Signal Processing Systems
Mekelweg 4,
2628 CD Delft
The Netherlands
https://sps.ewi.tudelft.nl/

SPS-2023-5561086

M.Sc. Thesis

ADS-B Based Trajectory Prediction for Aerial Vehicles

Haobo Wang

Abstract

The evolution of aerial vehicle technology necessitates robust trajectory prediction models. These models are crucial for maintaining safe airspace and enabling autonomous operations. Automatic dependent surveillance–broadcast (ADS-B) is a surveillance system that enables aircraft to receive data from navigation satellites and periodically broadcasts it, enabling it to be tracked. Moreover, using ADS-B data for more general aerial vehicles has become a popular trend because it can provide real-time high-resolution aircraft state information and share this information with other vehicles in real-time for the aviation safety ecosystem.

In this project, we delve into ADS-B-based trajectory prediction for both aircraft and drone motion trajectories with the overarching goal of improving prediction accuracy. We initially implement several model-based Kalman filters—including interactive multiple models (IMM)—to assess the accuracy of aircraft trajectory predictions across different model structures. The results reveal that the IMM filter outperforms the single model predictions in terms of root mean square error (RMSE).

Furthermore, we implement the Gaussian process (GP) with a sliding window scheme to predict online drone trajectories. Recognizing the high computational complexity of the GP, we also introduce a low-rank approximation method, structured kernel interpolation (SKI) GP, aiming to conserve computational resources. Finally, we compare the prediction performances of the IMM filter, classical GP, and SKI GP on real drone trajectories. The results highlight that the classical GP method enhanced prediction accuracy, achieving an RMSE of less than 1.7m, which is 50% lower compared to the model-based IMM filter. Additionally, the SKI GP realizes a 25% reduction in computation time compared to the classical GP, despite a slight compromise in prediction accuracy.

TU Delft

Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology

ADS-B Based Trajectory Prediction for Aerial Vehicles

Haobo Wang

Delft University of Technology
Faculty of Electrical Engineering, Mathematics and Computer Science
Department of Microelectronics & Computer Engineering
Signal Processing Systems Group

Delft University of Technology
Department of
Microelectronics

The undersigned hereby certify that they have read and recommend to the Faculty of Electrical Engineering, Mathematics and Computer Science for acceptance a thesis entitled **"ADS-B Based Trajectory Prediction for Aerial Vehicles"** by **Haobo Wang** in partial fulfillment of the requirements for the degree of **Master of Science**.

Dated: 27 October 2023

Chairman: ______________________________
dr. R.T. Rajan

Advisor: ______________________________
dr. R.T. Rajan

Committee Members: ______________________________
prof.dr.ir. A.J. van der Veen

____
dr. J. Sun

____
ir. Z.Li

Abstract

The evolution of aerial vehicle technology necessitates robust trajectory prediction models. These models are crucial for maintaining safe airspace and enabling autonomous operations. Automatic dependent surveillance–broadcast (ADS-B) is a surveillance system that enables aircraft to receive data from navigation satellites and periodically broadcasts it, enabling it to be tracked. Moreover, using ADS-B data for more general aerial vehicles has become a popular trend because it can provide real-time high-resolution aircraft state information and share this information with other vehicles in real-time for the aviation safety ecosystem.

In this project, we delve into ADS-B-based trajectory prediction for both aircraft and drone motion trajectories with the overarching goal of improving prediction accuracy. We initially implement several model-based Kalman filters—including interactive multiple models (IMM)—to assess the accuracy of aircraft trajectory predictions across different model structures. The results reveal that the IMM filter outperforms the single model predictions in terms of root mean square error (RMSE).

Furthermore, we implement the Gaussian process (GP) with a sliding window scheme to predict online drone trajectories. Recognizing the high computational complexity of the GP, we also introduce a low-rank approximation method, structured kernel interpolation (SKI) GP, aiming to conserve computational resources. Finally, we compare the prediction performances of the IMM filter, classical GP, and SKI GP on real drone trajectories. The results highlight that the classical GP method enhanced prediction accuracy, achieving an RMSE of less than 1.7m, which is 50% lower compared to the model-based IMM filter. Additionally, the SKI GP realizes a 25% reduction in computation time compared to the classical GP, despite a slight compromise in prediction accuracy.

Acknowledgments

Reflecting on my two years at TU Delft, I've experienced a challenging and exciting journey through my master's studies. As this chapter draws to a close, I recognize the wealth of valuable knowledge I have acquired, guiding me toward my future endeavors. There are many people I want to thank; without their assistance, the completion of my studies would not have been possible.

Firstly, I extend my sincere gratitude to my supervisor, Dr. Raj Thilak Rajan, whose guidance and valuable suggestions were indispensable when challenges arose in my thesis. Additionally, I owe much appreciation to my daily supervisor, Ir. Zhonggang Li. The insights gained from our weekly meetings were invaluable, and his patient guidance has been important to my learning. Moreover, I would like to express my appreciation to Prof. Dr. Ir. A.J. van der Veen and Dr. J. Sun for serving as committee members for my thesis defense.

Secondly, my heartfelt thanks go to the friends who have always supported and stood by me. Special thanks to Chengyan Wang, Bingxiang Zhong, Kunlei Yu, and Xi Chen, who have supported me and provided valuable comments during the thesis period.

Finally, I want to convey my deepest gratitude to my parents for their unconditional love and support. Without their inspiration, I could not have become who I am today.

Haobo Wang
Delft, The Netherlands
27 October 2023

Contents

List of Figures

List of Tables

List of Notation

The following list shows some symbols and operators which will be used in the thesis.

Symbol	Definition
a	Scalar
$\mathbf{a}$	Column vector
$\mathbf{A}$	Matrix
x	Position(scalar)
$\dot{x}$	Velocity (scalar)
$\ddot{x}$	Acceleration(scalar)
A_{ij}	Entries of the i-th row and j-th column of matrix $\mathbf{A}$
$\mathbf{1}_N$	All-one vector of length N
$\mathbf{0}_N$	All-zero vector of length N
$\mathbf{I}_N$	Identity matrix of size N
$\mathcal{N}(0,1)$	Normal distribution with mean 0 and variance 1
$\mathbb{R}^N$	Set of real vectors of length N
$\mathbb{R}^{N\times M}$	Set of real matrices of size N-by-M
$\otimes$	Kronecker product
$\hat{\cdot}$	Estimate
$(\cdot)^T$	Transpose operator
$(\cdot)^{-1}$	Inverse operator
$\|\cdot\|_2$	Euclidean norm
$\mathrm{Var}[\cdot]$	Variance operator
$\mathrm{Cov}[\cdot]$	Covariance operator

Introduction 1

1.1 Motivation

As the skies grow more populated with a diverse range of aerial vehicles, ensuring their safe and efficient operation becomes paramount. Aerial vehicles can be broadly classified into manned and unmanned categories. Manned aerial vehicles include commercial airplanes, which are pivotal for global transportation, connecting distant parts of the world by ferrying passengers and cargo. On the other hand, unmanned aerial vehicles (UAVs) such as drones have found applications in numerous fields such as photography, agriculture, surveillance, and increasingly, cargo delivery. Central to this challenge is the need for accurate trajectory prediction. There are many model-based and data-driven methods have been applied to tackle trajectory prediction. In terms of the trajectory data acquiring, there are several traditional technologies such as primary or secondary surveillance radars [37], but they have several limitations such as high operational cost, limited coverage, and low accuracy of tracking [66, 65]. The automatic dependent surveillance-broadcast (ADS-B) emerges as a revolutionary aviation technology that provides a robust foundation for addressing this need [23]. Initially designed for tracking aircraft by broadcasting their satellite-determined positions, ADS-B has evolved into a key part of advanced aerial navigation systems. By utilizing the high-resolution data provided by ADS-B broadcasts, sophisticated trajectory prediction algorithms can forecast the future paths of aerial vehicles with high precision. Such predictions not only enhance air traffic management capabilities but also foster safer coexistence in shared airspaces, be it among commercial jets, private aircraft, or the burgeoning fleet of UAVs. This paper delves into the mechanics and applications of ADS-B-based trajectory prediction for modern-day aerial vehicles, underlining its significance in the evolving dynamics of global aviation.

ADS-B plays a vital role in air traffic control, collision avoidance systems, and weather tracking. Figure 1.1 is from the FlightRadar24 website, which uses ADS-B for real-time tracking and historical data presentation of aircraft, allowing both air traffic controllers and the general public to access information about flights. Most aircraft are equipped with ADS-B Out, broadcasting essential parameters such as position, velocity, and heading. A global network of over 40,000 ADS-B receivers collects this data, offering enhanced precision and reliability compared to conventional radar systems. This technology provides additional safety benefits, including collision prevention, air traffic optimization, and increased situational awareness for both pilots and air traffic management personnel [13].

1.2 Trajectory prediction on aircraft

The growth of the global economy has led to a surge in demand for air transport. With predictions indicating a 4.4% annual growth in worldwide air transportation over the next two decades, and China's air traffic volume expected to rise by 3.5 times [47], the aviation industry faces significant challenges. The current system of fixed airspace sectors and routes has several problems such as structural solidification, cascading failures, and limited capacity. This system hinders air traffic optimization and is not aligned with future trajectory and performance-based airspace operation trends. To address the growing air traffic demands, complex systems, and varied operational settings, many nations and entities are embarking on

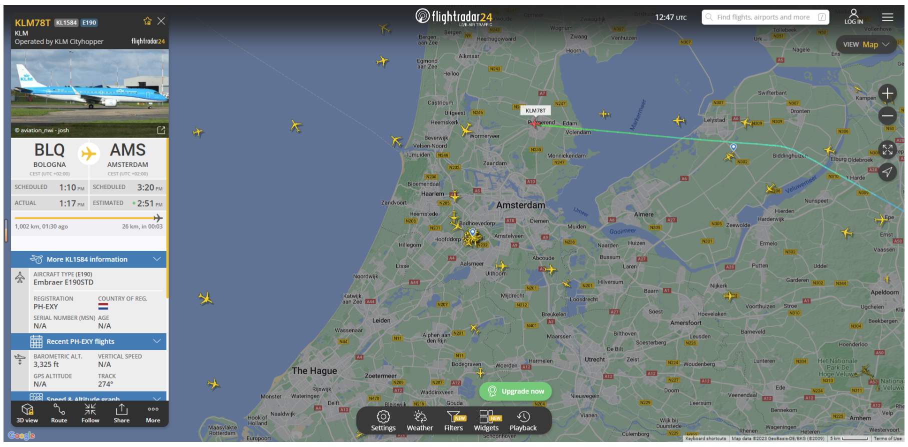


Figure 1.1: Aircraft tracking with ADS-B message on *Flightradar*

projects to modernize air traffic systems [44]. Whether the current sector-based approach or the anticipated trajectory-based method is considered, accurately predicting an aircraft's future path is vital. Precise trajectory predictions support decision-making tools like sequencing of arrivals and departures, conflict identification, understanding of airspace situations, and flight flow management, enhancing the predictability and safety of air traffic. For instance, the federal aviation administration (FAA) in the United States has heavily invested in "The Next-Generation of Aerial Transportation" initiative, aiming to enhance the safety and reliability of air travel over the recent decades [20]. Essential safety measures encompass air traffic control protocols, defining secure flight paths, and mechanisms for avoiding collisions. These measures outline the conditions under which planes can operate, and by doing so, diminish the risk of in-flight accidents. In this context, aircraft trajectory prediction emerges as a pivotal instrument to ensure safer skies.

One of the applications of aircraft prediction is air traffic management (ATM). It encompasses the entirety of services, infrastructure, and procedures designed to ensure the safe and efficient movement of aircraft throughout the skies and on the ground at airports. ATM oversees the safe, efficient, and orderly coordination of aircraft movements both in the sky and at airports. As Figure 1.2 shows the structure of ATM, it includes key components: air traffic services (ATS), air traffic flow management (ATFM), and airspace management (ASM) to maintain safety, facilitate smooth and predictable flight operations, reduce environmental impact, and optimize the use of airspace and ground facilities.

ATM is intricately linked with trajectory prediction, as accurately forecasting an aircraft's future path is fundamental to ensuring safety and efficiency in flight operations. Trajectory prediction provides air traffic controllers with foresight into an aircraft's position, enabling timely decisions to maintain safe distances between planes, optimize flight routes for fuel efficiency, reduce potential airspace congestion, and ensure punctual departures and arrivals. Essentially, trajectory prediction is central to the effective operation of the vast airborne system managed by ATM.

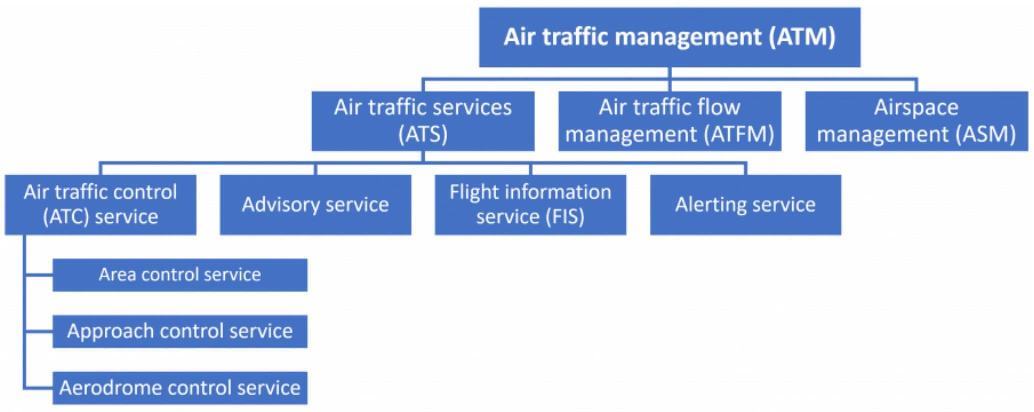

Air traffic management (ATM)
Air traffic services (ATS)
Air traffic flow management (ATFM)
Airspace management (ASM)
Air traffic control (ATC) service
Advisory service
Flight information service (FIS)
Alerting service
Area control service
Approach control service
Aerodrome control service

Figure 1.2: Air Traffic Management structure [51]

1.3 Trajectory prediction on UAV

Transitioning from traditional aviation, the rise of UAVs brings a new dimension to trajectory prediction on various applications [46, 24]. UAVs, commonly known as drones, operate in different altitudes and environments than commercial aircraft and often in closer proximity to obstacles and urban landscapes. Accurate trajectory prediction for UAVs is crucial to prevent collisions, especially in congested or uncharted areas. Additionally, with the potential integration of UAVs into broader airspace systems, their trajectories need to be seamlessly coordinated with other aircraft. The principles from traditional aircraft trajectory prediction in air traffic control(ATC) can be adapted and refined for UAVs, ensuring that as skies become more populated with diverse aerial vehicles, safety and efficiency remain paramount. The integration of UAVs into the U.S. National Airspace System (NAS) will introduce levels of complexity that will require holistic assessments of safety and risk mitigation [43].

However, it could also be challenging to acquire the position information for the UAVs. One of the competitive approaches is ADS-B technology, which offers significant benefits for the future ATC [17]. ADS-B technology allows aircraft to determine their position via satellite navigation and periodically broadcast it, enabling them to be tracked. By implementing ADS-B technology into UAVs, we can obtain real-time information such as position, velocity, and intent data. This information significantly enhances trajectory prediction accuracy and operational safety, particularly in airspace where multiple drones or other aircraft are present. Thus, the integration of ADS-B technology into UAVs can play a pivotal role in managing the rapidly growing drone traffic, paving the way for a safer and more efficient utilization of our shared airspace.

Trajectory prediction helps UAVs plan their route, reducing the risk of collisions and improving navigation. This is particularly crucial for collision avoidance systems in UAVs. Based on the "Sense, Detect, and Avoid (SDA)" concept from the study [34], as illustrated in Figure 1.3 with two UAV collision avoidance scenarios, trajectory prediction is vital in the detection phase. It helps predict the UAV's most possible position in the near future, working alongside onboard sensors to plan safe movements and make avoidance movements if necessary. Furthermore, it aids in ensuring smooth UAV operations in busy areas, avoiding unplanned interactions with other drones, buildings, or obstacles.

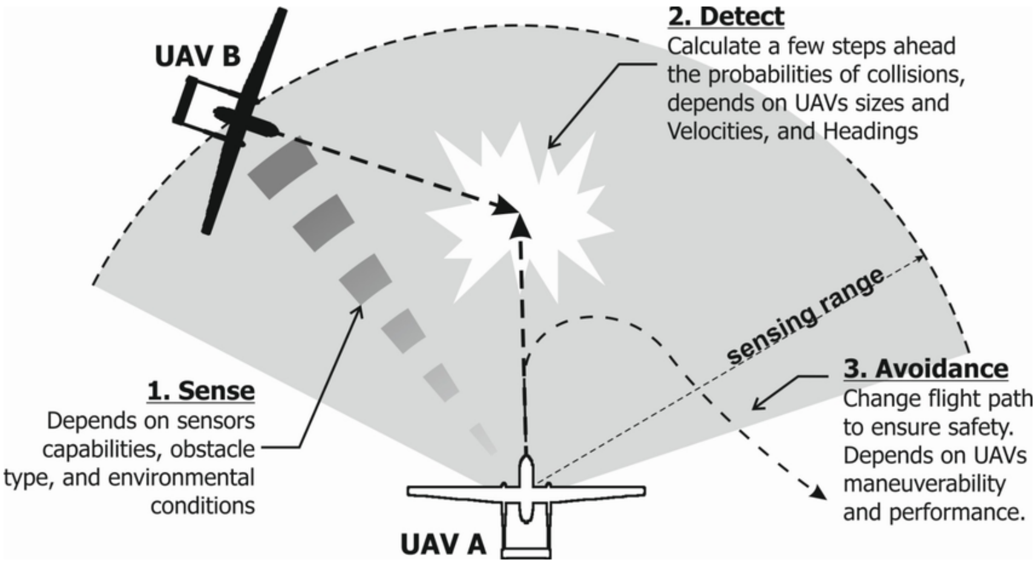

UAV B
2. Detect
Calculate a few steps ahead the probabilities of collisions, depends on UAVs sizes and Velocities, and Headings
sensing range
3. Avoidance
Change flight path to ensure safety. Depends on UAVs maneuverability and performance.
1. Sense
Depends on sensors capabilities, obstacle type, and environmental conditions
UAV A

Figure 1.3: Sense-Detect-Avoidance definition [22]

1.4 Objectives

In this thesis, we investigate the trajectory prediction methods for aerial vehicles both for aircraft and drone trajectories. We evaluate the trajectory prediction by exploiting ADS-B trajectory information. Based on the different motion characteristics, we take the different model-based and data-driven methods to predict the trajectories. The objectives can summarised as follows:

- Propose the interactive multiple models (IMM) filter to predict the ADS-B aircraft trajectories for higher prediction accuracy to compare with other single model Kalman filters.
- Propose the Gaussian process regression-based method with sliding window strategy to predict the ADS-B type drone trajectories for higher prediction accuracy compared with the IMM model-based method.
- Extend the GP with the low-rank approximation for low complexity and fast computation.
- Compare and analyze the trajectory prediction methods' performance on drone trajectory.

1.5 Outline

The thesis structure is structured as follows:

- In Chapter 2, some background knowledge is introduced including about ADS-B system, trajectory prediction methods, and dataset usage.
- In Chapter 3, we exploit the IMM filter along with other single model Kalman filters as compared to predict raw ADS-B message aircraft trajectory. We design the models for each model to describe different motions.
- In Chapter 4, we propose the Gaussian process regression with a sliding window to predict the ADS-B type drone trajectory. Furthermore, we implemented the low-rank approximation structured kernel interpolation on GP for the computational resources concern.

Finally, we compare the GP-based prediction method with the IMM filter on drone trajectory.

- In Chapter 5 we conclude the thesis, summarise the conclusion, and the following future work are discussed.

Literature Review 2

2.1 Automatic Dependent Surveillance-Broadcast (ADS–B)

Automatic dependent surveillance-broadcast (ADS-B) is a surveillance technology broadly used in air traffic control (ATC) scenarios. Using Mode-S transponders, aircraft periodically transmit ADS-B messages that contain critical navigational data such as ground and air position, velocity, call sign, operational status, and more. ADS-B allows ATC ground stations to continuously monitor aircraft even in areas beyond the reach of conventional radars. This is achieved through ground or space-based ADS-B receivers that can significantly expand coverage. Seen as a crucial element of the future air transport system, the use of ADS-B has been made compulsory by both EUCONTROL [12] in Europe and the FAA [55] in the U.S. since 2020.

