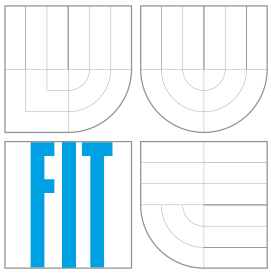
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

OPTIMIZATION OF AIRCRAFT TRACKER PARAMETERS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — MICHAL SAMEK
AUTHOR

VEDOUCÍ PRÁCE — doc. RNDr. PAVEL SMRŽ, Ph.D.,
SUPERVISOR

KONZULTANT — Dr. Stephan Schiffel
CONSULTING SUPERVISOR

BRNO 2015

Abstrakt

Diplomová práce se zabývá optimalizací systému pro sledování letadel, využívaného pro řízení letového provozu. Je popsána metodika vyhodnocování přesnosti sledovacího systému a přehled relevantních algoritmů pro sledování objektů. Dále jsou navrženy tři přístupy k řešení problému. První se pokouší identifikovat parametry filtrovacích algoritmů pomocí algoritmu Expectation-Maximisation, implementací metody maximální věrohodnosti. Druhý přístup je založen na prostých odhadech parametrů normálního rozložení z naměřených a referenčních dat. Nakonec je zkoumána možnost řešení pomocí optimalizačního algoritmu Evoluční strategie. Závěrečné vyhodnocení ukazuje, že třetí přístup je pro daný problém nejvhodnější.

Abstract

This Master's thesis deals with a configuration optimization of an aircraft surveillance system, which is being used in the air traffic control. We survey commonly used methodology for a performance evaluation of such surveillance systems and review relevant algorithms for target tracking. Three optimization approaches are explored. The first solution attempts to identify parameters of employed filtering algorithms by using the Expectation-Maximisation algorithm to find corresponding maximum likelihood estimates. The second approach employs a simple distribution fitting to the available measured and reference data. Evolution strategies are examined as the third option. Experimental evaluation shows that the third approach is the most suitable for the problem in hand.

Klíčová slova

Letecký tracker, Optimalizace, Kalmanův filtr, IMM algoritmus, Evoluční algoritmy, Evoluční strategie, CMA-ES

Keywords

Aircraft tracker, Optimization, Kalman filter, IMM algorithm, Evolutionary algorithms, Evolution strategy, CMA-ES

Citace

Michal Samek: Optimization Of Aircraft Tracker Parameters, diplomová práce, Brno, FIT VUT v Brně, 2015

Citation

Michal Samek: Optimization Of Aircraft Tracker Parameters, master's thesis, Brno, FIT BUT, Brno, 2015

Declaration

I confirm that this Master's thesis is my own work which I have carried out under supervision of Doc. RNDr. Pavel Smrž, Ph.D.

.....
Michal Samek
May 24, 2015

Acknowledgment

I would like to thank my Master's thesis advisor Doc. RNDr. Pavel Smrž, Ph.D. for leading my thesis. Furthermore, I would like to express my gratitude to Dr. Stephan Schiffel for his support, comments and overall guidance throughout the thesis; our meetings and discussions made this thesis possible. Next I would like to thank Sigurjón Árni Guðmundsson, Ph.D. for introducing me to the topic as well as his cooperation during the development. Last but not least, I want to thank my beloved ones for supporting and encouraging me during my exchange stay in Iceland where I worked on this thesis.

Contents

1 Introduction **5**

2 ATC Tracker Evaluation Methods **8**
2.1 Considered requirements for optimization 8
2.1.1 Horizontal position RMSE 9
2.1.2 Track velocity characteristics RMSE 9

3 Employed Methods and Algorithms **10**
3.1 Used Motion Models 10
3.1.1 Constant velocity model 11
3.1.2 Constant turn model 12
3.2 Optimal filtering 13
3.2.1 Kalman filter 13
3.2.2 Extended Kalman filter 15
3.3 Estimation of Multiple Model Systems 16
3.3.1 The Interacting multiple model algorithm 16
3.3.2 IMM Smoother 18
3.4 Evolutionary optimization 18
3.4.1 Introduction 18
3.4.2 Covariance-Matrix-Adaptation ES 20
3.5 Classification 21
3.5.1 Ensemble learning 21
3.5.2 Skewed data 22

4 Current Implementation and Configuration of the Tracker **23**
4.1 Structure and Implementation of the Tracker 23
4.1.1 Used Kalman filters 23
4.1.2 Sensor fusion 24
4.2 Current Configuration and Evaluation 26

5 Problem Definition **28**
5.1 Additional Criteria 28
5.1.1 Smoother heading while cruising 28
5.1.2 Adaptation of the IMM structures 29
5.2 Tracker Parameters 29

6 Optimization of Separate Kalman Filters **30**
6.1 Classification and Segmentation of the Trajectories . . . 30
6.1.1 Training set of trajectories . . . 31
6.1.2 Classification features . . . 31
6.1.3 Classification results . . . 32
6.2 Training Kalman Filters . . . 33

7 Empirical Estimation of Process and Measurement Noise **34**
7.1 Measurement Noise Estimation . . . 34
7.2 Process Noise Estimation . . . 35

8 Evolutionary Optimization **40**
8.1 Fitness Function . . . 40
8.2 Parameters to Be Optimized . . . 43
8.2.1 Measurement noise model . . . 43
8.2.2 Parameter scaling . . . 43
8.2.3 Parameter reduction . . . 44
8.3 Incorporating Smoothness of Heading . . . 45
8.3.1 Measuring smoothness of a curve . . . 45
8.3.2 Integration with the global fitness function . . . 46

9 Implementation **48**
9.1 Handling of Trajectory Data . . . 48
9.1.1 Trajectory class . . . 48
9.1.2 TrajectoryReconstruction class . . . 48
9.1.3 TrajectoryStub class . . . 49
9.2 Estimation Algorithms . . . 49
9.2.1 IProcessModel class and descendants . . . 50
9.2.2 IMeasurementNoise class and descendants . . . 50
9.3 Optimization end Evaluation . . . 50

10 Results **51**
10.1 Optimization Process . . . 51
10.2 Different Structures of the IMM . . . 53
10.3 Incorporating the optimized IMM structure . . . 55
10.4 Smoothness Incorporation . . . 56
10.5 Best configuration . . . 57

11 Conclusions **60**

A Toolbox Class Diagram **66**

B Optimized Parameters **67**

List of Figures

4.1 Overall structure of the aircraft tracker 24
4.2 Architecture of the IMM algorithm used in the aircraft tracker 25

7.1 Measurement noise standard deviation estimated from the data 35
7.2 Distribution of the interval between consecutive ADS-B recordings 36
7.3 Empirical estimation of process covariance matrices 39

8.1 Comparison of exponential and direct partial cost 42
8.2 Logarithmic mapping of P to x with both axes linear 44
8.3 Smoothness as differences in the neighbouring samples 46
8.4 Smoothness as differences to the smoothed curve 47

10.1 Evolution of object variables for different sizes of population 52
10.2 Optimized measurement noise standard deviation for radar1 56
10.3 Comparison of the optimized and original configuration on the whole dataset 58
10.4 Comparison of the optimized and original configuration on a smaller area around airport 59

List of Tables

2.1 Specification of the maximum RMSE in the horizontal position 9
2.2 Recommended performance requirements for the track velocity vector . . . 9

4.1 Physical characteristic of the used radars 25
4.2 Current configuration of the tracker 26
4.3 Evaluation of the current configuration 27

6.1 Arrangement of the feature vector 31
6.2 Distribution of the MOF in the training data 32
6.3 Confusion matrix for a three-class problem 33
6.4 Confusion matrix for the resultant ensemble 33

7.1 Fitted linear models to the standard deviation of radar's measurement error 37
7.2 Evaluation of the empirically estimated parameters 37
7.3 Evaluation of the IMM with empirically estimated parameters 38

9.1 Size comparison of Trajectory and TrajectoryStub objects 49

10.1 Typical ranges for the parameters involved in the optimization 51
10.2 Analysed IMM Structures 54
10.3 Evaluation of optimized configurations 55
10.4 Evaluation of the selected and optimized IMM structures 55
10.5 Evaluation of the smoothed configuration 57
10.6 Comparison of the smoothed and non-smoothed configuration 57

B.1 Best configuration for the tracker 67

Chapter 1

Introduction

Air transport is nowadays undoubtedly the fastest and the most comfortable way to transport people or freight on longer distances. Every day thousands of aircrafts cruise the skies all around the world. Surely, our airspace is becoming much more crowded than ever before and especially around the airports the concentration of airplanes is reaching its limits. In such a dense traffic, one cannot expect the pilots of the airplanes to do without any navigational assistance. This assistance or service is called Air Traffic Control (ATC) and is provided by ground controllers to the pilots not only in order to prevent collisions, but also to guide aircrafts through controlled airspace, organize the flow of air traffic, as well as provide the pilots with information on weather or other types of support. As such it is an area of great importance where any failure might be life-threatening.

ATC controllers are utilizing to work a great variety of devices that provide them with all necessary information – Air Traffic Management (ATM) systems. An essential part of this set of tools is a unit for tracking targets (aircrafts) called *tracker* in short. This device is used to visualize the current position and heading of an aircraft, which is then a subject for further evaluation.

The input data used for ATC trackers comes from primary and secondary surveillance radars – devices measuring the distance of an object by detecting reflected radio signals, and thus suffering from high measurement noise due to physical limitations of such a measuring principle; to name a few beam path, range, signal noise or interference with other signals. Sometimes the radar signal might be reflected by other objects than aircrafts causing false detections or outliers. Moreover, the measurement noise usually changes with the distance of the aircraft from the radar. The noise of such measurement can reach a magnitude of several hundred meters and these measurements are often the only information about the position of an aircraft the ATC controller has.

Significantly more precise position measurements can be extracted from the data provided by the Automatic Dependent Surveillance-Broadcast system (ADS-B). ADS-B is a modern surveillance technology enabling an aircraft equipped with such system to periodically broadcast its position determined via satellite navigation (GPS). Nevertheless, even these measurements might be inaccurate though their error is much lower.

Naturally, one can ask why air traffic control still uses radars instead of switching to modern systems such as ADS-B. The answer to this question lies in the key differences between these technologies. Whilst radar technology (used in primary surveillance radars) is non-cooperative, the ADS-B system relies on cooperation with the target. If the target does not reply to the ADS-B request sent by a ground station, there is no other way of determining the position of the aircraft. Total reliance on cooperative surveillance tech-

nology would be obviously a risky affair as the transponders might malfunction, can be turned off or the aircraft is simply not equipped with appropriate ADS-B unit although it is slowly becoming a common equipment. Moreover, due to the low receiving power of GPS signal and resultant susceptibility to either intentional jamming or unintentional radiofrequency interference, the whole ADS-B system can easily experience service interruptions. To maintain the reliable operation of air traffic surveillance systems, the use of primary radars, accompanied by secondary radars, is still a vital source of data.

Needless to say, we cannot directly use raw measurements supplied by radars and project it on the tracker screen. Such a trajectory would inevitably be very *bumpy* and all calculations using the raw data would give very imprecise results. For such scenarios, there are *filtering* algorithms that try to estimate the true state (position and other kinematic characteristics) of the observed process. The input data for these algorithms are noisy raw radar measurements from which they compute estimates that are more precise, robust to false detections, smoothed and follow more closely the motion patterns observed during flights. These algorithms are based on statistical theory so one needs to supply necessary parameters describing the examined phenomena. Correct identification of these statistical values have a great influence on trackers overall performance. However, it is not clear how one can derive these parameters. In this thesis, we compare different approaches to the problem of parameter estimation.

The purpose of this Master's thesis is an optimization – in a horizontal plane – of the ATC tracker that is currently being deployed at Keflavik airport (Iceland) implemented by Tern Systems ehf[1]. Tern Systems also supplied radar recordings along with corresponding ADS-B data that were used throughout the thesis.

The configuration of the tracker consists of various parameters that greatly influence the performance of the tracker. However, there is no analytical solution or another systematic approach to determine the best value for these parameters. The current configuration was hand-optimized by examining various trajectory plots and fiddling with these parameters in nearly *ad hoc* manner. Despite that researchers in Tern Systems managed to find a configuration that is good enough for the needs of ATC controllers. Moreover, the tracker fulfils all requirements of the ESASSP specification (see section 2). However, by employing the techniques of machine learning we aim to find even better configurations without the tedious *trial-and-error* approach.

This work covers three different approaches we applied to the problem of ATC tracker's performance optimization. The first approach is based on a separate optimization of the very core parts of the tracker – Kalman filters. For the optimization of these filters we intended to use the Expectation–maximization (EM) algorithm – an algorithm for finding the most likely estimates of parameters.

Besides that, we carried out the estimation of the parameters statistically describing the process and measurement model from the available data (radar measurements and corresponding reference values) – we shall refer to this method as an *empirical* parameter estimation – by fitting normal distributions to the samples of the process disturbances and measurement noise.

The last method investigates an application of a stochastic black-box optimization using Evolution Strategy algorithm – a versatile derivative-free evolutionary algorithm.

The thesis is organized as follows. In chapter 2 we give a brief overview of common performance evaluation methods for ATC trackers. Chapter 3 describes algorithms used in

[1] www.tern.is

the tracker or in the thesis. The current implementation of the tracker is examined in chapter 4. In the chapter 5 we defined the problem of ATC tracker optimization. Chapters 6, 7 and 8 elaborate on the proposed solutions. The implementation of the used approach is described in 9 and in 10 we present achieved results with the conclusions in 11.

Chapter 2

ATC Tracker Evaluation Methods

The assessment of tracker's quality can be carried out on recorded datasets[1] of radar measurements. The output of the tracker is then compared to the corresponding ADS-B data, which are then considered to be a ground truth – as we are unable to observe the true states directly.

Standards provide the minimum performance that an ATM surveillance system has to fulfil. The most recent one is the Eurocontrol[2] Specification for ATM Surveillance System Performance (ESASSP, EUROCONTROL-SPEC-0147)[3]. The performance requirements are expressed in terms of statistical measures (mostly Root Mean Square Error – RMSE) dependent on the specific situation of an aircraft. For all requirements, both mandatory and recommended performance measures are given[4].

The definition of the Root Mean Square Error reads:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} E_i^2}{n}}$$

where n is the total number of samples and E_i is the error of the i-th sample.

2.1 Considered requirements for optimization

For the optimization process it would be impractical and cumbersome to consider all requirements specified in the ESASSP for the evaluation, therefore only the following were recognized to be the main tuning metrics for the tracker configuration, i.e., they serve as a cost-function in the optimization process.

Outlier criteria Outlying target reports and false positives are excluded from the evaluation. The horizontal position outlier criteria for cooperative surveillance is horizontal position error greater than 1690 m.

[1] also referred to as opportunity traffic

[2] www.eurocontrol.int

[3] available from https://www.eurocontrol.int/articles/atm-surveillance-system-performance-esassp-specification

[4] The ATC tracker that is to be optimized mainly operates in the area with 3 NM horizontal separation with data from secondary surveillance radars (cooperative surveillance) – all requirements taken into consideration account for 3 NM horizontal separation with cooperative surveillance system

2.1.1 Horizontal position RMSE

The error on the provided horizontal position is the 2D Euclidian distance between the horizontal position provided by the surveillance system and the reference horizontal position of the corresponding aircraft at the time when the updated position was output/delivered. For each target report, we calculate horizontal position error which is later used for calculation of root mean square error for the whole track and the whole dataset. The required values are to be found in the table 2.1.

Table 2.1: Specification of the maximum RMSE in the horizontal position.

	for every track	globally (whole dataset)
mandatory performance	330 m	300 m
recommended performance	230 m	210 m

2.1.2 Track velocity characteristics RMSE

The error calculation for track velocity includes error for velocity amplitude and velocity angle (commonly referred to as heading) and follows the same principle as for horizontal position error, i.e., the tracker provided value is compared with the reference value. Here the ESASSP specification distinguishes between straight-line motion and manoeuvres. These types of motion are differentiated by the value of transversal acceleration whose threshold value is set to 1.5 m/s^2. Note that the specification again does not mention mandatory performance and both measures are defined globally for all tracks (whole dataset). Track velocity error characteristics are listed in the table 2.2.

Table 2.2: Recommended performance requirements for the track velocity vector. Specified as the maximum RMSE in the vector components.

	straight-line	manoeuvre
amplitude (speed)	4 m/s	8 m/s
angle (heading)	10°	25°

Chapter 3

Employed Methods and Algorithms

In this chapter, we provide an overview of the theoretical basics a reader should be familiar with. This covers broad fields of classification, optimal filtering (and corresponding motion models) and evolutionary optimization as they comprise the core parts of this thesis.

3.1 Used Motion Models

When tracking a manoeuvring target the main objective is to precisely estimate the target's trajectory, i.e., its kinematic state. Many of these tracking methods are model based – they represent the target's motion and its observations (measurements) by some mathematical models, usually *state-space* models of the following form:

$$x_{k+1} = f_k(x_k, u_k) + w_k$$

$$z_k = h_k(x_k) + v_k$$

where

- x_k, u_k and z_k are the target state, control input and target state observation correspondingly, at the discrete time t_k,
- w_k and v_k are process and measurement noise sequences,
- and f_k and h_k are functions defining the evolution of the state and its relation to the measurement (observation).

However, the usability of these models is very limited without coupling with appropriate estimation (filtering) algorithm. Depending on the nature of the functions f_k and h_k, these models might be classified as linear or non-linear models and, consequently, for their estimation a linear or non-linear estimation algorithm is used.

In the available literature dealing with tracking algorithms, one can find a great variety of motion models applicable to the problem of a manoeuvring target tracking. Several are described in [34], [36] and [5], most of them are also summarized in a survey [24]. These models differ in their complexity, characteristics of the motion they represent, organization of a state vector, type of observations and in the way how is the process noise injected. Generally, the process noise can be modelled as a continuous-time white noise signal or

a discrete-time random variable with a normal distribution. In the continuous-time domain, the process noise magnitude is expressed by corresponding Power Spectral Density (PSD), while in the discrete-time counterpart by its variance.

The motion models used throughout this thesis are described in the next sections. We note that all motion models use Cartesian coordinate frame in their state representations as well as observations.

3.1.1 Constant velocity model

The most basic model is a Constant Velocity (CV) model that assumes the object is moving at nearly constant velocity on a straight track. The process noise can be injected using random acceleration inputs that accounts for changes in speed. In such models the state usually constitutes of the object's position and velocities and can be organized as follows:

$$\mathbf{x}_k = \begin{bmatrix} \hat{x}_k & \hat{y}_k & \hat{v}_k^x & \hat{v}_k^y \end{bmatrix}^T$$

where $\mathbf{x}_k$ is a state vector at time step k, $\hat{x}_k$ and $\hat{y}_k$ are estimates of the position in XY coordinates; and $\hat{v}_k^x$ and $\hat{v}_k^y$ are estimates of the velocities in the X and Y direction correspondingly.

