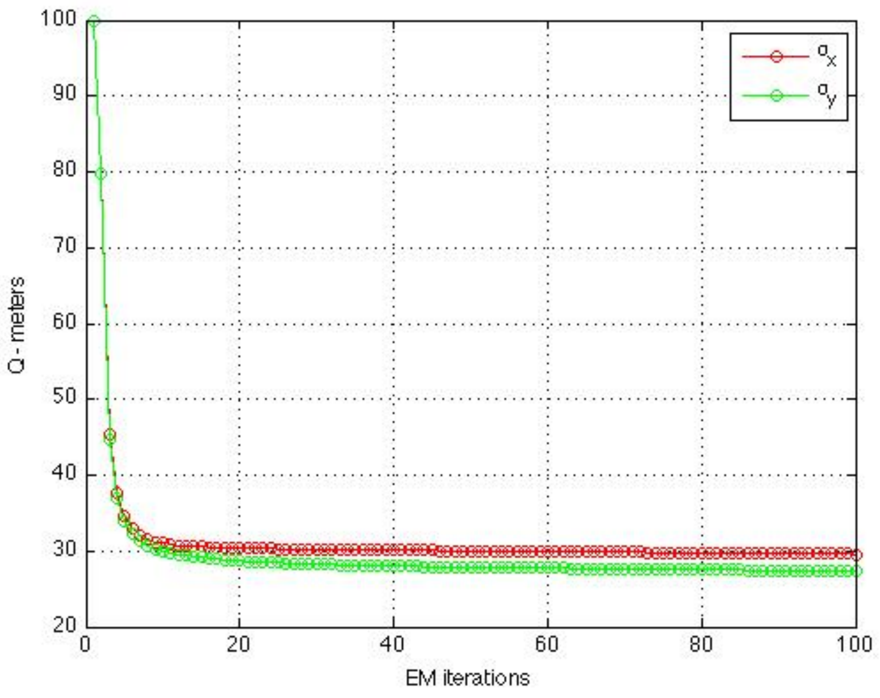


Figure 4.2: After learning the model's error covariance Q using 100 EM iterations where the true (σ_x, σ_y)=30m

Figure 4.1 shows a synthetic track where it flies with a constant velocity and a constant error covariance of $Q_{true} = (30^2)I$, where I is the identity matrix. Figure 4.2 shows the model error covariance after running the EM algorithm 100 iterations on the track in figure 4.1. We noticed how Q tends to get closer from the first initialisation $Q_1 = (100^2)I$ to the true value of the model's error covariance **.** This tells us that as long as the aircraft flies in a constant velocity we can estimate the true value of the Q given the whole track. We carried out this learning on an additional 12 tracks each has 300 track points, and estimated the average error for all tracks after 100 EM iterations is 5.4 meters shown in figure 4.3:

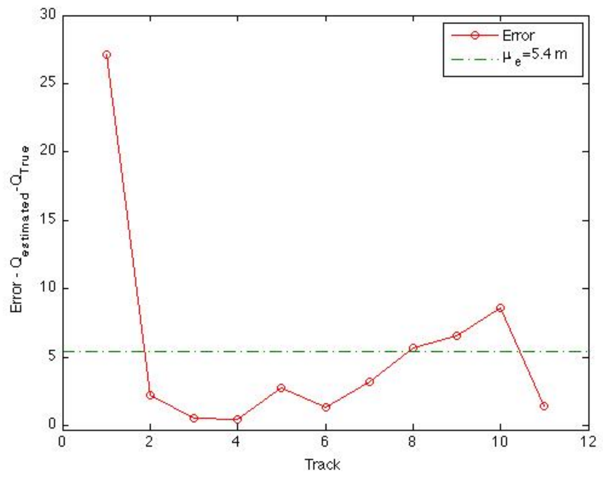


Figure 4.3: Error between the learned and the true state error covariance Q after running 100 EM iterations on 12 synthetic tracks

What if the aircraft change its direction? The error covariance will vary during the flight, therefore, learning one error for the whole track is not efficient as we mentioned above. Instead we need to learn the covariance at each time. We show in algorithm 4.3 the overall process of the EM algorithm for this type of online learning:

Algorithm 4.3: Overall Process of Learning for each time step t

```
iters = 0
MaxIterations= 10
m = 0 or 1                                                            Flight mode

for t=2 to N
    while iters < MaxIterations do
                                                                      E- step:
        $(\hat{x}_t, \hat{P}_t, \bar{x}_t, \bar{P}_t, L_t^m) =$
        $Filter(\hat{x}_{t-1}, \hat{P}_{t-1}, \bar{x}_t, \bar{P}, Q_t^m, m, H_m, A_m, t)$        Kalman Filtering

        $(\tilde{x}_{t-1}, \tilde{P}_{t-1}, \widetilde{PP}_t) = Smoother(\hat{x}_t, \hat{P}_t, R_t^{cv}, H_{cv}, A_{cv}, t)$   Kalman Smoothing
                                                                      M-step:
        $(Q_t^m, R_t^m) = ErrorEstimation(z, (\tilde{x}, \tilde{P}, \widetilde{PP}_t, t)^m)$   Re-estimating
                                                                      models' and
                                                                      observation errors
        $iters = iters + 1$
    end while
Next t
```

In algorithm 4.3, every time we get an observation, we run 10 EM iterations to find best Kalman filter's error covariances $(\mathrm{Q}_{\mathrm{t}}^{\mathrm{m}}, \mathrm{R}_{\mathrm{t}}^{\mathrm{m}})$ given all the observation up to time t before moving on to estimate the error covariance at time $t + 1$. In algorithm 4.4 below, we show the inside of error estimation tool, but with little crucial modification which is re-estimating the error covariances given a history window of 15 time steps (1 minute window of flight journey time) instead of averaging over all the whole journey time up to time t. There are two reasons behind this:

1. To better estimate the error covariances
2. To minimise the computation time

For example, assume that the aircraft was flying in a straight line with cross track error of 30m and along track error of 30m, then after 30 seconds it changes directions with little acceleration applied causing errors to increase from 30meters to 60m, after 1 minute the aircraft decides to fly back in a straight line with errors of 30m. If we use the whole history of flight time we would average the whole errors including the large errors that occurred during the turning mode. So when the aircraft errors go back to 30m after turning, the error estimating tool would still include the 60m error to be averaged and therefore, the errors will still be high. As a result, we create a history window of 15 steps to re-estimate the error covariances to not only avoids this issue; but also to minimise the computation time. The longer the window the slower the error estimator tool converges to the right error and vice versa. :

Algorithm 4.4: (Q_t^m, R_t^m) = $\mathrm{ErrorEstimation}(Z, \tilde{X}, \tilde{P}, \widetilde{PP}, t)$

```
Require: $Z$                                Observations matrix
Require: $\widetilde{X}$                    Smoothed states matrix
Require: $\widetilde{P}$                    Smoothed error covariance matrix
Require: $\widetilde{PP}$                   Smoothed error covariance matrix
                                            between time $t$-1 and $t$-2
Require: t                                  Length of the journey up to step t
m= 0 or 1                                   Flight mode
$D, E, F = 0$
HistWindow=15
                                            History window of 15 steps
$if$ t <= (HistWindow +2) then
  $N = t$
  $t = 2$
$else$
  $N = t$
  $t = t - HistWindow$
$end\ if$

while $t \leq N$ do
    $R = (z_t - H_m \tilde{x}_t^m)(z_t - H_m \tilde{x}_t^m)' + (H_m \tilde{P}_t^m H_m')$
    $t = t + 1$
end while
                                            Observation error re-estimation
$R_t^m = R/N$

while $t \leq N$
    $D = D + (\tilde{x}_{t-1} \tilde{x}_{t-1}' + \tilde{P}_{t-1})$
    $E = E + (\tilde{x}_t \tilde{x}_{t-1}' + \widetilde{PP}_t)$
    $F = F + (\tilde{x}_t \tilde{x}_t' + \tilde{P}_t)$
    $t = t + 1$
end while
                                            Model error re-estimation
$Q_t^m = \frac{(F - EA' - AE' - ADA')}{HistoryWindow - 1}$

return $Q_t^m, R_t^m$
```

Figure 4.4 shows a synthetic track with 2000 track points (2 hour flight time) where true

(σ_x, σ_y) =60m between time steps $500 \leq t \leq 600$ $1000 \leq t \leq 1600$ and 30m otherwise.

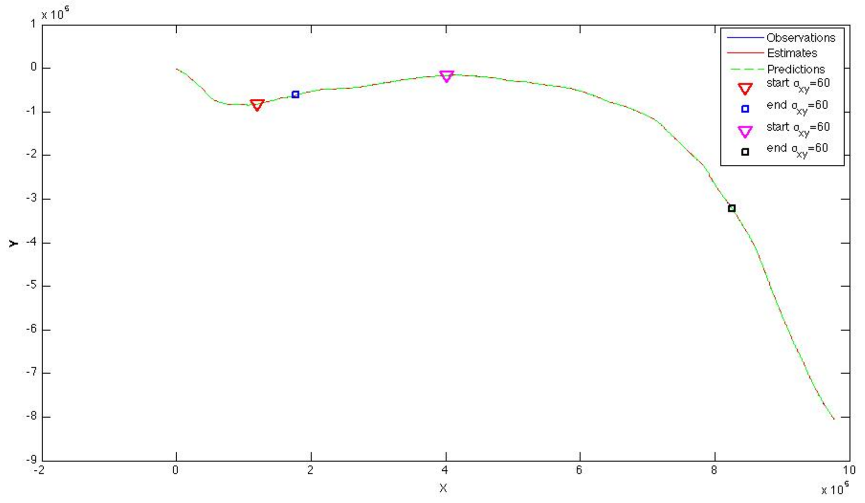


Figure 4.4: Synthetic track with different error covariances where (σ_x, σ_y)=60m between the triangle square markers and 30m otherwise

We then re-estimate Q in figures 4.5 and 4.6 using the history window of 15 steps and without it to compare them with the true values of the model's error covariance.

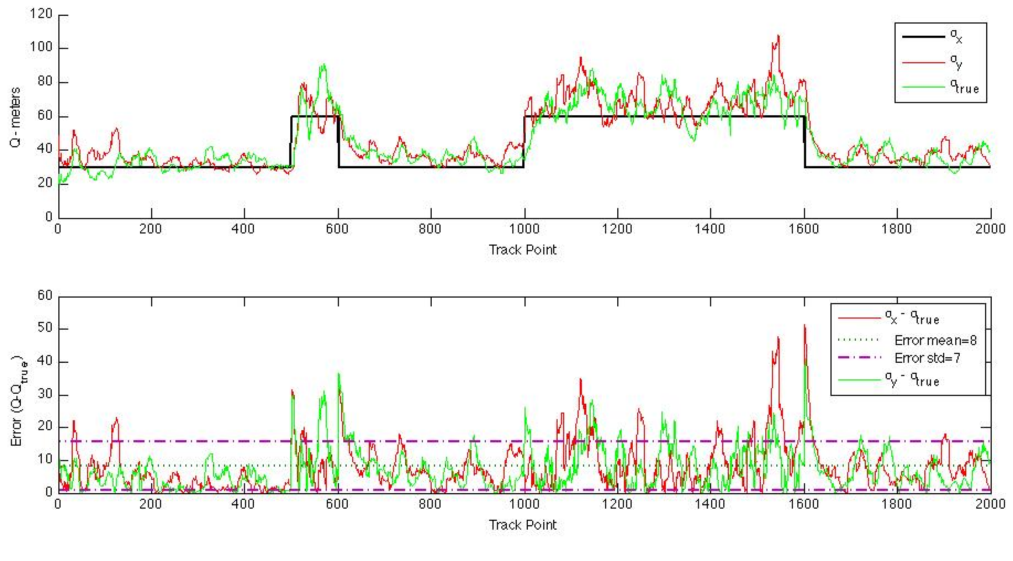


Figure 4.5: Learning the model's error covariance at each time step with a window of 15 time steps after 10 EM iterations for the synthetic track in figure 4.4. In addition to amount of error between the true Q and the estimated Q.

Figure 4.5 shows that after 10 EM iterations, Q gets closer to the true value marked in black where the average error between the true error covariance and the estimated one was 8 meters with a standard deviation of 7 meters as appose to figure 4.6 where the error estimator is having a hard time picking up the right errors with average error 13m and standard deviation 12.

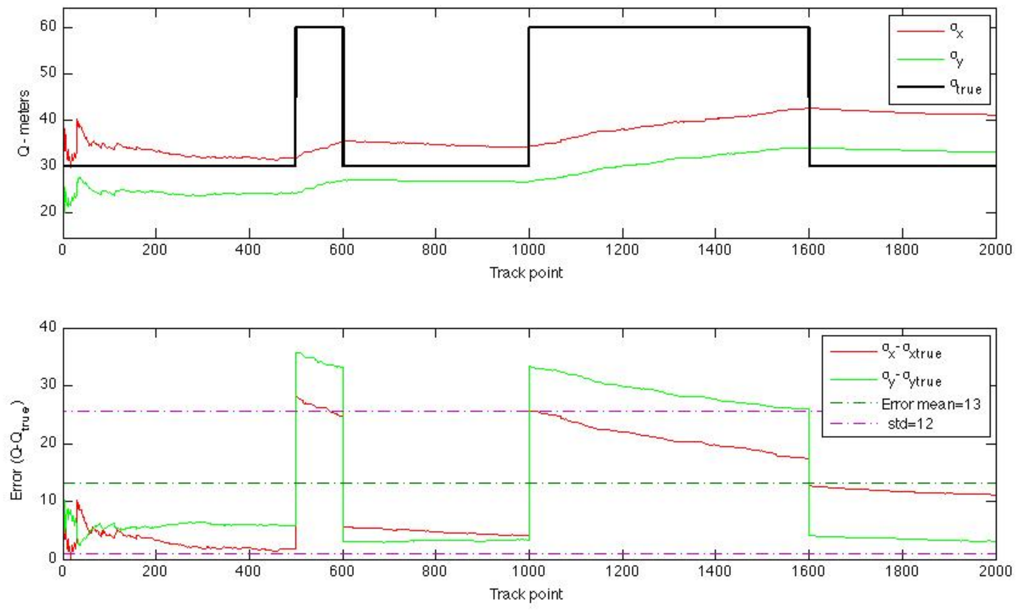


Figure 4.6: Learning the model's error covariance at each time step without a history window after 10 EM iterations for the synthetic track in figure 4.3. In addition to amount of error between the true Q and the estimated Q.

As far as for the computational time, we show in figures 4.7 and 4.8 the average time required to re-estimate Q at each time step with a history window and without it respectively.