Before ADS-B was introduced, ATC applications were heavily dependent on primary surveillance radar (PSR) and secondary surveillance radar (SSR). PSR provides the aircraft's radial distance and azimuth relative to the radar's location, while SSR gives information about the aircraft's altitude and identity. However, the inherent limitations of PSR and SSR technology have restricted further advancements in accuracy and coverage. As a result, ADS-B has been adopted to augment the situational awareness of ATC operators and pilots.
In terms of the trajectory estimation aspect, we will briefly introduce the ADS-B decoding message for position and velocity and its transmission rate for different types of messages.

2.1.1 Airbone Position Decoding

A standard airborne position message typically includes the aircraft's longitude, latitude, and altitude. Accurately decoding this airborne position is a crucial step for predicting the target's future trajectory. The challenge arises with the longitude and latitude components, which are encoded in the compact position reporting (CPR) format. There are two primary methods for decoding this: globally unambiguous position decoding and locally unambiguous position decoding.

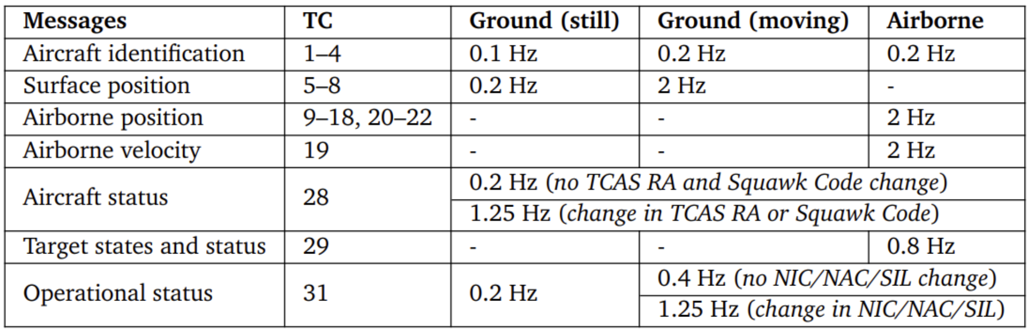
The globally unambiguous position decoding method requires a pair of "odd" and "even" messages to compute a globally correct position [54], and this process does not necessitate any prior information. Conversely, locally unambiguous position decoding requires a known reference position in close proximity to the decoded position, for instance, within a radius of 180 nautical miles. The main advantage of the latter approach is its ability to decode a position from each individual message.

2.1.2 Airborne Velocity Decoding

The airborne velocity message conveys velocity split into East-West, North-South, and vertical directions. This division provides a granular understanding of an aircraft's movement in three-dimensional space, which is crucial for precise trajectory prediction. Incorporating target velocity into these calculations offers additional information, which can be beneficial the prediction accuracy. Additionally, within the realm of civil aviation, it is common practice to compute a track angle, irrespective of altitude changes.

2.1.3 ADS-B messages transmission rate

The transmission rates of ADS-B messages differ depending on their type. The frequency of updates also changes based on the aircraft's status, such as whether it's airborne or grounded, and if it's stationary or in motion when on the ground. Figure 2.1 outlines the transmission rate of these distinct messages. It's noteworthy that the transmission rate for airborne position and velocity is set to 2Hz, which is regarded as the standard for updating the status of target motion. Notably, since this transmission is commonly from the aircraft's broadcast, for small UAVs this is considered to be a relatively low update rate for position and velocity.



Messages	TC	Ground (still)	Ground (moving)	Airborne
Aircraft identification	1–4	0.1 Hz	0.2 Hz	0.2 Hz
Surface position	5–8	0.2 Hz	2 Hz	-
Airborne position	9–18, 20–22	-	-	2 Hz
Airborne velocity	19	-	-	2 Hz
Aircraft status	28	0.2 Hz (*no TCAS RA and Squawk Code change*)		
		1.25 Hz (*change in TCAS RA or Squawk Code*)		
Target states and status	29	-	-	0.8 Hz
Operational status	31	0.2 Hz	0.4 Hz (*no NIC/NAC/SIL change*)	
			1.25 Hz (*change in NIC/NAC/SIL*)	

Figure 2.1: ADS-B message transmission rates (ADS-B version 2) [54]

2.2 Trajectory prediction methods

For the four-dimension trajectory prediction (longitude, latitude, altitude, time), use historical data to predict the future position information. According to the form of the prediction results, the trajectory prediction can also be divided into two other categories [27] :

- Deterministic prediction: This is considered as the nominal method and generally directly outputs the predicted four-dimensional trajectory information [41]. A deterministic model would provide a single, specific output for a given input The nominal method cannot perfectly describe the uncertainty of the aircraft's future behavior.

- Probabilistic prediction: On the other hand, probabilistic prediction models understand that the future is uncertain and represent this uncertainty as a probability distribution over possible outcomes. Probabilistic models are generally more robust to the inherent unpredictability of real-world systems and can provide a measure of the uncertainty or confidence in their predictions (Bayesian network, Gaussian Process [16])

Trajectory prediction methods for aerial vehicles can be also categorized into two types: model-based methods and data-driven methods. Model-based methods leverage mathematical and physical principles to simulate the behavior of aerial vehicles, taking into account factors like wind speed, vehicle dynamics, control input, and others. These models offer a high degree of interpretability, allowing for efficient and accurate prediction under known conditions. On the other hand, data-driven methods rely on historical flight data and machine learning techniques to forecast the trajectory. They're particularly useful in handling complex scenarios where traditional models may fail, as they can capture nonlinear relationships and consider a wide range of variables. These techniques have grown in popularity with the rise of AI and can be more flexible, but their accuracy can be significantly affected by the quality and volume of the data. In the Figure 2.2 is a brief overview to illustrate some common methods for model-based methods and data-driven methods.

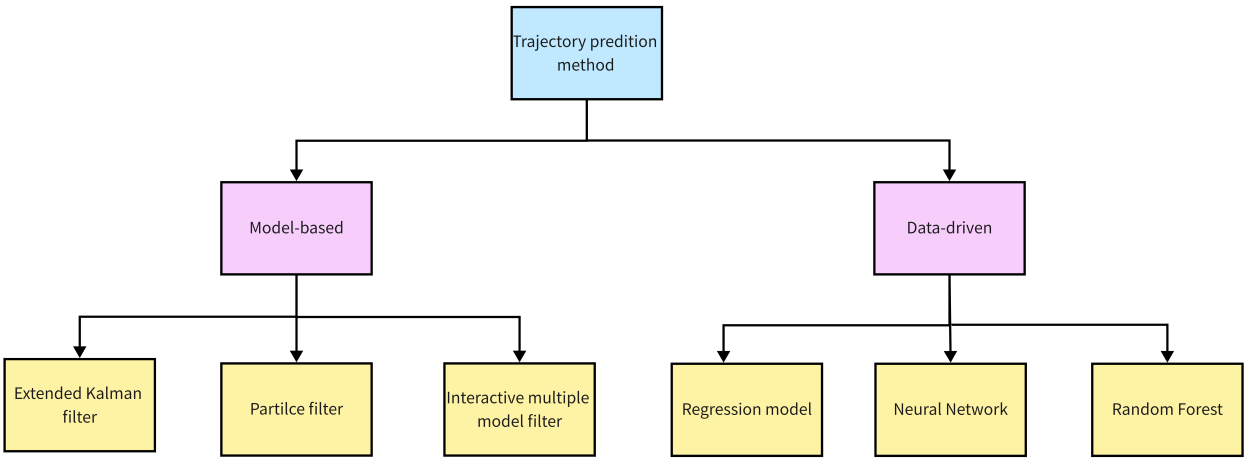


Figure 2.2: An overview of tracking methods for trajectory prediction

2.2.1 Model-based methods

The most commonly used model-based methods are related to the Kalman filter. These model-based methods need to construct the state transition matrix in the state equation through the equation of motion and study the relationship between position, historical position, speed, acceleration, angle, and other states at each time point in the future. According to different assumptions about whether the aircraft has a single flight mode or multiple modes in the prediction process, this type of method is divided into single-model methods and multi-model methods.

For the single model, the Kalman filter (KF) algorithm, particle filter algorithm, and hidden Markov model(HMM) and their improved algorithm are all applied in the trajectory prediction problem. KF [25] offers an iterative, optimal solution for linear target dynamics and measurement models. For nonlinear models, several filters have been introduced, including the extended KF [53], and the unscented KF [56]. In terms of multiple model models, it includes multi-model KF or the IMM filter [6]. Notably, the IMM filter is recognized as one of the most cost-efficient nonlinear filters [35]. Each of these approaches is reliant on an underlying target dynamics model. They select a model that aligns most closely with the average target trajectory and use model noise to account for any variations. These filters can merge data from various sensors, resulting in an enhanced and more dependable estimation of the target state. However, when it comes to handling highly maneuverable targets such as drones, these methods might encounter a larger noise variance.

Specifically, the EKF has been utilized to combine data from optical flow, IMU, and GPS for accurate online predictions for UAVs [40]. Additionally, dynamic state estimation based on the EKF has been proposed for trajectory planning for UAV clusters [60]. The Kalman filter has been employed using lidar data to detect small UAVs [19]. Moreover, the KF, EKF, and Particle Filter have been used for the ADS-B aircraft trajectory prediction as the same aircraft data set in our project [64].

Table 2.1: Model-based trajectory prediction method

Single-model method	KF:[25, 53, 56],EKF:[40],Particle filter:[32],HMM:[31]
Multi-model method	Multi-model KF:[48] Interactive multiple model:[6, 3]

2.2.2 Data-driven methods

Contrasting the model-based filters, various machine learning techniques, like regression models, neural networks, and random forest techniques provide alternative solutions. Table 2.2 provides a summary of some examples using different data-driven trajectory prediction methods. These techniques don't depend on kinetic equations but seek to learn patterns from historical positional and kinetic data to predict a target future location. These techniques have the advantage of learning complex data patterns without requiring explicit models. In terms of the regression model, a Gaussian Process-based trajectory prediction framework has been used with a change-point detection technique to note sudden changes [62]. Moreover, GP-based target tracking and smoother are developed [1] with online learning and parameter learning. For the neural network, the most commonly used methods include long short-term memory neural networks (LSTMs), convolutional neural networks (CNNs), depth neural networks (DNNs), and combinations for different networks. RNN has been proposed to predict UAV trajectories in 5G networks [61]. The LSTM network as one type of neural network method is well-suited for capturing long-term dependencies in sequences and has been successfully utilized in various time series prediction tasks. As a result, it has become the most commonly employed neural network for trajectory prediction. It has been used for flight trajectory prediction within ADS-B information [49].

Traditionally, trajectory prediction is modeled using deterministic prediction models which do not explicitly capture the sources of uncertainty that affect the prediction accuracy [7]. Some deep-learning-based prediction methods (CNN, LSTM) may predict the positions of UAVs at future moments accurately. However, uncertainty quantification is a fundamental problem. Those methods offer single trajectory predictions without the capability to express the uncertainties associated with those predictions. Modeling uncertainty in predictive analytics involves quantifying the uncertainty associated with input parameters in a predictive model and how it impacts the uncertainty in the predictions for the target variable. To achieve this, uncertainty quantification methods, such as Monte Carlo simulations, are utilized to express performance parameters as probability density functions (PDF). These PDFs represent the likelihood of different outcomes for the target variable based on varying input conditions. By propagating input uncertainties through deterministic models, the joint effect of stochastic factors on the predictions of the target variable is identified [45]. Moreover, Gaussian process regression is another good way to quantify the uncertainty in the trajectory prediction aspect. As a non-parametric Bayesian probabilistic prediction method, it can explicitly provide both mean prediction and associated measures of uncertainty due to the underlying probabilistic nature of the model [16]. Besides, the LSTM is more applicable for large data sets and may overfit in small data sets. On the contrary, GP is typically very effective as it can capture complex relationships with a relatively small number of points Also it becomes computationally expensive when dealing with a large amount of data due to its model complexity.

Table 2.2: Data-driven trajectory prediction method

Regression Model:	Linear Regression:[18, 26] Gaussian Process Regression:[16, 62, 1]
Neural Network:	LSTM:[63, 49] RNN:[61] CNN+LSTM:[33] Bayesian neural network:[38]
Other methods:	Random forest with clustering:[29] Neural Networks with clustering:[57]

2.3 Data set

To evaluate the trajectory prediction method on aerial vehicles, we access the different ADS-B type trajectory data sets, which could be categorized into aircraft trajectory and drone trajectory. The aircraft trajectory is transmitted as the ADS-B message with around 500 data samples. In terms of the drone trajectory, it includes 23 seconds of motion with a 400Hz sampling rate. To simulate the ADS-B transmission rate, we down sample it to 2Hz which includes 53 samples as a relatively small data set.

2.3.1 Aircraft data sets

To test the trajectory prediction performance on aircraft, We work with two realistic ADS-B data sets. The first data set is called *TrajAir* [39] contributed by the Air Lab from the robotics institute at Carnegie Mellon University. The data set is collected at the Pittsburgh-Butler Regional Airport. The other data set is collected by the faculty of aerospace engineering at TU Delft [21], which we call the *TUD* data set.

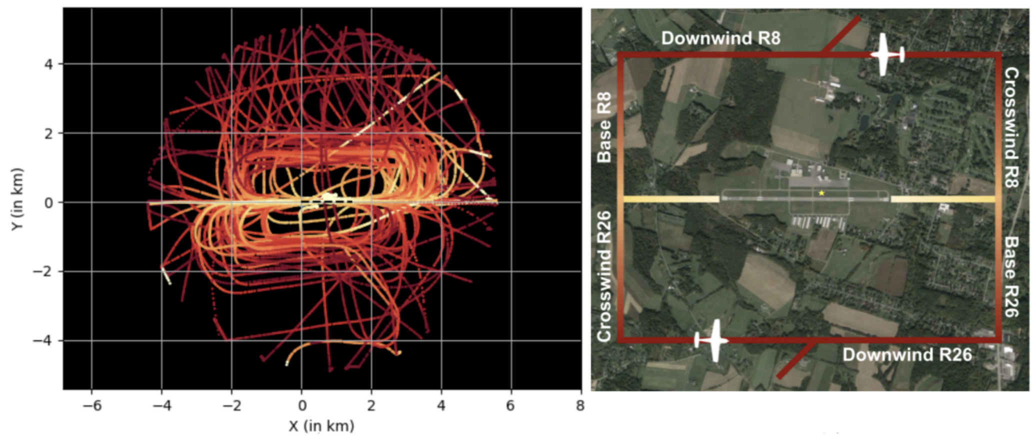
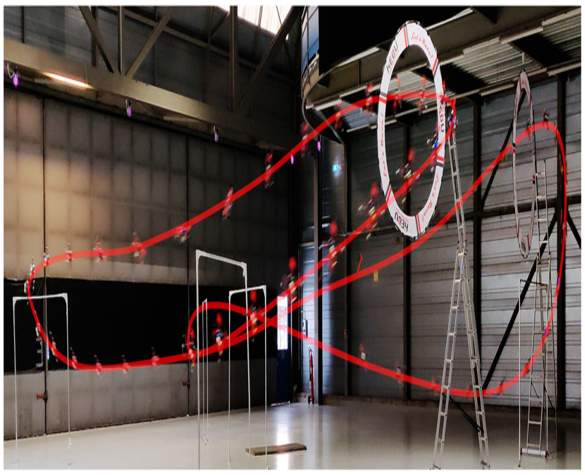


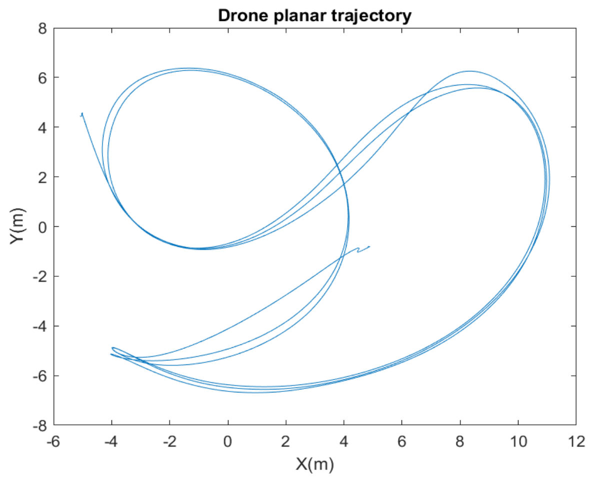
Figure 2.3: *TrajAir* data set: The left image displays a section of analyzed aircraft flight paths, while the right image depicts the standard traffic pattern and the labeling system used for aircraft runways. [39]

The data set includes detailed information about the status of an aircraft, such as timestamps, geographic coordinates, velocity measurements, track angles, altitudes, and vertical rates at regular intervals. Notably, the data does not capture the alternate transmission behavior of ADS-B. Figure 2.3 visualizes the trajectories of a group of aircraft during landing or takeoff, revealing distinct traffic patterns known as "lobes" around the airport. The right portion of the figure specifically highlights the "Left Traffic" patterns characterized by rectangular shapes and left turns relative to the direction of landing or takeoff. Additionally, the trajectories are color-coded to indicate lower altitudes with lighter shades [39].

2.3.2 Drone data set

In order to verify the effectiveness of the proposed method on the drone data set, a real-world drone trajectory has been used [14]. The drone trajectory is from a trajectory optimization research sponsored by the Swiss National Science Foundation (SNSF). The drone is flown in an in-door motion capture system with 2.5 laps of flying on an intended trajectory. The experiment setup is around $20m \times 20m$ to cover the motion of the drone. Within the on-board and off-board sensor equipment on the drone, the information on position, linear velocity and acceleration, rotation velocity, and acceleration has been collected along $3D$ dimensions. In this section, we only focus on the position information P_x, P_y, P_z as needed for trajectory planning. Figure 2.4a is the real setup of the indoor motion system for the quadcopter trajectory and Figure 2.4b is the planar trajectory in the simulation. The sampling rate for the position update is 400Hz, in order to simulate the ADS-B-based transmission for the drone trajectory prediction. We down sample the data set to 2Hz which corresponds to the standard protocol for the position update rate.


(a) 3D trajectory



(b) planar trajectory

Figure 2.4: Trajectories of drone

2.3.3 Prediction Evaluation

In our cases, the exact ground truth is not available for the above open-source trajectory data set. Therefore we will take the original trajectory data set as a reference(ground truth) to compare with the predicted trajectory. It represents a measure of discrepancy to evaluate how well the predicted model is performing relative to actual data.

2.4 Summary

In this chapter, we introduce the preliminary of ADS-B based trajectory prediction related to this project. The main content are as follows:

- Introduce the ADS-B message decoding protocol and transmission rate.
- Categorize the trajectory prediction method for aerial vehicles. Compare the specific methods of model-based and data-driven methods.
- Illustrate the trajectory sources including aircraft and drone data sets for evaluation.

Model-based Filters on Aircraft Motion Trajectory 3

Accurately tracking the position and movement of dynamic objects like aircraft, vehicles, and drones is an issue that demands attention. In the context of aircraft motion, Commercial airplanes usually maintain designated speed, heading, and altitude during en route flying, we assume it as a linear Gaussian state space mode. Hence, the traditional model-based filters like the Kalman filter [25] could be the optimal filter to tackle the problems. However, this can result in errors since the aircraft motion always include multiple dynamics model, such as in the case of an aircraft that switches from flying in a straight line to performing a maneuver turn. In such cases, the Kalman filter may not be able to accurately estimate the object's trajectory, leading to potentially dangerous situations.

The interacting multiple model (IMM) filter [35] then could be an improved filter to address this problem. In essence, the IMM filter is a combination of several Kalman filters, each representing a different mode of the system. It enables the use of multiple models simultaneously and smoothly switches between them based on the current behavior of the object. For example, an aircraft that is flying straight may be best tracked with a constant velocity or constant acceleration model, while an aircraft that is performing a maneuver may be better tracked with a constant turn model. The IMM filter switches between these models as needed, resulting in more precise trajectory estimation and better tracking performance, especially when the object being tracked experiences relatively large changes in speed or direction. It has been widely used in tracking applications such as radar [5] or sonar tracking [2] as well as robotics or navigation involved in dealing with uncertain or changing environments.