Since our observations are taken in a discrete-time domain and the motion model is linear, the discrete-time state and measurement models develops to:

$$x_{k+1} = Ax_k + w_k$$

$$z_k = Hx_k + v_k$$

where

- A is a state transition matrix,
- H is a measurement model,
- w_k and v_k are process and measurement noise samples with covariance matrices Q_k and R_k, correspondingly.

We intentionally left out the term counting for the control input as it is in the most cases unknown. A common approach is to assume there is no control input, even though an actual thrust is, obviously, at certain times present. The effect of such inputs is uniformly modelled as a process noise.

The measurements (observations) in our scenario consist of a position in X and Y coordinates, therefore, the measurement model (linear) is defined as:

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

The state transition matrix and process noise covariance matrices are obviously time-varying since we observe the process via irregularly sampled measurements. Such irregularity makes the transition matrix and process covariance matrix dependent on the length of the interval between measurements. The dynamics of the process is thus modelled in continuous-time domain and for every iteration of the filtering algorithm the continuous-time model is discretized.

Discretization of a random process

The continuous-time linear model is described by the differential equation:

$$\frac{dx(t)}{dt} = \mathbf{F}x(t) + \mathbf{L}w(t),$$

where

- $\mathbf{F}$ is a constant matrix characterizing the behaviour of the model,
- $\mathbf{L}$ is a constant noise effect matrix denoting which state components are corrupted by the process noise,
- $w(t)$ is a white process noise with power spectral density $\mathbf{Q_c}$.

With CV motion model and process noise entering the state as a random acceleration inputs, the matrices F and L read:

$$F = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad L = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$

In order to obtain a discrete-time equivalent (suitable for use with discrete time estimation algorithm), we first need to discretize/sample the continuous model. The matrices A_k and Q_k are then given as

$$\mathbf{A_k} = exp(\mathbf{F}\Delta t_k)$$

$$\mathbf{Q_k} = \int_0^{\Delta t_k} exp(\mathbf{F}(\Delta t_k - \tau))\mathbf{L}\mathbf{Q_c}\mathbf{L}^T exp(\mathbf{F}(\Delta t_k - \tau))^T d\tau$$

where Δt_k is the stepsize of the discretization. Numerical evaluation of $\mathbf{Q_k}$ can be computed by a matrix fraction decomposition. How these equations were derived and for further details the reader is referred to [19].

In the current implementation of the tracker, it is assumed that the disturbances in both axes have the same magnitude and, thus, white noise with the same power spectral density is perturbing estimates of the both velocities $\hat{v}_k^x$ and $\hat{v}_k^y$.

3.1.2 Constant turn model

The Constant Turn motion model (often referred to as Coordinated Turn model) aims at describing the turning motion with a constant turn rate, which also augments the state vector by a corresponding estimate $\hat{\omega}_k$. On contrary to the CV model, the CT model is not linear – the evolution of a state is governed by a non-linear *dynamic* function ([19]).

$$\mathbf{x_{k+1}} = \begin{bmatrix} 1 & 0 & \frac{sin(\hat{\omega}_k\Delta t)}{\hat{\omega}_k} & \frac{cos(\hat{\omega}_k\Delta t)-1}{\hat{\omega}_k} & 0 \\ 0 & 1 & \frac{1-cos(\hat{\omega}_k\Delta t)}{\hat{\omega}_k} & \frac{sin(\hat{\omega}_k\Delta t)}{\hat{\omega}_k} & 0 \\ 0 & 0 & cos(\hat{\omega}_k\Delta t) & -sin(\hat{\omega}_k\Delta t) & 0 \\ 0 & 0 & sin(\hat{\omega}_k\Delta t) & cos(\hat{\omega}_k\Delta t) & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{x_k} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} v_k$$

where $\hat{\omega}_k$ is the turn rate estimate at the time step k and v_k is a sample from univariate normal distribution with a variance Q_ω.

As the process noise is perturbing only the turn rate estimates, noise covariance is defined as:

$$\mathbf{L} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \end{bmatrix}^T$$

$$\mathbf{Q_k} = \mathbf{L} Q_\omega \mathbf{L}^T \Delta t$$

On account of non-linear dynamic function, one cannot use linear estimation algorithm, but a non-linear one instead. We note that the measurement model H is still the one introduced with the CV motion model, i.e., a linear one.

3.2 Optimal filtering

The term *optimal filtering* is herein in the meaning of a methodology used for state estimation of a process, which is observed indirectly via noisy measurements. The state of this process is constituted of the physical state, in our case described by the position of a moving object and its derivatives such as velocity or acceleration. Our observation of this state is often incomplete (for example we observe only position) and noisy as the sensor is encumbered by errors of various origin. The process of filtering then refers to removing the noise from measurements and computing estimates for the state given these corrupted measurements.

Several different estimators applicable to target tracking (alpha-beta, adaptive Kalman filter, IMM algorithm) are compared in [22]. In this experiment, the IMM clearly outperformed other state estimators examined and as such is still a prevailing algorithm for target tracking in the ATC domain.

In this section, we give an overview of the Kalman filter algorithm and the Interacting Multiple Mode (IMM) algorithm. Both present core parts of the tracking algorithm so their understanding is essential for the rest of the project.

State smoothing Another method for estimating the true state of a process is *smoothing*. Unlike filtering, it estimates the current state based on the complete sequence of process observations, i.e., to estimate a state it incorporates not only past but also *future* observations. It follows that this kind of estimation can only be performed off-line, when all process observations are available.

3.2.1 Kalman filter

The (linear) Kalman filter (originally introduced in [21]) is an optimal recursive algorithm to estimate the state of a process[1] based on noisy measurements observed over time. Thanks to its computational efficiency it is widely used in technology; especially engineering and econometric applications.

The Kalman filter operates in iterations that are usually referred to as *time* and *measurement update* iterations (or predictor and corrector accordingly). In the time update iteration we project the state estimate forward from time step n to step $n+1$ – to obtain an *a priori* estimate and compute the estimated error covariance. In the measurement update step, we incorporate the actual measurement and get *a posteriori* estimates.

The process which is being estimated by the filter is assumed to be governed by a linear dynamic system – such as CV model introduced in 3.1.1 – of the following form:

[1] sometimes referred to as a latent variable

$$x_k = Ax_{k-1} + w_{k-1}$$

$$z_k = Hx_k + v_k$$

Thanks to the recursive equation describing the underlying state dynamics, the Kalman filter can use the whole history of previous states without the need of explicitly storing them. This allows the filtering to be optimal, but still computationally feasible.

All of the above indicates that the underlying system of Kalman filters can be seen as a hidden Markov model with a continuous latent variable. Specifically,

- the true state x_k is unobservable,
- the current state x_k is dependent only on the previous state x_{k-1},
- the visible outputs of the system z_k are dependent only on the current state x_k.

This can be expressed by the following equalities:

$$p(x_k|x_{1:k-1}, z_{1:k-1}) = p(x_k|x_{k-1})$$

$$p(z_k|x_{1:k}, z_{1:k-1}) = p(z_k|x_k)$$

Computational background of the Kalman Filter

We assume that from our knowledge about the system and the state of the system in the previous time step we can predict the state for the current time step without the need for any measurement. We denote this *a priori* state estimate by $\hat{x}_k^-$. The error of the *a priori* estimate is defined as $e_k^- = x_k - \hat{x}_k^-$, whose expected value is referred to as *a priori* error covariance. The predicted state is distributed as follows:

$$p(x_k|x_{k-1}) \sim \mathcal{N}(Ax_{k-1} + Bu_{k-1}, Q_{k-1})$$

In the next step, we obtain a noisy measurement z_k as a form of feedback which we use to make our prediction of the current state more precise. To blend in the actual measurement we compute the *Kalman gain* or *blending factor* which expresses how much we trust the sensor that provided the measurement. This computation of Kalman gain is based on the *a priori* estimate covariance and measurement noise.

By incorporating the „weighted“ measurement we obtain an *a posteriori* state estimate; a sum of the *a priori* state estimate and a weighted term called *residual* or *innovation* reflecting the discrepancy between the predicted measurement $H\hat{x}_k^-$ and actual measurement z_k.

$$\hat{x}_k = \hat{x}_k^- + K(z_k - H\hat{x}_k^-)$$

Finally, we compute the *a posteriori* error covariance as expected value of the *a posteriori* estimate error $e_k = x_k - \hat{x}_k$. The objective of the filtering is then to minimize the *a posteriori* error covariance so that the difference in the estimated and actual state is low.

Further reading on the rationale of the Kalman filter is to be found in [27] and [11].

Discrete Kalman filter algorithm

As stated earlier, the algorithm works in two iterations. In the time update iteration, we compute the *a priori* state and error covariance estimates as follows. If we do not know the initial state of the system, we can „make a guess“ and set the error covariance large enough to reflect this uncertainty.

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_{k-1}$$

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

In the measurement update iteration, we begin by computing the Kalman gain, which is later used to integrate the actual measurement into the estimate. The value of the gain K_k is such, that it minimizes the a posteriori error covariance. How this equation was derived is explained in [7, p. 216].

$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1} = \frac{P_k^- H^T}{HP_k^- H^T + R}$$

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-)$$

$$P_k = (I - K_k H)P_k^-$$

From the equation for computing the Kalman gain, we see that as the measurement error covariance R approaches zero, the Kalman gain weights the residual more heavily. On the other hand, as the *a priori* estimate error covariance P_k^- approaches zero, the residual is weighted less heavily.

$$\lim_{R_k \to 0} K_k = H^{-1}$$

$$\lim_{P_k^- \to 0} K_k = 0$$

3.2.2 Extended Kalman filter

As soon as the underlying system (described by a dynamic function f) – as in the case of CT motion model (see section 3.1.2) – and/or the relationship between measurement and the process state (described by a function h) are not linear the linear Kalman filter is no longer capable of estimating the process state accurately. For such an estimation, one might use an Extended Kalman filter, which approximates the joint distribution of state and/or measurement by a Taylor series based transformations (usually first or second order).

$$x_k = f(x_{k-1}) + w_{k-1}$$

$$z_k = h(x_k) + v_k$$

The (possibly) non-linear functions f and h are used for computing the predicted state from the previous state estimate and the predicted measurement from the current state estimate. However, they cannot be used for estimation of the respective covariance matrices. Instead, one needs to compute matrices of partial derivatives (Jacobians), which are then used in the equations of the Kalman filter.

For further description of the algorithm refer to [6] or [4].

3.3 Estimation of Multiple Model Systems

The motion of a commercial aircraft can be usually segmented and classified into several classes corresponding to basic motion models in the literature referred to as Modes-Of-Flight (MOF). Most of the time, constant velocity is sustained and the aircraft is moving straight with only occasional turns. During landing and take-off, the aircraft performs considerably more manoeuvres and accelerates.

We see that the modes-of-flight an aircraft undergoes during a single flight can differ significantly. Thus, a precise description of the motion using single motion model is not possible. By using more models, we can describe different modes-of-flight more precisely.

Multiple model systems are considered to be systems with a discrete set of n models denoted by $M = \{M^1, M^2, ..., M^n\}$. We also know probabilities $p^{ij} = P\{M_k^j | M_k^i\}$ of switching from model M^i to model M^j and prior probabilities μ_0^j for each model M^j. From the probabilities p^{ij} we can construct a transition probability matrix of corresponding first order Markov chain that characterizes the mode transitions. The equations describing such a linear system are:

$$x_k = A_{k-1}^j x_{k-1} + B_{k-1}^j u_{k-1} + w_{k-1}^j$$

$$z_k = H_k^j x_k + v_k^j$$

where A, B, H are a state transition matrix, a matrix relating input to state and a measurement model matrix, respectively, and w and v are process and measurement noise. They have the same meaning as in section 3.2.1, but are related to a model that is being active in the specific time step. The meaning of x_k and z_k stays the same.

However, a wide range of multiple model algorithms have been developed, we herein deal with Interacting Multiple Model algorithm only as it is currently implemented in the ATC tracker of our interest. For a comprehensive exploration of another applicable algorithms the reader is referred to [25].

3.3.1 The Interacting multiple model algorithm

The IMM algorithm is a suboptimal algorithm to filter multiple model systems. Optimality could be achieved by applying optimal filters for every possible model sequences. For n models n^k optimal filters would have to be run to process the k-th measurement; which is computationally infeasible.

The IMM algorithm itself is divided into 3 consecutive computational steps.

Interaction During the interaction phase *mixing probabilities* $\mu_k^{j|i}$ for each model are calculated – expressing the probability of model M^j being active in the time step $k-1$ conditioned on model M^i being active in the current time step k.

$$\bar{c}_j = \sum_{i=1}^{n} p_{ij} \mu_{k-1}^i$$

$$\mu_k^{j|i} = \frac{1}{\bar{c}_j} p_{ij} \mu_{k-1}^i$$

where

- $\bar{c}_j$ is equal to marginal probability $P(M_k^j)$ of model M^j in the time step k; also referred to as normalization factor,
- p_{ij} is a probability of switching from model M^i to model M^j,
- μ_k^i is a probability of model M^i being active in the time step k,
- $\mu_k^{j|i}$ is a probability of model M^j being active in the time step k, conditioned model M^i was active in the time step $k-1$.

These mixing probabilities serve to estimating the initial means and covariances for each filter. For the j-th filter we compute the initial state mean $\hat{x}_{k-1}^{0j}$ as a sum of means from previous iteration weighted by the mixing probability $\mu_k^{i|j}$. The initial covariance P_{k-1}^{0j} is estimated similarly.

$$\hat{x}_{k-1}^{0j} = \sum_{i=1}^{n} \mu_k^{i|j} x_{k-1}^i$$

$$P_{k-1}^{0j} = \sum_{i=1}^{n} \mu^{i|j} \times \{P_{k-1}^i + [x_{k-1}^i - \hat{x}_{k-1}^{0j}][x_{k-1}^i - \hat{x}^{0j} k-1]^T\}$$

where x_{k-1}^i, P_{k-1}^i are the *a posteriori* estimates of state covariance at time step $k-1$ after measurement update

Filtering Next we perform the filtering with *mixed inputs* $\hat{x}_{k-1}^{0j}$ and P_{k-1}^{0j}. We denote the outputs (state mean, covariance) of Kalman filters $\hat{x}_k^j$ and P_k^j correspondingly. During the filtration we also compute the likelihood of the measurement for each filter as:

$$\Lambda_k^i = \mathcal{N}(\omega_k^i; 0, S_k^i)$$

where ω_k^i is the measurement residual (innovation) and S_k^i it's prediction covariance for model M^i in the measurement update step. The measurement prediction covariance S_k^i is the denominator in the equation for computing the Kalman gain $S_k^i = HP_k^- H^T + R$.

Combination In the last phase of the algorithm we combine the estimates for the state mean x_k and its covariance P_k according to μ_k^i; the probability of model M^i being active at current time step k.

$$c = \sum_{i=1}^{n} \Lambda_k^i \bar{c}_i$$

$$\mu_k^i = \frac{1}{c} \Lambda_k^i \bar{c}_i$$

$$x_k = \sum_{i=1}^{n} \mu_k^i \hat{x}_k^i$$

$$P_k = \sum_{i=1}^{n} \mu_k^i \times \{P_k^i [\hat{x}_k^i - x_k][\hat{x}_k^i - x_k]^T\}$$

3.3.2 IMM Smoother

The IMM smoother can be thought of as two instances of the IMM algorithm. One instance runs forward estimating the current state based on the past evidence $p(x_k|z_{1:k-1})$; while the second instance runs backward and incorporates the future measurements $p(x_k|x_{k+1:N})$. However, this approach is restricted to systems having the state transition matrix A invertible.

3.4 Evolutionary optimization

In this thesis, the evolutionary optimization is employed to optimize parameters of the IMM algorithm. In the following subsection, we give a general description of algorithms from the algorithm family of evolutionary optimization and introduce Evolution Strategy (ES) algorithm and its particular derivative – Covariance-Matrix-Adaptation ES – which is being used in this research.

3.4.1 Introduction

By the term *evolutionary optimization* one usually refers to an optimization technique inspired by a natural phenomenon such as selection, reproduction or mutation. For the evolutionary algorithms is characteristic their randomness – we categorize them as stochastic algorithms. They are being successfully applied in a variety of engineering tasks (mathematics, biology, finance) where they are rapidly becoming more and more popular [2].

The success of evolutionary optimization algorithms might be justified by their favourable properties:

1. They do not require the objective function to have continuous derivatives which is advantageous as real-world complex engineering problems often have discontinuities [2].

2. The evolutionary algorithms can be implemented to perform a parallel search of the search space – by evaluating multiple solutions at a time.

The second advantage is also important when an optimizer is initialized to a local optimum as a serial optimizer might not be able to get out of that local optimum. This issue can be usually dealt by multiple restarts. However, the gathered information about the search space is not shared among the optimization runs and thus this approach cannot be regarded as efficient. On the other hand when searching in parallel we sample and evaluate the search space at multiple points and use the acquired information to drive the optimizer to the most promising regions of the search space. Moreover, such a robustness makes the evolutionary optimization an appropriate tool for rugged objective functions with possibly many local optima [16].

Populations of individuals

The evolutionary algorithms operate on populations of individuals. An individual is, in this case, an arbitrary potential solution to the problem – a point in the search space. However, in order to utilize the evolutionary optimization the solution to the actual problem typically needs to be transformed into a representation suitable for evolutionary optimization. This is achieved by a mapping from the original problem context (phenotype) onto an individual's

genotype within the context of evolutionary optimization. With evolutionary algorithms, the phenotype can be encoded into real numbers, binary code or a tree structure [3].

In our scenario, the IMM algorithm parameters are real-valued number and, therefore, no encoding needs to take place. However, if we would have subjected for the optimization also the choice of motion models, then we would have to encode the chosen models into the corresponding genotype.

Reproduction

In every iteration[2], the evolutionary algorithm generates a new offspring population by selecting the best individuals (parents), with respect to the fitness function, as a seed for the production of new individuals. These descendants are randomly generated by processes representing mutation or recombination. Mutation herein stands for a randomly corrupted replication of the genotype and by recombination we refer to an exchange of randomly selected genotype information among the parents (two or more).

The rationale behind the mutation is to find potential better solutions based on the current best solutions. While by performing recombination we assume that by combining individuals with promising phenotype also the offspring will evince good phenotype; but since we cannot tell which genotype information is responsible for the specific phenotype we exchange the genotypes randomly.