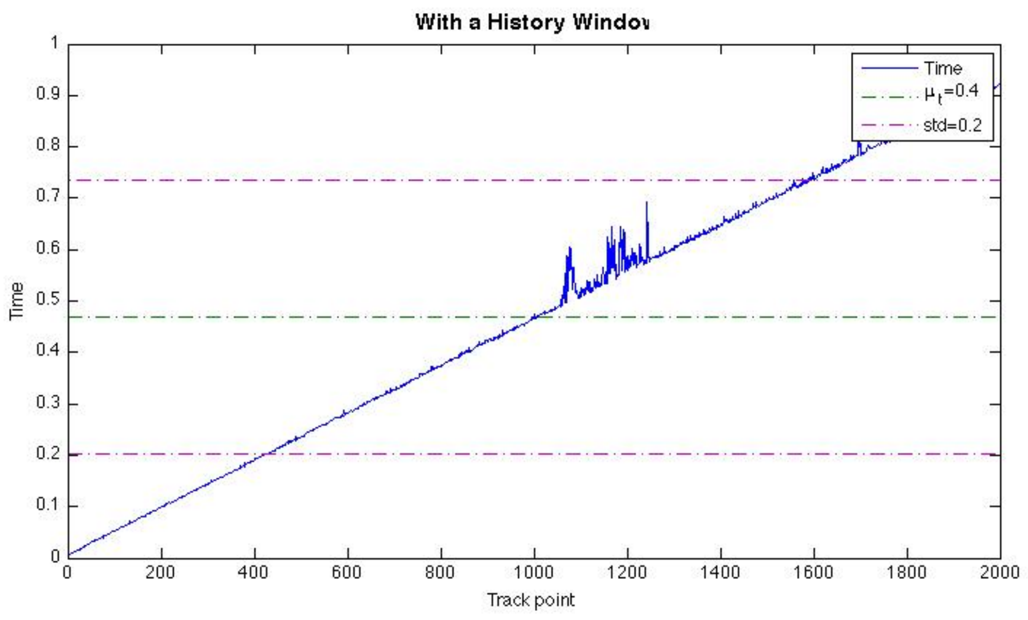


Figure 4.7: Time at each time step using the history window of 15 steps

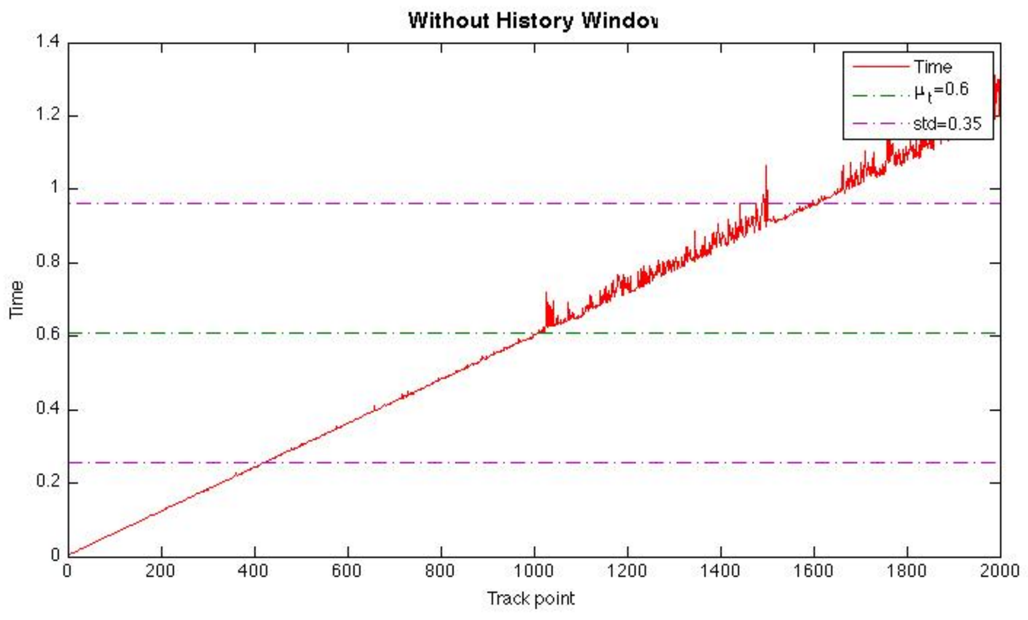


Figure 4.8: Time at each time step without using the history window

In figure 4.7 given a history window the average time was 0.4 seconds and a standard deviation 0.2 seconds where as in figure 4.8 the average time required to estimate the errors without the history window was 0.4 and standard deviation 0.35.

Because we are using one Kalman filter for the track in figures 4.1 and 4.4, it would be more efficient if we could use different Kalman filters and switch between them (as will be discussed next chapter). In the next section, we will implement two Kalman filters, which deals with two different flight modes m: Constant velocity (CV) and constant acceleration (CA) modes.

4.2 Learning Model Error covariance with Different Kalman Filters

First, we have implemented two Kalman filters: CV and CA Kalman filter. It is more efficient to have two different Kalman filters to be used when appropriate instead of having one flexible Kalman filter as we have seen in figure 4.1 and 4.4 in the previous section. Figure 4.9 illustrates why is it important to have two different Kalman. We used a real track given by NATS and apply our Kalman filters on it to predict and estimate the aircraft locations given the observations (blue line). When the aircraft flies in an arc shape (turning mode), our CA Kalman filter predicts more accurately than the CV Kalman filter where the RMSE for the state predictions (green crosses) equals to 65m whereas the CV KF gave RMSE= 73m. The reason for this is because CV Kalman filter models the aircraft as flying in a straight line whereas the CA Kalman filter models the aircraft flying in a circular arc.

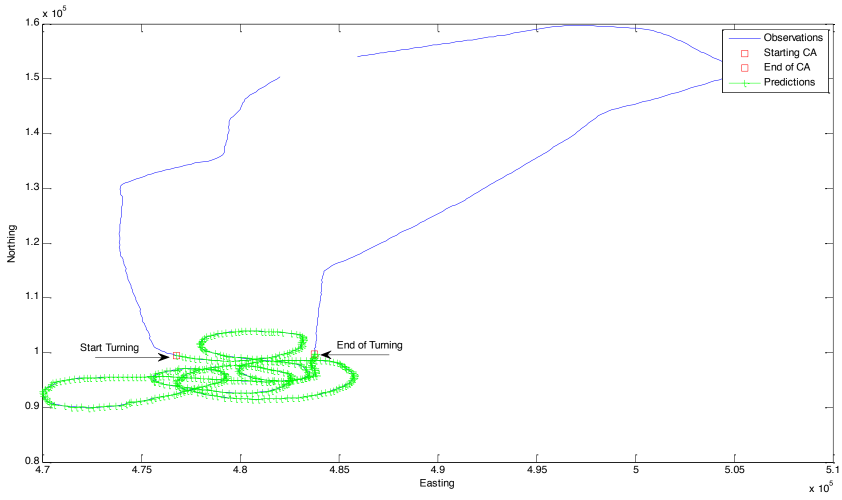


Figure 4.9: Real track Prediction showing the locations of the aircraft where it started to accelerate in (+).

After running several experiments on predicting ahead of time such as 2 minutes ahead, we found out that the amount of prediction error increases linearly with time this will provide inaccurate estimate of the P(I) therefore, unnecessary false alerts to the ATCs. Given the results for predicting ahead of time experiments, we chose to extend both Kalman filters to predict 5 steps ahead that is 20 seconds in the future. This prediction can not only provide us with the a clear idea where the prediction location might be but also raise an alert to ATCs that an aircraft is approaching the CAS boundary and may infringe it as appose to the current CAIT, where it alerts them when its already being infringed 8 seconds ago.

First, we will present both components and equations of the constant velocity Kalman filter and the constant acceleration Kalman filter. The Kalman filters' general equations and symbols are presented in chapter 2.

Kalman Filters' initializations for both Constant velocity and acceleration models:

We initialized the state estimates and predictions with the first observation for both Kalman filters $\bar{x}_1 = \hat{x}_1 = z_1$. The initial state's error covariance Q will be set with a large number since we are uncertain about the first prediction, however, the state's initial velocities and accelerations were set to 1 instead of zeros to avoid initial division by zero inside the Kalman filter. The error covariance for the state estimate and prediction $(\bar{P}_1, \hat{P}_1)$ were initialized with the observation error covariance matrix $R = \rho \times I$ by: $\bar{P}_1, = \hat{P}_1 = H'_m R H_m$ where m is the mode type (CV or CA). The state transition matrix A and the observation matrix H are both constant for both CV and CA models. Their definition will be in section 4.2.1. Also, since the radar rotates a whole revolution in 4 seconds collecting the locations of the aircraft in each revolution, the time interval we used is $T = 4$ seconds. In addition to the 1 step ahead prediction time interval T, a new time interval τ is defined for the multistep prediction where $\tau = 5 \times 4 = 20$ seconds (predicting the location of the aircraft 20 seconds ahead in the future).

4.2.1 Constant Velocity Model

The state of the aircraft in CV mode is defined as real valued vector $x = [x\ v_x\ \ y\ v_y\ ]$ where $[x\ \ y]$ are the state locations and $[v_x\ v_y]$ are the velocities in the x and y directions. The state model and its state transition matrix are defined as:

$$x_t = A_{cv} x_{t-1} + G_{cv}{}' w_{cv}$$

$$A_{cv} = \begin{bmatrix} 1 & T & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_{cv,\tau} = \begin{bmatrix} 1 & \tau & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \tau \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$G_{cv} = \begin{bmatrix} \frac{T^2}{2} & T & 0 & 0 \\ 0 & 0 & \frac{T^2}{2} & T \end{bmatrix}$$

The observation model for the CV Kalman filter and it transition matrix is defined as:

$$z_t = H_{cv}x_t + v$$

$$H_{cv} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

4.2.2 Constant Acceleration Model

The state of the aircraft in CA mode is defined as $x = [x\ v_x\ a_x\ y\ v_y\ a_y]$, where $[x\ y]$ are the state locations, $[v_x\ \ v_y]$ and $[a_x\ a_y]$ are the velocities and accelerations in x and y directions respectively. The state model is defined as:

$$x_t = A_{ca}x_{t-1} + G_{ca}{}'w_{ca}$$

$$A_{ca} = \begin{bmatrix} 1 & T & \frac{T^2}{2} & 0 & 0 & 0 \\ 0 & 1 & T & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & T & \frac{T^2}{2} \\ 0 & 0 & 0 & 0 & 1 & T \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_{ca,\tau} = \begin{bmatrix} 1 & \tau & \frac{\tau^2}{2} & 0 & 0 & 0 \\ 0 & 1 & \tau & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & \tau & \frac{\tau^2}{2} \\ 0 & 0 & 0 & 0 & 1 & \tau \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$G_{ca} = \begin{bmatrix} \frac{T^2}{2} & T & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{T^2}{2} & T & 1 \end{bmatrix}'$$

The observation model for the CA Kalman filter is defined as:

$$z_t = H_{ca}x_t + v$$

$$H_{ca} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

4.3 Further Testing on Synthetic Tracks

To generate the synthetic tracks and observations we used the underlying state and observation linear equations and they are:

$$x_t = A_m x_{t-1} + w$$

$$z_t = H_m x_t + v$$

where $w \sim \mathcal{N}(0, Q_m)\ and\ v \sim \mathcal{N}(0, R)$ and m is the type of flight mode (CV,CA). These tracks either have one state error covariance or different state error covariances for the whole track as shown in algorithm 4.5:

```
Algorithm 4.5:
$(X_{Syntheic}, Z_{Syntheic}) = \text{GenerateSyntheticTracks}(A_{CV}, A_{CA}, Q_{CV}, Q_{CA}, \text{R}, \text{H}, \text{N})$

Require: $A_{CV}, A_{CA}$          The state transition matrices for
                                   CV and CA models
Require: $Q_{CV}, Q_{CA}$          The state error covariances for CV
                                   and CA models
Require: $R$                       Observation error covariance
                                   matrix
Require: $H$                       Observation matrix
Require: $N$                       Total track length

$x_1 = 0$
$z_1 = 0$
$n = 2$
```

```
$while\ n \leq N$
  $if(500 \leq n \leq 600\ or\ 1000 \leq n \leq 1600)$        Between these time points apply
                                                              acceleration
        $w = randn \times Q_{CA}$                             Generate random errors from CA
        $x_t = A_{CA} x_{t-1} + w$                            error covariance
  $else$
        $w = randn \times Q_{CV}$                             Generate random errors from CA
        $x_t = A_{CV} x_{t-1} + w$                            error covariance

  $end$
 $v = randn \times R$                                         Generate observations random
 $z_t = H x_t + v$                                            error
$end$

Return $(x, z)$
```

4.3.1 Best errors for the whole track

For this particular section of learning the whole track, its only efficient to test it using one error covariance for the whole track as a result, we used the CV error covariance with random parameters to generate the track which is:

$$Q_{true} = \begin{bmatrix} \sigma_x^2 & 0 & 0 & 0 \\ 0 & \sigma_{vx}^2 & 0 & 0 \\ 0 & 0 & \sigma_y^2 & 0 \\ 0 & 0 & 0 & \sigma_{vy}^2 \end{bmatrix} = \begin{bmatrix} 1^2 & 0 & 0 & 0 \\ 0 & 0.1^2 & 0 & 0 \\ 0 & 0 & 1^2 & 0 \\ 0 & 0 & 0 & 0.1^2 \end{bmatrix}$$

In addition to white observational noise:

$R_{true} = \rho^2 I$ where $\rho = 50$ and I is an identity matrix

In figure 4.10, the observation track is generated by the true states, the state estimation and the smoothed track after 100 EM iterations.

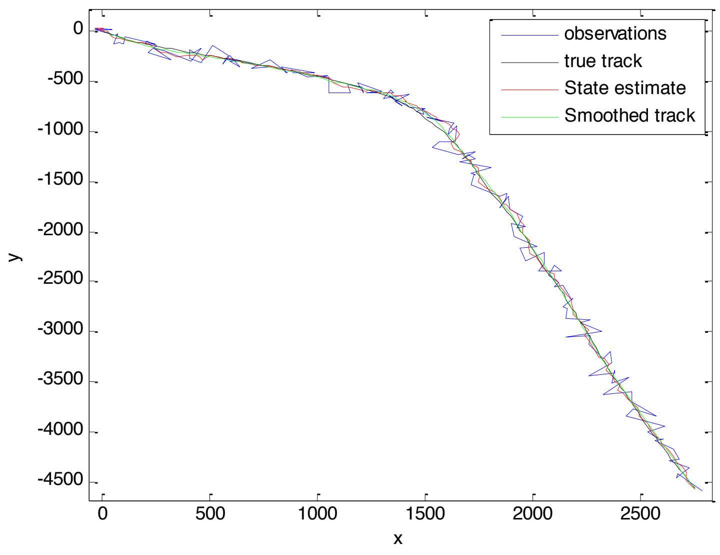


Figure 4.10: Synthetic track with one fixed error generated to test the EM algorithm for the whole track

We generated another sample track where we initialized the Kalman filter with large observation and state error as appose to the previous example:

$$R_1 = \begin{bmatrix} 100^2 & 0 \\ 0 & 100^2 \end{bmatrix} \text{ and } Q_1 = \begin{bmatrix} 100^2 & 0 & 0 & 0 \\ 0 & 2^2 & 0 & 0 \\ 0 & 0 & 100^2 & 0 \\ 0 & 0 & 0 & 2^2 \end{bmatrix}$$

The first observation and state estimate are initialized by the first true state $z_1 = \bar{x}_1 = x_1^{truth}, \bar{x} = [x \; v_x \; y \; v_y]$. In figures 4.11 and 4.12, we ran the EM algorithm 600 iteration to find the new best errors (Q^*, R^*).

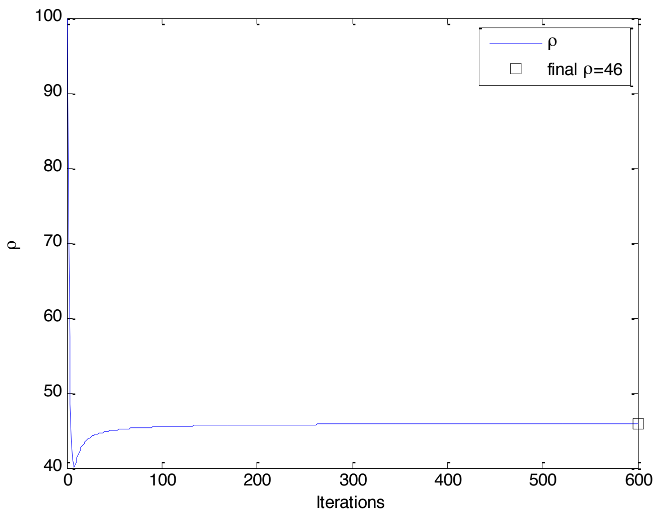


Figure 4.11: Best observation error ρ after 600 iterations

In figure 4.11, the true error of the observation which we generated was 50, and after 100 EM iterations it decreased from the initialized value 100 to 46 which is very close to the true value of the observation error.