3.1 Problem Formulation

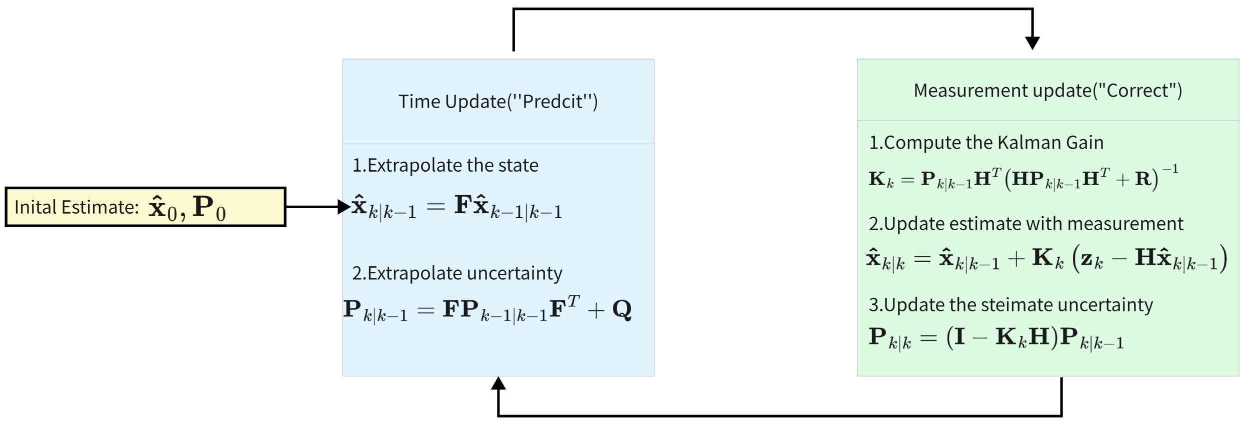
Given a sequence of ADS-B measurements of an aircraft's position over time, the goal is to predict the future trajectory of the aircraft with high accuracy, while accounting for uncertainties and possible changes in the aircraft's motion patterns. The prediction also takes into account the flight dynamics and constraints of the aircraft, such as its maximum speed, altitude, and turn rate.

The ADS-B measurements provide information about the aircraft's position, velocity, altitude, and heading, as well as other metadata such as the flight identification number and the aircraft type. However, the measurements may be noisy or incomplete, and there may be gaps in the data due to loss of signal or other factors.

To address these challenges, an IMM filter can be used, which combines multiple models to represent different possible motion patterns of the aircraft and dynamically switches between them based on the available measurements and the estimated probabilities of each model. Each model is characterized by a set of parameters that describe the aircraft's motion, such as speed, altitude, heading, and turn rate, and the filter estimates these parameters over time to predict the future trajectory. To simplify the problem, We took three motion models on IMM filter and the state space model of each is explained in the next section.

3.2 Kalman filter

Before introducing the IMM filter, the task of trajectory prediction can be considered as a state estimation problem. The most commonly used model-based method for it is the Kalman filter. It requires the algorithm to retrieve signals of interest from noisy data and construct a reasonable (regarding target dynamics) trajectory from available data. Here is the diagram of the Kalman filter in Figure 3.1, it shows the Kalman filter is a recursive algorithm including the predict and update steps to estimate the state and covariance.



Inital Estimate: $\hat{\mathbf{x}}_0, \mathbf{P}_0$
Time Update("Predcit")
1.Extrapolate the state
$\hat{\mathbf{x}}_{k|k-1} = \mathbf{F}\hat{\mathbf{x}}_{k-1|k-1}$
2.Extrapolate uncertainty
$\mathbf{P}_{k|k-1} = \mathbf{F}\mathbf{P}_{k-1|k-1}\mathbf{F}^T + \mathbf{Q}$
Measurement update("Correct")
1.Compute the Kalman Gain
$\mathbf{K}_k = \mathbf{P}_{k|k-1}\mathbf{H}^T(\mathbf{H}\mathbf{P}_{k|k-1}\mathbf{H}^T + \mathbf{R})^{-1}$
2.Update estimate with measurement
$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k(\mathbf{z}_k - \mathbf{H}\hat{\mathbf{x}}_{k|k-1})$
3.Update the steimate uncertainty
$\mathbf{P}_{k|k} = (\mathbf{I} - \mathbf{K}_k\mathbf{H})\mathbf{P}_{k|k-1}$

Figure 3.1: kalman filter diagram

Here we use the $\hat{\mathbf{x}}_{k|k-1}$ to denote the predicted state at the $k-$th step while $\hat{\mathbf{x}}_{k|k}$ denotes the corrected state estimate at the $k-$th step. Similar notations are used also for uncertainty for the estimated state $\mathbf{P}_{k|k-1}$ and $\mathbf{P}_{k|k}$. The $\mathbf{F}$ and $\mathbf{H}$ are denoted the state transition matrix and observation matrix. Also, the $\mathbf{Q}$ and $\mathbf{R}$ are denoted as process and noise measurement noise covariance. The matrix $\mathbf{K}_k$ is the Kalman gain computed at the $k-$th step.

Within the initial estimate, for the prediction step, it predicts the state estimate and covariance for the next step as the input for the update step. The equations for this step are as follows:

$$\hat{\mathbf{x}}_{k|k-1} = \mathbf{F}\hat{\mathbf{x}}_{k-1|k-1} \tag{3.1}$$

$$\mathbf{P}_{k|k-1} = \mathbf{F}\mathbf{P}_{k-1|k-1}\mathbf{F}^T + \mathbf{Q} \tag{3.2}$$

and for the update step, the filter incorporates the latest measurement to correct the predicted state estimate. This step comprises the following equations:

$$\mathbf{K}_k = \mathbf{P}_{k|k-1}\mathbf{H}^T\left(\mathbf{H}\mathbf{P}_{k|k-1}\mathbf{H}^T + \mathbf{R}\right)^{-1} \tag{3.3}$$

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k\left(\mathbf{z}_k - \mathbf{H}\hat{\mathbf{x}}_{k|k-1}\right) \tag{3.4}$$

$$\mathbf{P}_{k|k} = (\mathbf{I} - \mathbf{K}_k\mathbf{H})\,\mathbf{P}_{k|k-1} \tag{3.5}$$

3.3 State Space model

The State Space Model is a crucial part of applying the Kalman filter on trajectory prediction since is to provides a mathematical representation of the aircraft's motion that can be used to make predictions and estimate uncertainties. By modeling the aircraft's motion as a set of state variables that evolve over time, the filter can use the available measurements to estimate the values of these variables and make predictions about the future trajectory of the aircraft. Based on the aircraft dynamics and trajectory characteristics, we consider and build three models of movement which will be further used as the motion model in the IMM filter:

- Constant Velocity model (CV)
- Constant Acceleration model (CA)
- Constant Turn model (CT)

The general discrete-time state representation of a linear system is given in the following equation:

$$\mathbf{x}_{k+1} = \mathbf{F}\mathbf{x}_k + \mathbf{w}_k \tag{3.6}$$

where $\mathbf{x}_k$ is the state estimate, $\mathbf{F}$ is a state transition matrix from time k to $k+1$, and $\mathbf{w}_k$ is system process noise assumed to be Gaussian-distributed zero mean and white.
The observation equation follows as

$$\mathbf{z}_k = \mathbf{H}\mathbf{x}_k + \mathbf{v}_k \tag{3.7}$$

where $\mathbf{z}_k$ is the measurement vector, $\mathbf{H}$ is the matrix relating the state to observation quantities, and $\mathbf{v}_k$ is observation noise assumed to be Gaussian-distributed zero mean and having zero cross-correlation with the process noise $\mathbf{w}_k$. The observation matrix is used to map the state vector to the measurement vector.

Based on the assumption of process noise and observation noise it can be presented as follows where the $\mathbf{Q}$ and $\mathbf{R}$ are process and noise measurement noise covariance as KF :

$$\begin{aligned} p(\mathbf{w}) &\sim \mathcal{N}(0, \mathbf{Q}) \\ p(\mathbf{v}) &\sim \mathcal{N}(0, \mathbf{R}) \end{aligned} \tag{3.8}$$

Each model could has a different state estimate $\mathbf{x}$ and state transition matrix $\mathbf{F}$ which would be elaborated in the following context [30].

3.3.1 Constant Velocity model

The state vector for the constant velocity model is defined as $\mathbf{x} = \begin{bmatrix} x & \dot{x} & \ddot{x} & y & \dot{y} & \ddot{y} & \omega \end{bmatrix}^T$ $x, \dot{x}, \ddot{x}$ include the position velocity and acceleration for the x dimension, and $y, \dot{y}, \ddot{y}$ include the position velocity and acceleration for the y dimension. The ω is the turn rate in the xy dimension. To align with other models, for the constant acceleration and constant turn model, the state vectors are the same.

The state transition matrix for the CV model is defined for a linear equation as:

$$\mathbf{F}_{cv} = \begin{bmatrix} 1 & \Delta t & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & \Delta t & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \tag{3.9}$$

The observation vector $\mathbf{z}$ for each model is described the position and velocity for $2D$ information:

$$\mathbf{z} = \begin{bmatrix} x & \dot{x} & y & \dot{y} \end{bmatrix}^\top \tag{3.10}$$

Since the measurements only include the position and velocity for $2D$, whereas the observation matrix for the three models is the same :

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{3.11}$$

3.3.2 Constant Acceleration model

The constant acceleration kalman filter extends the state vector to include acceleration,the transition matrix $\mathbf{F}_{ca}$ as follows

$$\mathbf{F}_{ca} = \begin{bmatrix} 1 & \Delta t & \frac{1}{2}\Delta t^2 & 0 & 0 & 0 & 0 \\ 0 & 1 & \Delta t & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & \Delta t & \frac{1}{2}\Delta t^2 & 0 \\ 0 & 0 & 0 & 0 & 1 & \Delta t & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.12}$$

3.3.3 Constant Turn model

The state transition matrix for the constant turn model is to perform a constant speed turn maneuver along the trajectory by state vectors velocity and acceleration. Besides, the CT model is considered with a known constant turning rate ω in this case. The transition matrix $\mathbf{F}_{ct}$ as follows

$$\mathbf{F}_{ct} = \begin{bmatrix} 1 & \frac{\sin\omega\Delta t}{\omega} & 0 & 0 & -\frac{1-\cos\omega\Delta t}{\omega} & 0 & 0 \\ 0 & \cos\omega\Delta t & 0 & 0 & -\sin\omega\Delta t & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & \frac{1-\cos\omega\Delta t}{\omega} & 0 & 1 & \frac{\sin\omega\Delta t}{\omega} & 0 & 0 \\ 0 & \sin\omega\Delta t & 0 & 0 & \cos\omega\Delta t & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.13}$$

3.3.4 Observation noise covariance

Since we assume the observation noise as zero-mean Gaussian with covariance $\mathbf{R}$. In terms of the observation noise covariance $\mathbf{R}$, it reflects the noise in the measurements obtained from the system. Var_x, Var_y and Var_{vx}, Var_{vy} denotes the variance for the $2D$ dimension position and velocity respectively. The correlation between Cov_{xy} and Cov_{Vxy} has been taken into account.

$$\mathbf{R} = \begin{bmatrix} \mathrm{Var}_x & \mathrm{Cov}_{xy} & 0 & 0 \\ \mathrm{Cov}_{xy} & \mathrm{Var}_y & 0 & 0 \\ 0 & 0 & \mathrm{Var}_{vx} & \mathrm{Cov}_{Vxy} \\ 0 & 0 & \mathrm{Cov}_{Vxy} & \mathrm{Var}_{vy} \end{bmatrix} \tag{3.14}$$

3.3.5 Process noise covariance

Similar to the observation noise, with the process noise covariance $\mathbf{Q}$. It includes the state vector of position velocity and acceleration for x and y dimensions. It represents the expected uncertainty or noise in the prediction model, it affects how much the filter will rely on the measurements over time. Tuning these matrices properly is vital for the performance of the Kalman filter in estimating the states accurately and reliably. Therefore an estimated matrix Q is essential to obtain reliable results. We assume the case in which variance in acceleration σ_a^2 causes a variance in velocity and position, and the state variables are dependent based on [4].

Given the position update equation in discrete time:

$$x_{k+1} = x_k + v_k \Delta t + \frac{1}{2} a_k \Delta t^2 \tag{3.15}$$

The variance of $Var(x)$, $Var(v)$ the covariance $Cov(v, x)$ is derivated based

$$Var(x) = \sigma_x^2 = E\left(x^2\right) - E(x)^2 = E\left(\left(\frac{1}{2} a \Delta t^2\right)^2\right) - \left(\frac{1}{2} E(a) \Delta t^2\right)^2 = \frac{\Delta t^4}{4}\left(E\left(a^2\right) - E(a)^2\right) = \frac{\Delta t^4}{4}\sigma_{ax}^2 \tag{3.16}$$

$$Var(v) = \sigma_v^2 = E\left(v^2\right) - E\left(v\right)^2 = E\left((a\Delta t)^2\right) - (E(a)\Delta t)^2 = \Delta t^2\left(E\left(a^2\right) - E(a)^2\right) = \Delta t^2 \sigma_{ax}^2 \tag{3.17}$$

$$Cov(v, x) = E(xv) - E(x)E(v) = E\left(\frac{1}{2} a^2 \Delta t^3\right) - \left(\frac{1}{2} E(a)^2 \Delta t^3\right) = \frac{\Delta t^3}{2}\left(E\left(a^2\right) - E(a)^2\right) = \frac{\Delta t^3}{2}\sigma_{ax}^2 \tag{3.18}$$

The Y dimension has a similar assumption. Thus, the process noise covariance matrix **Q** is structured within the information of two-dimensional acceleration variance σ_{ax}^2 and σ_{ay}^2 and sampling interval Δt

$$\mathbf{Q} = \begin{bmatrix} \frac{\Delta t^4}{4} \cdot \sigma_{ax}^2 & 0 & \frac{\Delta t^3}{2} \cdot \sigma_{ax}^2 & 0 & \frac{\Delta t^2}{2} \cdot \sigma_{ax}^2 & 0 \\ 0 & \frac{\Delta t^4}{4} \cdot \sigma_{ay}^2 & 0 & \frac{\Delta t^3}{2} \cdot \sigma_{ay}^2 & 0 & \frac{\Delta t^2}{2} \cdot \sigma_{ay}^2 \\ \frac{\Delta t^3}{2} \cdot \sigma_{ax}^2 & 0 & \Delta t^2 \cdot \sigma_{ax}^2 & 0 & \Delta t \cdot \sigma_{ax}^2 & 0 \\ 0 & \frac{\Delta t^3}{2} \cdot \sigma_{ay}^2 & 0 & \Delta t^2 \cdot \sigma_{ay}^2 & 0 & \Delta t \cdot \sigma_{ay}^2 \\ \frac{\Delta t^2}{2} \cdot \sigma_{ax}^2 & 0 & \Delta t \cdot \sigma_{ax}^2 & 0 & \sigma_{ax}^2 & 0 \\ 0 & \frac{\Delta t^2}{2} \cdot \sigma_{ay}^2 & 0 & \Delta t \cdot \sigma_{ay}^2 & 0 & \sigma_{ay}^2 \end{bmatrix} \tag{3.19}$$

With notice, it's difficult to find the optimal values for the process noise covariance matrix **Q**, but defining a reasonable value for the **Q** is crucial to balance the fusion between predicted states and measurement values.

3.4 IMM Filter

The IMM filter is an advanced filter as a combination of multiple Kalman filters with different state space models as we defined above. The algorithm operates as follows it includes several important steps:

The Algorithm 1 shows the complete step of the IMM filter. It includes the mixing step for estimating the prior state and covariance, the filtering step for updating the posterior state and covariance, the model probability update step for updating the likelihood function for each model, and the estimate combination step for the final estimation. The details explanation is as follows.

Algorithm 1 Interacting Multiple Model (IMM) Algorithm

```
1: Initialization
2: Initialize the model set $\{\mathbf{F}^{(m)}, \mathbf{H}^{(m)}, \mathbf{Q}^{(m)}, \mathbf{R}^{(m)}\}_{m=1}^{M}$ for $M$ models
3: Initialize model probabilities $\mu_0^{(m)}$
4: Initialize state estimates $\hat{\mathbf{x}}_0^{(m)}$ and covariances $\mathbf{P}_0^{(m)}$ for each model
5: for $k = 1$ to $N$ do
6:     Mixing
7:     for $m = 1$ to $M$ do
8:         $\mu_{k|k-1}^{(m)} = \sum_{j=1}^{M} p_{jm} \mu_{k-1}^{(j)}$
9:         $\mu_{k-1}^{(j|m)} = \frac{p_{jm}\mu_{k-1}^{(j)}}{\mu_{k|k-1}^{(m)}}$
10:        $\hat{\mathbf{x}}_{k-1|k-1}^{(m)} = \sum_{j=1}^{M} \mu_{k-1}^{(j|m)} \hat{\mathbf{x}}_{k-1|k-1}^{(j)}$
11:        $\mathbf{P}_{k-1|k-1}^{(m)} = \sum_{j=1}^{M} \mu_{k-1}^{(j|m)} \left(\mathbf{P}_{k-1|k-1}^{(j)} + (\hat{\mathbf{x}}_{k-1|k-1}^{(j)} - \hat{\mathbf{x}}_{k-1|k-1}^{(m)})(\hat{\mathbf{x}}_{k-1|k-1}^{(j)} - \hat{\mathbf{x}}_{k-1|k-1}^{(m)})^T\right)$
12:    end for
13:    Filtering
14:    for $m = 1$ to $M$ do
15:        Prediction: $\hat{\mathbf{x}}_{k|k-1}^{(m)} = \mathbf{F}^{(m)} \hat{\mathbf{x}}_{k-1|k-1}^{(m)}$
16:                    $\mathbf{P}_{k|k-1}^{(m)} = \mathbf{F}^{(m)} \mathbf{P}_{k-1|k-1}^{(m)} (\mathbf{F}^{(m)})^T + \mathbf{Q}^{(m)}$
17:        Update: $\mathbf{K}_k^{(m)} = \mathbf{P}_{k|k-1}^{(m)} (\mathbf{H}^{(m)})^T ((\mathbf{H}^{(m)} \mathbf{P}_{k|k-1}^{(m)} (\mathbf{H}^{(m)})^T + \mathbf{R}^{(m)})^{-1}$
18:                $\hat{\mathbf{x}}_{k|k}^{(m)} = \hat{\mathbf{x}}_{k|k-1}^{(m)} + \mathbf{K}^{(m)} (\mathbf{z}_k - \mathbf{H}^{(m)} \hat{\mathbf{x}}_{k|k-1}^{(m)})$
19:                $\mathbf{P}_{k|k}^{(m)} = (\mathbf{I} - \mathbf{K}^{(m)} \mathbf{H}^{(m)}) \mathbf{P}_{k|k-1}^{(m)}$
20:    end for
21:    Model Probability Update
22:    for $m = 1$ to $M$ do
23:        Likelihood $\mathcal{L}_k^{(m)} = \mathcal{N}(\mathbf{z}_k - \hat{\mathbf{z}}_{k|k-1}^{(m)}; 0, S_k^{(m)})$        ▷ Gaussian likelihood function
24:        $\mu_k^{(m)} = \frac{\mathcal{L}_k^{(m)} \mu_{k|k-1}^{(m)}}{\sum_{j=1}^{M} \mathcal{L}_k^{(j)} \mu_{k|k-1}^{(j)}}$
25:    end for
26:    Estimate Combination
27:    $\hat{\mathbf{x}}_{k|k} = \sum_{m=1}^{M} \mu_k^{(m)} \hat{\mathbf{x}}_{k|k}^{(m)}$
28:    $\mathbf{P}_{k|k} = \sum_{m=1}^{M} \mu_k^{(m)} \left(\mathbf{P}_k^{(m)} + (\hat{\mathbf{x}}_{k|k}^{(m)} - \hat{\mathbf{x}}_{k|k})(\hat{\mathbf{x}}_{k|k}^{(m)} - \hat{\mathbf{x}}_{k|k})^T\right)$
29: end for
```

To begin with, the mixing phase computes mixed conditions for each motion filter based on mixing probabilities $\mu_{k-1}^{(j|m)}$ using Baye's theorem. The $m, j \in M$ where M denotes the total models. It illustrates the mixing probability of the model j corrected at time $k-1$, given the model m will be valid at time k. This mixing allows information to be shared among the models.

$$\hat{\mathbf{x}}_{k-1|k-1}^{(m)} = \sum_{j=1}^{M} \mu_{k-1}^{(j|m)} \hat{\mathbf{x}}_{k-1|k-1}^{(j)} \tag{3.20}$$

$$\mathbf{P}_{k-1|k-1}^{(m)} = \sum_{j=1}^{M} \mu_{k-1}^{(j|m)} \left(\mathbf{P}_{k-1|k-1}^{(j)} + (\hat{\mathbf{x}}_{k-1|k-1}^{(j)} - \hat{\mathbf{x}}_{k-1|k-1}^{(m)})(\hat{\mathbf{x}}_{k-1|k-1}^{(j)} - \hat{\mathbf{x}}_{k-1|k-1}^{(m)})^T\right) \tag{3.21}$$

Afterward, the filtering phase involves running a Kalman filter for each model prediction and update. These Kalman filters operate in parallel, each producing its own state estimate and covariance.

$$\hat{\mathbf{x}}_{k|k-1}^{(m)} = \mathbf{F}^{(m)}\hat{\mathbf{x}}_{k-1|k-1}^{(m)} \tag{3.22}$$

$$\hat{\mathbf{x}}_{k|k}^{(m)} = \hat{\mathbf{x}}_{k|k-1}^{(m)} + \mathbf{K}_k^{(m)}(\mathbf{z}_k - \mathbf{H}^{(m)}\hat{\mathbf{x}}_{k|k-1}^{(m)}) \tag{3.23}$$

Thus for each model, the model probability update phase computes the likelihood of the observed measurements based on the Gaussian function, the model probabilities are then updated according to a normalized likelihood measure.

$$\mu_k^{(m)} = \frac{\mathcal{L}_k^{(m)}\mu_{k|k-1}^{(m)}}{\sum_{j=1}^{M}\mathcal{L}_k^{(j)}\mu_{k|k-1}^{(j)}} \tag{3.24}$$

whereas $\mathcal{L}_k^{(m)}$ is the likelihood of model m at time k .It evaluates the Gaussian probability density function at the residual between the estimate and actual measurement with mean 0 and covariance $\mathbf{S}_k^{(m)}$.