Selection

To determine *how good* a particular individual is one defines a fitness function, which is based on the objective function whose optimum is to be found by the optimizer. This objective function is typically bound to the problem solved, and thus decoding the solution to the corresponding phenotype is often applied.

After evaluation of the individuals is finished, the algorithm selects either deterministically or probabilistically the parents for the next population – this selection is relative to the actual fitness value or the acquired ranking with respect to the fitness value.

Initialization and termination

Before running the optimization, one usually needs to define the first generation of the evolutionary process. Commonly, this is done randomly in order to cover the complete search space. On the other hand, if the human designer has some prior knowledge about the location of the optimum he can initialize the algorithm to search around that location.

To terminate the algorithm several stopping criterions can be set. Reaching the optimum is the most trivial one. Unfortunately, very often the optimal value is not known and therefore it is not possible the set such a criterion right. It is also hard to claim that the found solution represents the global optimum as one can never be sure there is not a better solution. What's more due to the stochastic nature of the algorithm the optimum might never be reached. Generally for these cases one can set a limit on the iteration count, fitness function evaluations count or investigate if the fitness value or value of phenotype does not stagnate.

[2] also referred to as generation in Evolution strategies

3.4.2 Covariance-Matrix-Adaptation ES

The CMA-ES algorithm is a stochastic, iterative technique for real-valued parameter optimization. It was first introduced by Nikolaus Hansen in [17] in 1996. Since then it has gained great popularity and nowadays is regarded as the state-of-the-art optimization scheme for noisy, rugged, multimodal and ill-conditioned functions with possible discontinuities. The algorithm, as other evolutionary algorithms, does not need the objective function to have continuous derivatives. Moreover, it is a parameter-free algorithm. Finding good parameters for the optimization process is a part of the algorithm design. The only parameter that is left for the user to define is the population size. The other parameters are adjusted during the optimization or their default values are set appropriately for most of the applications.

Population distribution

The main difference to a classic evolutionary algorithm is the representation of the population. In the CMA-ES we represent the population by a multi-variate normal distribution $\mathcal{N}(\mathbf{m}, \mathbf{C})$, where

- $\mathbf{m} \in \mathbb{R}^n$ is the mean of the sample distribution,
- $\mathbf{C}$ is corresponding symmetric $n \times n$ covariance matrix.

The sample distribution can be geometrically interpreted as an ellipsoid whose position is determined by the mean value $\mathbf{m}$ and shape by the covariance matrix $\mathbf{C}$.

This is the driving idea of CMA-ES, that the evolutionary operators of selection, recombination, and mutation comprise an implicit distribution for sampling the next generation. The optimization scheme then follows the sequence of steps:

1. Sample distribution $\mathcal{N}(\mathbf{m}, \mathbf{C})$ to generate a population.
2. Evaluate individuals from the current population.
3. Update parameters $\mathbf{m}$ and $\mathbf{C}$ of the sampling distribution.

The mean value $\mathbf{m}$ and covariance matrix $\mathbf{C}$ are updated after evaluation of the current population is finished. For this update only the best – newly generated – individuals are selected; CMA-ES is therefore referred to as non-elitist evolution strategy algorithm. The updated mean value $\mathbf{m}$ is computed as a weighted average of the best points. The weights are assigned by a ranking position in the population with respect to the defined fitness function. Similarly, the value of the covariance matrix $\mathbf{C}$ is updated in such a way that it fits the distribution of the best points.

Stepsize control

The value stepsize σ influences how far from the mean value $\mathbf{m}$ new individuals are sampled. Its value is automatically adjusted during the optimization via a technique called *cumulative step-size adaption* which is based on the *evolution path* method from evolution strategy algorithms. This technique employs the information on the correlation between consecutive generations. It increases the stepsize σ when the population mean $\mathbf{m}$ gets updated in the expected direction based on the direction of previous updates letting the algorithm make bigger *steps* in the direction towards the optimum. On the other hand, when the mean $\mathbf{m}$

is updated in another direction the value of stepsize σ is decreased in order to carry out a finer search.

Population size

The population size is the only parameter potential user needs to define. Its value significantly influences the properties of the optimization. By setting the population size to a smaller value, the algorithm favours fast convergence. By using larger populations (possibly in connection with optimization restarts [1]) the global search performance can be improved, however, the algorithm will take a longer time to converge.

For further details on the CMA-ES algorithm the reader is referred to [17], [15] or [18].

3.5 Classification

In the chapter 6, classification of trajectories is carried out. This classification is based on using classification decision trees [29], which are *grouped* into ensembles, as the whole ensemble might have better predictive performance than a single decision tree.

However, due to unequal distribution of the training data an extension to the ensemble learning – a boosting algorithm – is used, since training the classifier on a skewed data[3] might lead to favouring the majority class over the minority classes ([35]). A common approach to deal with skewed data is employing sampling techniques ([8], [9]).

Below we introduce the concept of ensemble learning and the Random Undersampling Boosting algorithm (RUSBoost). A more thorough introduction to classification can be found in the review [23] or [20]. For a comprehensive description of decision trees refer to [30].

3.5.1 Ensemble learning

Ensemble learning methods are learning algorithms that create a set of classifiers, which are afterward used to classify new data points ([31]). The final classification is aggregated of predictions of these classifiers, commonly referred to as *weak learners*. The rationale behind ensemble learning is a question if a set of weak learners with a moderate performance can form together one strong learner (a classifier with good performance). Experiments showed that it is often possible to construct a very good ensemble ([26]). A necessary and sufficient precondition for a strong learner to be more accurate than any of its individual weak learners is that the weak learners are accurate (meaning more accurate than random guessing) and differ in making errors on new data.

In what manner these classifiers are trained depends on the specific learning strategy, the following are the two most common [26]:

- *Bagging* uses for learning a weak learner different samples of observations or a different set of features. The aggregation of the predictions is done by averaging (regression) or taking the majority vote (classification).
- *Boosting*, on the other hand, trains weak learners on the whole training set, but assigns weights to the samples. In the beginning all sample have the same weight, but

[3] also referred to as imbalanced data

in latter iterations larger weight obtains samples that were misclassified in the previous iteration. The weighted sum of the weak learners' predictions is the resultant prediction of the ensemble. Weak learners' weights are given by their accuracy.

3.5.2 Skewed data

RUSBoost algorithm is a hybrid sampling/boosting algorithm which deals with the skewed data by applying random undersampling; a technique which randomly removes samples of the majority class. The algorithm itself is based on AdaBoost algorithm (introduced in [12]) and described in [32]. The key difference compared to AdaBoost is random undersampling of the majority class in each iteration of the boosting algorithm.

Chapter 4

Current Implementation and Configuration of the Tracker

The tracker of our concern is a multi-sensor tracker developed by Tern Systems and deployed in Keflavík, Iceland. It is a tracking system based on measurements from multiple radars whose purpose is to derive trajectory estimates for aircrafts in the controlled airspace, filtering out as much measurement noise as possible. The filtering techniques applied in these systems varied greatly throughout the history as the corresponding estimation theory was advancing. Nowadays, most modern radar tracking systems are based on some derivate of the IMM algorithm, which is also applied in the tracker developed by Tern Systems.

4.1 Structure and Implementation of the Tracker

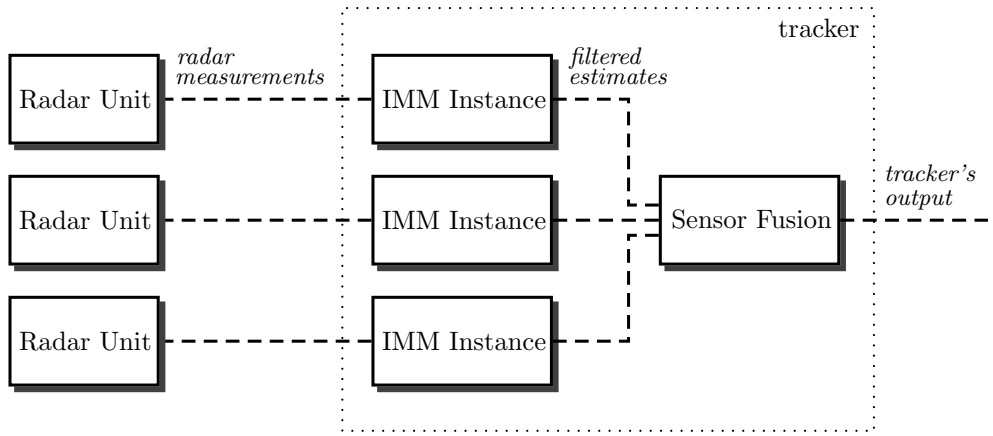
As noted above, the input data for the tracker is provided by three different sources. Throughout the thesis, we refer to these sources by labels *radar1*, *radar2*, and *radar3*. Their physical characteristics are listed in the table 4.1. For each trajectory, these sources provide unique sequences of track data encumbered by measurement noise of different magnitudes. Each of this sequences is filtered using a separate instance of the IMM algorithm, which itself consists of three different Kalman filters. The outputs of the Kalman filters are fused together by the IMM algorithm according to the mode transition probabilities and these IMM-fused outputs get eventually combined using a method described in 4.1.2. For reader's better understanding, these structures are also visualized in the figures 4.1 and 4.2. The first one is depicting the overall structure of the aircraft tracker, while the latter outlines the detailed structure of each IMM instance.

4.1.1 Used Kalman filters

For filtering radar measurements the implemented IMM algorithm combines outputs of three different Kalman filters. These Kalman filters compute their output in parallel and their configuration matches the following different types of motion (MOF):

- uniform motion (linear),
- accelerated motion (linear),
- manoeuvres (non-linear).

Figure 4.1: Overall structure of the aircraft tracker.



The Kalman filters for uniform and accelerated motion share the same linear CV model and differ in the magnitude of the process noise perturbing only velocity estimates. The magnitude of the process noise is defined via Power Spectral Density (PSD) and is naturally higher in the accelerated motion model. The state vector for linear models is composed of estimates of position and velocity and is arranged as:

$$\mathbf{x}_k = \begin{bmatrix} \hat{x}_k & \hat{y}_k & \hat{v}_k^x & \hat{v}_k^y \end{bmatrix}^T$$

where $\mathbf{x}_k$ is a state vector at time step k, $\hat{x}_k$ and $\hat{y}_k$ are estimates of the position in XY coordinates; and $\hat{v}_k^x$ and $\hat{v}_k^y$ are estimates of the velocities in the X and Y direction correspondingly.

For the filtering of manoeuvres, represented with CT model, an Extended Kalman filter is applied. Its state vector is augmented by an estimate of turn rate $\hat{w}_k$ and the process noise is assumed to be corrupting only estimates of the turn rate. The process noise is, in this case, modelled as a random variable with known variance – expressing the magnitude.

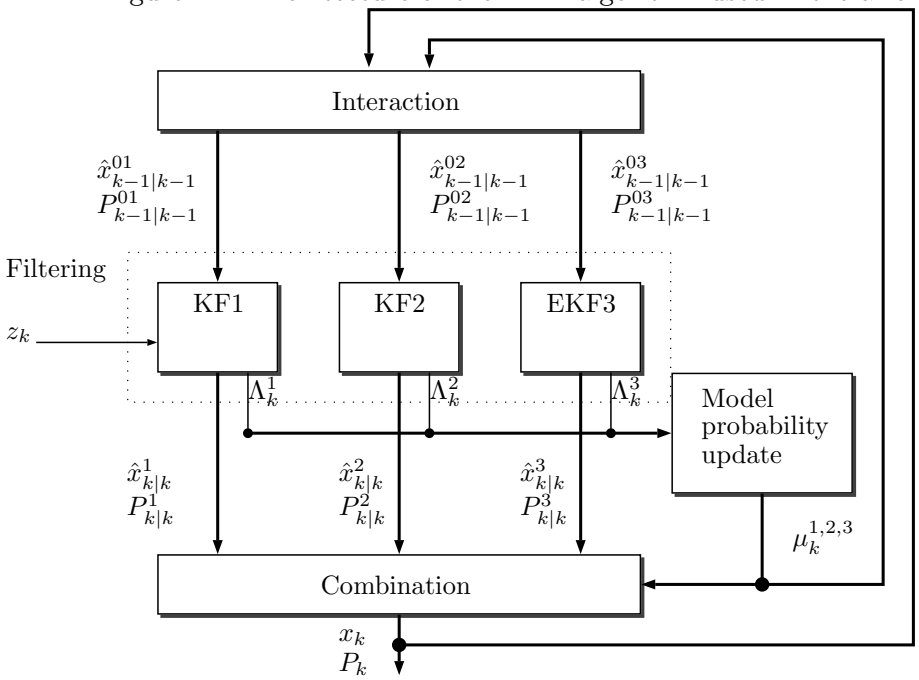
Coordinate conversion The filtering is carried out in a projected X,Y space relative to the position of the specific radar, so conversion from geographic coordinates (latitude, longitude, altitude) to local X,Y coordinates takes place. When fusing the filtered sequences, all of them need to be converted to the common coordinate frame resulting in another coordinate conversion.

4.1.2 Sensor fusion

As mentioned earlier we refer to the Tern ATC tracker as a multi-sensor tracker, because it uses several radar sources for tracking aircrafts. The advantages of such a setting are greater area coverage than what is possible with a single radar and higher reliability in case of a failure. To our application, the most important differences between radars are the maximum measuring distance and the magnitude of the measurement error (expressed via measurement error covariance).

The three radars in the table 4.1 corresponds to two physical radars only. The *radar2* and *radar3* represent a single device but differ in the maximum range and *precision*. By using a fixed-length representation (11 bits) for the range measurements the maximum range

Figure 4.2: Architecture of the IMM algorithm used in the aircraft tracker.



influences quantization error. The *radar2* has a lower maximum range and thus lower quantization error. For the *radar3* it is the opposite – by having higher maximum range also the quantization error rises.

Table 4.1: Physical characteristic of the used radars. The range is in *Nautical miles*, scan period is in *revolutions per minute*.

label	range [Nm]	scan period [rpm]
radar1	100	5
radar2	60	15
radar3	100	15

Hence, for every track we have measurements from several different radars (sensors) encumbered by the noise of different magnitudes. The current implementation does not distinguish between different radars – we can say that the configuration is generic for all radars, although the radars differ considerably. The filtered sequences are eventually fused together using the following equation[1] – linear combination weighted by respective noise error covariances:

$$\mathbf{x} = \mathbf{P}(\mathbf{P_1}^{-1}\mathbf{x_1} + \mathbf{P_2}^{-1}\mathbf{x_2})$$

where $\mathbf{P} = (\mathbf{P_1}^{-1} + \mathbf{P_2}^{-1})^{-1}$ is the error covariance of the fused measurement, $\mathbf{P_1}$ and $\mathbf{P_2}$ are error covariances of the filtered sequences and $\mathbf{x_1}$ and $\mathbf{x_2}$ are filtered state estimates. The equation for the fusion of three sensors can be derived in a similar fashion.

[1] subscript k indicating the number of a sample intentionally left out for better readability

Coasting

Since the measurement sequences from different radars are not synchronized or some of the measurements might be missing, the filtered measurement sequences must be time-aligned in order to being able to fuse them. One can think of this alignment procedure called *coasting* as a special type of interpolation when we take the last measurement preceding the timestamp specified for coasting and use the time-update step of Kalman filter algorithm described in 3.2.1.

4.2 Current Configuration and Evaluation

The current configuration of the IMM algorithm, presented in 4.2, is shared across all IMM instances coupled with different radar sources. In other words, to every radar corresponds an IMM instance with the same structure and configuration as for the other radars, despite possibly differing radar's characteristics.

Table 4.2: Current configuration of the tracker. Labels KF1-3 refer to the corresponding Kalman filters.

KF1 process noise PSD	$9 \times 10^{-6} m^2 s^{-4} Hz^{-1}$			
KF2 process noise PSD	$100 m^2 s^{-4} Hz^{-1}$			
KF3 process noise variance	$9 \times 10^{-6} deg^2$			
measurement noise variance	range [Nm]			var $[m^2]$
	< 40			50^2
	$\in [40, 60]$			300^2
	> 60			$1\,500^2$
IMM transition matrix		KF1	KF2	KF3
	KF1	0.80	0.15	0.05
	KF2	0.10	0.75	0.15
	KF3	0.10	0.05	0.85

In order to have figures for later comparison, table 4.3 shows evaluation results for the current configuration. The evaluation was performed on a traffic captured within one whole day. In total, 83 327 target reports were included in the evaluation of the fused output, only around 4% of all reports account for turns according to the transversal acceleration threshold of 1.5 m/s^2. The maximum range of the recorded trajectories was 100 Nm.

Table 4.3: Evaluation of the current configuration. Error statistics for velocity vector components are divided for straight-line motion (on the left in the particular column) and turns (on the right). For convenience, we also evaluated the performance of each radar separately.

source	horz. position RMSE [m]	speed RMSE [m/s]	heading RMSE [°]
radar1	387.59	7.04/7.28	2.96/9.44
radar2	160.10	8.42/5.26	3.90/5.63
radar3	256.21	9.78/8.51	3.55/7.48
fused	**256.95**	**4.56/6.00**	**2.33/11.41**

Chapter 5

Problem Definition

The problem of ATC tracker performance optimization can be stated as a multi-objective optimization problem. It must take into account the tracker's performance against the set of tuning metrics, such as those described in the chapter 2, highly depending on the particular scenario one selects for evaluation. Among these merits different levels of trade-offs can be identified and also each one of the considered magnitudes can drive the optimization problem to a different solution. For these reasons the process of optimization needs to be designed such that it considers all tuning metrics at the same time and takes into account most of the representative situations.

Our concern in this thesis is the part of the tracker that deals with the estimation in the horizontal position only. The models in the horizontal and vertical plane are typically filtered decoupled for the sake of simplicity and also due to the fact that the target motion in horizontal plane and vertical plane can be considered independent.

Generally, the tracking quality of such tracker highly depends on:

- the number of Kalman filters, i.e., the number of modes-of-flight used,
- the configuration of each Kalman filter,
- the configuration of the model transition probabilities for the IMM algorithm.

Therefore, the main goal of this thesis can be formulated as finding an improved configuration for the IMM algorithm leading to an increase in the tracker's performance. Apart from that there are two additional objectives outlined in the following section.

5.1 Additional Criteria

5.1.1 Smoother heading while cruising

An additional quality requirement was identified through ATC controller feedback. It was formulated as a requirement to increase the smoothness of track velocity angle (heading) while the aircraft is cruising (rectilinear motion at constant velocity). This might introduce a trade-off to the error in the horizontal position and heading which will be higher during manoeuvres.