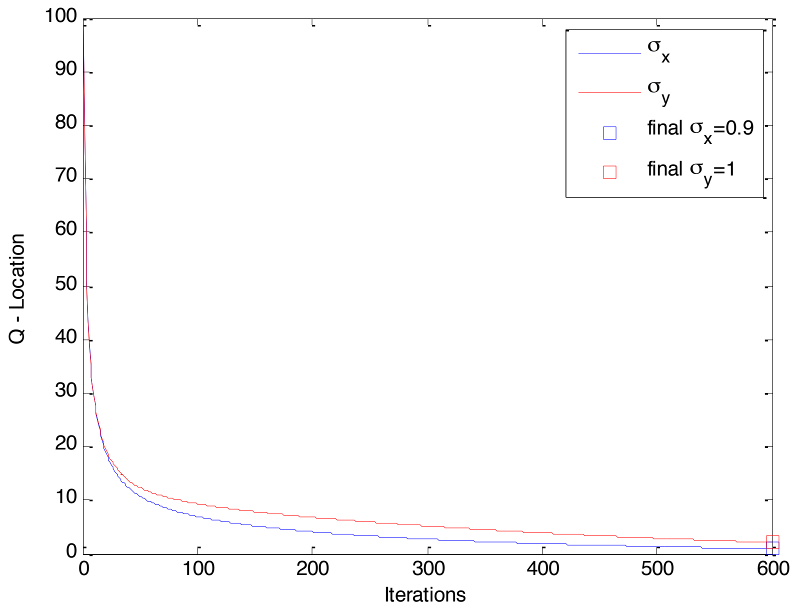


Figure 4.12: Estimated $\sigma_x$and $\sigma_y$after 600 EM iterations

The true values of the model's error covariance $\sigma_x = \sigma_y = 1$, we show in figure 4.12 that after 600 EM iterations the values of $\sigma_x$and σ_y dropped from 100 towards 1.

4.3.2 Best error at each time t

Since our research focuses on this type of learning, we will test our error estimation tool using a synthetic track with 2000 time steps with using both Kalman filters: CV and CA. They were generated using the following:

$$x_t^{truth} = Ax_{t-1}^{truth} + w_t$$

This track has two different model errors (Q_{true1}, Q_{true2}) in different time periods: $w_t \sim \mathcal{N}(0, Q_{CA})$ when $500 \leq t \leq 600$ and $1000 \leq t \leq 1600$ and $w_t \sim \mathcal{N}(0, Q_{CV})$ otherwise.

$$Q_{CV} = \begin{bmatrix} \sigma_x^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma_{vx}^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_{ax}^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma_y^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma_{vy}^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma_{ay}^2 \end{bmatrix} = \begin{bmatrix} 30^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 30^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \text{and}$$

$$Q_{CA} = \begin{bmatrix} 60^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 6^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 60^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 6^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2^2 \end{bmatrix}$$

The observations were generated using the x^{truth} in addition to white noise calculated as:

$$z_t = Hx_t^{truth} + v_t \text{ where } v_t \sim \mathcal{N}(0, R)$$

$$R = \rho^2 I \text{ where } \rho = 15 \text{ and } I \text{ is an identity matrix}$$

Figure 4.13 shows the observation, the true states, the state estimation and the smoothed track after 10 EM iterations calculated using the EM algorithm.

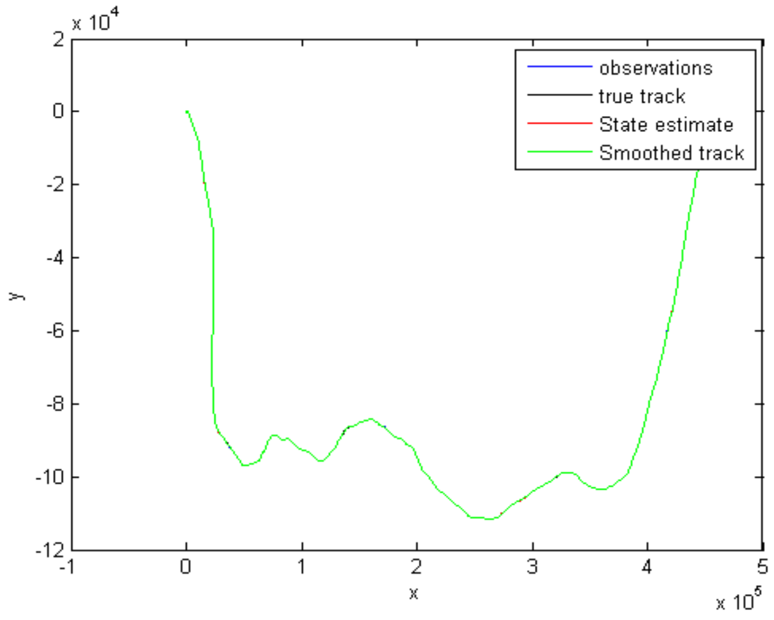


Figure 4.13: Synthetic track with two different model errors generated to test the EM algorithm at each time step

Constant Velocity

The constant velocity model assumes the aircraft is flying in a straight line with no acceleration applied. Here, we ran the EM algorithm for 10 iterations instead of 100, because error covariances stabilizes around the true value using 10 or less iterations.

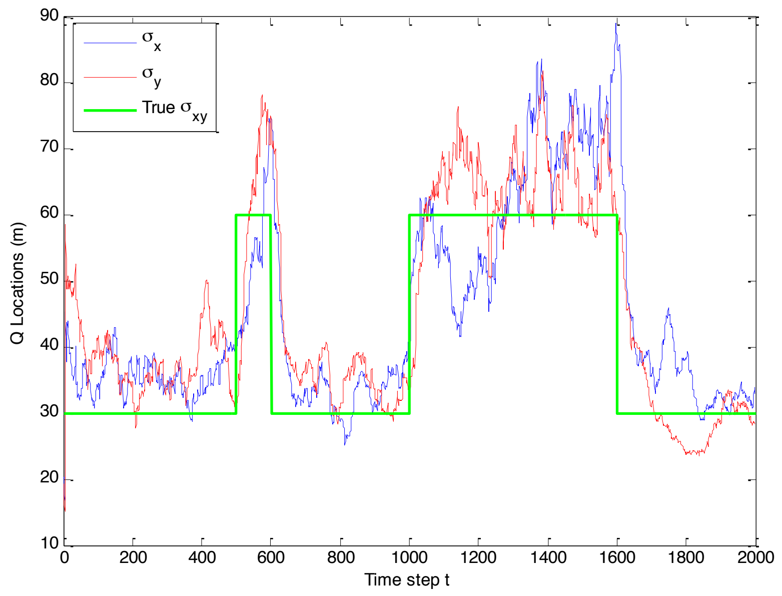


Figure 4.14: Best σ_x andσ_y after 10 EM iterations at each time t using CV Kalman filter

Using the history window method helped the error estimator to track the error as soon as the aircraft changed its direction. As can be seen in figure 4.14, between the time steps $1 \le t \le 500$ the uncertainties in the x and y locations were originally 30m and after running the EM algorithm 10 times at each time t we see that the error estimation tool indeed estimates approximately the true. Also when $500 \le t \le 600$ and $1000 \le t \le 1600$, the aircraft location errors increased from approximately 30 to around 60 which they originally were when we generated the true track.

Constant Acceleration

The constant acceleration model is being modeled as flying in either straight line or turning as long as there is an amount of acceleration applied during the flight. Here we use the same track in figure 4.13 to test our error estimation tool with CA Kalman filter in figure 4.14:

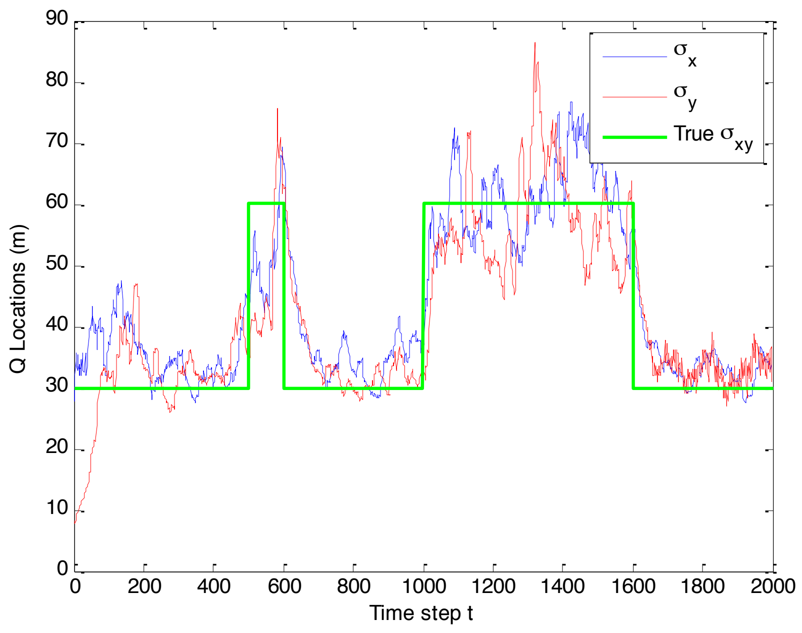


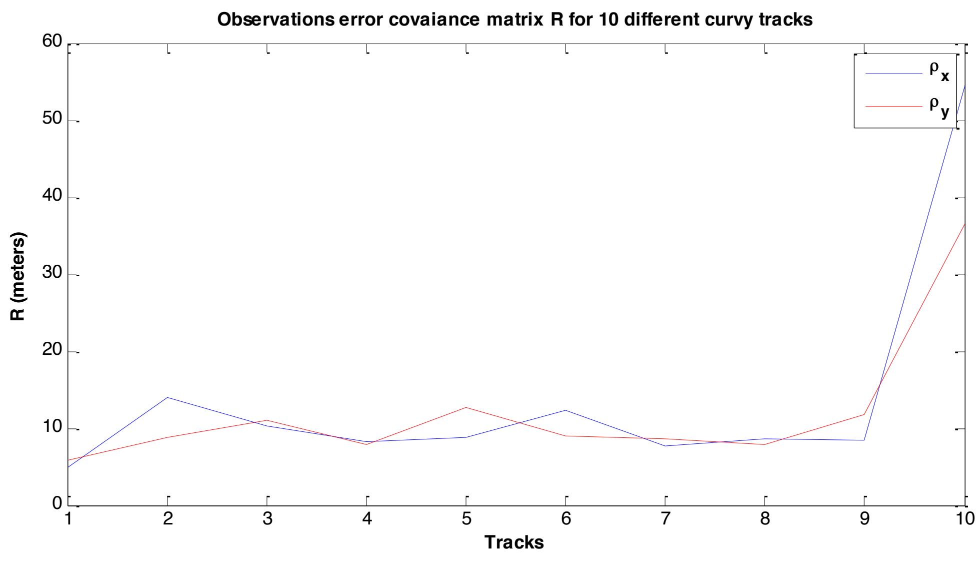
Figure 4.15: Best $\sigma_x$and $\sigma_y$after 10 EM iterations at each time t using CA Kalman filter

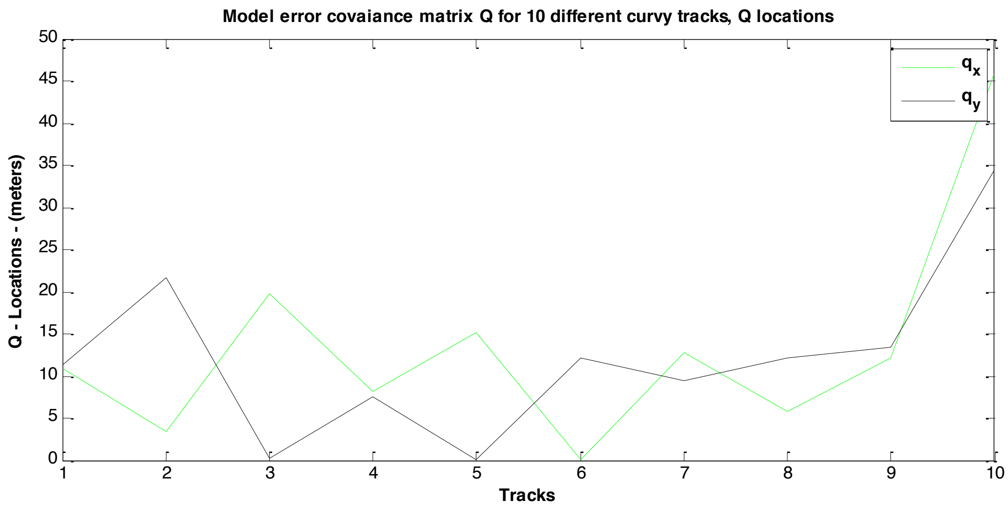
We can see that CA model also presents good estimation of the true error covariance during the whole flight. When the location uncertainty increases between 500 and 600 and between 1000 and 1600 in the true error covariance, the error estimation tool detects that and set the σ_{xy} closer to 60.

Now that we tested on synthetic track and were able to find the true errors in the x and y locations, we can now learn real tracks which we will present in the next section.

4.4 Testing on Real Tracks

For the real tracks we do not have any information regarding the amount of errors in the track for the observation system (radar) and the state system. Therefore, we applied the learning method to find out what are the typical amounts of errors in a sample of real tracks. We randomly selected 10 real tracks that are straight line and another 10 real tracks that have turns and learned their errors using the whole track learning method. In figures 4.16 and 4.17 we show these errors for the straight lines and the turning tracks respectively:





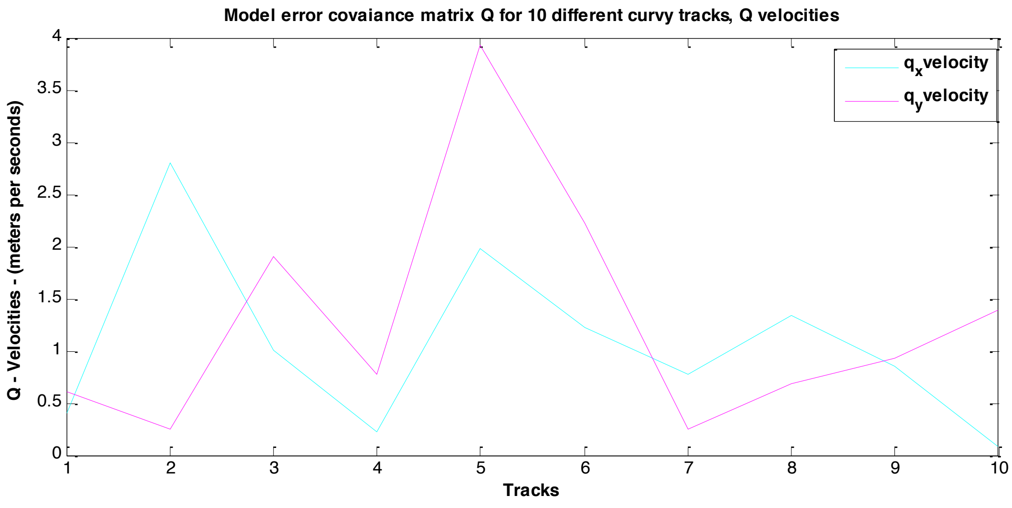
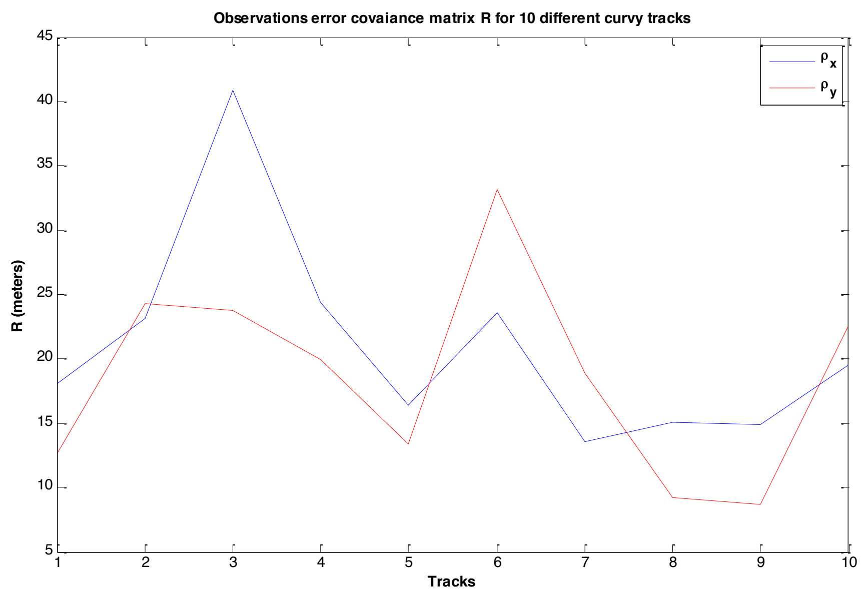
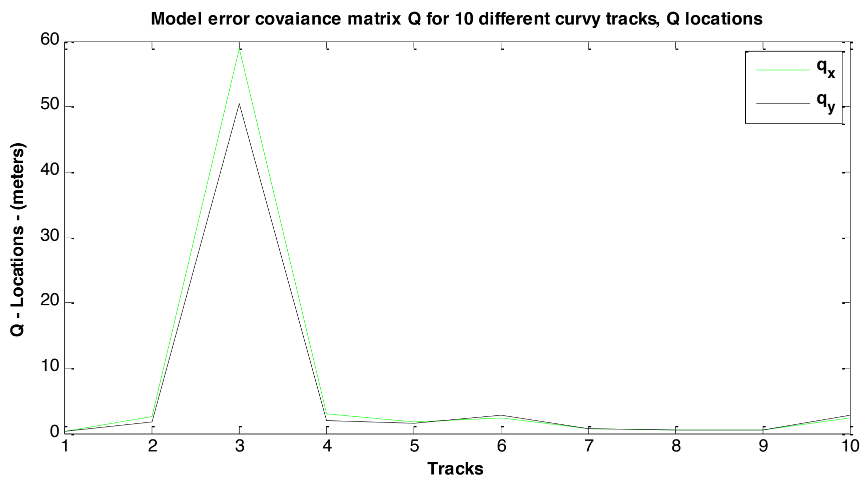