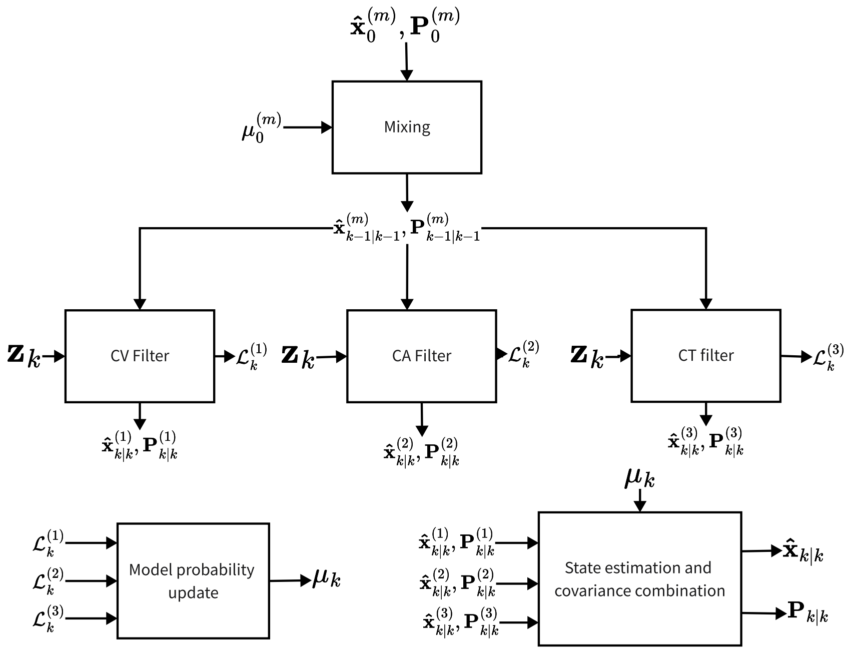
$$\mathcal{L}_k^{(m)} = \mathcal{N}(\mathbf{z}_k - \hat{\mathbf{z}}_{k|k-1}^{(m)}; 0, \mathbf{S}_k^{(m)}) \tag{3.25}$$

The final step involves combining the state estimates and covariances of all M models at time k based on the updated model probabilities to compute the overall estimate $\hat{\mathbf{x}}_{k|k}$ and covariance $\mathbf{P}_{k|k}$.

$$\hat{\mathbf{x}}_{k|k} = \sum_{m=1}^{M}\mu_k^{(m)}\hat{\mathbf{x}}_{k|k}^{(m)} \tag{3.26}$$

$$\mathbf{P}_{k|k} = \sum_{m=1}^{M}\mu_k^{(m)}\left(\mathbf{P}_k^{(m)} + (\hat{\mathbf{x}}_{k|k}^{(m)} - \hat{\mathbf{x}}_{k|k})(\hat{\mathbf{x}}_{k|k}^{(m)} - \hat{\mathbf{x}}_{k|k})^T\right) \tag{3.27}$$

The flow chart of IMM filter is shown as Figure 3.2: It briefly illustrates the IMM filter in our case with three models: CV, CA, and CT.



$\hat{\mathbf{x}}_0^{(m)}, \mathbf{P}_0^{(m)}$
$\mu_0^{(m)}$
Mixing
$\hat{\mathbf{x}}_{k-1|k-1}^{(m)}, \mathbf{P}_{k-1|k-1}^{(m)}$
$\mathbf{z}_k$
CV Filter
$\mathcal{L}_k^{(1)}$
$\mathbf{z}_k$
CA Filter
$\mathcal{L}_k^{(2)}$
$\mathbf{z}_k$
CT filter
$\mathcal{L}_k^{(3)}$
$\hat{\mathbf{x}}_{k|k}^{(1)}, \mathbf{P}_{k|k}^{(1)}$
$\hat{\mathbf{x}}_{k|k}^{(2)}, \mathbf{P}_{k|k}^{(2)}$
$\hat{\mathbf{x}}_{k|k}^{(3)}, \mathbf{P}_{k|k}^{(3)}$
μ_k
$\mathcal{L}_k^{(1)}$
$\mathcal{L}_k^{(2)}$
$\mathcal{L}_k^{(3)}$
Model probability update
μ_k
$\hat{\mathbf{x}}_{k|k}^{(1)}, \mathbf{P}_{k|k}^{(1)}$
$\hat{\mathbf{x}}_{k|k}^{(2)}, \mathbf{P}_{k|k}^{(2)}$
$\hat{\mathbf{x}}_{k|k}^{(3)}, \mathbf{P}_{k|k}^{(3)}$
State estimation and covariance combination
$\hat{\mathbf{x}}_{k|k}$
$\mathbf{P}_{k|k}$

Figure 3.2: IMM Filter flow chart

3.5 Performance Analysis

In order to test the IMM filter along with other kalman filter prediction performances, we first test it on the simulation dataset which combines with motion following the predefined model motion. The Gaussian noise has been added to test the filter's robustness. As Figure 3.3 shows, the simulation trajectory is generated following three constant motion models(CV, CT, CA) as different parts of the trajectory.

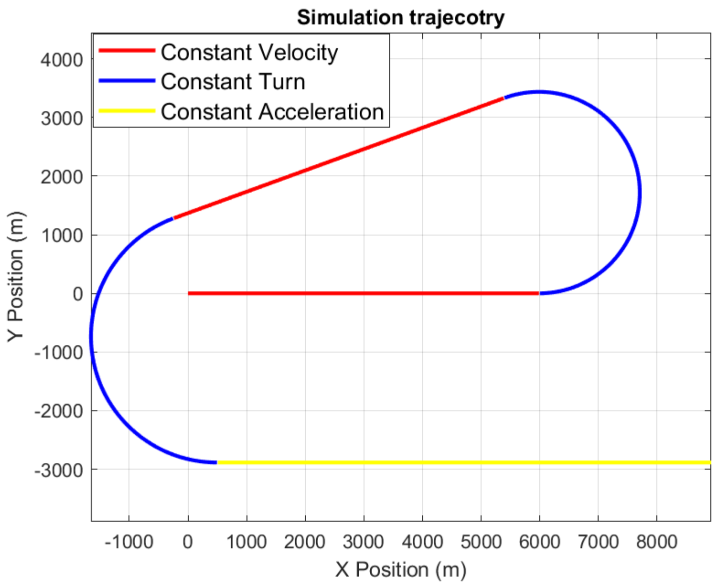
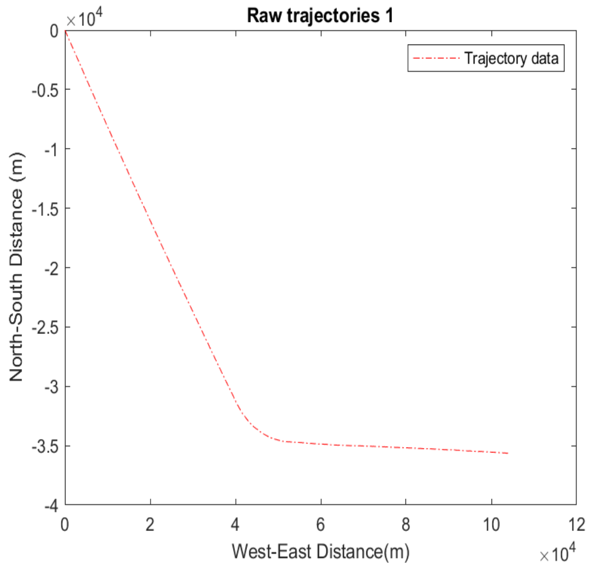


Figure 3.3: Simulation trajectory

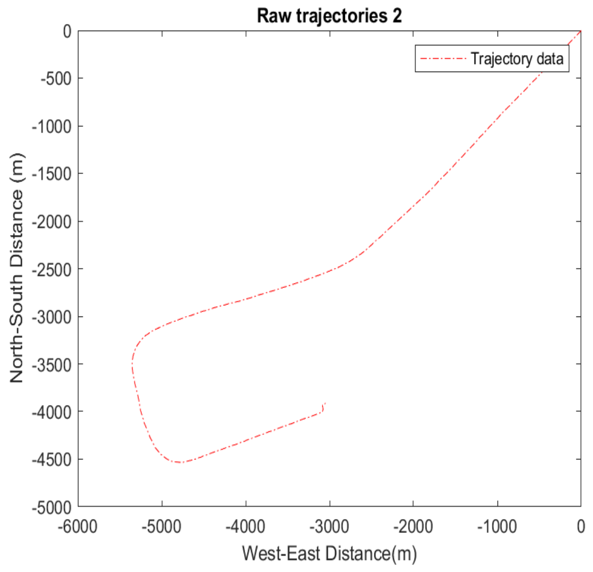
Besides, the real-world aircraft trajectory dataset has been used Figure 3.4 to further investigate the IMM performance. To ensure simplicity and obtain reasonable results, we selected two representative flight scenarios from each dataset. These scenarios are referred to as "trajectory 1" and "trajectory 2." We use the notation X_{data} to represent the new set of trajectories.

- **Trajectory 1:** This scene contains the trajectory of a jetliner, which flew in a linear motion for a certain period of time and then executed a lazy turn. The aircraft maintained its cruising speed and altitude during this period. The trajectory of the aircraft is shown in Figure 3.4a.

- **Trajectory 2:** It includes the landing trajectory of a GA aircraft. The aircraft made sharper turns and frequently changed its velocity. Compared to Scene 1, this trajectory exhibits more abrupt changes in states. This is depicted in Figure 3.4b.



(a) Aircraft trajectory 1



(b) Aircraft trajectory 2

Figure 3.4: Two flight trajectories. Trajectories 1 is from the TUD data set and Trajectories 2 is from the $TrajAir$ data set.

3.5.1 Root mean square error

Root mean square error (RMSE) is used to quantify the difference between the estimated positions (given by the filter models) and the actual positions The specific RMSE equation in the context of $2D$ trajectory is calculated as follows:

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}((\hat{x}_i - x_i)^2 + (\hat{y}_i - y_i)^2)} \tag{3.28}$$

Where N is the total number of data points or samples, $\hat{x}_i$ and $\hat{y}_i$ are the estimated trajectories, x_i and y_i are the observation trajectories.

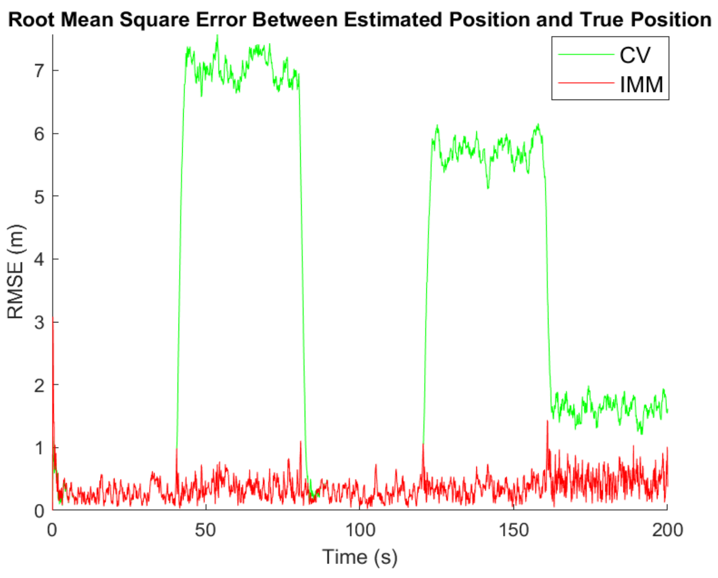
3.5.2 Model probability

Model probabilities indicate the likelihood that each model is the correct one for the current behavior of the target, based on the match between the estimations made by each model and the actual observations.

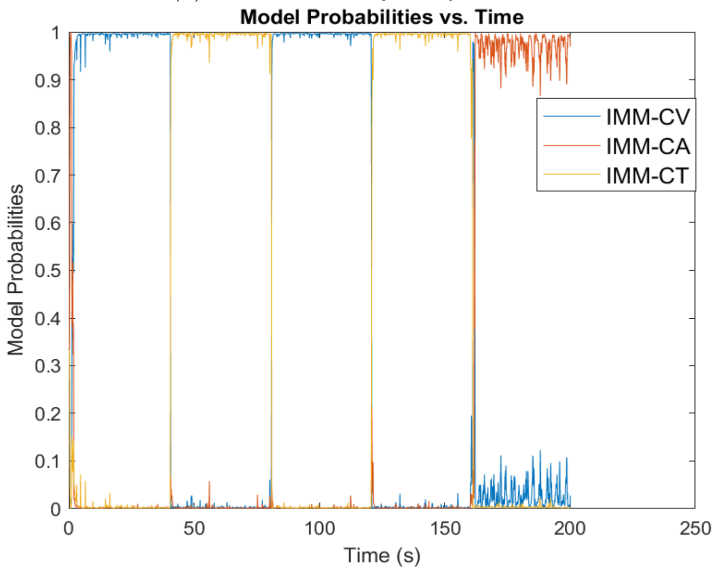
$$\mu_1 + \mu_2 + \ldots + \mu_i = 1 \tag{3.29}$$

Where μ_i is the model probability for model i, the model probabilities should add up to 1.

3.5.3 Simulation data set results



(a) Simulation trajectory RMSE

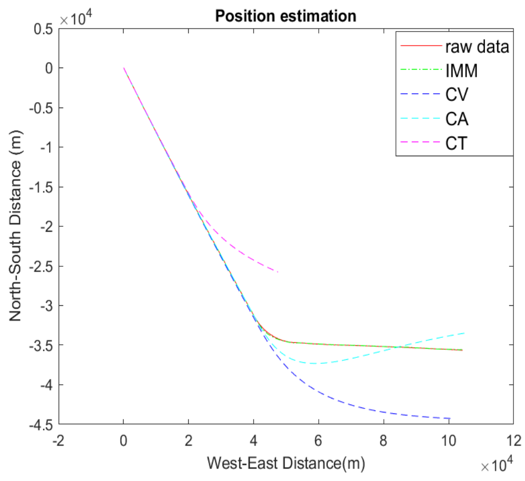


(b) Simulation trajectory model probability

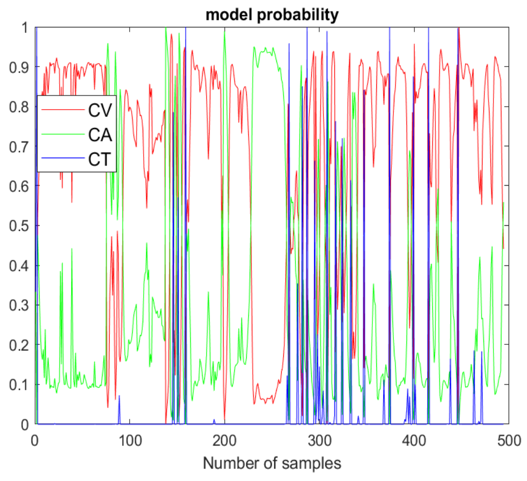
Figure 3.5: Simulation trajectory

The results depicted in Figure 3.5 display the RMSE over time on simulation trajectory for both the IMM filter and the constant velocity (CV) model over time, as well as the model probability over time. It is evident that the IMM model accurately recognizes the motion model across different trajectory segments. As observed from the model probability figure, the IMM can swiftly switch to the corresponding model based on the current motion dynamics. Furthermore, because it operates accurately under the correct motion model, the IMM maintains a substantially lower RMSE throughout the duration compared to the CV filter. In contrast, the CV filter only exhibits precise predictions during segments of constant velocity motion, resulting in a larger RMSE during other trajectory motion segments.

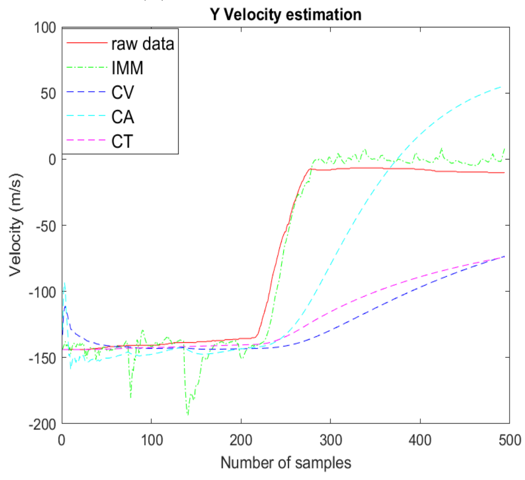
3.5.4 Real data set results



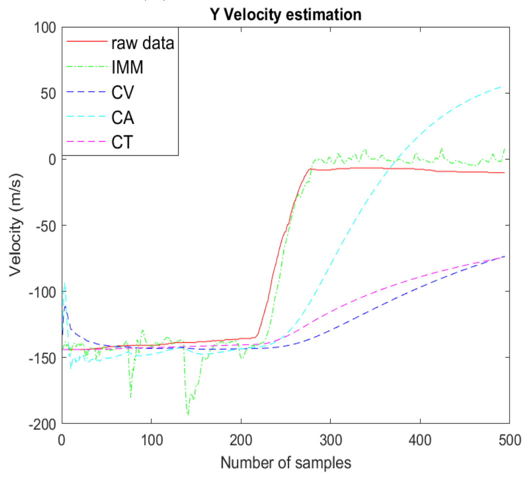
(a) Trajectory scene 1



(b) Model Probability



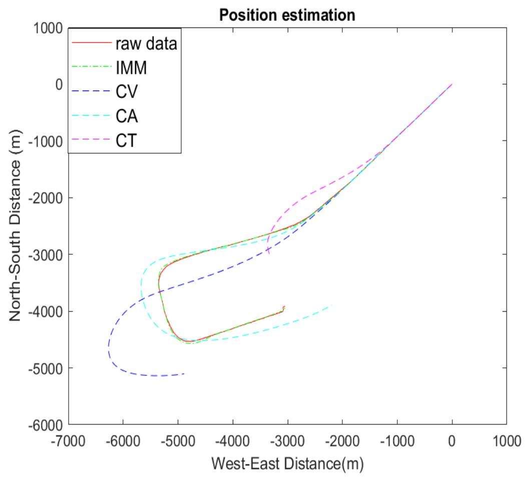
(c) Trajectory Velocity Estimation for x-axis position



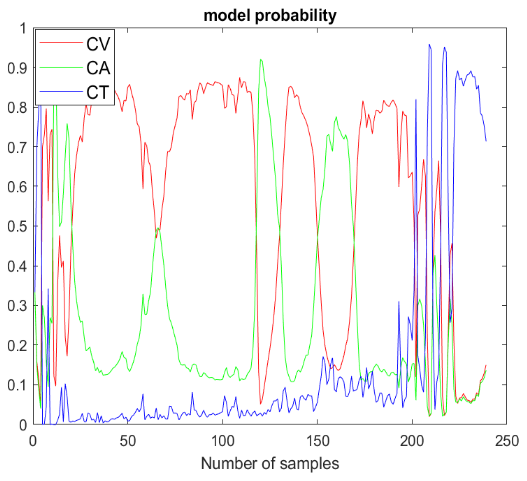
(d) Trajectory Velocity Estimation for y-axis position

Figure 3.6: Evaluation results for Trajectory 1

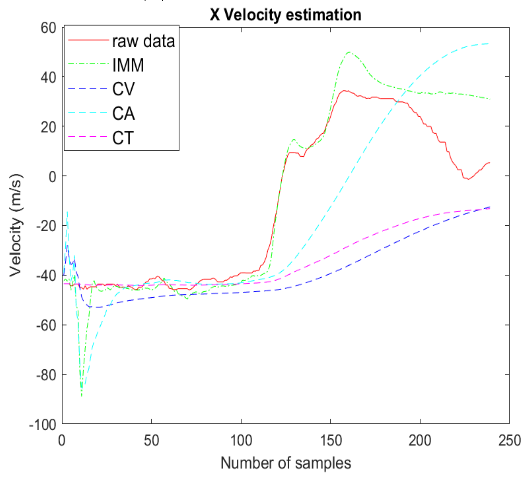
The results Figure 3.6 shows the estimation of the first real trajectory, model probability update for three models, and corresponding velocity estimation. In terms of trajectory prediction, we evaluated the IMM filter along with three single filters: CV, CA, and CT filters. It can be stated that the IMM filter outperforms the other three filters. From the analysis of the model probability, we observed that the three models change frequently while the aircraft is in motion, while the constant velocity model remains dominant. Additionally, based on the velocity estimation for both the x and y positions, we found that the IMM filter tracks the velocity better than any of the other three models at both positions. Overall, the results clearly demonstrate that the IMM filter provides superior performance compared to the single filters in terms of trajectory estimation and velocity tracking.