The intention is to have the heading progression smoother when the aircraft is farther from the airport as it undergoes there significantly fewer manoeuvres and we can also allow for higher error. On the other hand when the aircraft is within close proximity to the airport

it is preferable to make the estimates adapt quickly to the supplied measurements and, as a result, the progression is less smooth.

5.1.2 Adaptation of the IMM structures

In the air traffic control environment, the tracker is estimating the kinematic state of several dozen aircraft. For most of the flights the tracker obtains the flight data from multiple radars of varying precision, as shown in the table 4.1. For each radar an estimated state needs to be computed using the IMM algorithm, coasted and fused with the estimates computed from data supplied by remaining radars and processed in the same manner. This results in a significant computational load if running all of these procedures for tens of different airplanes.

Currently, for filtering output of all radars the very same IMM structure (set of Kalman filters) is used. However, it might be that for *radar1* we could employ a simpler IMM structure due to its higher measurement noise magnitude and lower scan rate. Nevertheless, the IMM structure should not be modified if the overall tracker's performance could be negatively influenced.

5.2 Tracker Parameters

With the current structure of the tracker, the parameters that are to be optimized are:

- process noise PSD for the *uniform* and *accelerated* Kalman filters,
- variance of the random noise in the *manoeuvring* Kalman filter,
- measurement noise covariance,
- transition matrix in the IMM algorithm.

All of these needs to be optimized for each radar source. We believe this will yield better results than using one generic configuration for all radars. Also, the actual structure of the IMM algorithm is a subject for optimization, so the set of parameters can vary depending on the particular structure selected.

Chapter 6

Optimization of Separate Kalman Filters

Both the IMM algorithm and Kalman filters are Markov models. The standard way of training parameters of a Markov model is to derive the maximum likelihood estimate of the parameters of the model given a set of output sequences (flight trajectories). In general, there are no efficient algorithms that find optimal solutions to the problem. However, there are efficient algorithms for finding good local solutions based on the Expectation-Maximization (EM) algorithm [10], [14].

Thus, our proposed solution to the problem was divided in the following steps.

1. Categorize existing radar data of flight trajectories into different classes by Mode-Of-Flight (MOF) such that the data can be used to train different Kalman filters (each corresponding to the particular MOF).
2. Train the Kalman filters separately so they perform well on the trajectories of the corresponding MOF (EM algorithm).
3. Learn the transition probability matrix of the IMM algorithm (Baum-Welch algorithm).

If this approach proves to be appropriate, then also different structures of the IMM algorithm will be optimized in the similar fashion in order to determine the best configuration possible.

6.1 Classification and Segmentation of the Trajectories

The goal of this task is to divide available trajectory data into segments with different MOF. These segments can be thought of as grouped measurements with the same MOF assigned, thus the problem might be stated as a multiclass classification of single measurements rather than trajectories.

The selection of MOF classes is arbitrary, it only depends on what motion models one plans to apply. We classified the data to the classes currently employed in the tracker, that is uniform and accelerated motion and other manoeuvres.

We approached this classification problem via techniques of supervised machine learning. An approach employing decision tree learning or other machine learning techniques was already successfully applied in similar works [13], [28]. In this work, we experimented with ensembles of decision trees.

6.1.1 Training set of trajectories

To train the model itself we needed to prepare a set of training trajectories that were correctly classified. This was achieved by picking 18 trajectories that were representative for the whole dataset and afterwards hand-classified to the classes specified above. While there are no recommendations on what should be considered as accelerated motion or manoeuvre we set a threshold for determining acceleration to be a value of 0.4 m/s^2. Similarly, segments with turn rate greater than 0.007 rad/s were classified as a turn. These thresholds were based on reviewing different plots with trajectories and corresponding velocity waveforms. While classifying acceleration and turns, also the nearest surrounding measurements (context) were classified as acceleration/turn although they might not have reached the threshold value.

6.1.2 Classification features

As a sample's features we experimented with turn rate, longitudinal and transversal accelerations computed – interpolated in one-second intervals – from the ADS-B recordings and IMM mode probabilities (used in [13]). Furthermore, we must take into account that the information is not contained in the current measurement only, but also in the neighbouring past and future measurements (context), so including past and future neighbouring measurements' features in the set of features might help better identify transitions between different modes. The layout of a feature vector is depicted in the figure 6.1, where

- μ_a^n stands for the a-th feature of a sample in step n,
- $\mathbf{x}_n$ denotes features of a sample in step n,
- N refers to the number of measurements,
- M refers to the number of a sample's features,
- k specifies the width of the context.

Table 6.1: Arrangement of the feature vector.

	$\mu^{n-k}_{1,2,\dots M}$	$\cdots$	$\mu^{n-1}_{1,2,\dots M}$	μ^n_1	μ^n_2	$\cdots$	μ^n_M	$\mu^{n+1}_{1,2,\dots M}$	$\cdots$	$\mu^{n+k}_{1,2,\dots M}$
$\mathbf{x}_{k+1}$										
$\mathbf{x}_{k+2}$										
$\dots$										
$\mathbf{x}_{N-k}$										

During the experiments, various parameters were tested. We studied the influence of using acceleration and turn rate in the feature set, IMM mode probabilities, different values of parameter denoting the width of context or depth of weak learners' decision trees.

Skewed data

We must point out that in the trajectories different modes-of-flight were not equally distributed. Most measurements belonged to the uniform model, much less to the accelerated

model and to manoeuvres as is shown in the table 6.2. For our purposes, we decided to use RUSBoost algorithm, which is also available from MATLAB Statistics Toolbox.

Table 6.2: Distribution of the MOF in the training data.

MOF	Count	Percent
uniform	3990	82.69%
accelerated	468	9.70%
manoeuvres	367	7.61%

Classification algorithm configuration

To evaluate different configurations and settings of the classifier, various scenarios were evaluated using 10-fold cross-validation. During each round of cross validation a confusion matrix was computed and used to further compare with other classifier instances.

We determined the best configuration is to have both longitudinal and transversal acceleration included in the features set. Including the IMM mode probabilities as in [13] did not yield better predictive results. A possible reason might be an incorrect configuration of the IMM algorithm which afterwards failed to recognize modes-of-flight correctly. The width of context included (on both sides) in the feature set was determined to give best results when set to 10 seconds (1 sample per second). A sufficient number of weak learners proved to be 50. The prediction is based on 42 features (21 samples x 2 transversal and longitudinal acceleration).

6.1.3 Classification results

The performance of a learned classifier is typically evaluated by a confusion matrix (see table 6.3 for a 3 class problem). The columns denote *predicted* class while rows denotes *actual* class. Cells in the confusion matrix (table 6.3) are explained as:

- T_X percentage[1] of correctly classified samples of class X,
- M_{XY} percentage of incorrectly classified samples of class X for class Y.

The overall accuracy on all classes can be then computed as:

$$ACC = \frac{T_1C_1 + T_2C_2 + T_3C_3}{C_1 + C_2 + C_3}$$

where $T_{1,2,3}$ are diagonal cells from confusion matrix and $C_{1,2,3}$ are the element counts of the corresponding classes.

For the final evaluation of the acquired model, we created a set of testing (validating) trajectories to provide an objective assessment – independent data that was neither used to train nor test the configurations. This was fulfiled by selecting 6 trajectories (different from the training trajectories) and hand classifying them.

With the configuration from 6.1.2 we achieved an accuracy of 93.23% on the validating set of trajectories. The corresponding confusion matrix is the table 6.4. The learned classifier achieves good performance on all classes – especially classification of the accelerated

[1] common is also number of samples instead of percentage

Table 6.3: Confusion matrix for a three-class problem.

		predicted		
		uniform	accelerated	manoeuvre
actual	uniform	T1	M12	M13
	accelerated	M21	T2	M23
	manoeuvres	M31	M32	T3

segments seems very precise. On the other hand, a significant proportion of manoeuvres was misclassified as uniform segments. We shall point out, that this evaluation is strongly dependent on the right classification of the trajectories. Thus, it is generally very difficult to reach total conformity with the human classifier.

Table 6.4: Confusion matrix for the resultant ensemble.

		predicted		
		uniform	accelerated	manoeuvres
actual	uniform	93.1271	4.2285	2.6444
	accelerated	3.5337	95.8692	0.5971
	manoeuvres	8.0655	1.1696	90.7649

Trajectory segmentation After carrying out the classification, we grouped measurements with identical mode-of-flight into segments (parts of original trajectories). It might be that during the segmentation very short segments are created (let's say 1 or 2 measurements long) however mode-of-flight usually does not change freely from sample to sample. Such segments were ignored for subsequent parts of the project.

6.2 Training Kalman Filters

As stated in section 5.2 the parameters that need to be optimized are the measurement noise covariance and the process noise covariance modelled in the continuous time domain with magnitude expressed via corresponding power spectral density. To train these parameters we intended to use the EM algorithm as is presented in [14] or [33]. However, during the implementation of the EM algorithm we realized that the algorithm is not applicable to our scenario as it assumes all parameters to be (time) invariant. Unfortunately, this assumption does not hold as the process noise is time-dependent (via irregular sampling) and measurement noise depends on the distance from the radar. Although it might be tempting to simply substitute the expressions for computing measurement and discretizing process noise to the equation for the computation of log-likelihood, this would not help as the partial derivative of log-likelihood with respect to the process noise power spectral density does not have a closed-form solution.

From the above it is clear that EM algorithm is not applicable to our case and to our best knowledge there are no other algorithms which could be used for parameter estimation of such continuous-time linear dynamical systems sampled via irregular measurements.

Chapter 7

Empirical Estimation of Process and Measurement Noise

In this chapter we examine the estimation of the sample measurement and process noise covariances by a simple distribution fitting[1] from the available data. For the measurement noise estimation we investigated the differences in the observed measurements and the true states represented by ADS-B recordings. While for the process noise estimation we explored the samples of the noise extracted from the ADS-B data.

7.1 Measurement Noise Estimation

To estimate the measurement noise we examined the dependency of standard deviation of horizontal position error on the actual distance from the radar. We fitted normal distributions[2] to the samples of measurement error in the position that were computed against ADS-B recordings and grouped by their distance from the radar in 500 m increments. In this way we obtained sample means and standard deviations for the error in the specified ranges. Eventually, we smoothed the resulting data by a 10-span moving average filter.

The plot in the figure 7.1 represents acquired results for all radars along with corresponding fitted linear models. The X axis represents the distance from the radar, on Y axis the dependent standard deviation of error in horizontal position is plotted. Each radar is represented by a different colour with a solid line, while for linear approximations a dotted line is used. In addition we plotted (in black dotted line) the stepping function currently used in the tracker.

The plot 7.1 proves the presumed dependency of the standard deviation of measurement noise on the distance from the radar. Obviously, in the sampled measurements it could be approximated by a linear function. We also investigated the estimated sample means which always lied around zero – assumption that the measurement noise is a white Gaussian thus seems correct.

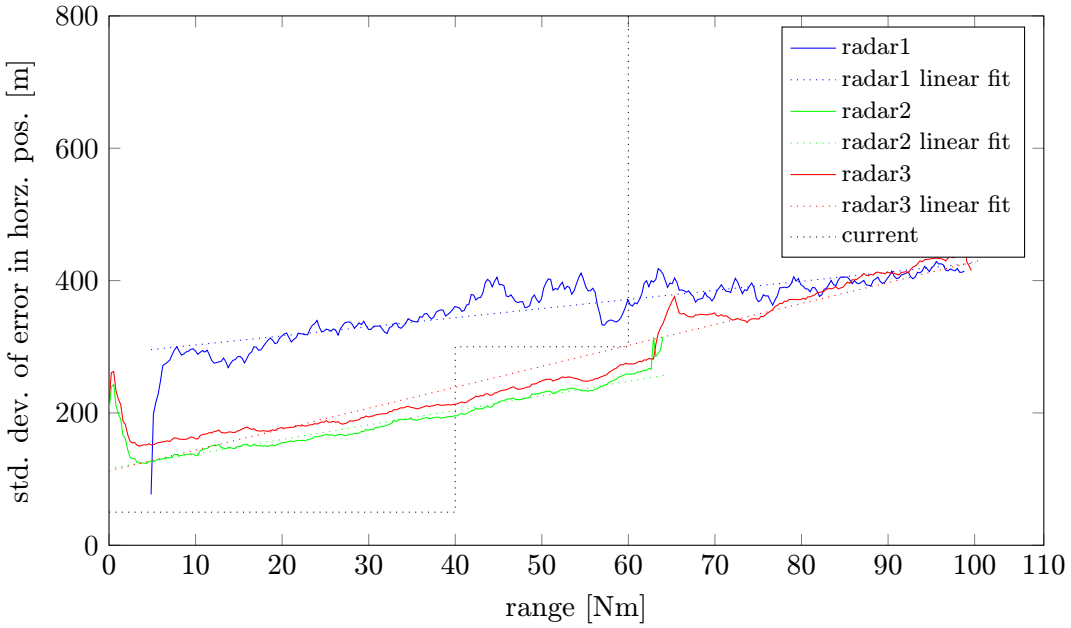
Furthermore, we can observe a sudden change in the estimated error for *radar3* in the range between 60 and 70 Nm possibly caused by a quantization error, described in 4.1.2.

The measurement noise variance in both axes can be determined by taking a square of estimated standard deviation. By the assumption that the measurement noise in one

[1] sample mean and sample variance

[2] measurements are assumed to be corrupted by a random noise with normal distribution

Figure 7.1: Measurement noise standard deviation estimated from the data. The X axis shows the distance from the radar (in Nautical miles), on Y axis a measurement noise standard deviation (in meters) is plotted.



direction is uncorrelated to the measurement noise in the other direction, the covariance matrix can be stated as a diagonal matrix with variances on its main diagonal.

7.2 Process Noise Estimation

The process noise was estimated in a similar fashion as the measurement noise. For this computation we used ADS-B recordings which are considered in our case as a *ground truth* and as such are best representing the process itself – we neglected the GPS measurement error.

This method is based on the equation describing the evolution of the state in linear dynamical systems[3] as is defined in the subsection 3.2.1.

$$x_k = Ax_{k-1} + w_{k-1}$$

We transformed the equation to the following form which can be used to calculate the process disturbance in every time step. For the non-linear model the term Ax_{k-1} is substituted with $f(x_{k-1})$ to comply with the non-linear evolution of the state. Note that with state represented by a column vector also process disturbance is represented by a column vector. During the estimation process we, thus, need to consider the *dependency of each element* of the noise vector on the interval length Δt.

$$w_{k-1} = x_k - Ax_{k-1}$$

We shall also think of the noise vector as a random vector consisting of 4 random variables; noise in position in X and Y coordinates and noise in velocity in X and Y coor-

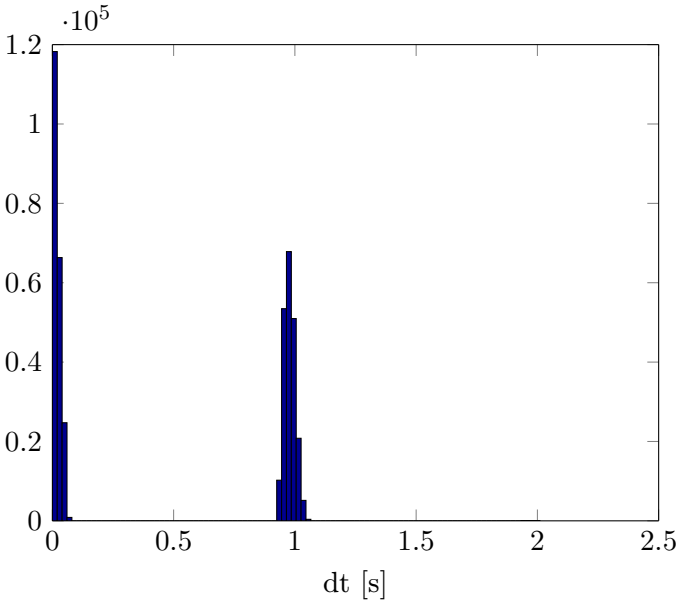
[3] there is no input to the system so the term Bu_{k-1} was left out

dinates[4]. The noise vector then represents an observation of this random vector. Now we can reformulate the problem as finding the dependency of this random vector's covariance on the length of the sampling interval Δt.

To estimate the covariance matrix for each Kalman filter currently used in the tracker we used the MOF-segmented trajectories (acquired in the chapter 6) as an input to the estimation process. In other words for estimating the process noise covariance of the Kalman filter for estimating the state during cruising, only segments of trajectories with uniform motion were used and so on.

However, if we had investigated the process noise only in the consecutive ADS-B recordings, we would have gotten estimates for the Δt intervals of the length of 1 second only, because of the distribution of the interval Δt in the ADS-B data – as is visualized via histogram in the figure 7.2. Without any further details on used ADS-B units, we can assume the standard period of ADS-B broadcast messages to be 1 second. In order to examine the dependency of the process noise with period Δt longer than 1 second, we did not consider only consecutive ADS-B recordings in the estimation process, but also recordings that are not neighbouring. By doing so, we acquired samples of the measurement noise along with different lengths of interval Δt. We again grouped the measurements according to the interval Δt and for each group computed a covariance matrix.

Figure 7.2: Distribution of the interval between consecutive ADS-B recordings. The ADS-B broadcast period is obviously 1 second. Additionally, we assume that the ADS-B units broadcast multiple redundant messages – explaining the high amount of messages received sooner than 0.1 seconds after the previous message.



The results of the process noise estimation are visualized in the figure 7.3 – note that only diagonal elements of the covariance matrices are plotted – variances of the random variables. Results for uniform, accelerated motion and manoeuvres are plotted simultaneously to visualize their magnitudes. While, variance in the position measurement is plotted

[4]in the case of non-linear (turn) model the turn rate disturbance accounts for the fifth random variable

with both axes linear, for plotting variance in the velocity we chose semi-logarithmic axes. It can be clearly seen that the process noise is significantly higher in the accelerated motion model and during manoeuvres. We can also observe that variances in the X and Y direction differ. Presumably, this is caused by an unequal distribution of the trajectories in the corresponding directions – most trajectories are recorded in eastward or westward direction.

The evaluation of the empirically estimated process and measurement parameters was performed only on the segments with a specific mode-of-flight.

Table 7.1: Fitted linear models (in meters) to standard deviation of radar's measurement error.

radar1	$\sigma_{R1} = 0.000\,7range + 290$
radar2	$\sigma_{R2} = 0.001\,1range + 115$
radar3	$\sigma_{R3} = 0.001\,7range + 112$

The standard deviation of a measurement error in the horizontal position for each radar was approximated using a linear model. Table 7.1 presents the fitted polynomials for each radar (both range and the standard deviation are in meters). The computed standard deviations were during evaluation transformed into covariance matrices and used within the estimation algorithm.