Figure 4.16: Learning the errors for 10 real tracks which are flying in a straight line





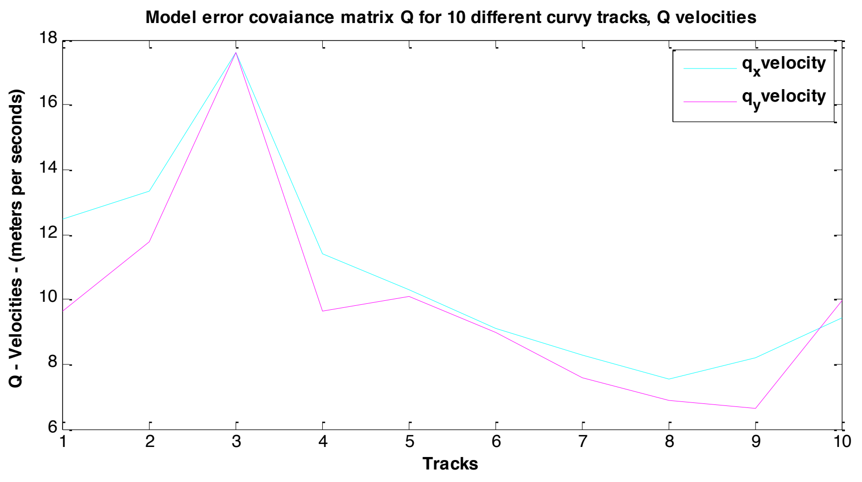


Figure 4.17: Learning the errors for 10 real tracks which are flying in a turning mode

We learn from these figures, that the average observation error ρ is approximately 15m, therefore, we will fix our observation error covariance to be $R = \begin{bmatrix} 15^2 & 0 \\ 0 & 15^2 \end{bmatrix}$ through out this research. In the next section we will use the learning at each time step method with both Kalman filters and test the probability of infringement methods we discussed in the previous chapter.

4.4.1 Constant velocity

We test our error estimation tool using this model and then estimate the probability of infringement given a 1 and 5 steps ahead predictions.

1-Step Prediction Learning

We have seen that the typical amount of error for the observation system (radar) was around 15m. We will use this error for the observation and fix it since the radar's error known to be almost constant. In figure 4.18, we applied the CV Kalman filter to a sample track and learned the model error covariance at each time step.

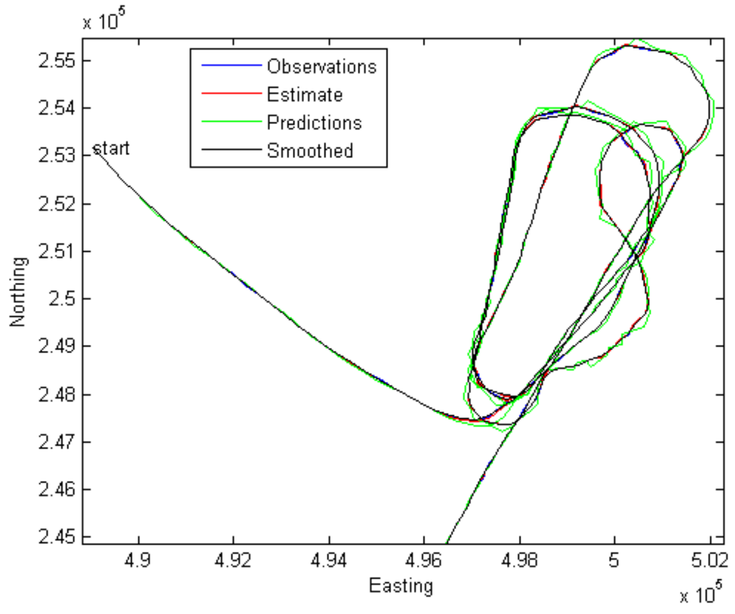


Figure 4.18: Real track after using CV model online learning

Figure 4.19 shows the model errors after we ran 30 EM iterations on the above track using CV model.

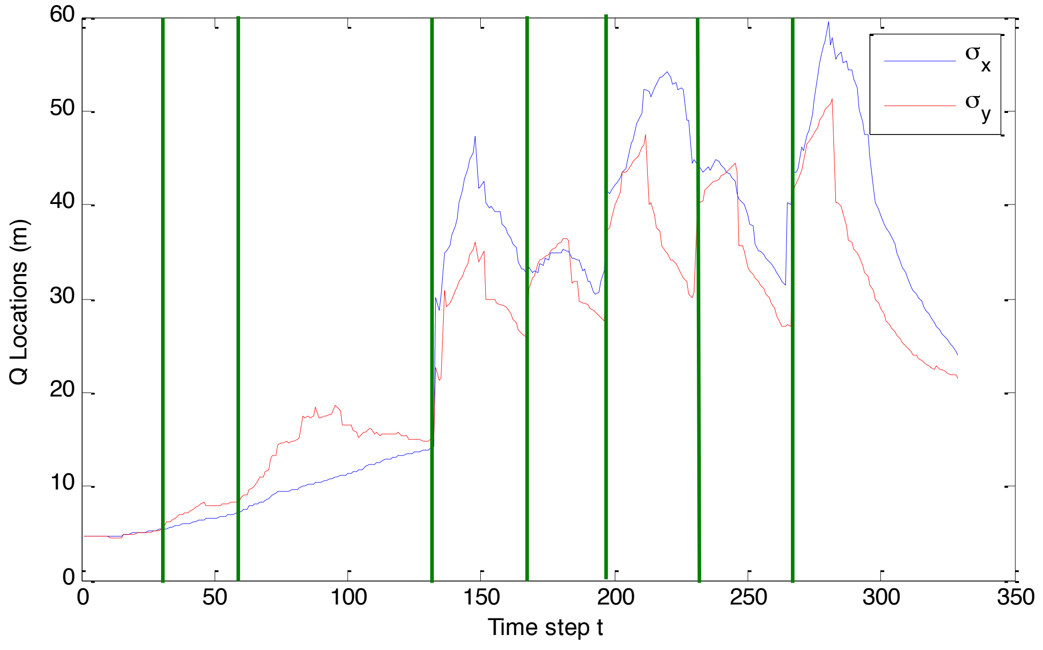


Figure 4.19: $\sigma_x$and σ_y after 20 EM iterations at each time step using the CV model

Notice in figure 4.19 that the location errors increase during the turning mode where the green vertical lines shows the locations on which the aircraft starts to turn, and since we are using the 15 step history window it will give it a chance to decrease back when it is flying in a straight line.

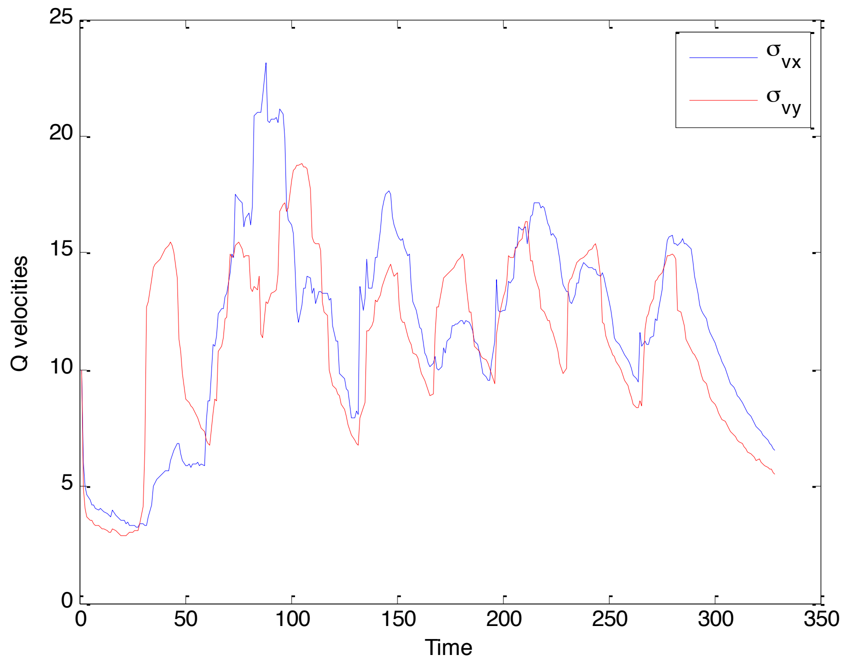


Figure 4.20: $\sigma_{vx}$and σ_{vy} after 20 EM iterations at each time step using the CV model

The uncertainties in the velocities shown in figure 4.20 increase because the aircraft is accelerating at it turns which means the velocity is no longer a constant. In the previous chapter, we mentioned the probability of infringement $P(I)$ would be accurate if we can make a good estimation of the model's error. In figure 4.20, we show the track we used to find the new $P(I)$ using the CV model and learning the errors at each time step:

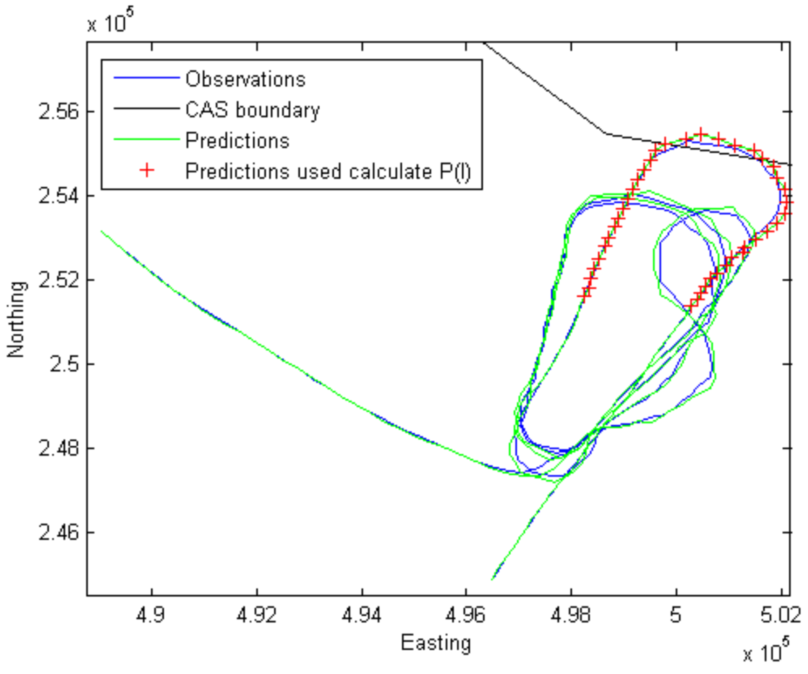


Figure 4.21: Partial real track showing the predictions which we will use to find their probability of infringements

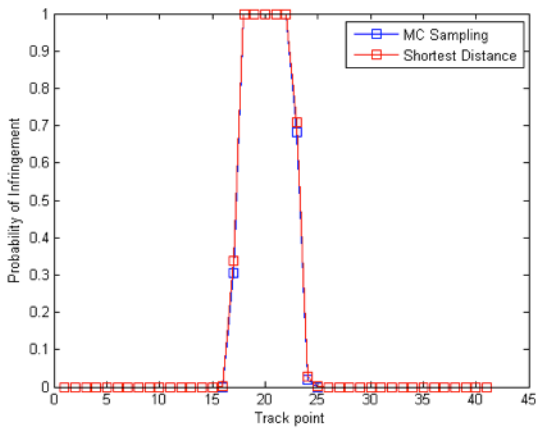


Figure 4.22: The probability of infringement using MC sampling and the shortest distance method after learning the errors for CV model

We can see in figure 4.21 that as the aircraft infringes the CAS boundary it gives a probability of at least $P(I) = 0.7$ and for the points inside CAS the $P(I) = 1$ in figure 4.22. We also calculated the probability of infringement using the whole track learning method for the same partial track in figure 4.21 and compared figures 4.22 with figure 4.23, and found out that even when the aircraft is inside the CAS it still did not give as high a probability as in the each time step learning where it gave $P(I) = 1$.

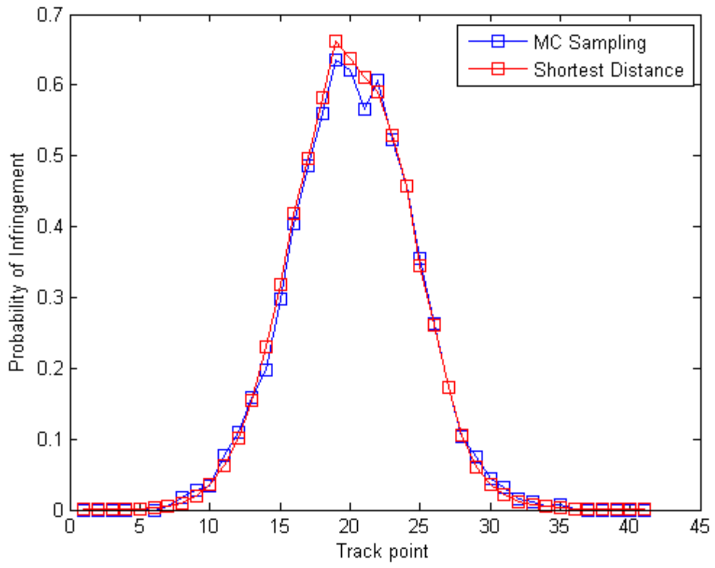


Figure 4.23: Probability of infringement after learning one error for the whole track using the CV model

The reason that the $P(I)$ in figure 4.23 are not high even when the aircraft is inside the CAS is because there was one average error for the whole track which was large enough for the system to assume that the aircraft could possibly be outside the CAS.

5-Step Prediction

Even though current CAIT warns ATCs when the infringement has already occurred, predicting one step ahead (4 seconds) for an infringement is not enough to warn them. Therefore, we want to predict more steps ahead such as 20 seconds and test the results by calculating the RMSE between the predictions and the observations. For simplicity we will use the track in figure 4.18 to show the prediction's RMSE and their probability of infringement.