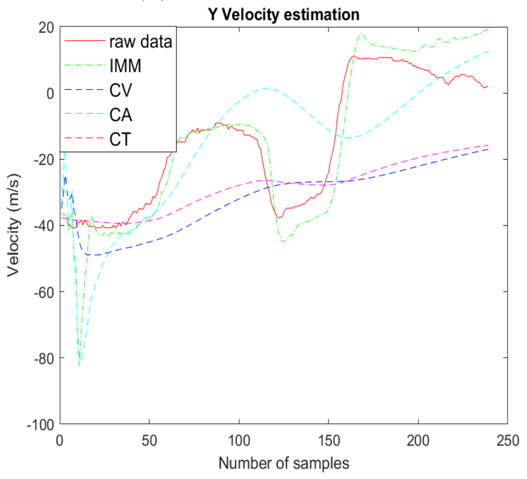
(a) Trajectory scene 2



(b) Model Probability



(c) Trajectory Velocity Estimation for x position

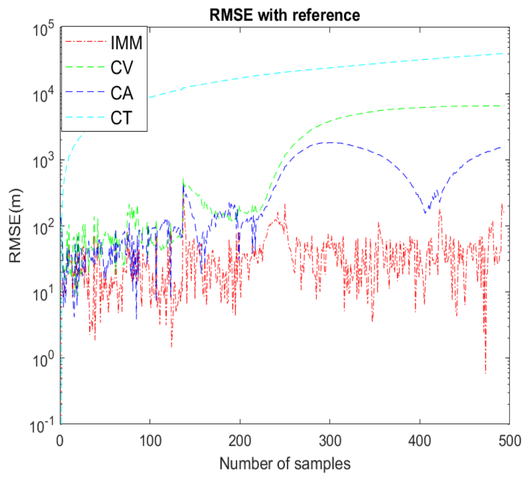


(d) Trajectory Velocity Estimation for y position

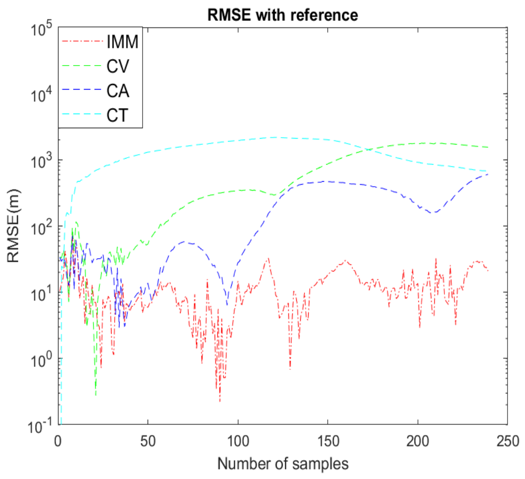
Figure 3.7: Evaluation results for Trajectory 2

The results of the second trajectory, as shown in Figure 3.7, further reinforce the superior performance of the IMM filter compared to the other single filters in terms of both trajectory estimation and velocity tracking. Analyzing the model probability, we observed that the constant velocity motion is initially dominant and eventually transitions to a turn model. This observation aligns with the intuitive understanding of the motion pattern exhibited by trajectory 2.

The figures referenced above, Figure 3.8 and Table 3.1, provide a detailed analysis of the RMSE for each data point, as well as the overall RMSE for two trajectories. These are analyzed in comparison with the four filters we previously discussed. Given that the trajectory measurements are in kilometers, significant differences can become apparent if the motion deviates from the intended trajectory. This is illustrated by the constant turn model, which struggles to accurately estimate the motion, resulting in a notable RMSE discrepancy with the true trajectory.



(a) Trajectory 1 RMSE for each data



(b) Trajectory 2 RMSE for each data

Figure 3.8: RMSE for each data sample

Table 3.1: Comparison of overall RMSE for trajectories by IMM, CV, CA, and CT filters (in kilometers)

RMSE (km)	IMM	CV	CA	CT
Trajectory 1	0.074	5.356	1.214	11.200
Trajectory 2	0.022	1.396	0.382	2.081

Overall, the analysis validates the performance of the IMM filter, as it is capable of accurately capturing the motion for both trajectories and providing reasonable estimations. This robust performance underscores the IMM filter's ability to adapt to varying motion dynamics and provides strong evidence for its suitability in trajectory-tracking applications.

3.6 Summary

In this chapter, model-based trajectory prediction methods have been implemented for the aircraft trajectory dataset. The main contribution can be summarised as follows:

- Define the state space models, the observation, and the noise covariance matrix of the interactive multiple models (IMM) filter.
- Implement the IMM filter to evaluate the trajectory prediction performance both on the simulation and real aircraft data set compared with three single Kalman filter constant velocity (CV), constant acceleration (CA), and constant turn (CV) models.
- Based on the prediction results, it show that the IMM filter exhibits superior prediction accuracy compared to the CV, CA, and CT filters. This indicates that the IMM filter excels in handling trajectories characterized by complex and varying motion patterns than the single model Kalman filter.

Gaussian Process on Drone Motion Trajectory 4

In chapter 3, we explore the trajectory prediction of aircraft motion. However, the limited degrees of freedom in aircraft restrict their maneuverability. For instance, they can't move instantaneously in any direction within their available degrees of freedom without first changing their orientation. Nevertheless, predicting UAVs' motion is also essential. One prime example of UAVs' motion is the trajectory of drones. With their increasing prevalence, accurate trajectory prediction becomes crucial for collision avoidance.

Moreover, within the context of air traffic control, predicting the trajectory of UAVs using the ADS-B transmission protocol enables seamless communication integration with manned aerial vehicles. Integrating UAVs into shared airspace ensures that they operate within established regulatory frameworks, fostering broader acceptance and trust in UAV operations. Therefore, in this chapter, we address trajectory tracking for drone motion.

To predict the drone trajectory, we first found that it's challenging to implement the model-based filter to accurately predict the trajectory of its high maneuver characteristic. Since the model mismatch could lead to potential large cumulative error. Thus, we investigate a data-driven method– Gaussian process regression (GP) to predict the future position of drone trajectory based on the past data set.

As we discussed beforehand, another important advantage of GP is it can explicitly quantify the uncertainty in the trajectory prediction aspect. The main sources of uncertainty come from drone environment factors, human intervention, and other unknown types. In the context of safety planning, the uncertainty on the prediction of future positions of drones is relevant for the detection of abnormal behavior of drones and for assessing collision risk in advance.

4.1 Problem Formulation

Suppose we have a trajectory training data set $D = \{(\mathbf{x}_i, f_i), i = 1, 2 \ldots n\}$ where $\mathbf{x}_i$ denotes a multidimensional input vector and f_i denotes a scalar output corresponding to $\mathbf{x}_i$, the i^{th} element in $\mathbf{x}$ and n denotes the sample size of the trajectory dataset. f_i values are assumed to be sample values of a non-linear and noisy process:

$$f_i = f(\mathbf{x}_i) \tag{4.1}$$

Gaussian process Regression is a non-parametric regression method that finds a distribution over the possible function $f(\mathbf{x})$ that is consistent with the observed data D [42]. A GP model is fully described by its mean function $m(\mathbf{x})$ and covariance function $k(\mathbf{x}, \mathbf{x}')$ as shown below:

$$f \sim \mathcal{N}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')) \tag{4.2}$$

The shape and smoothness of f are determined by the covariance function, as it controls the correlation between all pairs of output values.

The goal of the GP is to predict the value of the function f at any input $\mathbf{x}_*$. By definition [42], we have:

$$\left[\begin{array}{c} f \\ f(\mathbf{x}_*) \end{array}\right] \sim \mathcal{N}\left(0, \left[\begin{array}{cc} K & K_*^T \\ K_* & K_{**} \end{array}\right]\right) \tag{4.3}$$

where covariance sub-matrices, K, K_*^T, K_*, K_{**}, are calculated from Equation 4.4 to Equation 4.6 [42] and the mean function is assumed to be 0 i.e. $m(\mathbf{x}) = 0$.

$$K = \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & \cdots & k(\mathbf{x}_1, \mathbf{x}_n) \\ \vdots & \ddots & \vdots \\ k(\mathbf{x}_n, \mathbf{x}_1) & \cdots & k(\mathbf{x}_n, \mathbf{x}_n) \end{bmatrix} \tag{4.4}$$

$$K_* = \begin{bmatrix} k(\mathbf{x}_*, \mathbf{x}_1) & k(\mathbf{x}_*, \mathbf{x}_2) & \ldots & k(\mathbf{x}_*, \mathbf{x}_n) \end{bmatrix}, \tag{4.5}$$

$$K_{**} = k(\mathbf{x}_*, \mathbf{x}_*), \tag{4.6}$$

where $k(\mathbf{x}, \mathbf{x}')$ denotes the covariance between two inpts vector $\mathbf{x}$ and $\mathbf{x}'$. The covariance value is determined by the kernel function discussed below.

The prerequisite for GP is that the kernel functions need to be a positive-definite function. There are several kernel functions used in the literature as covariance functions. In [10] an overview of popular kernel functions for GP is provided. We choose the squared exponential (SE) function because it offers the smoothness in regression models which aligns well with trajectory modeling.

$$k(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left(-\frac{1}{2}\frac{(\mathbf{x} - \mathbf{x}')(\mathbf{x} - \mathbf{x}')^T}{\sigma_l^2}\right)\Bigg) \tag{4.7}$$

In Equation 4.7, σ_l is the characteristic length scale, and σ_f determines the standard deviation of the output. The parameters, σ_l and σ_f are called hyperparameters and form a set values $\theta = \{\sigma_l, \sigma_f\}$ [8]

The learning process of GP consists of tuning the values of θ to maximize the posterior probability of $\mu(f \mid \mathbf{x}; \theta)$, This is done by maximization of the log marginal likelihood with respect to hyperparameters in θ.

$$\theta^* = \arg\max_\theta \log \mu(f \mid \mathbf{x}; \theta) \tag{4.8}$$

The conditional probability distribution of $f(\mathbf{x}_*)$ given the observed data, D, has a multivariate normal distribution and is defined as:

$$f(\mathbf{x}_*) \mid f \sim \mathcal{N}\left(K_* K^{-1} f, K_{**} - K_* K^{-1} K_*^T\right) \tag{4.9}$$

where $K, K_*$and K_{**} are define from Equation 4.4 to Equation 4.6. The best estimation of $f(\mathbf{x}_*)$ is the mean $E[f(\mathbf{x}_*)]$ and the uncertainty of prediction is captured in the covariance $\operatorname{Cov}[f(x_*)]$ which are determined as follows:

$$E[f(\mathbf{x}_*) \mid f] = K_* K^{-1} f, \tag{4.10}$$

$$\sigma^2(\mathbf{x}_*, D) = \operatorname{Cov}[f(\mathbf{x}_*) \mid f] = K_{**} - K_* K^{-1} K_*^T \tag{4.11}$$

In conclusion, the GP could be beneficial for regressing highly maneuverable trajectories, such as drone trajectories. These trajectories often exhibit rapid changes in direction and velocity, resulting in correlations between different dimensions of position. The GP is well-suited to capture these correlations, enabling the sharing of information across dimensions and leading to improved accuracy in trajectory estimation.

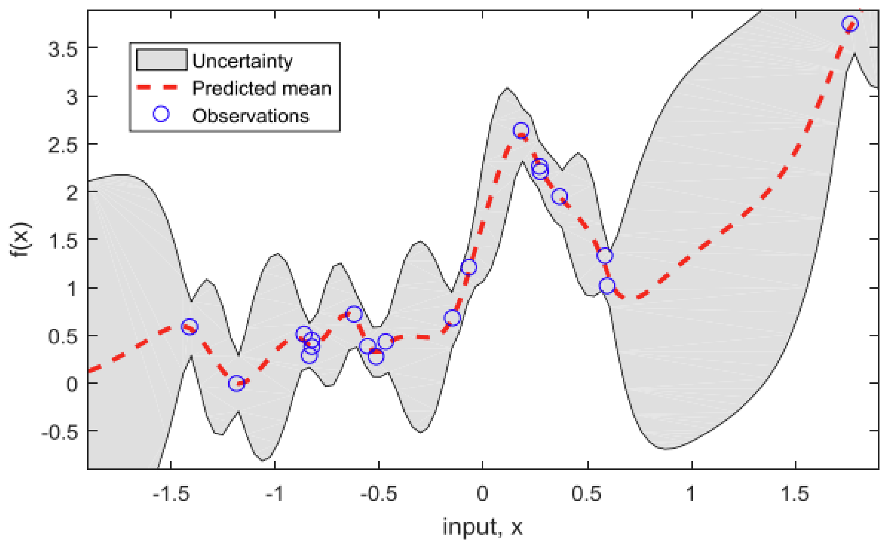


Figure 4.1: An example of one-dimensional GPR using zero mean and squared exponential covariance function [15]

4.2 Online trajectory prediction

There are two types of trajectory prediction methods: online learning and batch learning. In online learning, the model updates its predictions based on new incoming data points. In contrast, batch learning, also known as offline learning, is trained using the entire dataset at once. The motivation for using online learning, specifically in the context of drone trajectory prediction, is as follows:

Firstly, drone trajectory prediction often requires real-time predictions to be effective, especially for tasks like collision avoidance or path planning. Online learning can provide more timely predictions as the model can be updated as soon as new data becomes available.

Secondly, training a model using batch learning on a large dataset can be computationally expensive and time-consuming. If the drone's trajectory data is continuously recorded, this dataset can quickly become very large. Online learning, on the other hand, can be more computationally efficient as it only needs to process the new data as it arrives.

Moreover, in some cases, only the recent past is relevant for predicting the immediate future trajectory of the drone. Older data points might not contribute much to the prediction accuracy and can even be misleading if the drone's dynamics or the environment have significantly changed. Therefore, online learning allows the model to adapt to the most up-to-date information and make more accurate predictions.

4.2.1 Gaussian process with sliding window

Given the dataset D, described as ADS-B type includes $3D$ trajectory position. where $D = [x^1, y^1, z^1], [x^2, y^2, z^2], \ldots, [x^n, y^n, z^n]$, n denotes the total sample numbers. The problem is to use the observed data to predict the position of the target for the following time steps

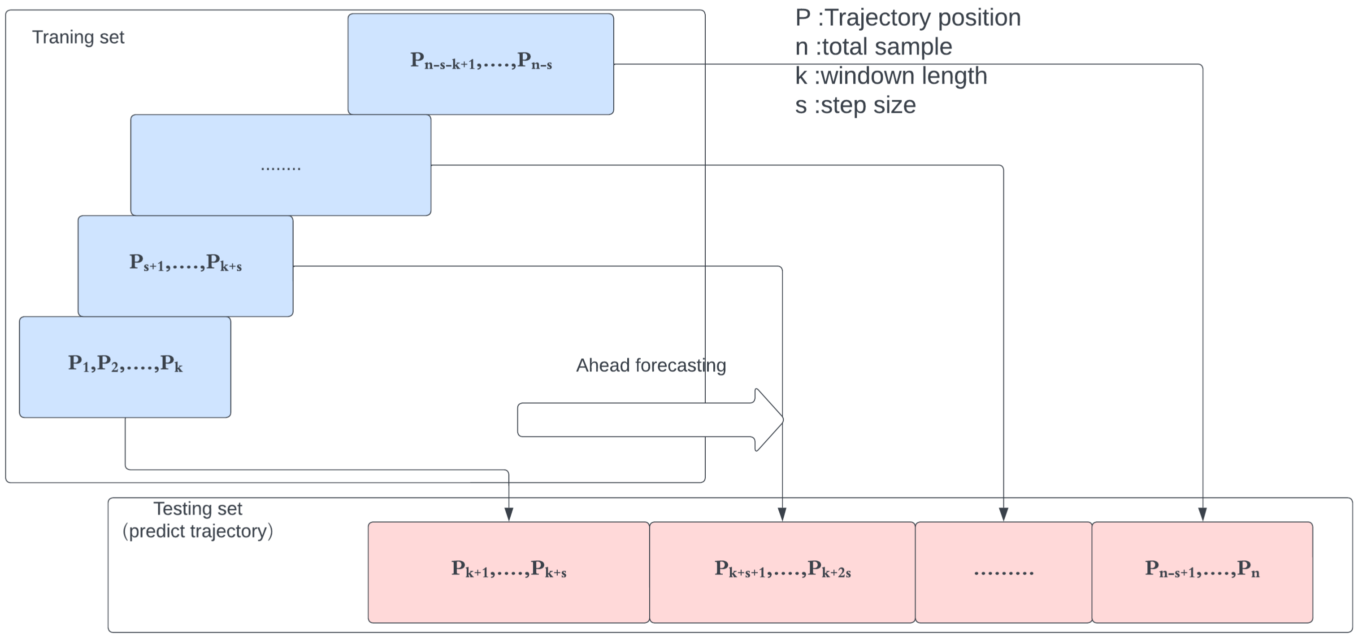


Figure 4.2: Slide window scheme

More specifically, the online application of GP faces the significant hurdle of computational complexity, which grows at a cubic rate proportional to the number of training points. For evolving systems, this can lead to rapidly escalating computational costs. A possible solution is using a sliding window approach to manage computational demands. The sliding window scheme for trajectory prediction is shown as Figure 4.2, it can be observed that $\mathbf{p}_i$ denotes the index for the ith $3D$ position, denotes as $\mathbf{p}_i \in \{x_i, y_i, z_i\}$. Within n total samples, the k and s represent the window length and step size respectively.
For the first iteration, the first k data sample was taken as the training set to

$$\mathbf{P}_{\text{train1}} = [\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_k] = \begin{bmatrix} x_1 & x_2 & \ldots & x_k \\ y_1 & y_2 & \ldots & y_k \\ z_1 & z_2 & \ldots & z_k \end{bmatrix} \tag{4.12}$$

The GP model assumes that position matrix $\mathbf{P}$ for each dimension $d \in x, y, z$, t_{train} denotes the time stamps for the training set. The data follows the Gaussian distribution for each training iteration:

$$\mathbf{P}_d(t_{train}) \sim \mathcal{N}\left(m_d(t_{train}), k_d\left(t_{train}, t'_{train}\right)\right), \quad d \in \{x, y, z\} \tag{4.13}$$

where $m_d(t_{train})$ is a mean function, $k_d(t_{train}, t_{train}{}')$ is a kernel function and train the GP model and then predict the next following s steps position

$$\mathbf{P}_{\text{test1}} = [\mathbf{p}_{k+1}, \mathbf{p}_{k+2}, \ldots, \mathbf{p}_{k+s}] = \begin{bmatrix} x_{k+1} & x_{k+2} & \ldots & x_{k+s} \\ y_{k+1} & y_{k+2} & \ldots & y_{k+s} \\ z_{k+1} & z_{k+2} & \ldots & z_{k+s} \end{bmatrix} \tag{4.14}$$

For each dimension $d \in x, y, z$, the predictive distribution in the test set $P_d(t_{test})$ is a Gaussian distribution for each testing iteration:

$$\mathbf{P}_d(t_{test}) \mid t_{train}, \mathbf{t} \sim \mathcal{N}\left(\mathbf{P}_d(t_{test}), \text{var}\left[(\mathbf{P}_d(t_{test}))\right]\right), \quad d \in \{x, y, z\} \tag{4.15}$$

where $\mathbf{P}_d(t_{test})$ is the predictive mean and $var(\mathbf{P}_d(t_{test}))$ is the predictive variance, given by the formulas for the posterior of a Gaussian process.