The dependency of elements in the process noise covariance matrices on the interval Δt were approximated using linear or quadratic polynomials for each MOF. For every MOF we thus obtained a polynomial expressing the dependency of the particular element in the covariance matrix on Δt.

The evaluation was performed for each radar and each MOF (uniform, accelerated, manoeuvre) separately. That means we were not evaluating the performance of the IMM combining all modes-of-flight, instead we evaluated the performance of a single KF matching the corresponding MOF. The results are listed in the following table 7.2.

Table 7.2: Evaluation of the empirically estimated parameters.

	MOF	horizontal position RMSE [m]	speed RMSE [m/s]	heading RMSE [°]
radar1	uniform	454.8	97.44	8.54
	accelerated	358.3	21.22	20.83
	manoeuvres	296.8	8.58	31.49
radar2	uniform	155.1	9.19	8.24
	accelerated	101.5	12.01	9.67
	manoeuvres	113.6	9.19	7.85
radar3	uniform	296.1	9.67	9.07
	accelerated	125.56	14.65	9.97
	manoeuvres	157.31	14.29	9.57

The evaluation results are rather poor – especially, for the *radar1* on the trajectories with uniform motion, which should be easiest to estimate, the performance is of a really

low quality. Also, the heading cannot be considered to be estimated accurately enough. For *radar2* and *radar3*, the estimates of position and heading seem satisfactory, but the error in the velocity estimates is still fairly high.

Apparently, these results might be influenced by the precision of the measuring devices, and thus do not reflect the actual performance of the estimation algorithm. This would explain the acceptable performance of radars *radar1* and *radar2* in the horizontal position and worse performance on velocity vector estimates as the position is contained in the actual measurement, while the velocity components are estimated by the algorithm.

Next, we incorporated these empirically estimated parameters to the IMM algorithm (with original mode transition matrix) and ran the evaluation again with the results in 7.3. Again, the evaluation did not yield satisfactory results, as the errors reached high magnitudes in all of the considered performance measures. Obviously, the tracker's performance was not improved.

There are several possible reasons explaining this failure:

1. Incorrectly classified trajectories into the MOF classes. The classification might be correct from the human designer's point of view, but is not suitable for the algorithm.

2. Estimation of parameters from a non-representative sample of data.

3. The very non-linear dependencies in the parameters void the use of *empirical* estimation.

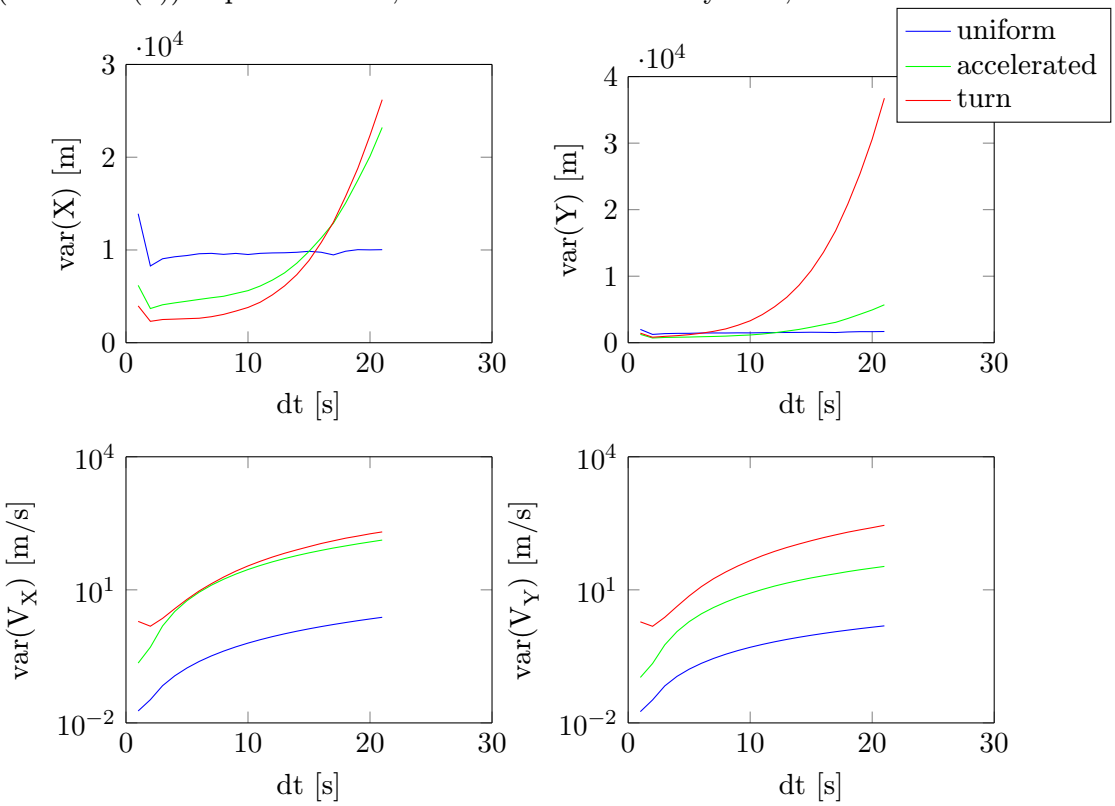
4. This kind of estimation assumes the process noise is the same for each radar. However, it might be that each radar observes the process differently and thus this assumption might not hold.

To sum up, this experimental approach again did not result in an improved configuration of the tracker and it is probably not suitable for the problem dealt. Nevertheless, we managed to get a good estimation for the measurement noise and thus for the following chapters we have a rough idea of its magnitudes.

Table 7.3: Evaluation of the IMM with empirically estimated parameters.

source	horizontal position RMSE [m]	speed RMSE [m/s]	heading RMSE [°]
radar1	411.59	28.32	10.39
radar2	180.34	12.03	9.38
radar3	284.02	15.91	10.82
fused	**322.75**	**17.84**	**9.29**

Figure 7.3: Diagonal elements of process noise covariance matrix estimated for uniform (blue), accelerated (green) motion and turns (red). The x-axes represents either variance (abbr. $var(x)$) of position in X,Y coordinates or velocity in X,Y coordinates.



Chapter 8

Evolutionary Optimization

The last approach examined in this thesis is the evolutionary optimization – a stochastic black-box optimization. We choose technique referred to as Covariance-Matrix-Adaptation Evolution Strategy (CMA-ES) from the family of Evolution Strategy (ES) algorithms.

The optimization using ES was already successfully applied for optimization of an IMM filter for radar tracking in [5], however, this optimization was carried out on a simulated trajectory data. In this work, we used real trajectory data as a training set for the optimizer.

The ES algorithm itself can be very easily parallelized making the optimization process more efficient especially when running on a multi-core system/cluster. In order to implement ES we need to define a fitness function (cost function that is to be minimized), develop a reasonable parameter encoding and decide on other design settings which are covered in the following subsections.

8.1 Fitness Function

The definition of a global fitness function should encompass the performance in all evaluated figures. In a multi-objective optimization, it is necessary to consider all goals simultaneously so that the solution is driven to the point where all evaluated figures achieve the best performance possible. In the general case, this is often not possible as different goals might be conflicting – in the meaning that optimizing one can lead to worsening the performance of another one. This makes it difficult to compare and rank the solutions directly. Therefore, one usually tries to find a set of alternative solutions, in the literature referred to as the Pareto-optimal front, where each solution represents best solution for one particular goal. Nonetheless, the problem of an ATC tracker optimization is so complex and computationally expensive, that generating complete Pareto front would be infeasible. In order to overcome this difficulty we defined the global fitness function as a summation of costs for each optimized figure transforming the multi-objective optimization problem into a single-objective one.

The fitness function is used for the evaluation of a single individual, i.e., a single configuration of the IMM algorithm. This configuration is used to initialize an instance of the IMM algorithm, which is then used for filtering training trajectories. The output is compared to the corresponding reference values computed from ADS-B recordings as is described in 9.1.2. The magnitude of errors is expressed by root mean square average computed for each figure separately.

The figures included in the fitness function are:

- horizontal position error,
- velocity vector amplitude,
- velocity vector angle (heading),
- *incorporation of the smoothness of heading is described in 8.3.*

Following criteria for global fitness function were identified as necessary in order to attain multiple objectives:

1. The magnitude of the figures included in the evaluation varies significantly. The error in horizontal position is usually in the order of hundreds, while the error in velocity vector components is occasionally greater than 10. To deal with these differences, it was necessary to carry out appropriate scaling so that all terms have similar influence on the overall cost. This was achieved by substituting individual cost terms by their partial cost expressing the level of attainment to some *objective value*. These objective values were defined for every evaluated figure and every radar individually. Their values were empirically estimated by performing several optimization test runs and evaluating the acquired results. The partial cost for goal x and the global fitness function for n goals is then defined as:

$$pc_x = \frac{c_x - obj_x}{obj_x}$$

$$COST = \sum_{x=1}^{n} pc_x$$

2. It might be that the algorithm is more successful in optimizing one figure, while other figures are hard to optimize. Moreover, the optimizer can minimize specific figure far below the objective value, thus making the partial cost negative which, in turn, can make the overall cost seem better while other figures' cost increases. To avoid this shortcoming, instead of taking the partial cost directly, we use its exponential. As seen in the figure 8.1 by doing so the individual cost decreases quickly as it is converging to the objective value, but once the objective is reached the cost decreases at a much slower pace and stays always positive. For this reason, it is necessary that all objective values are *sensible* as setting the objective to a very small value can easily cause error terms of very high magnitudes. Using exponentials also makes the cost function more sensitive to the changes in parameters. The final form of the cost function is:

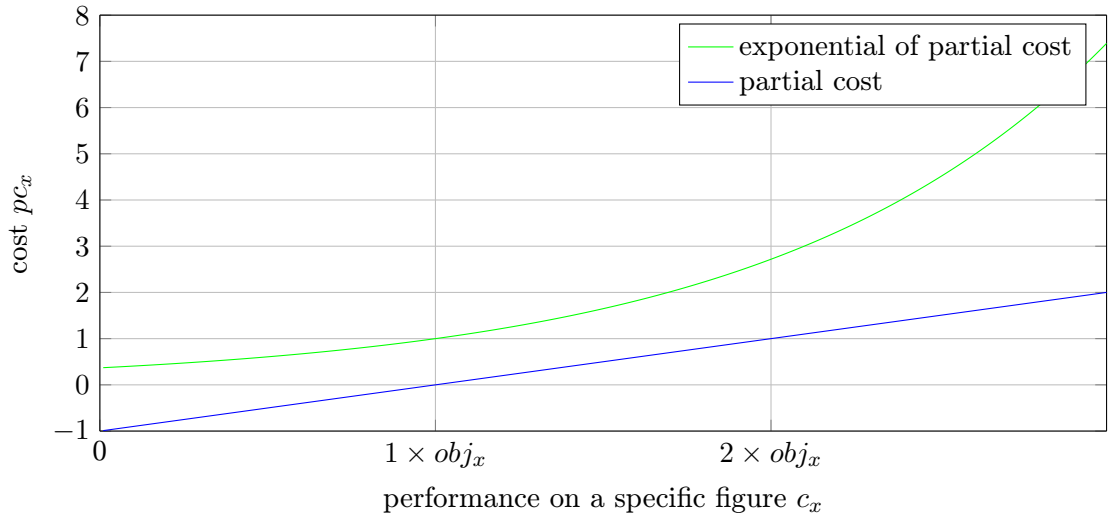
$$COST = \sum_{x=1}^{n} exp(\frac{c_x - obj_x}{obj_x})$$

   where c_x is the cost of an individual figure x and obj_x is its objective value.

3. Later, during experimenting with the algorithm we noticed that the optimizer sometimes found solutions that were *odd* and *unrealistic* – setting process noise magnitudes to the values, that were clearly wrong or constructing transition matrices where particular state transitions had been assigned zero probability. Visual evaluation of this

configuration also gave poor results, however, the fitness/cost of this configuration was very good. By inspecting the fitness evaluation, we figured out this was caused by introducing new outliers, which are excluded from the evaluation by the ESASSP specification (see chapter 2). The performance of these *wrongly* optimized configurations was so poor, that the algorithm marked most measurement as outliers and the cost was computed from a small non-representative sample. We addressed this issue by a *penalty term* that was set to a high value when new outliers were introduced.

Figure 8.1: Comparison of exponential and direct partial cost.



The global fitness function is evaluated on a set of trajectories. Yet using a big dataset of trajectories for the optimization would make the fitness function evaluation unnecessarily expensive. On the other hand, by choosing just a few trajectories we could easily run into over-fitting. Moreover, the distribution of a straight-line motion and turns in the training set significantly influences how well the tracker will perform on these classes of motion. We chose to use a set of 15 trajectories, carefully picked so that they were representative for all possible scenarios. And from these we even selected only *interesting* segments, trying to make the set as diverse as possible with all kinds of movements equally covered. In our training set, there is a considerably greater proportion of the turning to straight-line motion segments resulting in heavier weighting of the cost terms during manoeuvres. This strategy of using only a small subset of trajectories resulted in an assumption, that if the optimized IMM configuration performs well with this small set, the performance on the complete dataset will also be improved.

Furthermore, we believe that using a generic set of parameters for different radars cannot yield optimal results due to their varying physical characteristics and the corresponding measurement noise magnitudes. Moreover, varying coverage area implies that different segments of tracks are captured by a different radars, possibly resulting in a varying process noise characteristics. For these reasons in each optimization run, we work with measurements from a single radar only to optimize the performance of this particular radar.

Also, computing the output of the IMM algorithm for all three radars, coasting the states to common timestamps and fusing everything together would make the evaluation of the fit-

ness function very cumbersome and expensive. We assume that by optimizing the tracker for each radar separately, also the fused output of all radars will be optimized.

8.2 Parameters to Be Optimized

Depending on the structure of the IMM algorithm the number of parameters can vary greatly. The more motion models we incorporate in the IMM, the bigger the corresponding transition matrix is. Also each of these models requires supplying parameters describing the process and measurement noise. The configuration of the tracker currently deployed in Keflavik contains 15 design parameters; 3 describing process noise in the used motion models, 3 specifying the measurement noise magnitudes and 9 counting for the mode transition matrix.

8.2.1 Measurement noise model

For the purposes of the optimization we used linear models for the approximation of the measurement noise standard deviation, which is obviously *nearly* linearly dependent on the actual distance from the radar, as is visible from the figure 7.1. Such a simple measurement model was used for *radar1* and *radar2*, where the measurement error is assumed to have the same linear dependency on the whole range of the radar. However, for *radar3* we used 2 linear models, as the dependency clearly changes at the range of 60 Nm, possibly, because of higher quantization error or different encoding of the measured values. In this case, one linear model is used to approximate the measurement error up to the distance of 60 Nm and for the greater distances another linear model is used. The *radar3* has, therefore, four design parameters[1] influencing the measurement noise approximation, while *radar1* and *radar2* have only two parameters.

8.2.2 Parameter scaling

Not only the number, but also the magnitude and nature of these parameters may vary. The probability values in the transition matrix lie within the interval $(0, 1)$. However, the magnitudes of the process and measurement noise only need to be positive and as such are theoretically unconstrained. On the other hand, from our experience we can set constraints on them in the form of their *typical ranges*. These ranges are afterwards used for scaling so that the typical range of each parameter lies within values 0 and 1. In other words, we map the space of the typical parameters values inside the interval $< 0, 1 >$ using a convenient function that is invertible as we also need to map the scaled values back to the original parameter range.

From the definition of all the parameters enumerated above it is obvious that their valid values are always positive. For that reason, it is convenient to use for such a scaling functions from the logarithmic family. The functions we used are defined as:

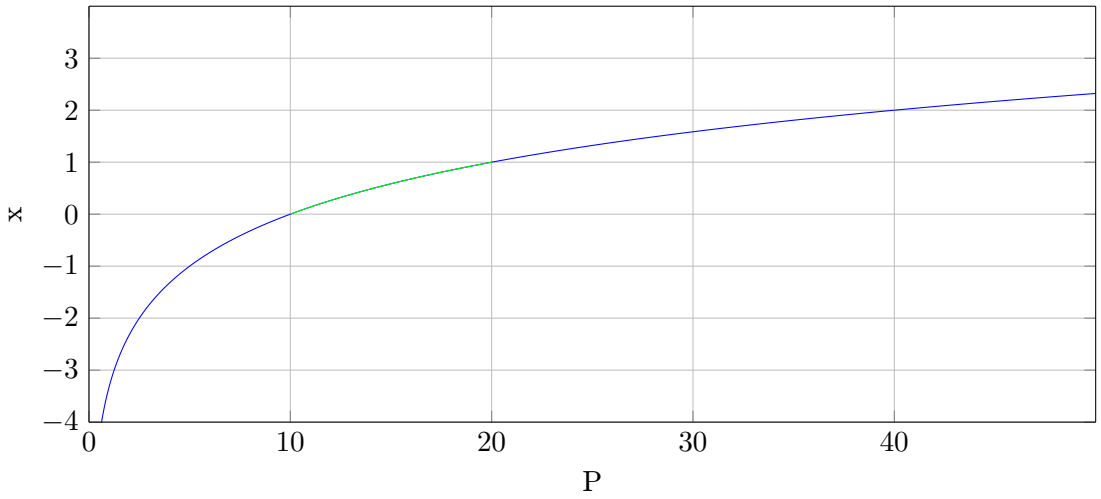
$$x = \frac{log\frac{P}{min}}{log\frac{max}{min}} = log_{\frac{max}{min}}\frac{P}{min}$$

$$P = min(\frac{max}{min})^x$$

[1] coefficients of the linear polynomials

where P is the value of the parameter with typical boundaries specified by min and max. x is then its scaled value.

Figure 8.2: Logarithmic mapping of P to x with both axes linear.



In the figure 8.2 is visualized the shape of the scaling function. The typical range of the parameter P, embossed in green, is set to lie within values 10 and 20. Apparently, the typical range is nearly linearly mapped to the interval (0,1) and the value of P is never negative. Another convenience of such mapping is certain tolerance for estimating the typical range of the parameter as also values close to the boundaries are *within a reach* of the optimizer.

8.2.3 Parameter reduction

The complexity of the black-box optimization problem is usually influenced by the properties of the objective function (e.g. roughness, smoothness) as well as the dimensionality of the searched space. Especially by reducing the dimensionality we can significantly influence the time needed by the stochastic search to converge. In our scenario, we can reduce the parameters determining the mode transition matrix by introducing a *proportional* notation. Instead of representing each element of transition matrix by an individual parameter, we define proportional relations among the matrix's elements, reducing the number of parameters by $\sqrt{n}$ for matrix with n elements.