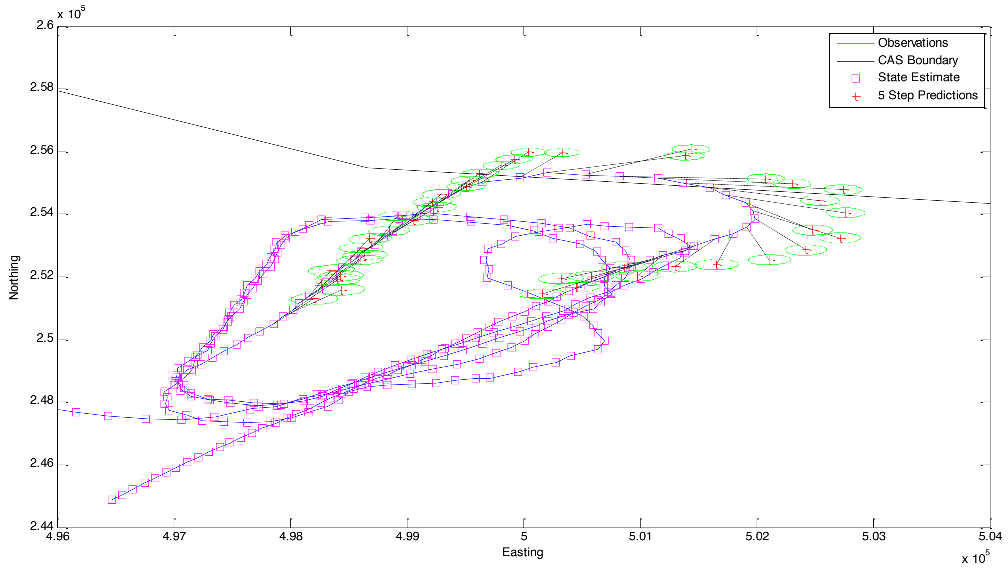


Figure 4.24: 5 steps ahead prediction using the CV Kalman Filter. The green ellipses represent the 5th step prediction's error covariance

Figure 4.24 shows state estimates (square) and their 5 step predictions (cross) where the black dashed lines correspond to each state estimate's prediction trajectory. As long as the aircraft flies in a straight line, we can predict an accurate locations of it, however, when the aircraft starts to change its direction the CV Kalman filter still predicts as if it was flying in a straight line and therefore, gives inaccurate predictions. We used these predictions and their error covariance (green ellipses) from figure 4.21 to calculate their probability using both probability of infringement methods.

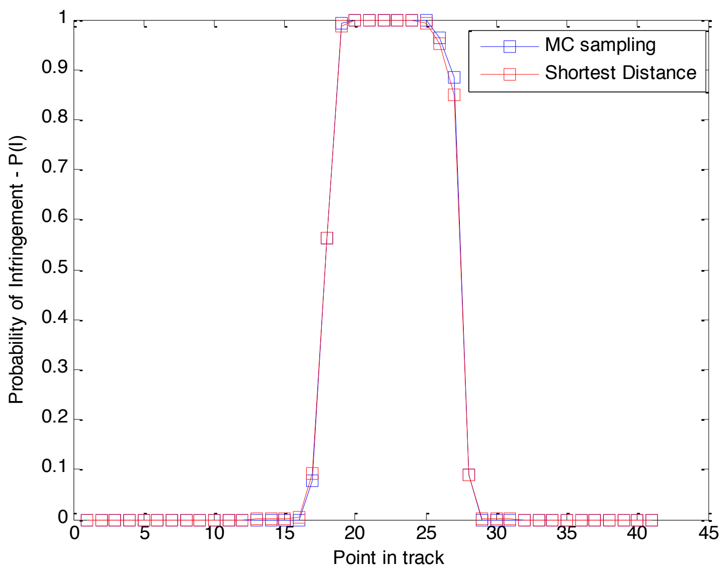


Figure 4.25: Probability of infringement for the 5th step predictions plotted in figure 4.24 using the CV Kalman filter

Figure 4.25 shows that even if we predict 5 steps ahead, we still get the same probability of infringement as using the 1 step ahead prediction which is more certain about the location. This suggests we can predict the infringement 20 seconds in advance with minimal errors where the RMSE for the 5 step predictions (red crosses) was 712m.

4.4.2 Constant acceleration

Here we will test our learning tool with the CA Kalman filter again using a real track and then find the probability of infringement given 1 and 5 steps ahead predictions.

1-Step Prediction

Here we use the CA Kalman filter/smoother for tracking on the same track as in figure 4.18; we also learned it at each time step and plotted their errors in figure 4.26, 4.27 and 4.28:

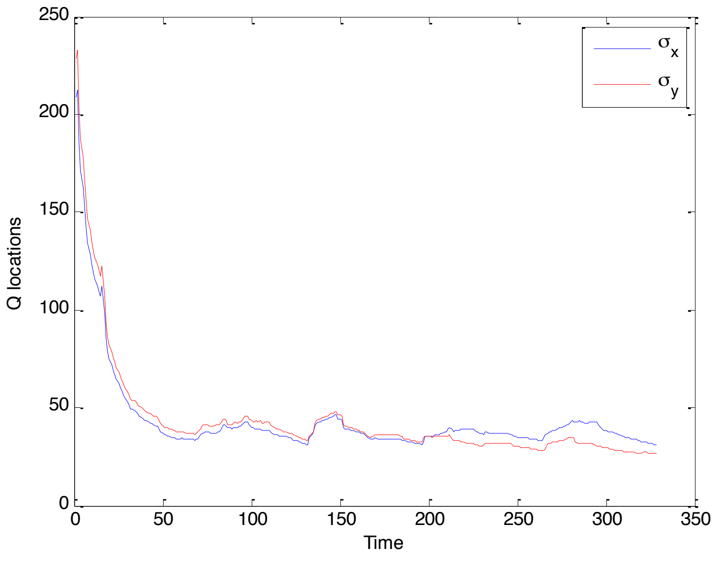


Figure 4.26: $\sigma_x$and σ_y after 20 EM iterations at each time step using the CA model

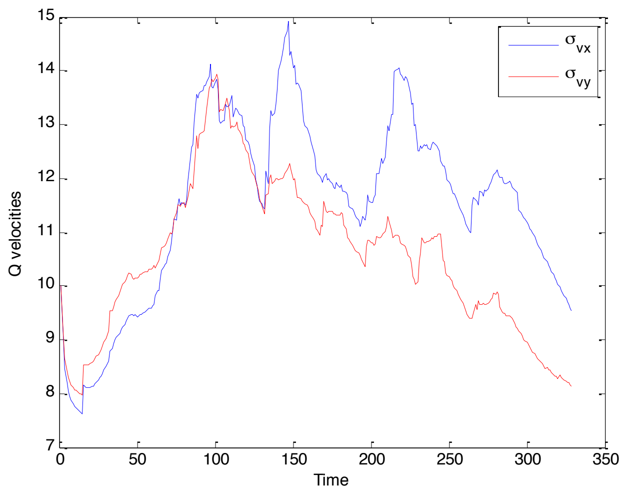


Figure 4.27: $\sigma_{vx}$and σ_{vy} after 20 EM iterations at each time step using the CA model

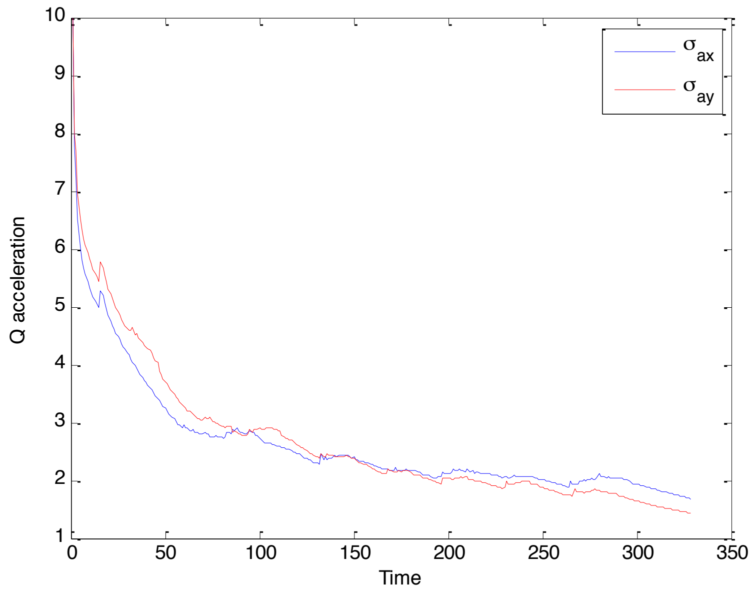


Figure 4.28: $\sigma_{ax}$and σ_{ay}after 20 EM iterations at each time step using the CA model

The CA model takes into consideration the velocity's uncertainty which we saw in figure 4.28. Also, notice in figure 4.27, the velocities' uncertainty were decreased compared to the velocities' uncertainty in CV model in figure 4.20 because here the CA model assumes that the velocity is not constant where as in the CV model when the aircraft changes its heading, it is no longer certain where the aircraft will be therefore load all the errors in the velocities. After learning the errors for the CA model, we then tested this model to find the $P(I)$ using each time step learning.

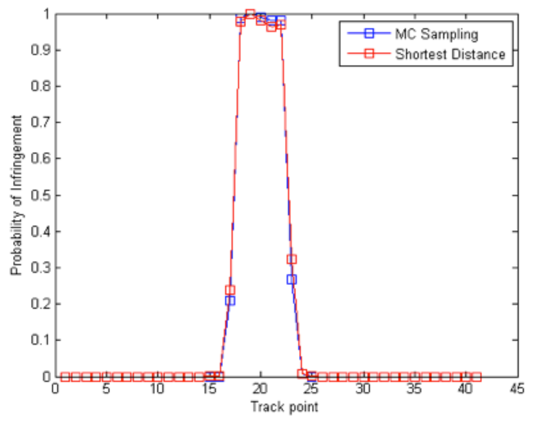


Figure 4.29: The probability of infringement using MC sampling and the shortest distance method after learning the errors using the CA model

Figure 4.29 shows the probability of infringement using the CA model after learning the errors at each time step. The five points on the track which fell inside CAS has a $P(I) = 1$, while the aircraft is turning this is when an acceleration is applied to leave CAS, the location of the aircraft is predicted to be somewhere close to the true location therefore, gives $P(I) = 0.25 - 0.3$ instead of 0.6 in the CV model.

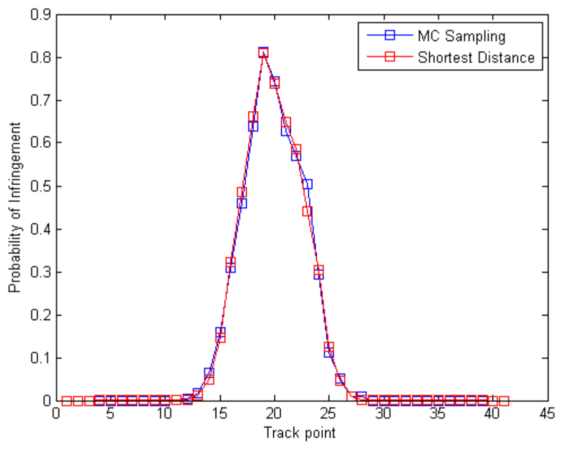


Figure 4.30: Probability of infringement after learning one error for the whole track using the CA model

Figure 4.30 shows the probability of infringement for the CA model when we used one fixed best error for the whole track. Again, here the probability of infringement is not equal to 1 when the aircraft is inside CAS because the one error for the whole track we found to be the best, is large enough to assume that the aircraft could possibly be outside CAS.

5-Step Prediction

We show in figure 4.31 a partial track we used to find their 5-step predictions, calculate their $P(I)$ and their RMSE.

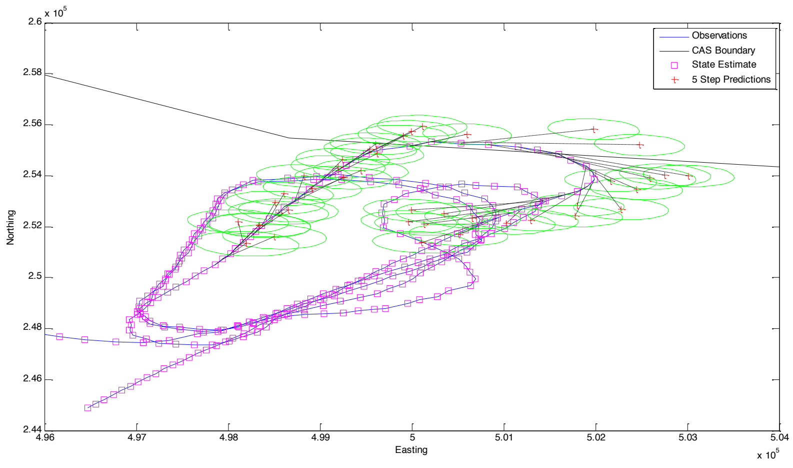


Figure 4.31: 5 steps ahead prediction using the CT Kalman Filter. The green ellipses represent the 5th step prediction's error covariance

By comparing figures 4.31 and 4.24, it clearly shows that the CA Kalman filter predicts better when the aircraft starts to turn as opposed to the CV Kalman filter which assumes that the aircraft will always fly in a straight line. Also in figure 4.31, notice the prediction error covariances (green ellipses) are larger than the CV prediction error covariances shown in figure 4.24, the reason for this increase is that CA prediction error covariance includes acceleration uncertainty added to the prediction location as appose to the CV location error covariance where only the velocity's uncertainty is added to the location. Again, we estimate the $P(I)$ given these predictions and their uncertainties shown in figure 4.32.

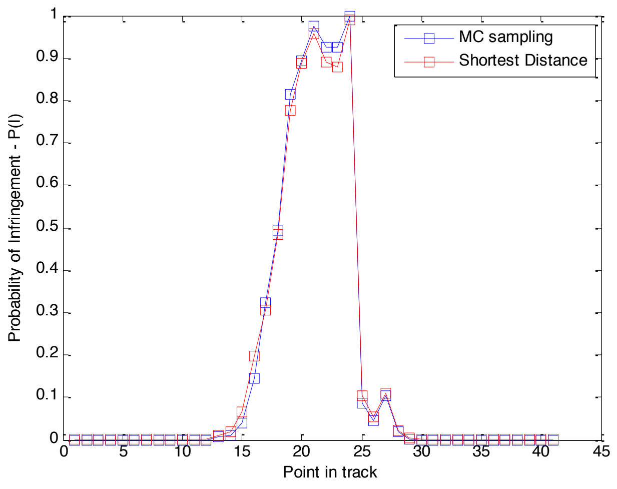


Figure 4.32: Probability of infringement for the 5th step predictions plotted in figure 4.22 using the CT Kalman filter

Comparing figure 4.32 to figure 4.29 which is the $P(I)$ for the 1-step ahead CA KF prediction, we see that 5- step prediction probability of infringement is performing similar to the 1-step model. This tells us that we are certain about the probability of infringement 20 seconds in advance as it is for the 4 seconds. Moreover, the aircraft was turning during the infringement therefore, CA KF predicted better in terms of the locations which gave RMSE=619m as opposed to the CV KF for which was RMSE=723m.

4.5 Conclusion

In this chapter, we proposed two models for tracking aircraft; the constant velocity and the constant acceleration models. Each model works in parallel for any track and both produce the optimal state estimate and error covariance. However each model performs differently in terms of the state components and the amount of error in them. After implementing these models, we learned their errors using the EM algorithm given the whole track on synthetic tracks. We then extended this learning method so that it learns the model's error at each time step and then tested it on the synthetic tracks, knowing the amount of true error for the track, and we succeeded in finding the approximately true errors when we learned them at each time step using 20 EM iterations. After seeing the results and how it can find the true errors of the model, we learned the models' errors for both models (CV and CA) given the whole track and found out approximately the amount of errors in the observation and in the model. We fixed the amount of observation error to be 15m after learning it for several real tracks using N EM iterations. After running 20 EM iterations using both models on the real track, we learned the amounts of uncertainties present during flight time and notice that we were given a good estimation especially after we tested the probability of infringement on both models. Since these errors were learned, we were able to apply the probability of infringement methods to predict future infringements more accurately from 4 to 20 seconds ahead. The Kalman filter performs generally well when we predict one step ahead, but as we predict ahead in time the amount of errors increase linearly, as a result, when we predict 5 steps ahead Kalman filter can project the prediction slightly further away from observed location at that time. The CA Kalman filter, however predicts closer to the observation than the CV model.

As we mentioned before each model performs well depending on the aircraft's behaviour at that time for example, the constant velocity works and predict future infringements well as long as the aircraft flies in a straight line, and If so the amount of errors will increase as we have seen in figure 4.14. As a result, we will review in the next chapter a switching method where we will be able to use in order switch to the proper model when appropriate.