As for the sliding window scheme, for the next following iteration, the next training set $\mathbf{P}_{\text{train2}}$ will take the next s new data from $\mathbf{p}_{k+1}$ to $\mathbf{p}_{k+s}$ and discard the old data from $\mathbf{p}_1$ to $\mathbf{p}_s$ to keep the window length fixed as K, and then the next testing set $\mathbf{P}_{\text{test2}}$ will predict the trajectory position from $\mathbf{p}_{k+s+1}$ to $\mathbf{p}_{k+2s}$. Since it is a recursive algorithm to take the newest k data samples to predict the following s future position. If we assume there are m iterations, then the observed data is the combination of the overall training sets and the predicted trajectory is the combination of the overall testing sets

$$\mathbf{P}_{\text{observe}} = [\mathbf{P}_{\text{train1}}, \mathbf{P}_{\text{train2}}, \dots, \mathbf{P}_{\text{trainm}}] = [\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_{n-s}] \tag{4.16}$$

$$\mathbf{P}_{\text{predict}} = [\mathbf{P}_{\text{test1}}, \mathbf{P}_{\text{test2}}, \dots, \mathbf{P}_{\text{testm}}] = [\mathbf{p}_{k+1}, \mathbf{p}_{k+2}, \dots, \mathbf{p}_n] \tag{4.17}$$

which denotes that the predicted trajectory $\mathbf{P}_{\text{predict}}$ records the consistent position from the end of the first test data $\mathbf{p}_{k+1}$ until the last dataset samples $\mathbf{p}_n$.

Algorithm 2 Online GP with Sliding Window for 3D Trajectory Prediction

```
 1: Inputs: $n$ total samples, window length $k$, step size $s$, positions $\mathbf{P} = \{\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_n\}$ where
    $\mathbf{p}_i \in \mathbb{R}^3$, timestamps $T = \{t_1, t_2, \dots, t_n\}$.
 2: Outputs: Predicted positions $\mathbf{P}_{\text{predict}} = \{\mathbf{p}_{k+1}, \mathbf{p}_{k+2}, \dots, \mathbf{p}_n\}$ where $\mathbf{p}_i \in \mathbb{R}^3$.
 3: for $i = 1$ to $n - k$ with step $s$ do
 4:     // Form the current training set
 5:     Form training set $\mathbf{P}_{\text{train}} = \{\mathbf{p}_j\}_{j=i}^{i+k-1}$,$T_{\text{train}} = \{\mathbf{t}_j\}_{j=i}^{i+k-1}$.
 6:     // Train a GP model on the training set
 7:     Define GP with RBF kernel for 3D data.
 8:     Optimize model hyperparameters (length scale, kernel variance)
 9:     Train GP model $\mathbf{f}$ on $\mathbf{P}_{\text{train}}$, $T_{\text{train}}$.
10:     // Predict the future 3D position
11:     Predict and store future positions $\mathbf{P}_{\text{predict}} = \{\mathbf{f}(t_m)\}_{m=i+k}^{i+k+s-1}$,$T_{\text{test}} = \{\mathbf{t}_m\}_{m=i+k}^{i+k+s-1}$
12: end for
```

Algorithm 2 presents the pseudocode for online GP using a sliding window approach to predict trajectories. In each iteration, the training set comprises the time indices T_{train} as inputs and the $3D$ positions $\mathbf{P}_{\text{train}}$ as outputs. The GP model is trained with these data, and its hyperparameters are optimized by maximizing the marginal likelihood. For the test set, future time indices are used as inputs to predict future $3D$ positions. After each prediction, the predicted positions are accumulated in a sequence throughout the loop, forming the predicted trajectory, denoted as $\mathbf{P}_{\text{predict}}$.

4.3 Simulation

Based on posterior distributions of GP models from Equation 4.10 and Equation 4.11, both prediction uncertainty and accuracy can be studied. The mean of the predictive distribution is considered the most probable prediction and thus used to compute the accuracy metrics with the ground truth data. The accuracy of the predictions is measured by RMSE for each dimension:

$$\text{RMSE}(x) = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (\hat{x}_i - x_i)^2} \tag{4.18}$$

Where N is the total number of data points or samples, $\hat{x}_i$ is the estimated sample, x_i is the observation sample, whereas for the y and z dimensions the equation is similar with Equation 4.18 Given the predicted vector $\hat{\mathbf{v}} = [\hat{x}, \hat{y}, \hat{z}]$ and the true vector $\mathbf{v} = [x, y, z]$, the RMSE in vector form for the $3D$ position is:

$$\text{RMSE}(\mathbf{v}) = \sqrt{\frac{1}{N} \sum_{i=1}^{N} \|\hat{\mathbf{v}}_i - \mathbf{v}_i\|^2} \tag{4.19}$$

where $\hat{\mathbf{v}}_i$ and $\mathbf{v}_i$ are the predicted and reference vectors for the i-th sample, respectively, and N is the number of samples. The notation $\|\cdot\|$ denotes the Euclidean norm.

The uncertainty is quantified by computing the standard deviation(σ square root of the variances from Equation 4.11 of the samples prediction, indicating the ambiguity of the predicted position. Instead of using the sliding window scheme, we investigate trajectory prediction using the growing window scheme, as illustrated in Figure 4.3. In the growing window approach, the window size expands by s steps with each iteration, rather than discarding the earliest data points. This leads to higher computational complexity compared to the sliding window scheme.

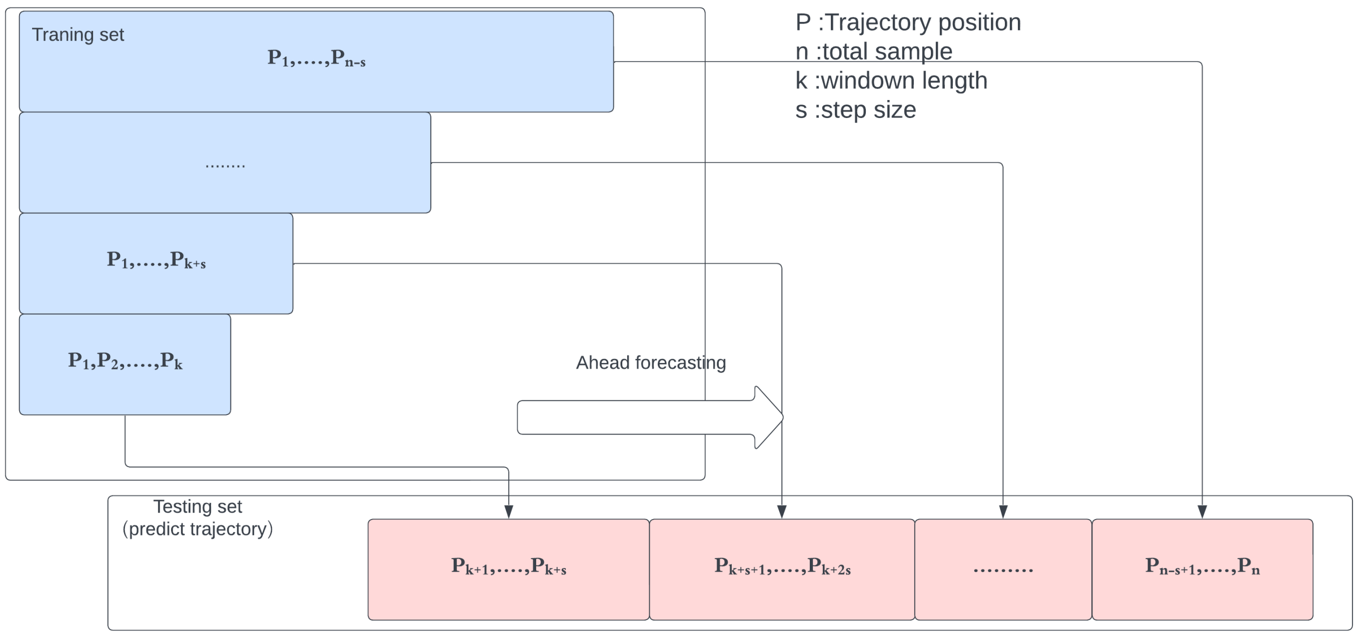


Figure 4.3: Growing window scheme

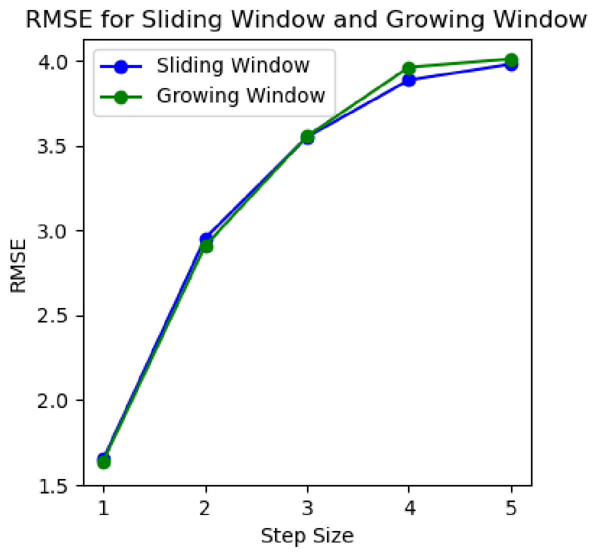


Figure 4.4: RMSE for slide and grow

Figure 4.4 shows the RMSE between the proposed sliding window strategy with a fixed window length k of 15 and a growing window with each iteration within the step size ranging from 1 to 5. We discovered that antiquated data fail to contribute relevant information for accurate predictions and may even introduce irrelevant information. This could lead to a larger RMSE when compared to the sliding window, particularly as the step size increases.

Furthermore, when examining computational complexity, the sliding window's complexity is approximately $\mathrm{O}\left((\mathrm{N/s})\cdot \mathrm{k}^3\right)$ with the fixed window length k, while the growing window's complexity is approximately $\mathrm{O}\left((\mathrm{N/s})\cdot \mathrm{m}^3\right)$, where growing window length $m = k + i$ and $i = 1, 2, 3 \ldots$. Thus, the computational complexity of the growing window is significantly higher, especially when handling larger datasets. This factor warrants considering the sliding window strategy as a preferable option when computational efficiency is a concern.

4.3.1 Simulation Results

For the parameter selection of the window length, a careful balance must be struck. If the window is too small, the model might not be able to accurately capture the dynamic characteristics of the drone's movement. Besides, considering the dataset availability and motion characteristics, we choose window length $k = 15$.

To motivate the step size related to the real-world project, since the max speed of drones ranges between 20-30 m/s, with top-tier models capable of reaching speeds up to 40 m/s [11]. In the context of collision avoidance consideration, these high-end drones feature obstacle detection systems with a range of approximately 15-30 meters e.g DJI Mavic Air 2S is able to detect the obstacle 28m away. To effectively utilize these systems for collision prevention, it becomes practical to predict the drone's trajectory between 0.5 to 2 seconds in advance. Accordingly, in our experiment, we examine time steps ranging from 0.5 to 2.5 seconds, incrementing each step by 0.5 seconds, to evaluate the performance of trajectory prediction. Through this approach, we aim to align our prediction capabilities with the operational characteristics of obstacle detection systems, thereby evaluating the effectiveness of collision avoidance mechanisms.

In terms of the results for the trajectory prediction with different step sizes, Table 4.1 is the table results of test trajectory RMSE for different step sizes in three dimensions. we observe the smaller step size could obtain better prediction results but lead to more computations, so it is a trade-off between accuracy and computational efficiency. However, due to its highly maneuver character, we found that prediction RMSE is relatively high, especially after a 2-second look-ahead may indicate it's challenging to predict the trajectory sufficiently for a long time look-ahead time. Table 4.2 is the mean uncertainty for the trajectory prediction for the different step sizes. It denotes a similar trend observing the increasing uncertainty with the step size increase.

Table 4.1: RMSE comparison for different step size prediction in $3D$

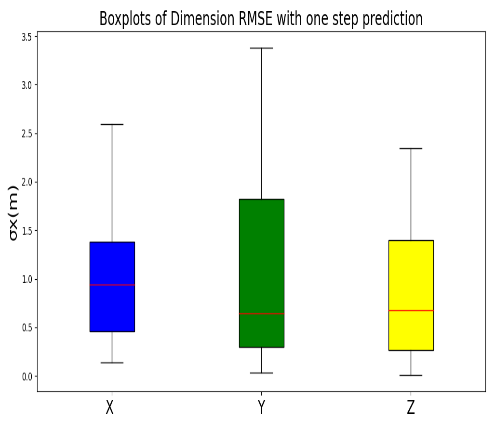
	Step Size				
RMSE(m)	1	2	3	4	5
X(m)	0.973	1.486	2.086	2.234	2.512
Y(m)	1.189	1.718	2.218	2.553	2.794
Z(m)	0.712	1.512	1.812	1.945	2.186
Overall(m)	1.693	2.855	3.503	3.888	4.342

Table 4.2: Mean uncertainty for different step size prediction in $3D$

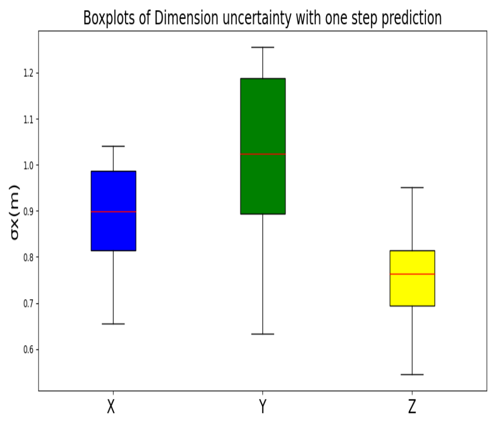
	Step Size				
Mean uncertainty(m)	1	2	3	4	5
X(m)	0.988	1.125	1.743	2.134	2.343
Y(m)	1.002	1.237	1.564	1.962	2.285
Z(m)	0.678	0.885	0.993	1.142	1.348

Figure 4.5a and Figure 4.5b are box plots that show the visualization prediction results with a step size s of 1 and a window length k of 15 regarding the median value and spread of RMSE and uncertainty for the given case over time. From the plots, we observe that the median value of RMSE for the x is around 0.8 meters, while the RMSE median for the y and z dimensions is lower at approximately 0.6 meters. However, the maximum RMSE and upper quartile indicate the y-dimension RMSE range is larger than the other two dimensions. In terms of uncertainty, the x dimension exhibits an average of around 0.9 meters and y shows a higher median value of 1.05 meters and a wider spread. The z dimension has a small variance range with a median value of 0.75, indicating that the position prediction in the z dimension is more confident which is due to the motion changing in the z dimension being relatively small compared with the other two dimensions. According to the trajectory characteristics, the results adequately reflect the motion trajectory for all three dimensions.

Figure 4.6 presents the outcomes of the GP online learning case with the same parameter as Figure 4.5a and Figure 4.5b. It shows the comparison between the mean predict trajectory (blue curve) and ADS-B type reference (red curve), and the shaded region around the mean prediction is the uncertainty region(95% confidence interval) From these results, it is noticeable that the online prediction effectively reflects the dynamic changes in motion across all three dimensions. Comparisons with the ground truth values show a reasonable alignment between the predicted and actual values, underscoring the predictive capability of this method. However, certain limitations are apparent. The predicted trajectory does not smoothly trace the target motion, especially when the direction of motion alters. This leads to a trajectory estimate that is both sharp and rough, consequently lowering the accuracy of the prediction. This shortcoming can be attributed to the data sampling rate of 2Hz. The 0.5-second interval between two measurements proves insufficient to accurately capture the drone's motion, resulting in a coarse trajectory prediction.

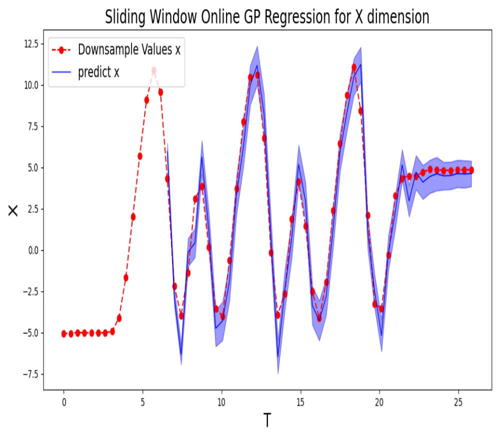


(a) RMSE quantification for three dimensions

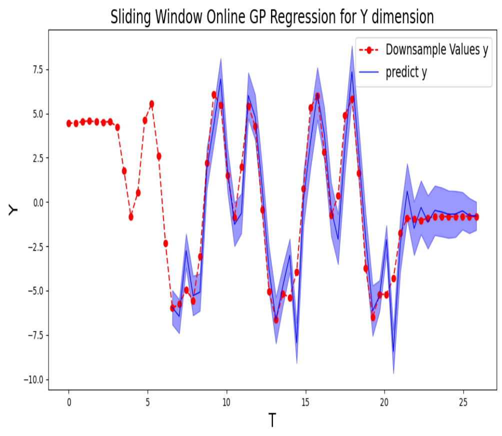


(b) Uncertainty quantification for three dimensions

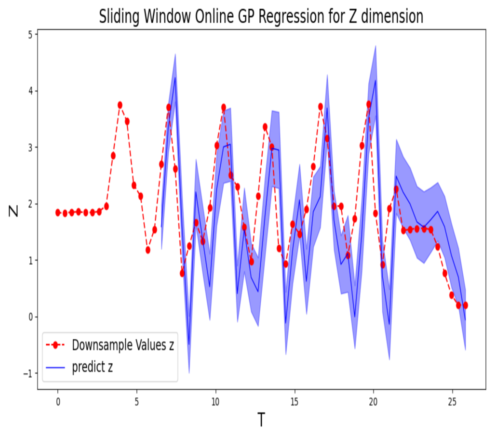
Figure 4.5: Box plots for three dimensions



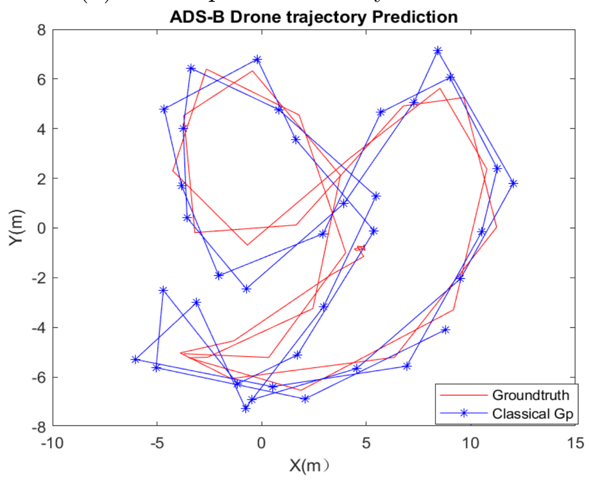
(a) Online prediction for x dimension



(b) Online prediction for y dimension



(c) Online prediction for z dimension



(d) Online prediction for xy planar

Figure 4.6: Evaluation results for online prediction for one step prediction

4.4 Low-rank approximation Gaussian process

In the drone's dynamic environment, the ability to quickly predict trajectories in real-time is a critical factor in ensuring safe and efficient operations. Utilizing GP can provide powerful tools for generating predictive models based on existing data. However, as the volume of data increases, the computational complexity of standard GP regression can become prohibitively high with the computation complexity $O(n^3)$ and memory storage $O(n^2)$, leading to delays in trajectory predictions and, consequently, potential safety risks and inefficiencies in drone operations.