Let vector R be an arbitrary row of the mode transition matrix consisting of three models – its elements p_1, p_2 and p_3 define the probability of switching from the specified mode to modes M_1, M_2 and M_3 correspondingly.

$$R = \begin{bmatrix} p_1 & p_2 & p_3 \end{bmatrix}$$

The corresponding row in the proportional transition matrix then reads:

$$R_{prop} = \begin{bmatrix} a_1 & a_2 \end{bmatrix}$$

$$a_1 = \frac{p_1}{p_2}, a_2 = \frac{p_1}{p_3}$$

To construct the mode transition matrix from the proportional transition matrix we simply solve a system of linear equations (the last equation holds from the definition of the mode transition matrix):

$$p_1 - a_1 p_2 = 0$$

$$p_1 - a_2 p_3 = 0$$

$$p_1 + p_2 + p_3 = 1$$

8.3 Incorporating Smoothness of Heading

As already mentioned in the chapter 5 another design goal is to make the waveform for track velocity angle smoother when the aircraft is cruising. However, it is not clear how this *smoothness* should be measured and thus in this section we develop a suitable formula.

We should also consider that by making the heading progression smoother we might get more noisy estimates about the position or velocity and thus the estimation error might increase. Clearly, there is a trade-off between a steady smooth curve and a dynamic curve quickly adapting to the changes. We shall also point out that it is not desirable to smooth the heading progression all over the track, but only within areas farther from the airport. This (locally *smoothing*) behaviour can be achieved by setting the measurement noise magnitudes reasonably larger for certain ranges, i.e., higher distances from the radar. Subsequently, such increased measurement noise makes the IMM algorithm *trust* the sensors less and considers its predictions of straight-line motion more *precise*, altogether possibly resulting in a *smoother* track.

For these particular reasons, it is impractical to model the measurement error to be linearly dependent on the distance from the radar, as a linear function cannot arbitrarily change its value. When running the optimization with the heading smoothness incorporated, we used stepping functions for the representation of the relationship between the measurement error and the distance from the radar.

Straight-line motion detection For the detection of a straight-line motion we used transversal acceleration estimates calculated from interpolated ADS-B data. A threshold value of 1.5 ms^{-2} is suggested by the ESASSP specification to differentiate straight-line from turning motion.

8.3.1 Measuring smoothness of a curve

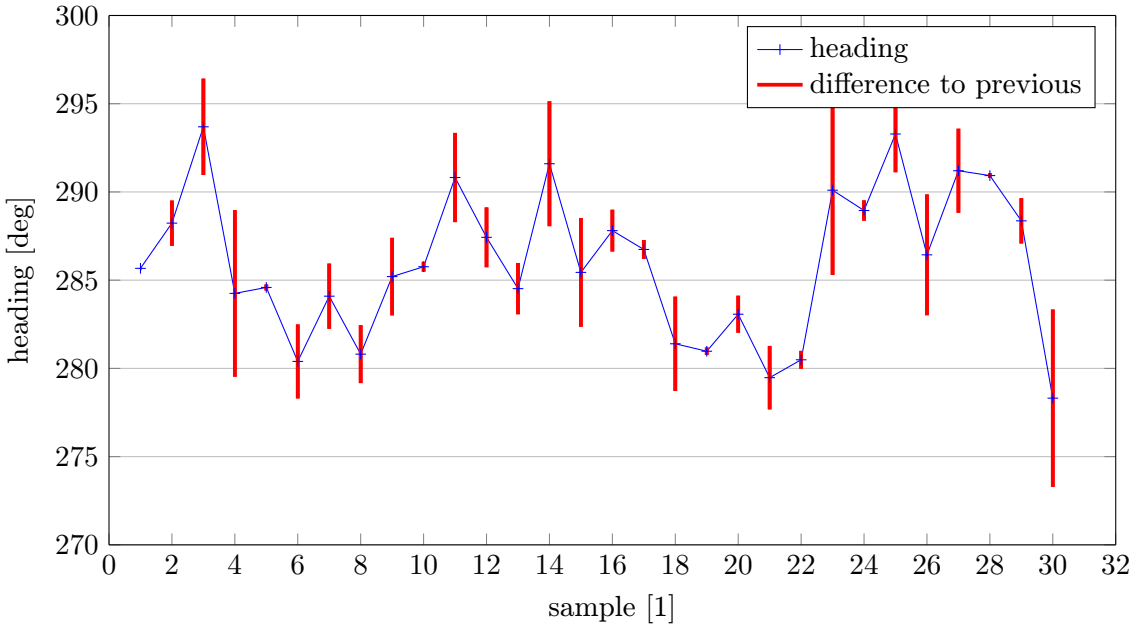
Smoothness as differences in the neighbouring samples

Our initial idea how to measure the actual smoothness was to involve in the computation the differences between neighbouring samples. This was rooted in the fact that samples belonging to a smooth curve does not change significantly. The smoothness of the whole curve is then defined as an average over these differences – we used the root mean square to weight the peaks heavier. In this fashion, to increase the smoothness of an arbitrary curve one would attempt to minimize samples' differences. The formula reads:

$$smoothness(X) = \sqrt{\frac{\sum_{i=2}^{n}(X_i - X_{i-1})^2}{n}}$$

where X denote the sequence of heading estimates and n is the number of these estimates.

Figure 8.3: Smoothness as a differences in the neighbouring samples.



In the figure 8.3 we can observe the results of the proposed method. The actual heading samples are plotted with a blue line. The red columns at each sample denote the difference to the previous sample. This method might seem sufficient, however, its drawback is the comparison to the preceding sample only. This does not guarantee that the curve will really be smoother on the wider range. Instead of incorporating more samples or developing other workarounds we developed another method.

Smoothness as differences of the samples to the corresponding smoothed curve

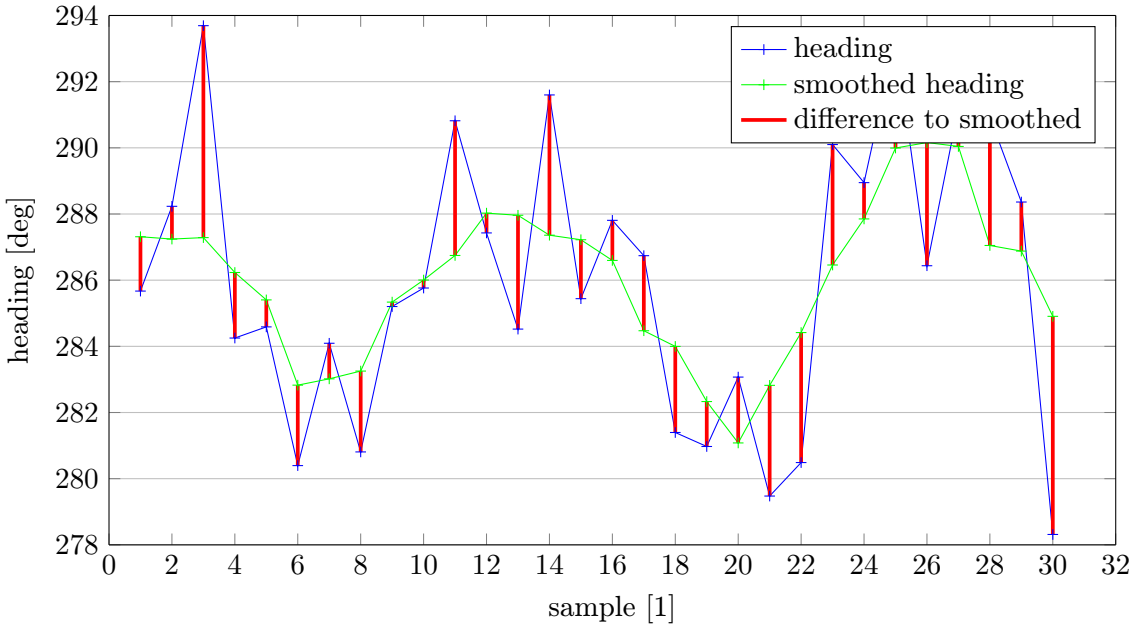
This method evaluates the smoothness of a curve by computing how much the curve differs from its smoothed equivalent. For every sample we determine the difference against corresponding sample in the smoothed curve. The measure for the whole curve is again computed as an RMS average over the differences. To compute the smoothed heading we used the IMM smoother (described in 3.3.2).

The figure 8.4 demonstrates the new method on the same sample data as used in the figure 8.3. We clearly see that peaks and sudden changes in the heading correspond to much higher differences than with the previous method.

8.3.2 Integration with the global fitness function

The smoothness of heading was integrated with the fitness function in a very similar manner as other figures. We again took the exponential of the partial cost computed for the smoothness, but this time we restricted the resultant cost by a stepping function. Once the desired level of smoothness is reached, the optimizer should not seek even more *smoothing* configuration as it would mostly give rise to higher errors.

Figure 8.4: Smoothness as a differences to the smoothed curve.



Chapter 9

Implementation

For the actual implementation we decided to use Matlab environment as its high-level technical computing language and interactive environment allows for fast development of algorithms and data processing tasks. The outcome of this master thesis is a Matlab toolbox for the ATC tracker optimization containing tools to work with a trajectory data, various structures of the IMM algorithm and its optimization. The toolbox was developed with Matlab 2014b.

Even though Matlab does not offer full object orientation, we chose this methodology for implementation because of the ability to define classes' responsibilities naturally and possibility of writing code that can be easily reused. The latter was essential during designing the class for the estimation algorithms that would allow easy incorporation of different motion models and IMM structures.

The overall design of the toolbox can be explored in the appendix A in form of a class diagram. A brief description of the important classes follows.

9.1 Handling of Trajectory Data

9.1.1 Trajectory class

One of the base classes in the toolbox is the `Trajectory` class. It is more or less just a convenient class that wraps around trajectory data. Each trajectory is in the system represented by an instance of this class which encapsulates corresponding measurements from different radars which are treated separately. Methods available to instances include functions for conversions between geographic coordinate system and local Cartesian system; and methods for trajectory splitting.

9.1.2 TrajectoryReconstruction class

We assume that to all tracks used throughout the project we have corresponding ADS-B data available. However, ADS-B recordings are taken at different times and with different period than the radar plots. In the toolbox we often need the reference track defined in the times when the radar plots were captured – some kind of a curve fitting, obviously, needs to be performed. The class `TrajectoryReconstruction` serves as a helper for computing reference trajectory data by interpolating the ADS-B data – to this task it uses Shape

Language Modelling toolbox[1]. The available reference quantities include measurements of position (both geographic and Cartesian), velocity vector components (amplitude and angle), acceleration (longitudinal and transversal) and turn rate. All of these are modelled with least square cubic splines. The spline knots are distributed uniformly in the time domain with a period of 5 seconds (heading), 10 seconds (position and speed) or 20 seconds (altitude). The accelerations and turn rate are computed from interpolated speed and heading recordings. These varying periods were set by examining the corresponding plots and were influenced by the rate of change of the specific measures in order to prevent the resultant curve from *oscillating* where the interpolated value changes slowly.

9.1.3 TrajectoryStub class

The design of this class was motivated by the needs of evaluating the fitness value of each individual during the evolutionary optimization. To calculate the fitness of an individual, one needs to run the estimation algorithm on the converted radar measurements and compare it to the reference data in order to get the error terms. Obviously, the estimated output differs with every individual. However, the reference data and radar measurements does not change. To keep the design of the `Trajectory` class *clean*, we decided not to encapsulate computed reference values and converted radar measurements in the `Trajectory` class instances, but to create another class where these precomputed data would reside.

By including measurements from the radar that is being optimized the size of the `TrajectoryStub` object is smaller than the corresponding `Trajectory` object – see the figure 9.1. This is beneficial for the parallel evaluation of individuals as the size of the data that needs to be broadcasted on the Matlab workers is also smaller.

Table 9.1: Size comparison of seven training trajectories represented by the Trajectory and TrajectoryStub objects [in Bytes].

class	size [B]					
`Trajectory`	204844	303948	310780	225500	213948	304188
`TrajectoryStub`	3176	17672	25544	17960	17960	23432

9.2 Estimation Algorithms

The IMM algorithm is in the toolbox represented by its own class `IMMFilter`. Its main functionality is obviously the algorithm itself, which was implemented abstract enough to allow for easy use of various process models (see section 9.2.1) or models of sensor noise (see section 9.2.2). This was achieved by defining common interfaces via abstract classes `IProcessModel` and `IMeasurementNoise`. Another functionality of the `IMMFilter` class is the method for coasting, i.e., predicting states.

The `IMMSmoother` class is derived from the `IMMFilter` class. `IMMSmoother` adds to the forward IMM algorithm (implemented by `IMMFilter`) its backward alternative, and thus, smoothes the estimates. These two classes share the same interface and as such can be interchangeably used.

[1] available from http://www.mathworks.com/matlabcentral/fileexchange/24443-slm-shape-language-modeling

For the fusion of filtered or smoothed estimates one can utilize the `SensorFuser` class, which is in the toolbox used to fuse the estimates of each radar supplied by the coupled `IMMFilter` or `IMMSmoother` instances. During the fusion process, each instance provides the estimates for the corresponding radar, then these estimates are coasted to the common timestamps and converted to the common local coordinate frame. Afterwards, the fusion of the track segments where 3 data sources are available is carried out, followed by the fusion of remaining segments with 2 data sources available. Eventually, the segments supplied by one source only are copied to the resulting track.

9.2.1 IProcessModel class and descendants

Various (both linear or non-linear) motion models are represented by the descendants of the class `IProcessModel` and are required to implement methods for computing the process noise covariance, state transition matrix or dynamic function (along with Jacobian) or providing the measurement model matrix.

The motion models included in the toolbox are implemented by classes `CVMotionModel`, `CTMotionModel` and `CTAcceleratedModel`. The instances of `CVMotionModel` describe constant velocity motion models with process noise injected in the velocity estimates. The constant turn motion is modelled by `CTMotionModel` and `CTAcceleratedModel`. The first one has turn-rate estimates perturbed by the process noise, while the latter one also velocity estimates.

9.2.2 IMeasurementNoise class and descendants

The measurement noise is in the toolbox represented by classes implementing interface `IMeasurementNoise`, which defines a method for computing the measurement noise covariances only. Two descendants are included in the toolbox. Class `PolyMeasurementNoise` estimates the sensor error by a linear function, while in the class `StepMeasurementNoise` this is approximated by a stepping function whose value changes in the predefined distances.

9.3 Optimization end Evaluation

The `Optimizer` class wraps methods used throughout the optimization process; the parameter scaling, mode transition matrix transformation, definition of the fitness function and integration of the CMA-ES algorithm. The `Evaluator` class implements all that is necessary for carrying out the evaluation of the specific figures from the aforementioned ESASSP specification. Its methods are also utilized by the `Optimizer` instances for evaluating the fitness function.

The CMA-ES algorithm is in the toolbox implemented by the stand-alone function `cmaes`[2] provided by the author of the CMA-ES algorithm – Nikolaus Hansen.

[2] available from https://www.lri.fr/ hansen/cmaesintro.html

Chapter 10

Results

In this chapter we deal with the results gained by employing techniques of evolutionary optimization to the problem of the tracker's performance optimization. In the first part we provide the reader with the details on the optimization algorithm configuration – especially the population size. In the following section we present various IMM structures possible for utilization in the tracker and evaluate which one to choose for each particular radar. After the *optimal* IMM structure is defined, we list the optimized parameters for that configuration along with the corresponding performance. In this part we left out the integration of the smoothness of heading in order to optimize the tracker's performance defined by its error magnitudes as much as possible. How the results changed after this requirement was incorporated is presented in the second part.

10.1 Optimization Process

The optimization was run within the Matlab environment on a computing cluster with 24 cores, making the parallel optimization very effective. The optimization algorithm itself has a few tuning parameters. For every parameter left free for the optimization we also needed to define a typical range, a boundaries within which the *optimal* value of parameter should reside. This was necessary for the purposes of scaling described in 8.2.2. The table 10.1 below summarizes all parameters used throughout different models along with their boundary values.

Table 10.1: Typical ranges for the parameters involved in the optimization.

Used in	Parameter	Typical range
CV motion model	Process noise PSD - uniform	$[10^{-6}, 10^{-2}]$
	Process noise PSD - accelerated	$[10, 300]$
CT motion model	Variance in the turn rate	$[10^{-6}, 0.1]$
Measurement noise model - linear	Std. dev. of error polynomic coefficients $y = ax + b$	$a \in [10^{-4}, 0.02]$ $b \in [20, 150]$

After variables encoding, the initial solution point x^0 and the initial standard deviation (stepsize) σ^0 must be chosen. We started by trying to improve the current configuration locally choosing the value of σ^0 rather small, i.e., 0.2. After that we also started

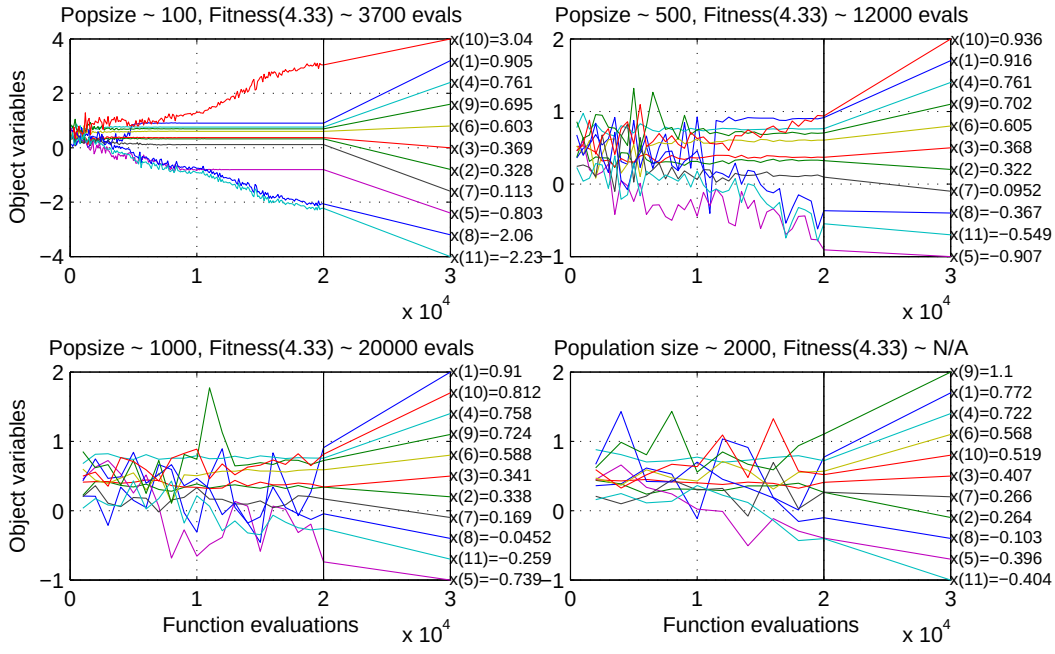
the optimization runs from random initial points with larger value of 0.4. We note that the parameter vector is scaled to lie within $[0, 1]$. However, most of the time they converged to the solutions that were close to the default configuration. That made us focus more on improving the current configuration as it was clearly close to the optimum.