5 Switching Kalman Filter

In chapter 4 we proposed two Kalman filters one for each model: Constant Velocity and Constant Acceleration. We have seen that one Kalman filter can predict the locations of the aircraft better than the other whenever the behaviour of that aircraft matches the modeling assumptions of the respective Kalman filter. For instance, the CA Kalman filter predicts better than the CV Kalman filter when the aircraft is turning but not in a straight line. We need to make sure that our model is capable of predicting the locations more accurately irrespective of whether the aircraft is accelerating or flying in a constant velocity and whether this varies during flight. To do that we need a switching method that can assist our model to switch to the appropriate Kalman filter when flight behaviour changes. In this chapter, we first discuss the reasons behind developing a switching tool, then review the Switching Kalman Filter (SKF) proposed by Murphy (2000), which we use in our research. The switching Kalman filter is a model used to switch between two or more Kalman filters which produces minimum RMSE of the future prediction. Given this model we will try to find the probability of future infringement using the SKF. Finally, we will show several examples of using the SKF on real tracks and compare its results to the use of the Kalman filters separately.

5.1 Why The Switching Kalman Filter

The Kalman filter is an efficient tool for tracking and prediction when applied to linear dynamical models as mentioned previously in Kalman Filters section 2.5 chapter 2. It uses the previous state estimates to predict the future hidden state, when the next observation becomes available the Kalman filter then correct the prediction and its uncertainty.

Suppose we have more than one model $\lambda^m = (\theta^m, \pi^m)$ each has its own distinct parameters $\theta = \{A, H, Q, R\}$ and initial values π , can we use one flexible Kalman filter for these models? Yes we could but there is a downside to it, which is over fitting the errors. Here we assume that we have one fixed parameters for the entire flight. This assumption causes poor prediction, as a result inaccurate probability of future infringement. To clarify this issue, suppose the Kalman filter tracks an aircraft shown in figure 5.1 which flies in a constant velocity mode, it will perform well as long as the aircraft does not change the direction of its motion. If it does, the Kalman filter will then result in bad estimations of the errors and provides inaccurate predictions.

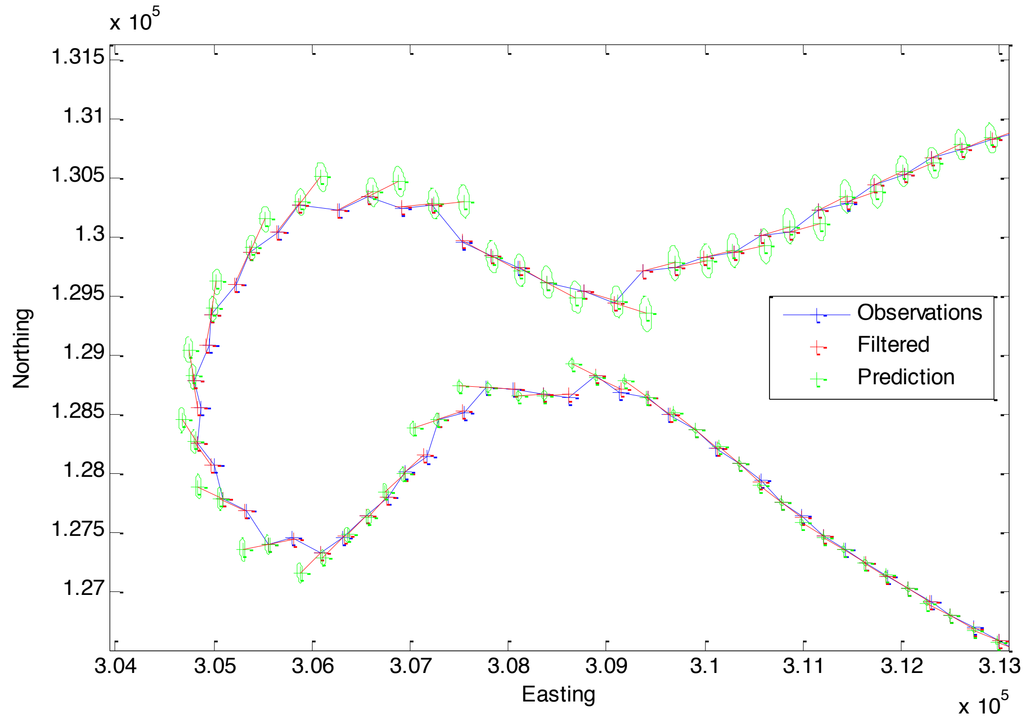
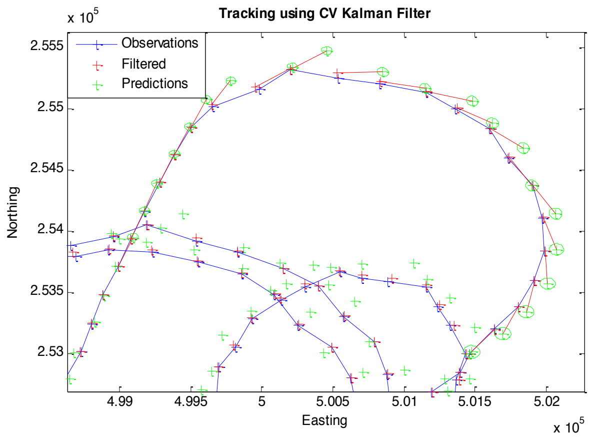


Figure 5.1: Constant Velocity Kalman Filter on a real aircraft track showing the increase of the error when the aircraft turns. The aircraft starts flying in the bottom right.

Notice in figure 5.1 that the aircraft was moving initially in a straight line with a constant velocity (lower right corner) then at some point it changes its direction or mode of flight to a constant acceleration mode. When the aircraft was flying in a constant velocity, the predictions are estimated with lower uncertainties or "prediction error covariance" as its transits from one state estimate to another. However, when the acceleration was applied during the turning mode, the Kalman filter starts its prediction phase given the previous state which was in constant velocity mode therefore, assumes the aircraft is still flying in straight line. The inaccuracies occur during the correction phase, when the Kalman filter has observed a new location for the aircraft which turns out to be far away from the prediction, as a result, the residual and its covariance will increase.

Clearly for this particular problem we need more than one Kalman filter or more flexible model where each one deals with specific specific mode. In response to this we implemented the two separate Kalman filters (CV and CA) in the previous chapter. Figure 5.2 shows both Kalman filter's predictions and estimations. The CV Kalman filter assumes the aircraft is flying in a straight line, therefore, always predicts the locations straight ahead. The CA Kalman filter however, performs better than the CV model when an acceleration is applied.



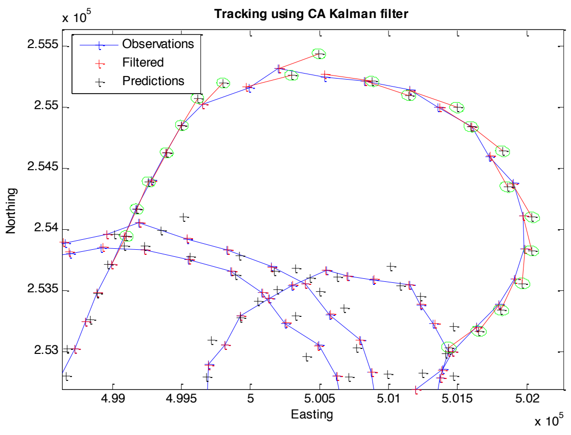


Figure 5.2: After using two separate Kalman filters to track an aircraft, CV KFs (left) predicts further away than the CA KF (right)

The prediction RMSE for the CA Kalman filter was 62m whereas the CV Kalman filter's prediction RMSE was 92m.

Given these Kalman filters we need to use the most appropriate one of the two which gives lower prediction RMSE. As a result, these Kalman filters then have to switch at some point whenever a specific model should be used as our most reliable tracker. This property of switching distinguishes the SKFs from the default Kalman filter where it minimizes the root mean squared error (RMSE) for the predictions when a sudden direction change occurs during tracking.

5.2 Implementation

5.2.1 Objective

Our research objective is to find the probability of infringement $P(I)$ in the future more accurately using a SKF. We examined each KF separately and now we need to switch between them when appropriate as in figure 5.3. The KF with the smaller prediction RMSE will be our most reliable tracking tool at that time.

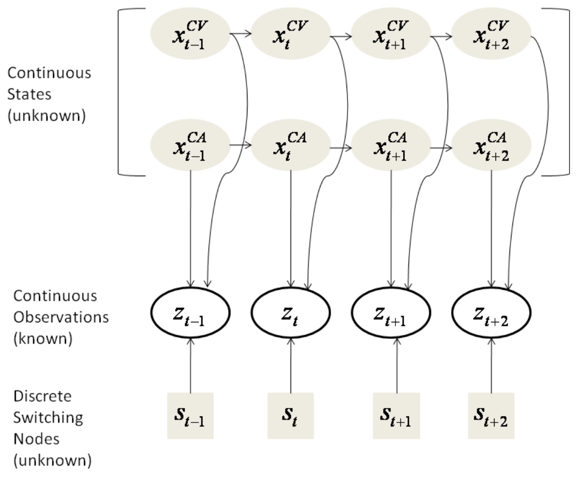


Figure 5.3: Switching Kalman Filter process where two kalman filters run in parallel given the observations. The KF which gave higher probability of the observation and lower RMSE is chosen by $S = CV\ or\ CA$

In figure 5.3, the shaded oval and square nodes correspond to the hidden state (calculated in the Kalman filters) and the switching nodes (learned in the HMM chapter 2) respectively. The most probable switching state sequence S defines which of the hidden states gave higher probability of observation at any time t.

5.2.2 Switching Process

The switching process occurs after we find the most probable switching state given the observation and the model. Switching nodes are assumed to be discrete and they are either in CV or CA mode, meaning that at any time t the model is in either the CV or the CA state. The optimal tool for learning these discrete states is the HMM reviewed in chapter 2. The observation sequence that we will use in the HMM is the probability of the error we calculate in the Kalman filter:

$$P\ (e_t|r_t, J_t) = \mathcal{N}(e_t|r_t, J_t)$$

where r is the residual which is the difference between the observation and the prediction at time t and J is the residual covariance.

We have also seen in chapter 2 that the HMM solves the three basic problems where it produces the outputs α, β, where α-β is the probability that a specific model generated the observations and γ is the maximum likelihood of being in a specific state S. The key idea in the SKF's is to pick the Kalman filter which gives lower RMSE in the predictions, In order to determine that, we need to find γ the maximum likelihood of being in state CV or CA in the future. We use this probability to weight our model's error covariance matrix Q inside the error estimator tool (M-step: learning phase). Algorithm 5.1, shows the modified version of this learning process:

Algorithm 5.1: $(Q_t^m, R_t^m) = \mathrm{ErrorEstimation}\left(Z, \tilde{X}, \tilde{P}, \widetilde{PP}, t, W_t^m\right)$

```
Require: $\mathrm{Z}$                      Observation matrix
Require: $\tilde{\mathrm{X}}$                      Smoothed states matrix
Require: $\tilde{\mathrm{P}}$                      Smoothed error covariance matrix
Require: $\widetilde{\mathrm{PP}}$                     Smoothed error covariance matrix between time t-1 and t-2
Require: $W_t^m$                    Weight given by the HMM for each mode where $W^m = \gamma^m$
m= 0 or 1                     Flight mode CV=0, CA=1
$\mathrm{D, E, F} = 0$
$\mathrm{t} = 1$
R=0
HistWindow=15                 History window of 15 time steps back

if $\mathrm{t} <= \mathrm{HistWindow}$ then
  N= $t$
  $\mathrm{t} = 2$
else
  $\mathrm{N} = \mathrm{HistWindow}+2$
  $\mathrm{t} = \mathrm{t} - \mathrm{HistWindow}$
end if

while $t \leq N$ do
    R= R+$\mathrm{W_t^m}[(z_t - \mathrm{H_m}\tilde{\mathrm{x}}_t^{\mathrm{m}})(z_t - \mathrm{H_m}\tilde{\mathrm{x}}_t^{\mathrm{m}})' + (\mathrm{H_m}\tilde{\mathrm{P}}_t^{\mathrm{m}}\mathrm{H_m}')\}$
    $t = t + 1$
end while
                              Observation error covariance re-estimation
$R_t^m = \frac{\mathrm{R}}{(\sum_{\mathrm{t=t}}^{\mathrm{N}} \mathrm{W_t^m})}$

While $t \leq N$
    $D = D + \mathrm{W_t^m}(\tilde{x}_{t-1}\tilde{x}_{t-1}' + \tilde{P}_{t-1})$
    $E = \mathrm{E} + \mathrm{W_t^m}(\tilde{x}_t\tilde{x}_{t-1}' + \widetilde{PP}_t)$
    $F = \mathrm{F} + \mathrm{W_t^m}(\tilde{x}_t\tilde{x}_t' + \tilde{P}_t)$
    $t = t + 1$
end while
                              Model error covariance re-estimation
$Q_t^m = \frac{(F - EA' - AE' - ADA')}{(\sum_{\mathrm{t=t}}^{\mathrm{N}} \mathrm{W_t^m})}$
return $Q_t^m, R_t^m$
```

This version of learning the model's error is analogous to the original learning tool used in the KF error estimator tool we have seen in chapter 4, except here we have weighted the equations using W^m. The overall process of Filtering, smoothing and learning are presented in figure 5.4:

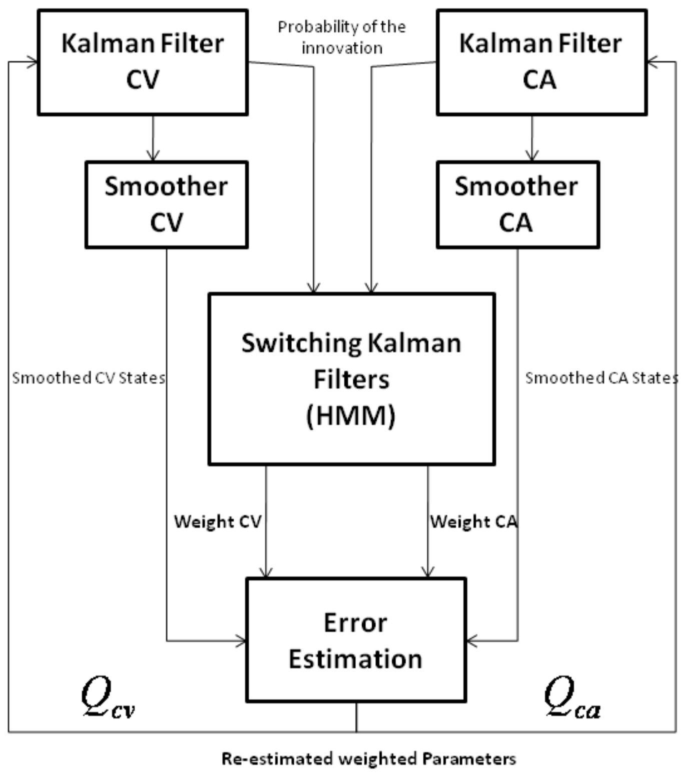


Figure 5.4: Switching Kalman filters process

Figure 5.4 shows the optimal filtering process which goes as following:

Filtering/smoothing:

1. Both Kalman filters state estimates and parameters are initialized to zero
2. The filters then predict using the previous state estimate or zero if no information of the previous state estimate is available.
3. The time increments and the Kalman filters receive a new observation from the radar. The filters calculate the innovation that is the difference between the

prediction and observation and finds its covariance, the average error for the prediction and the observation, these are essential to find γ in the HMM.

4. The optimal state estimates and their covariance are then the outputs of the Kalman filters.
5. The state estimates are then smoothed in the Kalman smoother, to filter out any additional noise. The smoothed estimates are crucial for learning the parameters.

Learning (switching state/new parameters):

1. The HMM will use the probability of the innovation calculated in the KFs to find the maximum likelihood of state at time t gamma γ_t
2. The smoothed estimates from the Kalman smoother along with the weights γ_t from the HMM are used to re-estimate the Kalman filters' parameters (Q_t^m, R_t^m).

We iterate through this process until the radar is no longer observing the aircraft.