In GP regression, the complexity of predictions is largely influenced by the inversion of the covariance (kernel) matrix, a process that is computationally intensive, scaling cubically with the number of data points. Low-rank approximations emerge as a powerful tool in mitigating this computational burden by approximating the full-rank kernel matrix with a matrix of lower rank, therefore simplifying the inversion process to a problem with reduced dimensions. Inducing methods serve as integral strategies in the facilitation of low-rank approximations in GP regression. These methods construct a reduced-rank representation of the kernel matrix by judiciously selecting a subset of data points, referred to as inducing points, to represent the entire data set, thereby trading off a degree of approximation accuracy for reduction in computational complexity. Through this selection process, they effectively distill the essence of the data into a compact, representative set, and generate an approximation of the kernel matrix that retains its critical structure and information but at a reduced computational cost. Consequently, inducing methods are considered to be a part of low-rank approximation techniques as they inherit the principle of reducing the rank of the kernel matrix to achieve computational efficiency while preserving the predictive power of the GP model as much as possible.

4.4.1 Inducing method

In terms of inducing points, there are several methods by replacing the exact $k(\mathbf{x}, \mathbf{z})$ to $\tilde{k}(\mathbf{x}, \mathbf{z})$ for fast computation. The most common inducing method such as the prominent subset of regressors (SoR) [50] and fully independent training conditional(FITC) [52] using the following approximate kernels

$$\tilde{k}_{\mathrm{SoR}}(\mathbf{x}, \mathbf{z}) = K_{\mathbf{x},U} K_{U,U}^{-1} K_{U,\mathbf{z}} \tag{4.20}$$

$$\tilde{k}_{\mathrm{FITC}}(\mathbf{x}, \mathbf{z}) = \tilde{k}_{\mathrm{SoR}} + \delta_{\mathbf{xz}} \left(k(\mathbf{x}, \mathbf{z}) - \tilde{k}_{\mathrm{SoR}} \right) \tag{4.21}$$

Given a collection of m inducing points, denoted as $U = [\mathbf{u}_i]_{i=1}^{m}$, the covariance matrices $K_{\mathbf{x},U}$, $K_{U,U}^{-1}$, and $K_{U,\mathbf{z}}$ are derived from the original kernel function $k(\mathbf{x}, \mathbf{z})$. These matrices have dimensions $n \times m$, $m \times m$, and $m \times n$ respectively. The SOR method results in an $n \times n$ covariance matrix, denoted K_{SoR}, which is of rank no greater than m. This indicates that SoR corresponds to a Gaussian process with a finite basis, making it degenerate. On the other hand, the FITC approach incorporates a diagonal correction, leading to a full rank covariance matrix, K_{FITC}. Due to this property, FITC often offers a more accurate approximation and is generally favored for practical applications. It's worth noting that the original kernel function, $k(\mathbf{x}, \mathbf{z})$, is characterized by parameters $\boldsymbol{\theta}$. Thus, when using an inducing point method, kernel learning typically involves optimizing the SoR or FITC marginal likelihoods in relation to $\boldsymbol{\theta}$.

Besides the above two methods, a new unifying framework for inducing points method called structured kernel interpolation (SKI) [58] has been proposed. It not only improves the scalability and accuracy of fast kernel methods but also naturally combines the advantages of inducing point and structure exploiting approaches.

4.4.2 Fast structure exploiting inference

Moreover, fast Structure-Exploiting Inference has been widely used within GP where certain structured properties of the covariance (or kernel) matrix are exploited to accelerate computations. The most common method includes Toeplotz and Krnoecker methods exploit the existing structure of the GP covariance matrix K to scale up inference and learning without approximations. Cooperating with the inducing methods can further reduce the computational complexity and storage limitation. In our case, we are dealing with the $3D$ dimension position separately with GP for prediction. Therefore, the covariance matrix would exhibit a Toeplitz structure due to the properties of the stationary kernel (RBF) with respect to $1D$ input data.

4.4.3 Toeplitz method

Given N training instances on a regularly spaced $1D$ input grid, a stationary kernel k will give rise to a Toeplitz covariance matrix K, meaning that every diagonal of K is the same (note also K is a covariance matrix, so is symmetric as well as Toeplitz)

$$\mathbf{K} = \begin{bmatrix} k_0 & k_1 & \dots & k_{N-2} & k_{N-1} \\ k_1 & k_0 & \dots & k_{N_3} & k_{N-2} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ k_{N-2} & k_{N-3} & \dots & k_0 & k_1 \\ k_{N-1} & k_{N-2} & \dots & k_1 & k_0 \end{bmatrix} \tag{4.22}$$

Where k_i represents the elements generated from a kernel function applied to the respective indices of input data.

The Toeplitz matrix $\mathbf{K}$ can be embed into a $2(N+1) \times 2(N+1)$ circulant matrix $\mathbf{C}$ [59] defined as:

$$C = \left[\begin{array}{ccccc|ccccc} k_0 & k_1 & \dots & k_{N-2} & k_{N-1} & k_{N-2} & k_{N-3} & \dots & k_2 & k_1 \\ k_1 & k_0 & \dots & k_{N-3} & k_{N-2} & k_{N-1} & k_{N-2} & \dots & k_3 & k_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ k_{N-2} & k_{N-3} & \dots & k_0 & k_1 & k_2 & k_3 & \dots & k_{N-2} & k_{N-1} \\ k_{N-1} & k_{N-2} & \dots & k_1 & k_0 & k_1 & k_2 & \dots & k_{N-3} & k_{N-2} \\ \hline k_{N-2} & k_{N-1} & \dots & k_2 & k_1 & k_0 & k_1 & \dots & k_{N-4} & k_{N-3} \\ k_{N-3} & k_{N-2} & \dots & k_3 & k_2 & k_1 & k_0 & \dots & k_{N-3} & k_{N-2} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ k_2 & k_3 & \dots & k_{N-2} & k_{N-3} & k_{N-4} & k_{N-3} & \dots & k_0 & k_1 \\ k_1 & k_2 & \dots & k_{N-1} & k_{N-2} & k_{N-3} & k_{N-2} & \dots & k_1 & k_0 \end{array}\right] \tag{4.23}$$

The benefit of this embedding is that circulant matrices have a special property that can be diagonalized by the Fourier matrix. This means we can solve the linear system using the fast Fourier transform(FFT) to achieve a faster in $O(n \log n)$ computations and $O(n)$ memory.

4.4.4 Structured Kernel Interpolation

Structured Kernel Interpolation(SKI) as one inducing points method for fast computational through kernel interpolation has been widely implemented on large datasets [9, 36]. Unlike the SOR method Equation 4.20 to approximate the $n \times m$ matrix $K_{X,U}$ of cross covariances for the kernel evaluated at the training and inducing inputs X and U, it can be interpolated on the $m \times m$ covariance matrix $K_{U,U}$. By doing so we could replace the $K_{U,U}^{-1}$ as a computationally expensive task by $K_{U,U}$. For example, if we wish to estimate $k(\mathbf{x}_i, \mathbf{u}_j)$, for input point $\mathbf{x}_i$ and inducing point $\mathbf{u}_j$, we can start by finding the two inducing points $\mathbf{u}_a$ and $\mathbf{u}_b$ which most closely bound $\mathbf{x}_i.\mathbf{u}_a \leq \mathbf{x}_i \leq \mathbf{u}_b$ ($D = 1$ in our case and a Toeplitz $K_{U,U}$ from a regular grid U). We can then form $\tilde{k}(\mathbf{x}_i, \mathbf{u}_j) = w_i k(\mathbf{u}_a, \mathbf{u}_j) + (1 - w_i) k(\mathbf{u}_b, \mathbf{u}_j)$, with linear interpolation weights w_i and $(1 - w_i)$, which represent the relative distances from $\mathbf{x}_i$ to points $\mathbf{u}_a$ and $\mathbf{u}_b$. More generally, we form

$$K_{X,U} \approx W K_{U,U}, \tag{4.24}$$

where W is an $n \times m$ matrix of interpolation weights that can be extremely sparse.

Substituting our expression for $\bar{K}_{X,U}$ Equation 4.24 into the SoR approximation for $K_{X,X}$, we find:

$$\begin{aligned} K_{X,X} &\overset{\text{SoR}}{\approx} K_{X,U} K_{U,U}^{-1} K_{U,X} \overset{\text{Eq.(4.24)}}{\approx} W K_{U,U} K_{U,U}^{-1} K_{U,U} W^\top \\ &= W K_{U,U} W^\top = K_{\text{SKI}}. \end{aligned} \tag{4.25}$$

In the SKI example scheme depicted in Figure 4.7, we utilize a sparse matrix W to facilitate an efficient approximation of the exact GP inference. In this method, only a subset of entries in the matrix are non-zero, thereby creating a sparse representation that significantly reduces computational demands. This is achieved by multiplying the sparse matrix W with the Toeplitz matrix $K_{U,U}$ of size $m \times m$ to approximate the GP inference with a lower rank. As a consequence, the computation complexity of SKI GP is approximately about $O(n + m \log m)$ [58] which is less compared with classical GP with $O(n^3)$ complexity.

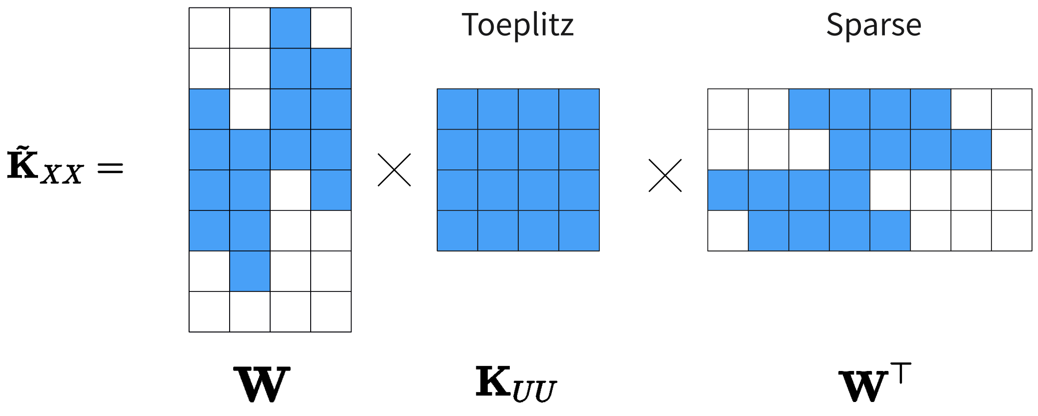


Figure 4.7: Structured Kernel Interpolation

Finally, we notice that all inducing methods are within the unified conceptual framework of SKI. When we consider a GP with a zero mean and devoid of noise, the predictive mean $\bar{f}_*$ can be expressed in a linear fashion in two distinct manners. Firstly, it can be represented as a $\mathbf{w}_X(\mathbf{x}_*) = K_{\text{SKI}}^{-1} K_{X,\mathbf{x}_*}$ weighted aggregation of the observations denoted by $\mathbf{y}$. Secondly, it manifests as a weighted summation of the cross-covariances between training and test inputs, denoted by $K_{X,\mathbf{x}_*}$, where the weights are encapsulated in the vector $\boldsymbol{\alpha} = K_{\text{SKI}}^{-1}\mathbf{y}$. Therefore, the equation for the predictive mean can be succinctly written as:

$$\bar{f}_* = \mathbf{y}^\top \mathbf{w}_X(\mathbf{x}_*) = \boldsymbol{\alpha}^\top K_{X,\mathbf{x}_*}.$$

4.5 Comparison Results

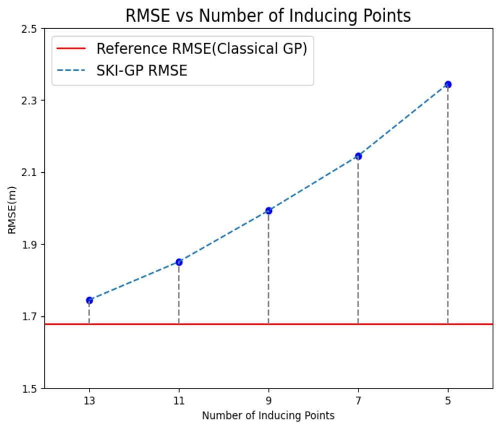
Here are some simulation parameters for the Gaussian process prediction of the drone data set listed below. According to the implementation standard of the ADS-B system, the drone position sampling rate is 2Hz including $3D$ position samples. In terms of window length, considering the data set availability and motion characteristics, we choose a window length equal to 15. Because if the window is too small, the model can not be able to accurately capture the drone's movement dynamic. To optimize the hyperparameter of the GP kernel(length scale, variance), we implement maximum likelihood estimation with a stochastic gradient descent algorithm to achieve it. Finally, we implement the Monte carlo simulation with 60 runs.

Table 4.3: Key simulation parameter for Gaussian process

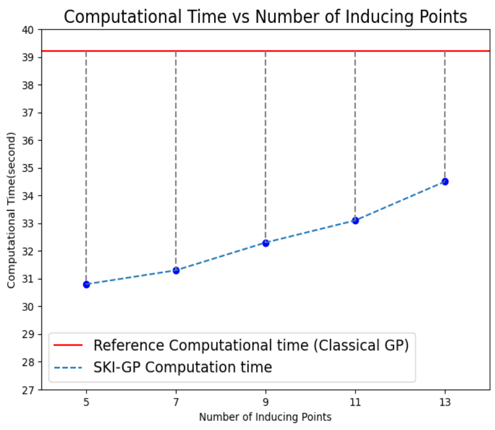
Simulation Parameter	value
Total trajectory position samples	53
Position sampling rate (Hz)	2
Window length number	15
Prediction step size(second)	0.5
Number of Monte Carlo runs	60
hyperparameter optimization iteration	50

Besides the classical GP, we also explore the low-rank approximation prediction performance on GP. We implement SKI GP based on the sliding window Gaussian process as shown before.

The key parameter in the SKI is the inducing points, these points are used to construct an approximation of the full covariance matrix in order to make inference and learning more computationally efficient. To balance the computational complexity and prediction performance, we explore the results with different inducing points set up as Figure 4.8a and Figure 4.8b. Contrasting with the classical GP, incorporating inducing points through SKI indeed reduces computational time. However, this comes at the expense of prediction accuracy, reflected in a higher RMSE. Moreover, there is an evident trend: as the number of inducing points decreases, the computational time benefits become more pronounced, but this is accompanied by a rise in prediction RMSE. Additionally, it's worth noting that our implementation was on a relatively small dataset. Thus, the reductions in computational time observed with SKI might not appear as pronounced as they would when applied to larger datasets.



(a) RMSE vs Number of Inducing Points



(b) Computational Time vs Number of Inducing Points

Moreover, the method outlined above predicts the trajectory using a 2Hz sampling rate. To investigate the predictive performance in more detail, we extended our approach to facilitate predictions at a higher resolution, specifically a 400Hz sampling rate through trajectory interpolation. This adjustment aligns with the original data's sampling rate.

This initiative necessitated adjustments in the testing indices for the GP, a modification readily achievable given the flexible nature of the GP. As illustrated in Figure 4.9, the one-step trajectory prediction varies significantly between the two approaches for the GP method. Given that the ADS-B data maintains a standard 2Hz position sampling rate, predictions at this rate will forecast the position for the next single sample in the test set. In contrast, the 400Hz prediction leverages trajectory interpolation to capture fine-grained details of the trajectory within each step size, forecasting 200 samples for each step in the test set, thereby providing a much richer picture of potential flight paths.

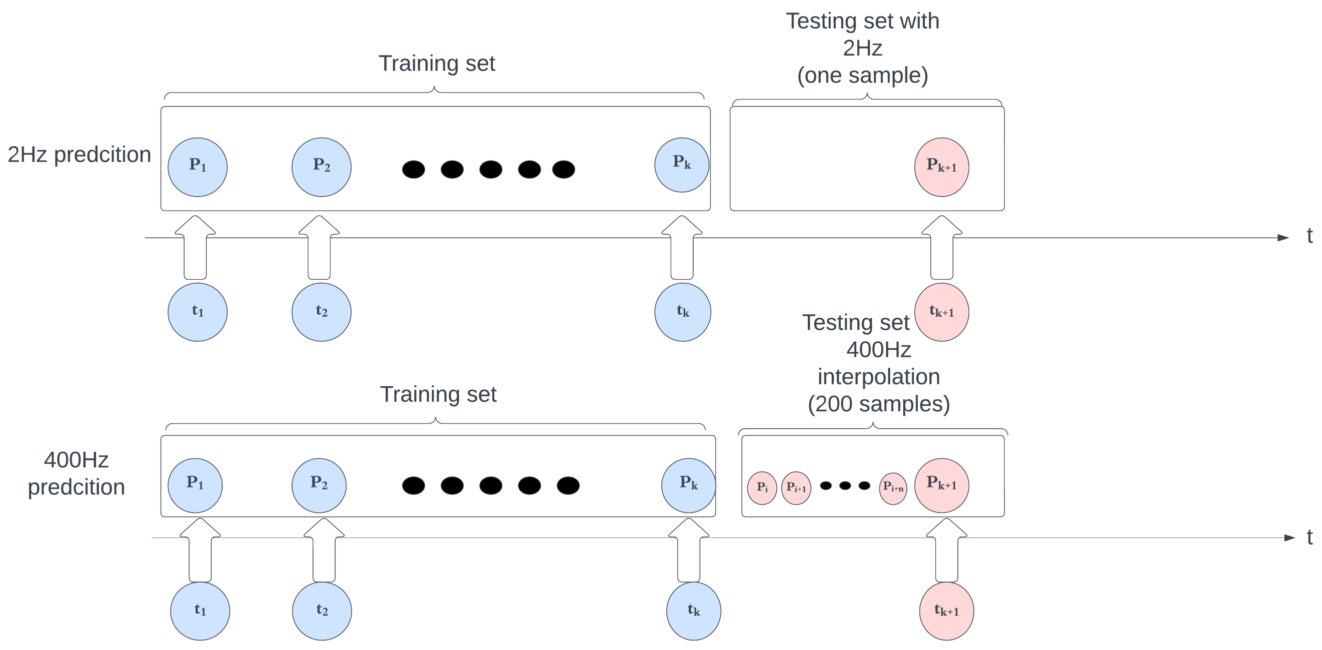
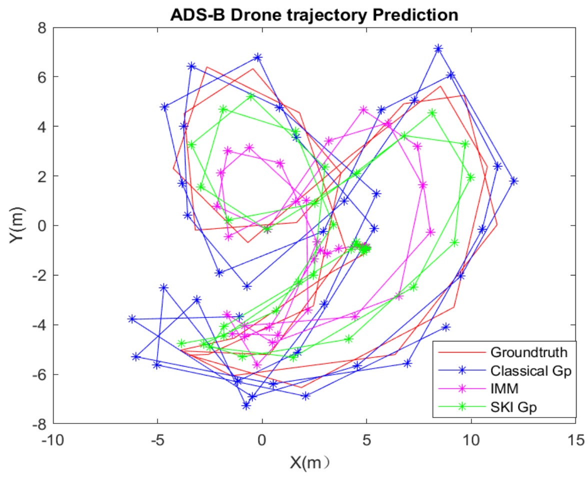


Figure 4.9: Gaussian process Prediction scheme for 2HZ and 400Hz

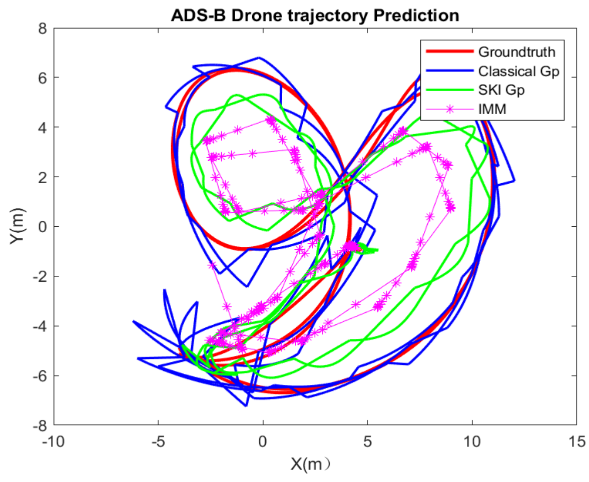
To achieve a 400Hz prediction with trajectory interpolation using the IMM filter with 2Hz data, the process involves altering the filter's prediction step to operate at a finer temporal granularity. The pseudocode is shown as Algorithm3. Initially, the transition matrices are configured to work with a time step of 0.5 seconds, corresponding to the 2Hz sampling rate. To adapt this to a 400Hz sampling, the time step in the transition matrices is reduced to 0.0025 seconds, allowing for the generation of predictions at 200 finer intervals within each 0.5-second step, thereby increasing the resolution to 400Hz. Most of the part of the code is the same as the standard IMM filter. The mixing, model probability, and estimate combination are the same as Algorithm1. However, in the prediction and update step, each prediction cycle now encompasses 200 iterative prediction steps before the filter arrives at the next update step, following which the cycle repeats for the next time step. We implement this method to take advantage of the higher rate to furnish more detailed predictions between updates which could potentially be more representative of the actual trajectory.