Population size

In order to determine the appropriate population size and corresponding size of parental population we need to take into account several aspects. Firstly, considering the dimensionality of the problem dealt, which is in the most complex scenario higher than 10, we experimented with populations of several hundreds to thousands individuals in order to cover the searched space as much as possible. On the other hand the population size has a tremendous impact on the time needed to evaluate and rank all generated individuals. Furthermore, in conjunction with the dimensionality, stepsize σ^0 and the size of a parental population it significantly influences the count of iterations needed for the algorithm to converge – to reach a stopping criterion.

To estimate the optimal setting we performed several test optimizations with various population sizes and examined how long (measured by the count of fitness evaluations) it took the algorithm to converge. The initial solution for these test runs was the current configuration of the tracker (thus the search space has 11 dimensions) with σ^0 set to 0.1. The size of parental population was left to be one half of the offspring population.

Figure 10.1: Evolution of object variables for different sizes of population. To each plot we added how many function evaluations were needed for the algorithm to reach fitness function value 4.33. *Note that with the population size of 2000 individuals, this objective was not reached.*



The results of these test runs are plotted in the figure 10.1 where the plots show the evolution of object variables through the optimization with population sizes of 100, 500, 1 000 and 2 000 individuals. The different rate of convergence is clearly visible. For this particular

reason we consider the algorithm converging when the value of the majority of its object variables stops to change *significantly*. We see that with smallest population the algorithm converged after approximately 15 000 function evaluations, for the population of 500 individuals we reckon maybe slightly more than 25 000 function evaluation would be needed. We can observe clear correlation between the population size and the number of function evaluations necessary for convergence. From the plots we assume that with populations of 1 000 and 2 000 individuals the algorithm is far from converging.

To evaluate the efficiency and performance for varying population sizes, we examined how many fitness function evaluations were performed in order to reach *objective* value of 4.33. Surprisingly, the smallest population reached the value after performing only 3 700 evaluations, followed by 500-individuals population reaching the objective after 12 000 evaluations. The biggest population did not reach the objective even after 20 000 evaluations. However, we note that the optimization instance with 20 000 individuals had not converged and the algorithm might have found later much better solution than the instances, that have already converged to, *possibly*, local optima.

The results presented in the following section were achieved by using populations of 500 individuals for all optimizing scenarios.

Stopping criteria

There are several criteria defined, that make the optimization algorithm stop whenever any of them is met. Naturally, one can declare the algorithm to stop when some objective value is reached, however, in our scenario it is not clear what is the cost of the global optimum. Similarly, the count of algorithm iterations (generations) or function evaluations might be restricted.

During our experiments we considered the algorithm converged when the change in the variable vector was too low (10^{10} smaller) compared to the initial standard deviation or when the fitness function value was stagnating.

The opposite situation, when the change in the variable vector was more than 10^3 times larger then initial standard deviation was recognized as a divergence.

10.2 Different Structures of the IMM

In total we analysed 3 different IMM structures. They are summarized in the table 10.2 along with a description of dynamic models they comprise and parameters that were optimized using CMA-ES optimization.

The structure currently implemented in the tracker is denoted by the label *IMM3*, i.e., two linear CV motion models for estimating uniform and accelerated straight-line motion with varying process noise magnitudes and one non-linear CT motion model, internally estimating the turn rate – note that in this CT model only turn rate is perturbed by a random noise. Although standard deviation of the measurement noise is in the tracker represented by a stepping function (refer to 4.2), here we used linear function to model the dependency of a measurement noise on the distance from radar as it might provide more accurate estimates.

Another structures subjected to optimization are consisting of two Kalman filters only making the IMM algorithm estimation less computationally expensive. The first structure *IMM1* is the simplest one, with two linear CV motion models. We did not expect it to

yield comparable results as the more sophisticated structure *IMM3*, however, for *radar1* such a simple setting might be sufficient.

In the last structure analysed (*IMM2*) we exchanged the more noisy (accelerated) linear model with a CT motion model similar to the one currently used in the tracker, but the process noise is perturbing not only turn rate but also velocity, theoretically combining noisy linear model with CT model containing *noisy* turn rate.

Table 10.2: Analysed IMM Structures.

Label	Parameters	Description
IMM1	6	• Two linear CV models (uniform, manoeuvres) with varying process noise variances • Linear model for measurement noise estimation • Mode transition matrix 2x2
IMM2	6	• One linear CV model (uniform) with low process noise variance • One non-linear CT model (manoeuvres) with estimated turn rate • Linear model for measurement noise estimation • Mode transition matrix 2x2
IMM3	11	• One linear CV model (uniform) with low process noise variance • One linear CV model (acceleration) with high process noise variance • One non-linear CT model (manoeuvres) with estimated turn rate • Linear model for measurement noise estimation • Mode transition matrix 3x3

The table 10.3 presents the evaluation results for optimized configurations of the IMM structures listed in the table 10.2. This evaluation was performed on the same dataset as the one used for the evaluation of the current configuration (to be found in the section 4.2).

For the radar *radar1* we see that using both simpler IMM structures (*IMM1*, *IMM2*) comprising tow Kalman filters only yielded very similar results in all evaluated figures to the more complex settings of *IMM3*. On the contrary, for *radar2* and *radar3* the current IMM structure (*IMM3*) clearly outperformed the two simpler ones, especially, in the heading error evaluated for the segments with turning motion.

We believe these differences are due to varying physical characteristics of the radars – especially, higher scan rate of *radar2* and *radar3* as well as varying error magnitudes of all three radars. These differences makes each radar observe a slightly different process. For example, *radar1* – low scan rate – captures significantly fewer measurements during manoeuvres which usually do not last long. As a result the *radar1* more or less captures just straight segments – a fragments of the whole trajectory. This makes the poor ability of estimating turns less significant. *Radar2* and *radar3* – with higher scan rate – clearly benefit of using non-linear structure to precisely estimate the track when turning, as they can incorporate more measurements and the estimated turn rate is in return more accurate.

Table 10.3: Evaluation of optimized configurations. Error statistics for velocity vector components are divided for straight-line motion and turns.

	IMM struct.	horz. pos. RMSE [m]	speed RMSE [m/s]	heading RMSE [°]
radar1	**IMM1**	**378.40**	**4.48/5.69**	**2.02/15.73**
	IMM2	379.21	4.71/6.44	2.28/10.91
	IMM3	378.10	4.38/5.82	2.00/16.62
radar2	IMM1	156.60	4.43/5.19	2.26/15.65
	IMM2	150.9	4.72/4.57	2.91/11.62
	IMM3	**146.50**	**5.07/4.89**	**3.21/3.92**
radar3	IMM1	261.70	9.37/7.76	3.47/16.51
	IMM2	244.9	5.70/5.74	3.04/15.86
	IMM3	**245.90**	**5.71/6.00**	**2.61/7.76**

10.3 Incorporating the optimized IMM structure

In the above we assessed the performance of the optimized parameters for each radar separately. Further, we examined how the tracker's output (fused from all three radars) is influenced based on which IMM structure we choose for the estimation of *radar1*. This is summarized in the table 10.4. Obviously, the quality of the fused output increased compared to the original configuration listed in 4.3 for all IMM structures. From the above results we assumed that IMM structures *IMM1* and *IMM2* are nearly equivalent and both will have similar influence on the overall performance. However, the resulting performance differed significantly.

Table 10.4: Evaluation of the selected and optimized IMM structures. Error statistics for velocity vector components are divided for straight-line motion and turns.

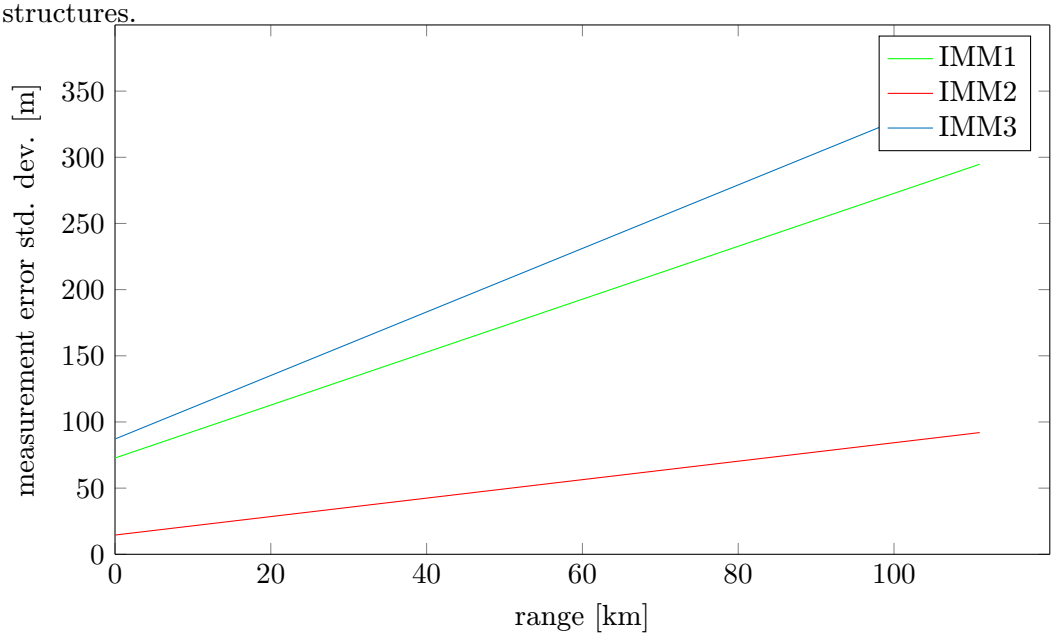
radar1	horz. pos. RMSE [m]	speed RMSE [m/s]	heading RMSE [°]
IMM1	**240.70**	**3.68/5.91**	**1.58/11.07**
IMM2	289.73	3.69/5.97	2.29/13.24
IMM3	242.23	3.69/5.94	1.57/11.33

Our presumption for the cause of this issue was very varying magnitude of the measurement noise estimated for *radar1*. As seen in the figure 10.2, for structure *IMM2* it was notably smaller than with remaining configurations *IMM1* and *IMM3*. Our hypothesis was confirmed by comparison of the state estimation error covariance matrices produced by IMM algorithm with all IMM structures on the input data from *radar1*. To compare the covariance matrices we used the following principle:

$$A > B \rightarrow x^T(A - B)x > 0, \forall x \neq 0$$

In other words $A > B$ if and only if the matrix $A - B$ is positive definite. The results of this comparison carried out on several trajectories showed that the covariances with the opti-

Figure 10.2: Optimized measurement noise standard deviation for *radar1* and various IMM structures.



mized structure *IMM2* were significantly smaller. This resulted in a heavier weighting of *noisier radar1* during the fusion process, which worsened the overall performance. The remaining *IMM1* and *IMM3* yielded larger covariance matrices and thus *radar1* was in these configurations weighted considerably lesser.

Possible solution could be modification of the global fitness function so that it takes into account the magnitude of covariances for the particular radars, however this would be hard to define as it is not clear what the *appropriate* magnitudes are. Other solution could be optimization of the whole tracker at once, i.e., optimizing configuration of all radars and defining the fitness function to evaluate the fused output.

To sum up, the optimized parameters for the current IMM structures implemented yielded good results as the quality of tracker's output increased. However, for the *radar1* a simpler IMM structure referred to as *IMM1* could be used and thus the computational load would decrease without worsening the output's quality. Configuration parameters for this optimized structure are to be found in the table B.

10.4 Smoothness Incorporation

In order to fulfil the requirement for a smoother heading while the aircraft is cruising we redefined the cost function as is described in 8.3. We also changed the model of measurement noise – for this optimization scenario we modelled the measurement error by a stepping function with steps at 40 and 60 Nm – as is currently implemented in the tracker. The interval boundaries 40 Nm and 60 Nm were kept and only corresponding magnitudes were left free for optimization. The cost of the heading smoothness was computed for measurements farther than 40 Nm from the radar. Otherwise, the optimization was run in a very same manner as in the previous sections. However, this time we optimized only the IMM structure assumed to be most efficient; that is structure *IMM1* for *radar1* and *IMM3* for *radar2* and *radar3*.

In the table 10.5 we can examine acquired results. We see that the error statistics increased a little as we anticipated.

Table 10.5: Evaluation of the smoothed configuration. Error statistics for velocity vector components are divided for straight-line motion and turns.

source	horz. pos. RMSE [m]	speed RMSE [m/s]	heading RMSE [°]
radar1	382.1	5.92/6.03	4.11/12.03
radar2	152.3	5.48/5.18	2.78/5.00
radar3	248.9	5.69/5.72	2.84/9.41
fused	242.8	3.85/5.65	1.85/11.41

In the next table 10.6 we can evaluate how much the smoothness of the trajectory increased based on the developed measuring function described in 8.3.1. Obviously, the smoothed configuration did not yield any significant improvement over the original optimized configuration. This failure in obtaining configuration that would provide smoother trajectory might be explained that the objective defined as *minimal changes in the heading while the aircraft is cruising* is already covered by an objective to minimize the error in heading. Clearly, during cruising the aircraft maintains a straight-line motion. The reference track during cruising, therefore, consists of a straight-line motion only and as such, represents the *smoothest* track. By minimizing the error in the heading, we are also minimizing the differences to the reference heading estimates, which is basically a definition of our smoothness-measuring function. In other words, the configuration that was optimized without the incorporation of the heading smoothness, can already be regarded as a smoothing configuration. The RMS average of differences in filtered heading and smoothed heading, in 10.6, and also the overall error[1] in heading for straight-line motion, prove this hypothesis.

Table 10.6: Comparison of the smoothed and non-smoothed configuration.

source	non-smoothed	smoothed
radar1	1.339	1.151
radar2	2.005	1.874
radar3	1.867	1.750
fused	1.349	1.266

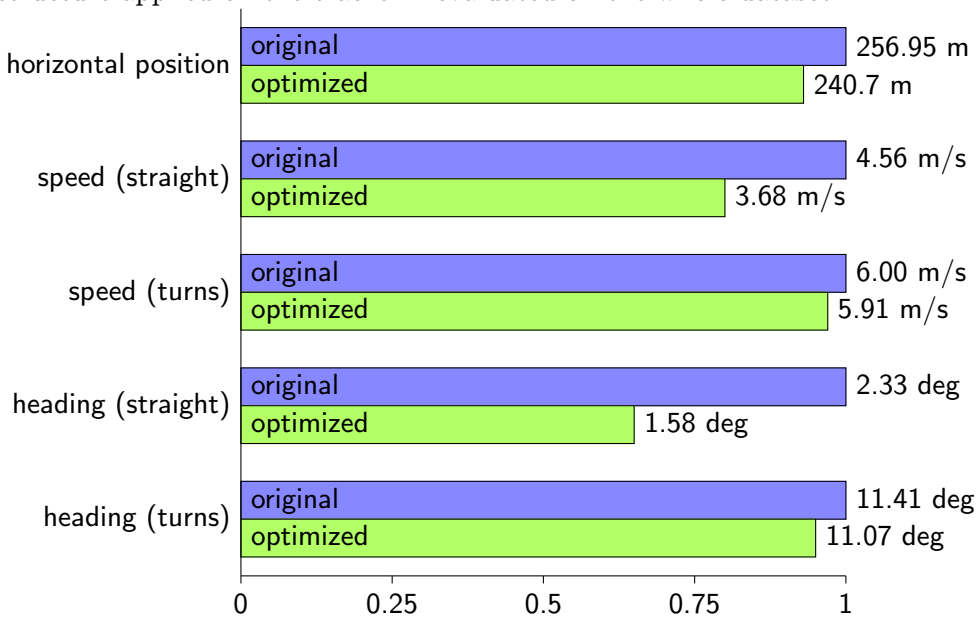
10.5 Best configuration

Based on the experimental results described above we determined the best configuration and structure for the tracker. For estimation of radar *radar1*, the IMM structure labelled as *IMM1* is recommended; while for *radar2* and *radar3* we recommend more sophisticated IMM structure *IMM3*. The detailed configuration for these structures is to be found in the appendix B.

[1] difference to the reference value

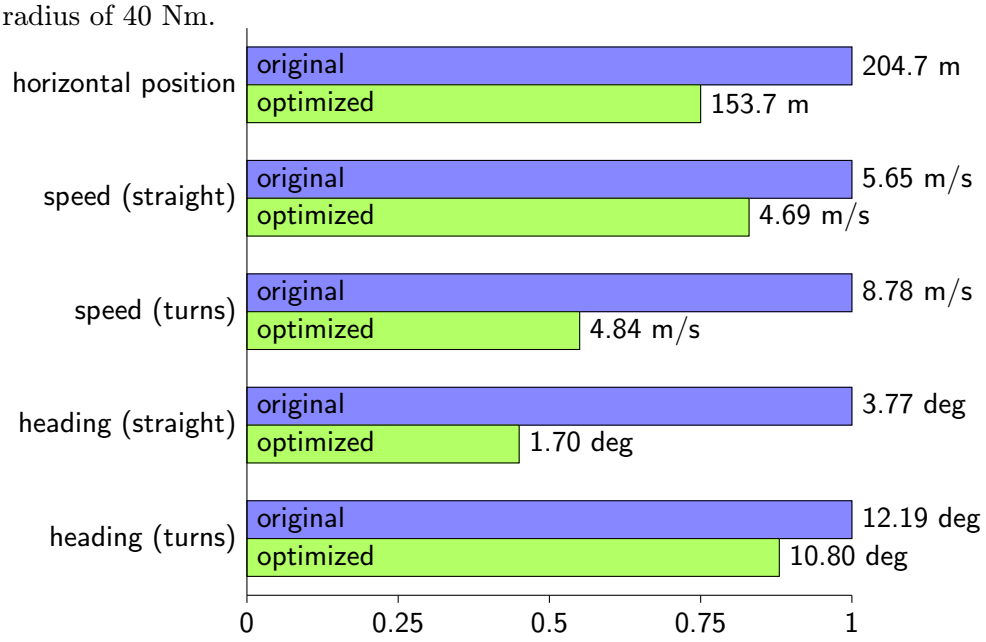
The direct comparison of the optimized and original configuration is in the figure 10.5. This chart presents results obtained by running the ESASSP evaluation on the whole dataset of trajectories – used throughout the thesis.