Algorithm 5.2 shows the optimal prediction and filtering process:

Algorithm 5.2: Overall Process of the SKF Algorithm

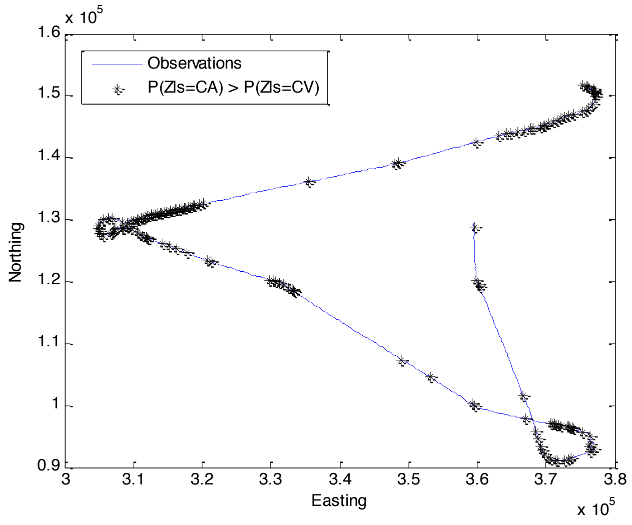
```
Require: $Z$                                                  Observations matrix
Require: Q                                                    State error covariance
Require: R                                                    Observation error covariance
Require: A                                                    State transition matrix
Require: H                                                    Observation transition matrix
Require: N                                                    Total Length of journey time step
$\bar{x}_1 = z_1$
$\bar{P}_1 = HRH'$
iters = 0
MaxIterations= 100
$\mathrm{m} = \text{number of models}$
for t=2 to N

 While iters < MaxIterations do
                                                              E-step
    $(\hat{x}_t, \widehat{P}, \bar{x}_t, \overline{P}, L_t^{cv}) =$                  Kalman Filtering
    $\mathrm{FilterCV}(\hat{x}_{t-1}, \widehat{P}_{t-1}, \bar{x}_t, \overline{P}, Q_t^{cv}, R_{cv}^*, H_{cv}, A_{cv}, t)$   Constant Velocity Kalman Filter
    $(\hat{x}_t, \widehat{P}, \bar{x}_t, \overline{P}, L_t^{ct}) =$
    $\mathrm{FilterCA}(\hat{x}_{t-1}, \widehat{P}_{t-1}, \bar{x}_t, \overline{P}, Q_t^{ct}, R_{ct}^*, H_{ct}, A_{ct}, t)$   Constant Acceleration Kalman Filter

                                                              Kalman Smoothing:
    $(\tilde{x}_{t-1}, \widetilde{P}_{t-1}, \widetilde{PP}_t) = \mathrm{SmootherCV}(\hat{x}_t, \widehat{P}_t, R_t^m, H_{cv}, A_{cv}, t)$   Constant Velocity Kalman Smoother

    $(\tilde{x}_{t-1}, \widetilde{P}_{t-1}, \widetilde{PP}_t) = \mathrm{SmootherCA}(\hat{x}_t, \widehat{P}_t, R_t^m, H_{ct}, A_{ct}, t)$   Constant acceleration Kalman Smoother
                                                              Switching process:
    $(\alpha_{1:t}, \beta_{1:t}, \gamma_{1:t}) = \mathrm{HMM}(L_t^{cv}, L_t^{ct}, t)$          calculating maximum likelihood of being in
    $W_{1:t} = \gamma_{1:t}$                                  specific state $m$ at time $t$
                                                              M- step:
    $(Q_t^m, R_t^m) = \mathrm{ErrorEstimation}((\tilde{x}_{t-1}, \widetilde{P}_{t-1}, \widetilde{PP}_t)^m, W_t^m)$   Re-Estimating Models' errors using the
    $\mathrm{iters} = \mathrm{iters} + 1$                     weight $W_t$
 end while
Next t
```

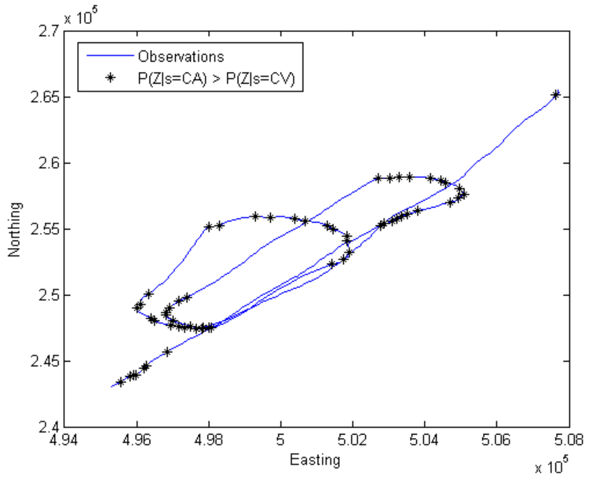
5.3 Testing

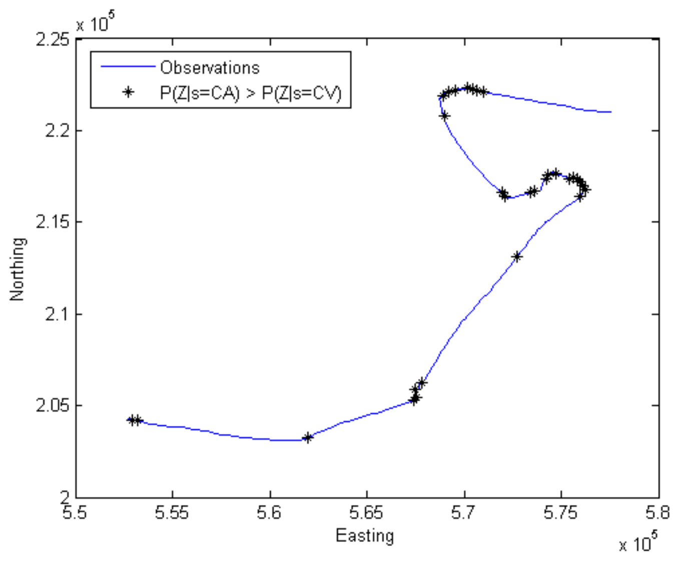
1- Step Prediction

We tested the SKF on a sample of real tracks and calculated the probability of being in state CA or CV at any time t. We also calculated the 1- Step ahead prediction RMSE using the SKF, CV Kalman filter and CA Kalman filter. Figure 5.5 shows 4 sample tracks where the observations marked with (*) are when the observations probability is higher in mode CA than it is in mode CV:

$$P(z_t|S_t = CA) > P(z_t|S_t = CV)$$







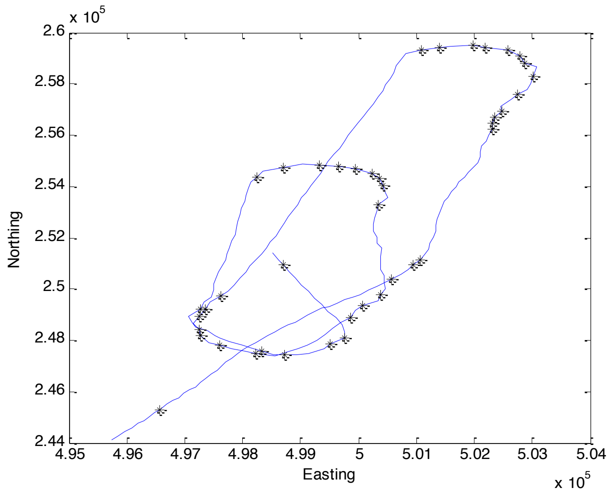


Figure 5.5: Real tracks showing in (*) the observations which have higher probability in CA than the CV model

Whenever there is acceleration, the CA model has a higher probability than the CV model which tells us that when to switch to CA model. The rest of the observations that are not marked in (*) are the observations when their probability are higher in CV mode than it is in the CA mode. Given these tracks in figure 5.5, we calculated their prediction RMSE before using the SKF and after then we compared the results which are listed in table 5.1:

Table 5.1: RMSEs for tracks in figure 5.5 for the two models before then after using the SKF

Track	Model	RMSE Prediction (m)
1st Track	CV	133
	CA	151
	SKF	128
2nd Track	CV	73
	CA	82
	SKF	65
3rd track	CV	65.1
	CA	75.9
	SKF	56.8
4th track	CV	84
	CA	99
	SKF	74

We noticed from table 5.1 that 1-step ahead prediction RMSE for the SKF performs better than CV or CA Kalman filters. We used the same tracks in table 5.1 and an additional 10 real tracks, where each of which have at least 300 time steps and calculated their prediction error (difference between the observation and the prediction at each time step) instead of taking the averaged error over all tracks. To determine which of the three models: the SKF, CV KF and the CA KF is significantly better than the other, we used the signed Wilcoxon rank test which performs a paired two-sided signed rank test of the null hypothesis that the model errors in each model come from the same continuous, symmetric distribution with zero median, against the alternative that they do not come from the same distribution (Lowry, 2000). In order to reject or accept the null hypothesis, we will test the p value generated by this test which is the probability that the error came from the same distribution against the significant value which is $\alpha = 5\%$. If the $p < \alpha$, we will reject the null hypothesis otherwise accept it.

We applied the signed wilcoxon rank test on a large matrix where each row corresponds to a time step error (difference between the prediction and the observation) for all 14 tracks and the columns correspond to the three models: the SKF, CV KF and CA KF. In order to assure the independency between time steps (samples), we applied the test on every 20th row (i.e. a separation of 80 seconds) .

Figures 5.6, 5.7 and 5.8 show plots of each 20th time step error for one model against the error at the same time step for another model:

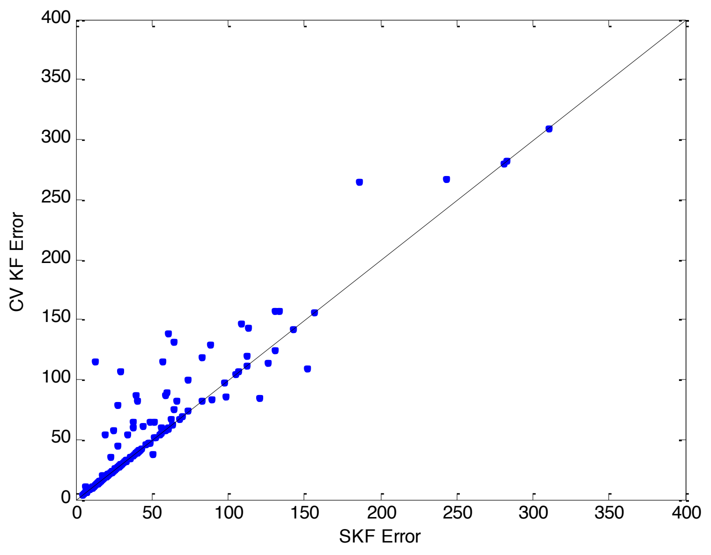


Figure 5.6: 1-step ahead prediction errors using SKF, CV Kalman filter where it appears that the SKF performs better compared to the CV KF.

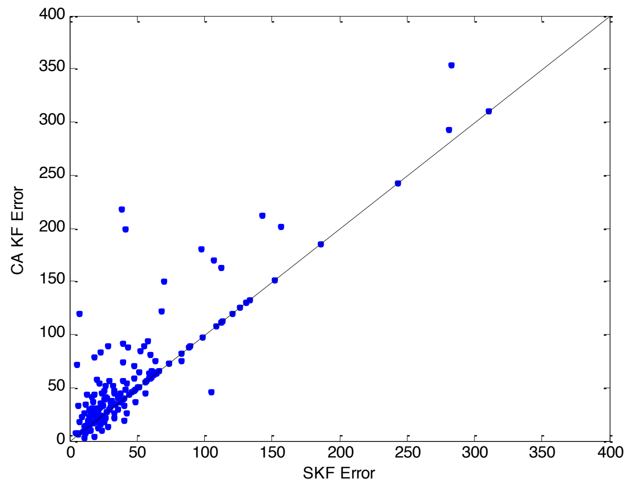


Figure 5.7: 1-step ahead prediction errors using SKF, CA Kalman filter where it appears that the SKF performs better compared to the CA KF.

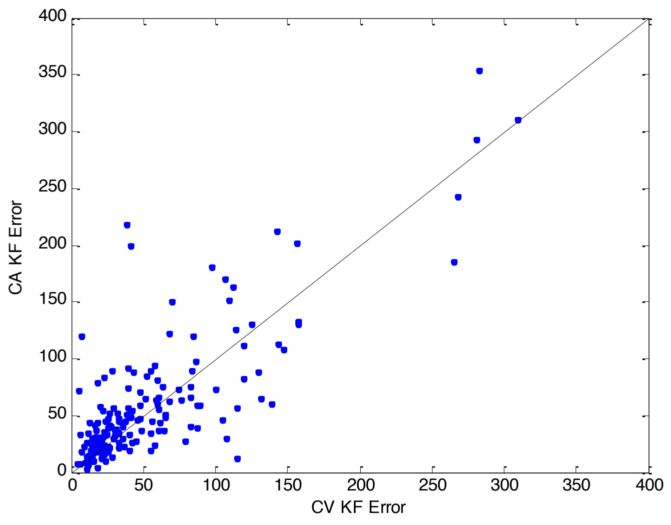


Figure 5.8: 1-step ahead prediction errors using CV KF and CA Kalman filter where it we are unclear of which one of the KFs is better than the other.

Figures 5.6 and 5.7 show that the SKF is significantly better than the CV and CA KFs where the propensity of the errors is higher or equal in both KFs than it is in the SKF. The signed wilcoxon rank test made on each pair of models errors returned $p_{\ (SKF,CV\ KF)}$= 1.3109e-006 and $p_{(SKF,CA\ KF)}$=1.2171e-011 which tells us to reject the null hypothesis that these errors come from the same the distribution. Figure 5.8 , is not clear visually whether one KF is better than the other however, the signed wilcoxon rank test returned a $p_{(CV\ KF, CA\ KF)}$ = 0.0188 therefore, there is a significant difference at the 5% level.

Since one step ahead prediction is not enough for future warning, we therefore, used the SKF to predict 5 steps ahead instead of one which allows us to predict future change; this will permit us with better estimates of future CAS infringements.

5- Step Prediction

Given the both Kalman filters and observation at time step t, we calculate the probability of that observation given the models. The Kalman filter with the higher probability will tell us if there will be a switch at time step $t+1$, using the same probability we will extend it to be able to switch at time $t+5$ given the prediction at time t. For example, if at time step t we are in state CV model and the probability of the observation at time $t+1$ is higher for the CV Kalman filter, then the probability of being in state CV at time step $t+5$ is higher for the CV than the CA. In this section, we will use the signed Wilcoxon rank test to test which of the three models is performing better than the other when predicting 5 steps ahead. Using the same tracks as in the previous section and predicting 5 steps ahead, we generated a similar matrix as in the 1-step ahead prediction error on which to apply the test but this time, the errors are the difference between the 5 step ahead prediction and the observation at time $t+5$. Again to assure the independency between the samples, we applied the test on every 20th time steps. Figures 5.9, 5.10 and 5.11 show plots the 5th step ahead prediction error for one model against the errors at the same time step for another model where the SKF appears to be significantly better than both CV KF and CA KF as shown in figures 5.9-5.10 whereas in figure 5.11 it is unclear which whether either of the CV or CA KFs are actually different than each other in performance:

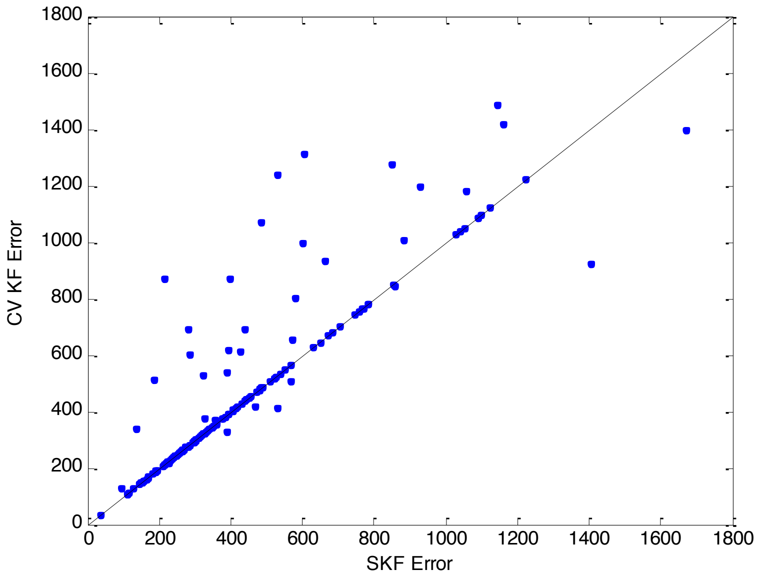


Figure 5.9: 5-steps ahead prediction errors for each time step of all 14 tracks using the SKF and CV where it is appears that the SKF is better than the CV KF.

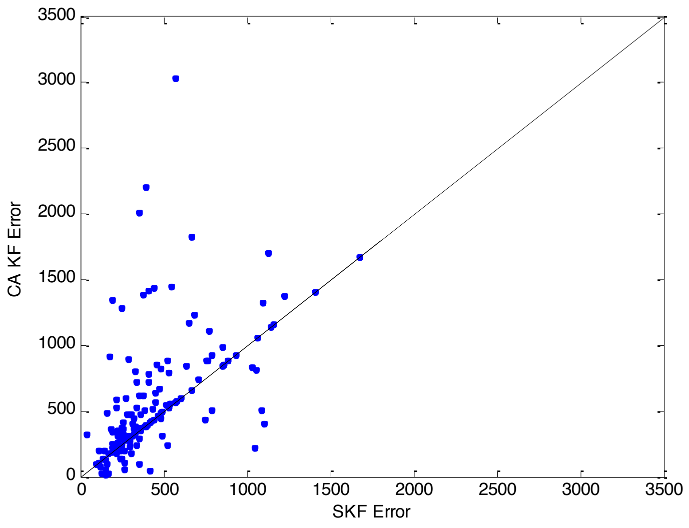


Figure 5.10: 5-steps ahead prediction errors for each time step of all 14 tracks using the SKF and CA where it is appears that the SKF is also better than the CA KF.

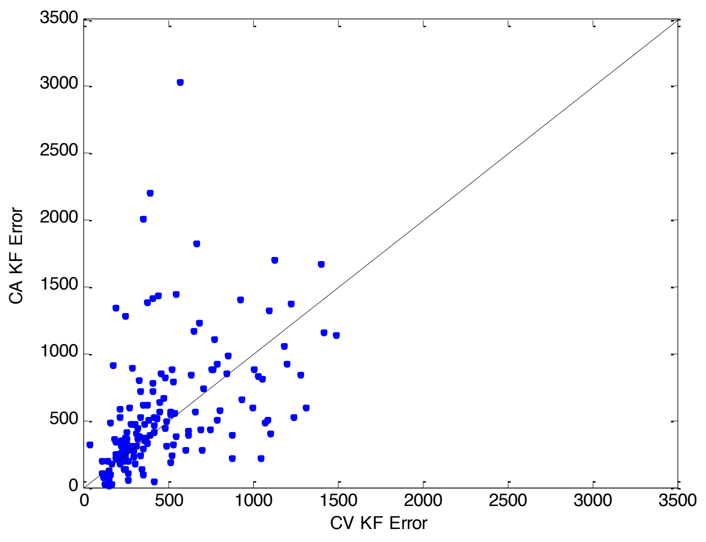


Figure 5.11: 5-steps ahead prediction errors for each time step of all 14 tracks using the CV and CA KFs where it is unclear whether one KF is better than the other.

Figures 5.9 and 5.10 show that the propensity of errors are also higher or equal in the CV and CA KFs section of the plot than it is in SKF. Here the signed wilcoxon rank test made on each pair of models 5th step ahead errors returned $p_{(SKF,CV\ KF)}$= 9.1838e-005 and $p_{(SKF,CA\ KF)}$=1.7169e-006 which tells us to reject the null hypothesis that these errors come from the same the distribution. Figure 5.11 again is not clear visually whether one KF is better than the other however, its signed wilcoxon rank test returned a $p_{(CV\ KF,CA\ KF)}$ = 0.0423 therefore, there is a significant difference at the 5% level.

Since we are interested in finding the probability of infringement at time $t + 5$, in figure 5.12, we show a partial track which we will use to estimate this probability. The SKF switches between the CV and CA Kalman filter's predictions at each time step when appropriate. Predictions with smaller ellipses or error covariances correspond to the CV Kalman filter and the large ellipses correspond to the CA Kalman filter.

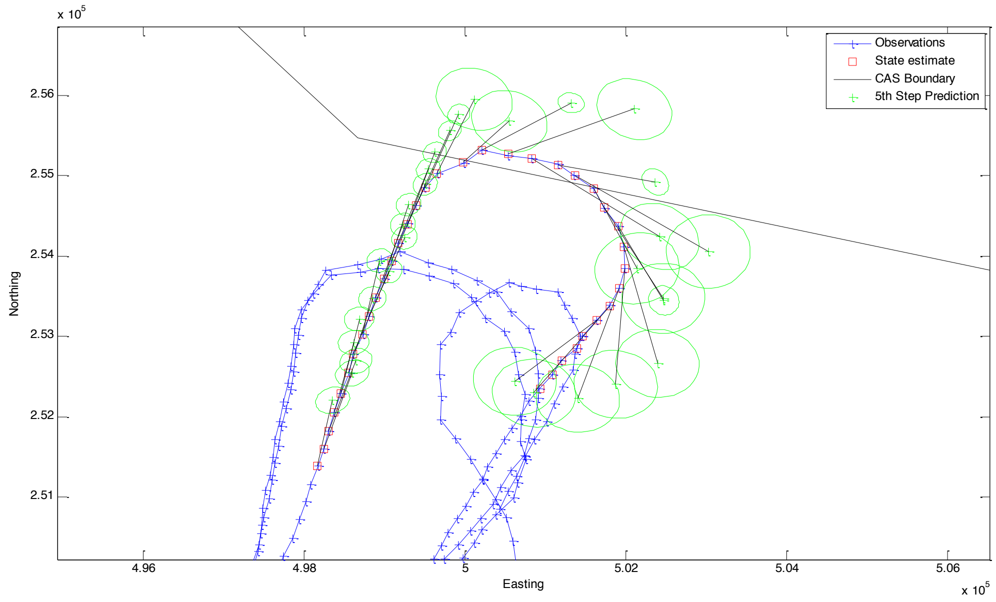


Figure 5.12: 5th step predictions and their error covariances using the SKF

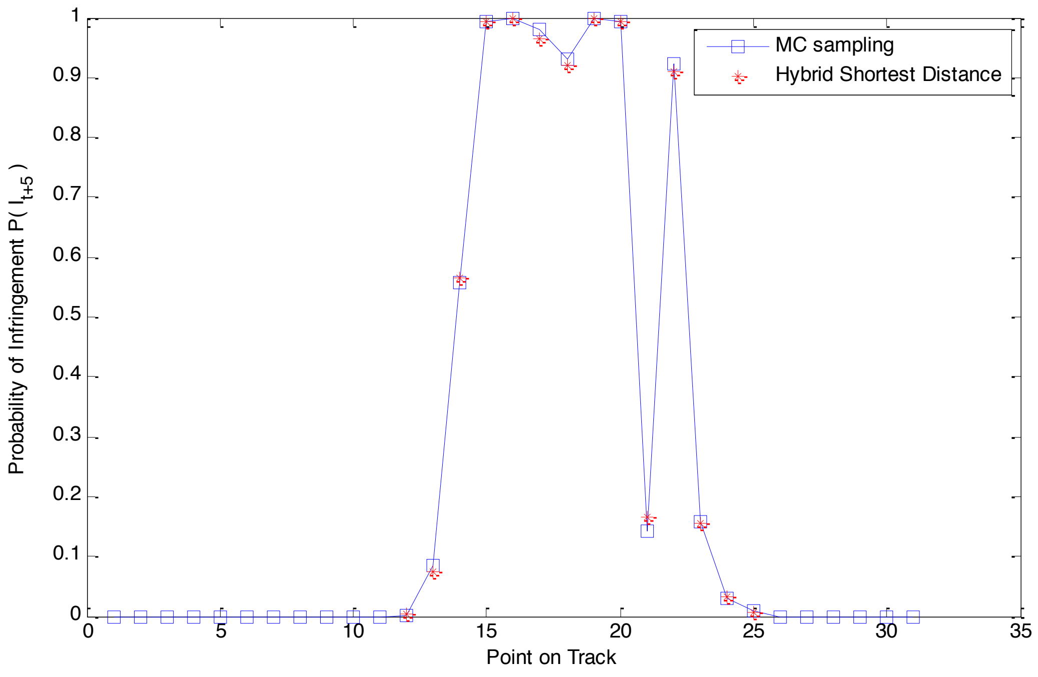


Figure 5.13: Probability of infringement using the Hybrid method for the 5th steps ahead predictions - SKF

We calculated the probability of infringement for the 5th step predictions marked in (+) in figure 5.12 and plotted their probabilities in figure 5.13. The probability of infringement using the MC sampling plotted in figure 5.13 correspond to the whole predictions marked in green crosses. We used the hybrid method to find the probability of infringement using the conditions we proposed in chapter 3. Since the predictions were not close to the CAS corner and did not exceed the threshold given by condition 2, we estimated the $P(I_{t+5})$ using the shortest distance which is marked in (*). By looking at these probabilities and the partial track in figure 5.12 we see that there were 6 observations showing that the aircraft has infringed CAS boundary thus the probabilities does give estimate of $P(I_{t+5}) \geq 0.9$.

5.4 Conclusion

We have reviewed the method of switching between Kalman filters, where the switching process occurs when one Kalman filter gives less RMSE for the prediction than the other Kalman filter. We determined that by finding the probability of the observation given the model, the model which produces the highest probability is chosen one at that time. We also modified the error estimator tool used in the previous chapter to include the weights given by the HMM. After implementing the SKF, we extended it so it can predict 5 steps ahead instead of one. We then found the probability of infringement using the hybrid method. The SKF was found to give superior 1 and 5 steps ahead predictions of the aircraft location, permitting better estimates of the probability of CAS infringement to be made. The SKF gave the minimum prediction error compared to the CV and CA Kalman filters shown in figures 5.6-5.11. We then applied the hybrid method to find the probability of infringement after 5 steps ahead which gave better results shown in figure 5.13 instead of using one Kalman filter for the whole track.

6 Conclusion

In this chapter, we summarise our research objective, the methods we used and their results after using synthetic and real tracks.

6.1 Summary

Controlled airspace infringements are a major safety threat to air traffic controllers and pilots. These threats occur for numerous of reasons, a common one is that pilots tend to misunderstand the ATC's clearance and think it was intended for him/her. Therefore, it was essential to develop a model which alerts ATCs for any occurring infringements. However, the downside of this system is that it only raises a warning to the ATC if the aircraft has already infringed the CAS. Developing a probabilistic model which predicts future infringements can be very helpful and effective if it can increase the warning time and reduces the safety threat ATCs face every day. As a result, a research study was conducted to be able to predict future infringements and assign a probability of infringement to them $P(I)$, this method is called the shortest distance method. We extended this study by removing the assumption that CAS boundary can only be a straight line in order to find the $P(I)$ better reflecting the real life situation. We developed a hybrid method which uses Monte Carlo sampling in addition to the shortest distance method to find the probability of future infringements. The reason for combining the two methods is because MC sampling was an effective way of finding the $P(I)$ when the prediction is near the CAS boundary corner. However, because the MC sampling can be slow, we can use the shortest distance method almost all the time as long as it is away from CAS corner which is defined by a flexible threshold. Another advantage of using the hybrid method aside from effectively finding $P(I)$ in the future, is that we only need a prediction and its error covariance to find

$P(I)$. Since aircraft flying within uncontrolled airspace tend to change their directions and fly in unexpected flight path, the uncertainty about their future locations and their error covariance increases at each time step. Therefore, we were able to predict and implement an online learning for the error covariance using the EM algorithm: the E-step we used the Kalman filter and smoother to predict the location and its error covariance and the M- step where we re-estimated the Kalman filter's error covariances (R, Q). We test our Kalman filter, smoother and error covariance estimation tool on synthetic tracks where we know their true error covariances, and found out that we can approximately estimate their true error covariance at each time step using a small number of EM iterations. Again, because the aircraft can change directions and learning their errors is not enough, we also needed to predict their locations more accurately. By having one flexible Kalman filter, it could give inaccurate predictions, as a result we implemented two Kalman filters which can predict one and 5 steps ahead: CV Kalman filter (aircraft flies in a straight line) and the CA Kalman filter (aircraft is turning or accelerating). Having both Kalman filters with their own error estimation tool running in parallel, we needed a model which can switch between the Kalman filters whenever appropriate. As a result, we implemented the switching Kalman filter that can switch between CV and CA Kalman filters effectively. After comparing all three models, we find that the SKF can be best between CV and CA with little overlapping of the predictions' RMSEs between the SKF and CV Kalman filter.

Since most aircraft within UCAS are not flying to a specific flight path and the pilot can change their intentions at any time in the future, the model is limited on how far it can predict locations in the future and be certain about it because the amount of prediction's uncertainty increases linearly as we predict ahead in time.

6.2 Future Work Recommendation

The probability of future infringements $P(I)$ alone cannot inform us if the aircraft will indeed infringe the CAS. Our model needs to produce a correct decision whether or not an infringement will occur in the future. This decision needs to be enforced more by assigning weights to the $P(I)$ we calculated in our model, such weights can be like the time of the day, is it holiday? Is the weather in UCAS good or bad for flying under visual flight rules. These weights can change our decision about whether an infringement could occur or not. One recommended method we see that can be used on how to assign weights to the $P(I)$ would be multilayer neural networks.

Finally, a probabilistic controlled airspace infringement tool can increase airspace safety and decrease the workload on ATC by detecting them in advance which allows them to resolve any possible future conflict or mid-air collision. We look forward to seeing our models incorporated in current controlled airspace infringement tools.

7 Bibliography

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *JOURNAL OF THE ROYAL STATISTICAL SOCIETY, SERIES B , 39* (1), 1-38.

Erzberger, H., Paielli, R., Isaacson, D., & Eshow, M. (1997). 'Conflict Detection and Resolution In the Presence of Prediction Error'. Moffett Field.

Ghahramani, Z., & Roweis, S. (1998). A Unifying Review of Linear Gaussian Models. *11* . London: University College London.

Hayward, R., & Howell, R. (2008). Proposed Specification of Algorithms for an Enhanced for Short Term Conflict Alert. *Operational Analysis Department* (5).

Kuchar, J. K., & Yang, L. C. (1997). Survey of Conflict Detection and Resolution Modeling Methods. *Navigation, and Control Conference*, (pp. 11-13). New Orleans.

Lowry, R. (2000 , 13-Decemeber). *The Wilcoxon Signed-Rank Test.* From http://faculty.vassar.edu/lowry/PDF/c12a.pdf

Mcdonald-Wallis, K. (2009). An Approach into the Probabilistic Prediction of the Movement of Uncontrolled Aircraft to Improve UK Aviation Safety. *MSc. Dissertation* .

Murphy, K. (1998). *Switching Kalman Filter.* DEC/Compaq Cambridge Research Labs.

Paielli, R., & Erzberger, H. (1997). Conflict probability Estimation for Free Flight. *Journal of Guidance, Control and Dynamics , 20* (3), 588-596.

Prandini, M., & Watkins, O. (2005). *Probabilistic Aircraft Conflict Detection.* Technical Report.

Prandini, M., Hu, J., Lygeros, J., & Sastry, S. (2000). A probabilistic approach to aircraft conflict detection. *Intelligent Transportation Systems , 1* (4), 199 - 220.

Rabiner, L. R. (1989). A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, (pp. 257--286).

Reckhouse, W. (2010, 10 22). Optimisation of Short Term Conflict Alert Safety Related Systems. Exeter.

Yang, L. C., & J. K. Kuchar. (1997). Prototype Conflict Alerting Logic for Free Flight. *AIAA Journal of Guidance , 20* (4).

Yang, L., & Kuchar, J. (1998). Using Intent Information in Probabilistic Conflict Analysis. *Navigation, and Control Conference*, (pp. 10-12). Boston.

Yu, B. M., Shenoy, V. k., & Sahani, M. (2004, October 19). Derivation of Kalman Filtering and Smoothing Equations .