Figure 10.3: Comparison of the evaluation of optimized and original configuration and structure applied on the tracker – evaluated on the whole dataset.



More dramatic improvements can be seen on the performance evaluation in the figure 10.4 – depicting evaluation performed for a smaller area around the airport where most of the manoeuvres take place. This area was defined by a circle with a radius of 40 Nm. This significant difference in improvement over the previous plot is caused by the selection of trajectories included in the training set. As explained in the chapter 8.1 by careful selection of trajectories we can influence how the tracker will perform on various segments of trajectories (uniform, accelerated, manoeuvres).

Figure 10.4: Comparison of the evaluation of optimized and original configuration and structure applied on the tracker – on a smaller area around airport defined by a circle with radius of 40 Nm.



Chapter 11

Conclusions

In this thesis, we dealt with the optimization problem of an aircraft tracker which uses IMM algorithm with a set of Kalman filters. The tracker fuses outputs of several radars, each coupled with an IMM algorithm instance. The tracker's original configuration did not take into account different characteristics of the radars and defined a generic settings that were applied to all IMM algorithm instances.

To evaluate the tracker performance we studied the ESASSP specification, every ATC tracker should fulfil. For our purposes we extracted several performance measures: root-mean-square error in the horizontal position and root-mean-square error in the track velocity vector components – amplitude and heading – which were used for evaluation of different configurations and structures of the IMM algorithm.

Our initial approach was based on the idea of training the Kalman filter separately on sets of segmented trajectories by a mode-of-flight. We assumed that if a particular filter performs well on a class of trajectories it is designed for, the whole set of Kalman filters (the IMM algorithm) will perform well too. For the optimization of Kalman filters, we planned to use the well-known EM algorithm commonly used for training Markov models. To learn the transition matrix of IMM algorithm we expected Baum-Welch algorithm (derivate of EM algorithm for Hidden Markov Models) to give good results. The prerequisite for this approach is a correct segmentation of the available recorded data into classes by the type of motion/mode-of-flight: uniform, accelerated and other manoeuvres. While completing this task, we have shown that ensemble learning along with boosting algorithms are an effective way to classify trajectories based on radar data, even if the data is skewed and noisy. We determined what the best features for such classification are and also what is the right number of surrounding measurements features to include in the feature set. The trained model, with best configuration found, was tested on real data with accuracy around 94% that was deemed as satisfactory. However, due to the insufficient initial research in the area of parameter estimation for continuous-time models our original approach proved to be impracticable since the derivation of the log-likelihood function – needed for the EM algorithm – does not have a closed-form solution. Further investigation in the field yielded no results, from which we assume that so far no appropriate algorithm was published to this estimation problem.

As an experiment, we tried to estimate the process noise and measurement noise empirically from the available data. Both noise terms are of time-varying nature. For the estimation of the process noise, which is dependent on the length of the sampling interval, ADS-B data were used as they represent, in our scenario, true samples of the process itself. The estimation procedure was based on computing covariance matrices to the samples of

process noise – computed as the differences of coasted states (i.e., predicted and thus without process noise) and true states (with process noise) represented by ADS-B recordings. The process noise was in this manner estimated for each mode-of-flight by carrying out the estimation procedure on the mode-of-flight-segmented trajectories. Similar approach was applied to the estimation of the measurement noise, where we fitted normal distributions to the samples of the measurement noise represented by the deviations of the radar measurements from the reference values (spline interpolated ADS-B data). The acquired results showed nearly linear dependency of the measurement noise standard deviation on the range (distance from the radar). To evaluate this empirically acquired estimations we plugged the constructed models for process and measurement noise into the IMM estimator and carried out the ESASSP evaluation. However, the yielded results failed to meet the demanded qualities both in the terms of horizontal position error as well as error in the groundspeed vector.

Lastly, we examined utilization of a stochastic black-box optimization method, Covariance-Matrix-Adaptation ES. In order to utilize the methodology of evolutionary computation we defined a fitness function for evaluating each individual's *performance* expressing its level of attainment to the specified accuracy statistics. This cost measure was computed on 15 selected trajectories covering as many possible scenarios as possible – uniform segments, accelerating segments as well as segments where the aircraft was turning or performing more complex (combined) maneuvers. The subterms of the global fitness function were error measures in the horizontal position and velocity vector components. The designed cost function conveniently combines all objective measures into one abstraction. We also showed how to scale the individual cost terms or parameters such that they reside within a certain boundaries; and the encoding used for the IMM mode transition matrix reduced the number of parameters by n in the case of $n \times n$ transition matrix.

This optimization approach resulted in an optimized configuration for each radar. We simultaneously proved that it is desirable to have a distinct configuration for each radar source as they differ in the measurement noise magnitude and also the observed process evinces different characteristics. The separate evaluation of the optimized configurations yielded good results, and also the quality of the fused output was improved significantly when evaluated on the available datasets. The error statistics went down in all interested measures by tens of percent, and the results are deemed satisfactory by Tern Systems – the supplier of the tracking software. Specifically, the RMSE in horizontal position was 240.7 m (originally 257.0 m); on straight segments of the track the RMSE in groundspeed vector amplitude was 3.68 m/s (originally 4.56 m/s) and heading 1.58° (originally 2.33°); and on segments with turns the RMSE in groundspeed vector amplitude was 5.91 m/s (originally 6.0 m/s) and heading 11.07° (originally 11.41°).

Also, the additional requirements were fulfilled. Originally, the radar source *radar1* was coupled with an IMM instance consisting of three Kalman filters. We simplified this IMM instance by reducing the number of employed Kalman filters while preserving the quality of the estimator's output. The tracker with the simplified structure is thus more computationally efficient which is highly beneficial when tracking multiple targets. The second additional objective was to smooth the track when the aircraft is cruising. However, by reformulating the requirement we came to a conclusion that this is already covered by the objective of minimizing the error in heading and thus this optimization objective was dropped.

The outcome of this thesis is a Matlab toolbox that eases work with trajectory data, estimation algorithms and corresponding optimization. The estimation algorithms are

implemented in a very abstract way that allows incorporation of different motion models or even totally different processes. The included estimation algorithms are based on the IMM algorithm and encompass both IMM forward filter and IMM forward-backward filter (smoother). Included is also a variety of motion models to enable experiments with various structures of the IMM. The optimization classes are based on the CMA-ES algorithm that can be easily substituted by another one if needed. The toolbox automates the process of finding a good configuration for the IMM algorithm applied to maneuvering target tracking, and greatly helps to decide on the suitable structure of the IMM algorithm itself.

There are many possible extensions to this thesis as we dealt only with the performance in the horizontal plane and configuration for commercial aircraft. Obviously, the tracker needs to estimate also the altitude of an aircraft. For this estimation a single linear Kalman filter is currently applied and, naturally, its configuration could be optimized using the implemented toolbox. Furthermore, the optimized configuration for the tracker performs well on commercial aircraft only, as other kinds of aircraft evince different motion patterns and thus should be modelled by different motion models. For example, small private airplanes or military aircraft undergo acceleration of considerably different magnitudes or perform turns at higher turning rates and, consequently, the process noise magnitude in the corresponding Kalman filters would differ. With military aircraft the configuration would be even harder to optimize as their motion can exhibit very sudden changes at nearly all estimated measures. Probably, application of substantially different motion models would be inevitable. Moreover, the problem of estimating the position of military aircraft is hard due to low scan rate of the radars, which, in return, have very poor ability to observe or even predict the kinematic state of fast targets, such as fighter planes.

Bibliography

[1] Anne Auger and Nikolaus Hansen. A restart cma evolution strategy with increasing population size. In *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, volume 2, pages 1769–1776. IEEE, 2005.

[2] Thomas Bäck and Michael Emmerich. Evolution strategies for optimisation in engineering applications. In *Proc. 5th World Congress on Computational Mechanics*, 2002.

[3] Thomas Back, David B Fogel, and Zbigniew Michalewicz. *Handbook of evolutionary computation*. IOP Publishing Ltd., 1997.

[4] Yaakov Bar-Shalom, X Rong Li, and Thiagalingam Kirubarajan. *Estimation with applications to tracking and navigation: theory algorithms and software*. John Wiley & Sons, 2004.

[5] Juan A Besada, Jesús García, Gonzalo de Miguel, Antonio Berlanga, José M Molina, and José R Casar. Design of imm filter for radar tracking using evolution strategies. *Aerospace and Electronic Systems, IEEE Transactions on*, 41(3):1109–1122, 2005.

[6] Gary Bishop and Greg Welch. An introduction to the kalman filter. *Proc of SIGGRAPH*, 8:41, 2001.

[7] R.G. Brown and P.Y.C. Hwang. *Introduction to random signals and applied Kalman filtering: with MATLAB exercises and solutions*. Introduction to Random Signals and Applied Kalman Filtering: With MATLAB Exercises and Solutions. Wiley, 1997.

[8] Nitesh V Chawla. Data mining for imbalanced datasets: An overview. In *Data mining and knowledge discovery handbook*, pages 853–867. Springer, 2005.

[9] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16(1):321–357, 2002.

[10] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977.

[11] Tristan Fletcher. The kalman filter explained. online, http://www.tristanfletcher.co.uk/LDS.pdf, 2010.

[12] Yoav Freund and Robert E Schapire. A desicion-theoretic generalization of on-line learning and an application to boosting. In *Computational learning theory*, pages 23–37. Springer, 1995.

[13] Jesus García, Oscar Pérez Concha, José M Molina, and G de Miguel. Trajectory classification based on machine-learning techniques over tracking data. In *Information Fusion, 2006 9th International Conference on*, pages 1–8. IEEE, 2006.

[14] Zoubin Ghahramani and Geoffrey E Hinton. Parameter estimation for linear dynamical systems. Technical report, Technical Report CRG-TR-96-2, University of Totronto, Dept. of Computer Science, 1996.

[15] Nikolaus Hansen. The cma evolution strategy: a comparing review. In *Towards a new evolutionary computation*, pages 75–102. Springer, 2006.

[16] Nikolaus Hansen and Stefan Kern. Evaluating the cma evolution strategy on multimodal test functions. In *Parallel problem solving from nature-PPSN VIII*, pages 282–291. Springer, 2004.

[17] Nikolaus Hansen and Andreas Ostermeier. Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation. In *Evolutionary Computation, 1996., Proceedings of IEEE International Conference on*, pages 312–317. IEEE, 1996.

[18] Nikolaus Hansen and Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary computation*, 9(2):159–195, 2001.

[19] Jouni Hartikainen, Arno Solin, and Simo Särkkä. Optimal filtering with kalman filters and smoothers. *Department of Biomedica Engineering and Computational Sciences, Aalto University School of Science: Greater Helsinki, Finland*, 16, 2011.

[20] Trevor Hastie, Robert Tibshirani, Jerome Friedman, and James Franklin. The elements of statistical learning: data mining, inference and prediction. *The Mathematical Intelligencer*, 27(2):83–85, 2005.

[21] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Journal of Fluids Engineering*, 82(1):35–45, 1960.

[22] K Kastella and M Biscuso. Tracking algorithms for air traffic control applications. *Air Traffic Control Quarterly*, 1996.

[23] Sotiris B Kotsiantis, I Zaharakis, and P Pintelas. Supervised machine learning: A review of classification techniques, 2007.

[24] X Rong Li and Vesselin P Jilkov. Survey of maneuvering target tracking. part i. dynamic models. *Aerospace and Electronic Systems, IEEE Transactions on*, 39(4):1333–1364, 2003.

[25] X Rong Li and Vesselin P Jilkov. Survey of maneuvering target tracking. part v. multiple-model methods. *Aerospace and Electronic Systems, IEEE Transactions on*, 41(4):1255–1321, 2005.

[26] Richard Maclin and David Opitz. Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research*, 1999.

[27] Peter S Maybeck. *Stochastic models, estimation, and control*, volume 3. Academic press, 1982.

[28] O Perez, Jesús Garcia, and José M Molina. Neuro-fuzzy learning applied to improve the trajectory reconstruction problem. In *Computational Intelligence for Modelling, Control and Automation, 2006 and International Conference on Intelligent Agents, Web Technologies and Internet Commerce, International Conference on*, pages 4–4. IEEE, 2006.

[29] J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

[30] Lior Rokach. *Data mining with decision trees: theory and applications*. World scientific, 2007.

[31] Lior Rokach. Ensemble-based classifiers. *Artificial Intelligence Review*, 33(1-2):1–39, 2010.

[32] Chris Seiffert, Taghi M Khoshgoftaar, Jason Van Hulse, and Amri Napolitano. Rusboost: improving classification performance when training data is skewed. In *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, pages 1–4. IEEE, 2008.

[33] Robert H Shumway and David S Stoffer. An approach to time series smoothing and forecasting using the em algorithm. *Journal of time series analysis*, 3(4):253–264, 1982.

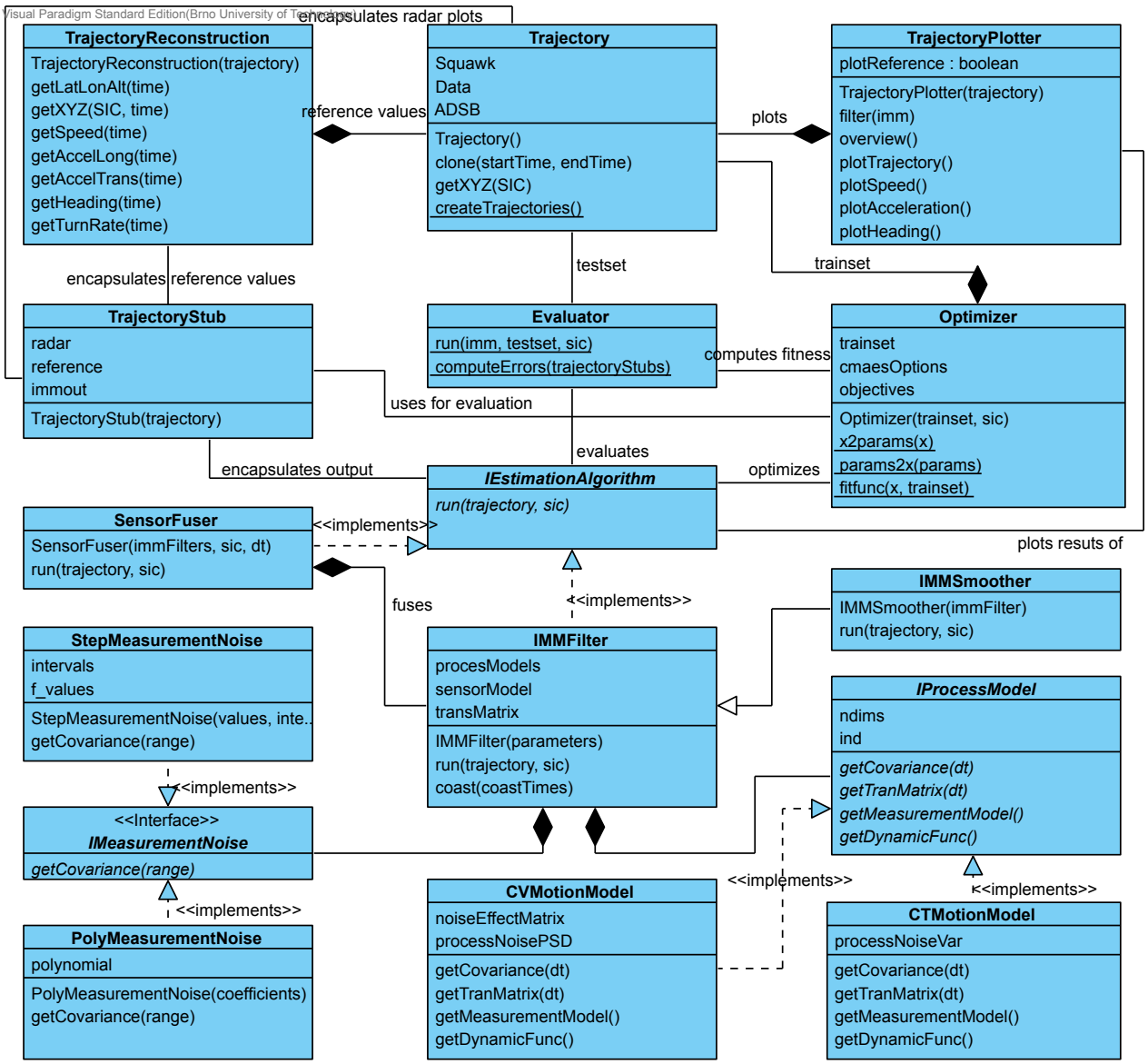
[34] H Wang, T Kirubarajan, and Y Bar-Shalom. Precision large scale air traffic surveillance using imm/assignment estimators. *Aerospace and Electronic Systems, IEEE Transactions on*, 35(1):255–266, 1999.

[35] Gary M Weiss and Foster J Provost. Learning when training data are costly: the effect of class distribution on tree induction. *J. Artif. Intell. Res.(JAIR)*, 19:315–354, 2003.

[36] Murali Yeddanapudi, Yaakov Bar-Shalom, and Krishna R Pattipati. Imm estimation for multitarget-multisensor air traffic surveillance. *Proceedings of the IEEE*, 85(1):80–96, 1997.

Appendix A

Toolbox Class Diagram



Appendix B

Optimized Parameters

Table B.1: Table presenting best configuration for the tracker. For each radar specific configuration is matched. The IMM structure applied is marked by IMM structure label next to each radar's label. This labeling corresponds to the one introduced in 10.2.

radar1 (IMM1)	KF1 process noise PSD	1.03			
	KF2 process noise PSD	28.98			
	Meas. noise std. dev.	$\sigma_R = 0.002 * range + 72$			
	IMM transition matrix		KF1	KF2	
		KF1	0.56	0.44	
		KF2	0.61	0.39	
radar2 (IMM3)	KF1 process noise PSD	0.04			
	KF2 process noise PSD	12.23			
	KF3 turn rate variance	0.000 42			
	Meas. noise std. dev.	$\sigma_R = 0.002 * range + 24$			
	IMM transition matrix		KF1	KF2	KF3
		KF1	0.93	0.01	0.06
		KF2	0.01	0.37	0.61
		KF3	0.01	0.35	0.64
radar3 (IMM3)	KF1 process noise PSD	0.03			
	KF2 process noise PSD	14.08			
	KF3 turn rate variance	0.000 017			
	Meas. noise std. dev. $< 60Nm$	$\sigma_R = 0.0007 * range + 32$			
	Meas. noise std. dev. $>= 60Nm$	$\sigma_R = 0.0016 * range + 8$			
	IMM transition matrix		KF1	KF2	
		KF1	0.98	0.01	0.01
		KF2	0.52	0.01	0.47
		KF3	0.04	0.25	0.71