Algorithm 3 Interacting Multiple Model (IMM) Algorithm with 400Hz Prediction

```
1: Initialization
2: Initialize the model set $\{\mathbf{F}^{(m)}, \mathbf{H}^{(m)}, \mathbf{Q}^{(m)}, \mathbf{R}^{(m)}\}_{m=1}^{M}$ for $M$ models $\Delta t_{400\text{Hz}} = 0.0025$s
3: Initialize model probabilities $\mu_0^{(m)}$
4: Initialize state estimates $\hat{\mathbf{x}}_0^{(m)}$ and covariances $\mathbf{P}_0^{(m)}$ for each model
5: for $k = 1$ to $N$ do
6:     for $i = 1$ to 200 do
7:         Mixing                                                  ▷ Same as Algorithm 1
8:         High-Rate Prediction and Update Loop
9:         Prediction using combined state estimate: $\hat{\mathbf{x}}_{k,i|k,i-1}^{(m)} = \mathbf{F}\hat{\mathbf{x}}_{k,i-1|k,i-1}$
10:        if $i\%200 == 0$ then
11:            Update step with 2Hz measurement:
12:            Update: Compute Kalman gain and update $\hat{\mathbf{x}}_{k,i|k,i}$ and $\mathbf{P}_{k,i|k,i}$
13:            Update estimate: $\hat{\mathbf{x}}_{k,i|k,i}^{(m)} = \hat{\mathbf{x}}_{k,i|k,i-1}^{(m)} + \mathbf{K}_{k,i}^{(m)}(\mathbf{z_k} - \mathbf{H}^{(m)}\hat{\mathbf{x}}_{k,i|k,i-1}^{(m)})$
14:        end if
15:        Model Probability Update                                ▷ Same as Algorithm 1
16:        Estimate Combination                                    ▷ Same as Algorithm 1
17:        Store prediction: Save $\hat{\mathbf{x}}_{k,i|k,i-1}$ and $\hat{\mathbf{x}}_{k,i|k,i}$ (when available) to output trajectory
18:    end for
19: end for
```



(a) 2Hz planar trajectory prediction comparison



(b) 400Hz planar trajectory prediction comparison

Figure 4.10: Trajectory prediction xy planar

Table 4.4: Prediction Method Comparison for 2Hz

2Hz	IMM	Classical GP	SKI GP(m=9)
RMSE(m)	3.127	1.678	1.993
Computational time(s)	2.68	39.23	32.31

Table 4.5: Prediction Method Comparison for 400Hz

400Hz	IMM	Classical GP	SKI GP(m=9)
RMSE(m)	3.013	1.282	1.671
Computational time(s)	6.14	40.12	33.21

The following Figure 4.10 shows the prediction of xy planar outcomes when compared with the IMM method and the GP method on the drone dataset. We extend the IMM filter from the $2D$ case in chapter 3 to the $3D$ case by including the state vector of z dimension (position, velocity, acceleration). In terms of the analysis on the RMSE, the IMM filter struggles to accurately track the drone's trajectory. It is due to the three models (constant velocity, constant acceleration, and constant turn) cannot capture the drone's dynamics over the ADS-B position transmission period (0.5 seconds). Unlike aircraft, drone motion is more maneuverable, which poses a challenge for the IMM tracking method. In comparison, the Gaussian Process method provides more accurate tracking than the IMM method.

Table 4.4 and Table 4.5 present prediction results for the ADS-B drone trajectory for 2Hz and 400Hz resolution using three distinct methods: the IMM filter, Classical GP, and SKI GP with an inducing point count of 9. The simulations were run on a laptop with the following specifications: AMD Ryzen 7 4800H CPUs and 16GB DDR4, 3200 MHz of RAM. From these results, we can infer the following:

- The IMM filter as a model-based method, generally requires less computational time compared to the GP methods. This efficiency stems primarily from its Markov property, where predictions are based solely on the current state, without being influenced by previous states in the sequence. However, the individual models used in IMM — namely, the CV, CA, and CT models — being linear and implemented through a Kalman filter framework, can not accurately represent the non-linear dynamics of a drone's motion, especially at a relatively low sampling rate of 2Hz. This inability to precisely capture rapid maneuvers or changes in the drone's dynamics leads to a higher RMSE when compared to GP methods, which have the flexibility to learn more complex patterns directly from the data.

- Comparing the Classical GP and structured kernel interpolation (SKI) GP methods, since the SKI employing the Toeplitz method reduces computational complexity to $O(n + m \log m)$ compared to the $O(n^3)$ complexity inherent in the Classical GP. From our results, it is apparent that the SKI GP's low-rank approximation technique facilitates a computational time saving of about 25% compared to the classical GP.

  However, this computational efficiency comes with a trade-off in terms of trajectory prediction accuracy. Utilizing fewer inducing points restricts the model's flexibility, hindering its ability to fully capture the variations in the data, which leads to somewhat flatter predictions along individual dimensions. Consequently, when we inspect the multi-dimension trajectory synthesized from these predictions, we notice a slight shrinkage compared to the trajectory derived from classical GP predictions, indicative of a marginal reduction in trajectory prediction accuracy.

- In addition to the standard 2Hz trajectory predictions, we extended our analysis to explore the potentials of a finer 400Hz resolution through trajectory interpolation, directly contrasting the outcomes with the original 400Hz trajectories generated through three distinct methods. Thanks to the inherent flexibility of the GP. Using the GP method, we noticed that the 400Hz predictions looked much smoother and closely followed the original path, more so than when we used 2Hz predictions. This meant that the GP method was not just creating smoother paths but also more accurate ones, as shown by the lower RMSE values we got.

Conversely, the predictions made using the IMM filter didn't show the same level of smoothness and detail. The issue here is that the IMM filter relies on predefined models that don't fully capture the complex movements of a drone, it could more rely on the constant velocity or constant acceleration model which shows more like a straight-line trajectory. Therefore, despite using a higher resolution of 400Hz, the IMM method couldn't recreate the finer details of the drone's actual path accurately.

4.6 Summary

In this chapter, Gaussian process regression has been implemented on the ADS-B type drone motion trajectory to evaluate the prediction performance along with model-based prediction methods. The main contribution can be summarised as follows:

- Propose a sliding window scheme on the GP method with tuned parameters for trajectory prediction.
- Explore the low-rank approximation approach in the GP method for trajectory prediction, utilizing structured kernel interpolation to save on computational resources.
- Propose trajectory interpolation for both GP and IMM methods, aiming to yield finer-grid trajectories with more detailed information.
- Evaluate and compare the performance of trajectory predictions using the IMM filter, classical GP, and SKI GP on real drone trajectories. The prediction results demonstrate that GP significantly enhances prediction accuracy when compared to the IMM filter, resulting in a 50% reduction in RMSE. Furthermore, the utilization of SKI GP yields a 25% reduction in computation time as compared to the classical GP method.

5 Conclusion and Future Work

5.1 Conclusion

In this thesis, we explore both model-based and data-driven methods for trajectory prediction based on ADS-B data. We implement these methods to analyze both aircraft and drone trajectories to test their performance across different levels of maneuverability and motion trajectories. The trajectory prediction performance is evaluated by the prediction accuracy and computation time. The main contributions are as follows:

- We propose the Kalman model-based algorithm to predict the aircraft trajectory with ADS-B position and velocity information. The interactive multiple models(IMM) have been implemented within three defined model motions (constant velocity, constant acceleration, and constant turn). The parameters eg. noise covariance has been estimated based on the model assumption. From the results evaluated by simulation and real aircraft dataset, the IMM filter obtained high accuracy on the aircraft trajectory compared with other single-model Kalman filters.
- In terms of the data-driven method, we propose the Gaussian process (GP) to quantify both accuracy and uncertainty. The sliding window scheme has been implemented on the GP with tunes parameter for the online trajectory prediction. It could save computational resources with updated data points in a fixed window length.
- We implement the structured kernel interpolation as the low-rank approximation approach in the GP method for trajectory prediction to further reduce the computational complexity. We further investigate the influence of the parameter-inducing number on computation time and accuracy.
- Finally, as a key contribution to our project, we assess and compare the efficiency of trajectory predictions using the IMM filter, classical GP, and SKI GP on ADS-B type real drone trajectories after data preprocessing. Besides, we also extend the predicted trajectory with higher resolution with proposed trajectory interpolation methods. Our findings indicate that the IMM filter, being model-based, operates with reduced computational time but with a compromise in accuracy. On the other hand, the classical GP method shows enhanced accuracy but demands greater computational resources. The SKI GP offers a balance: it slightly compromises accuracy but reduces computational time compared to the classical GP.

5.2 Future Work

Based on the thesis content, there are still some remaining works that could be further investigated.

Firstly, the ADS-B trajectory dataset we implemented is relatively small in size. This limitation is particularly noteworthy for the GP method, as it is data-driven and could benefit from a larger dataset. Additionally, the computational savings from low-rank approximation would be more apparent with a larger training set. Moreover, in our thesis, we implemented the GP method on each dimension of information separately, as our investigation revealed the limited correlation between the dimensions in this dataset. Future research could explore the use of multi-output GP to consider the correlation between different dimensions on larger datasets.

Secondly, in the context of low-rank approximation GP, we employ SKI to facilitate rapid computation. However, several alternative methods, such as sparse spectrum Gaussian processes (SSGP) [28] and sparse pseudo-input Gaussian processes (SPGP) [52], could also be utilized for GP trajectory prediction. For instance, SSGP leverages Fourier features, which can be advantageous for high-dimensional datasets, while SPGP allows for the explicit optimization of inducing points. Consequently, comparing the performance of different low-rank approximation methods on trajectory prediction problems could prove valuable.

Bibliography

[1] W. Aftab and L. Mihaylova. A learning gaussian process approach for maneuvering target tracking and smoothing. *IEEE Transactions on Aerospace and Electronic Systems*, 57(1):278–292, 2020.

[2] V. S. Babu, M. R. Shankar, S. Majumdar, and S. K. Rao. Imm-unscented kalman filter based tracking of maneuvering targets using active sonar measurements. In *2011 International Conference on Communications and Signal Processing*, pages 126–130. IEEE, 2011.

[3] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan. *Estimation with applications to tracking and navigation: theory algorithms and software*. John Wiley & Sons, 2001.

[4] A. Becker. Kalman filter from the ground up, May 2023.

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

[6] H. A. Blom and Y. Bar-Shalom. The interacting multiple model algorithm for systems with markovian switching coefficients. *IEEE transactions on Automatic Control*, 33(8):780–783, 1988.

[7] E. Casado, C. Goodchild, and M. Vilaplana. Identification and initial characterization of sources of uncertainty affecting the performance of future trajectory management automation systems. In *Proceedings of the 2nd International Conference on Application and Theory of Automation in Command and Control Systems*, pages 170–175, 2012.

[8] T. C. T. Christopher. *Analysis of dynamic scenes: Application to driving assistance*. PhD thesis, Institut National Polytechnique de Grenoble-INPG, 2009.

[9] K. Dong, D. Eriksson, H. Nickisch, D. Bindel, and A. G. Wilson. Scalable log determinants for gaussian process kernel learning. *Advances in Neural Information Processing Systems*, 30, 2017.

[10] D. Duvenaud. *Automatic model construction with Gaussian processes*. PhD thesis, University of Cambridge, 2014.

[11] Elizabeth Ciobanu. How Fast Can Drones Fly? (An In-Depth Guide). https://www.droneblog.com/how-fast-drones-fly/.

[12] European Union. COMMISSION IMPLEMENTING REGULATION (EU) 2020/587. https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32020R0587, 2008. Accessed: 2023-07-12.

[13] Flight Radar. An introduction into ADS-B. https://www.flightradar24.com/blog/ads-b/. Accessed: 2023-08-22.

[14] P. Foehn, A. Romero, and D. Scaramuzza. Time-optimal planning for quadrotor waypoint flight. *Science Robotics*, 6(56):eabh1221, 2021.

[15] S. A. Goli, B. H. Far, and A. O. Fapojuwo. Vehicle trajectory prediction with gaussian process regression in connected vehicle environment. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 550–555. IEEE, 2018.

[16] R. Graas, J. Sun, and J. Hoekstra. Quantifying accuracy and uncertainty in data-driven flight trajectory predictions with gaussian process regression. *11th SESAR Innovation Days*, 2021.

[17] G. Gui, F. Liu, J. Sun, J. Yang, Z. Zhou, and D. Zhao. Flight delay prediction based on aviation big data and machine learning. *IEEE Transactions on Vehicular Technology*, 69(1):140–150, 2019.

[18] M. G. Hamed, D. Gianazza, M. Serrurier, and N. Durand. Statistical prediction of aircraft trajectory: regression methods vs point-mass model. In *ATM 2013, 10th USA/Europe Air Traffic Management Research and Development Seminar*, pages pp–xxxx, 2013.

[19] M. Hammer, M. Hebel, M. Laurenzis, and M. Arens. Lidar-based detection and tracking of small uavs. In *Emerging Imaging and Sensing Technologies for Security and Defence III; and Unmanned Sensors, Systems, and Countermeasures*, volume 10799, pages 177–185. SPIE, 2018.

[20] S. M. Hashemi, R. M. Botez, and T. L. Grigorie. New reliability studies of data-driven aircraft trajectory prediction. *Aerospace*, 7(10):145, 2020.

[21] W. Huygen. Ads-b signal integrity and security verification using a coherent software defined radio: Mitigation of the threat of maliciously injected signals in ads-b networks. 2021.

[22] Y. I. Jenie, E.-J. Van Kampen, C. C. de Visser, and Q. P. Chu. Selective velocity obstacle method for cooperative autonomous collision avoidance system for unmanned aerial vehicles. In *AIAA Guidance, Navigation, and Control (GNC) Conference*, page 4627, 2013.

[23] S.-L. Jheng, S.-S. Jan, Y.-H. Chen, and S. Lo. 1090 mhz ads-b-based wide area multilateration system for alternative positioning navigation and timing. *IEEE sensors journal*, 20(16):9490–9501, 2020.

[24] Z. Jia, M. Sheng, J. Li, D. Niyato, and Z. Han. Leo-satellite-assisted uav: Joint trajectory and data collection for internet of remote things in 6g aerial access networks. *IEEE Internet of Things Journal*, 8(12):9814–9826, 2020.

[25] R. E. Kalman. A new approach to linear filtering and prediction problems. 1960.

[26] S. T. Kanneganti, P. B. Chilson, and R. Huck. Visualization and prediction of aircraft trajectory using ads-b. In *NAECON 2018-IEEE National Aerospace and Electronics Conference*, pages 529–532. IEEE, 2018.

[27] J. K. Kuchar and L. C. Yang. A review of conflict detection and resolution modeling methods. *IEEE Transactions on intelligent transportation systems*, 1(4):179–189, 2000.

[28] M. Lázaro-Gredilla, J. Quinonero-Candela, C. E. Rasmussen, and A. R. Figueiras-Vidal. Sparse spectrum gaussian process regression. *The Journal of Machine Learning Research*, 11:1865–1881, 2010.

[29] T.-H. Le, P. N. Tran, D.-T. Pham, M. Schultz, and S. Alam. Short-term trajectory prediction using generative machine learning methods. In *Proceedings of the ICRAT 2020 Conference, Tampa, FL, USA*, volume 15, 2020.

[30] X. R. Li and V. P. Jilkov. Survey of maneuvering target tracking. part i. dynamic models. *IEEE Transactions on aerospace and electronic systems*, 39(4):1333–1364, 2003.

[31] Y. Lin, J.-w. Zhang, and H. Liu. An algorithm for trajectory prediction of flight plan based on relative motion between positions. *Frontiers of Information Technology & Electronic Engineering*, 19(7):905–916, 2018.

[32] I. Lymperopoulos and J. Lygeros. Sequential monte carlo methods for multi-aircraft trajectory prediction in air traffic management. *International Journal of Adaptive Control and Signal Processing*, 24(10):830–849, 2010.

[33] L. Ma and S. Tian. A hybrid cnn-lstm model for aircraft 4d trajectory prediction. *IEEE access*, 8:134668–134680, 2020.

[34] D. R. Maroney, R. H. Bolling, M. E. Heffron, and G. W. Flathers. Experiemntal platforms for evaluating sensor technology for uas collision avoidance. In *2007 IEEE/AIAA 26th Digital Avionics Systems Conference*, pages 5–C. IEEE, 2007.

[35] E. Mazor, A. Averbuch, Y. Bar-Shalom, and J. Dayan. Interacting multiple model methods in target tracking: a survey. *IEEE Transactions on aerospace and electronic systems*, 34(1):103–123, 1998.

[36] C. Menzen, M. Fetter, and M. Kok. Large-scale magnetic field maps using structured kernel interpolation for gaussian process regression. In *2023 26th International Conference on Information Fusion (FUSION)*, pages 1–7. IEEE, 2023.

[37] S. Neemat and M. Inggs. Design and implementation of a digital real-time target emulator for secondary surveillance radar/identification friend or foe. *IEEE Aerospace and Electronic Systems Magazine*, 27(6):17–24, 2012.

[38] Y. Pang and Y. Liu. Probabilistic aircraft trajectory prediction considering weather uncertainties using dropout as bayesian approximate variational inference. In *AIAA Scitech 2020 Forum*, page 1413, 2020.

[39] J. Patrikar, B. Moon, J. Oh, and S. Scherer. Predicting like a pilot: Dataset and method to predict socially-aware aircraft trajectories in non-towered terminal airspace. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 2525–2531. IEEE, 2022.

[40] C. G. Prevost, A. Desbiens, and E. Gagnon. Extended kalman filter for state estimation and trajectory prediction of a moving object detected by an unmanned aerial vehicle. In *2007 American control conference*, pages 1805–1810. IEEE, 2007.

[41] M. Radanovic, M. A. P. Eroles, T. Koca, and J. J. R. Gonzalez. Surrounding traffic complexity analysis for efficient and stable conflict resolution. *Transportation research part C: emerging technologies*, 95:105–124, 2018.

[42] C. E. Rasmussen, C. K. Williams, et al. *Gaussian processes for machine learning*, volume 1. Springer, 2006.

[43] I. Roychoudhury, L. Spirkovska, M. Daigle, E. Balaban, S. Sankararaman, C. Kulkarni, S. Poll, and K. Goebel. Real-time monitoring and prediction of airspace safety. Technical report, 2018.

[44] S. Ruiz, J. Lopez Leones, and A. Ranieri. A novel performance framework and methodology to analyze the impact of 4d trajectory based operations in the future air traffic management system. *Journal of Advanced Transportation*, 2018:1–17, 2018.

[45] S. Sankararaman and M. Daigle. Uncertainty quantification in trajectory prediction for aircraft operations. In *AIAA guidance, navigation, and control conference*, page 1724, 2017.

[46] S. Sawadsitang, D. Niyato, P.-S. Tan, and P. Wang. Joint ground and aerial package delivery services: A stochastic optimization approach. *IEEE Transactions on Intelligent Transportation Systems*, 20(6):2241–2254, 2018.

[47] E. Schulz. *Global Networks, Global Citizens.* Airbus, Blagnac Cedex, France, 2018. Airbus, Ed.

[48] C. E. Seah and I. Hwang. A hybrid estimation algorithm for terminal-area aircraft tracking. In *AIAA guidance, navigation and control conference and exhibit*, page 6691, 2007.

[49] Z. Shi, M. Xu, Q. Pan, B. Yan, and H. Zhang. Lstm-based flight trajectory prediction. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2018.

[50] B. W. Silverman. Some aspects of the spline smoothing approach to non-parametric regression curve fitting. *Journal of the Royal Statistical Society: Series B (Methodological)*, 47(1):1–21, 1985.

[51] SKYbrary. Air Traffic Management. https://skybrary.aero/articles/air-traffic-management-atm. Accessed: 2023-08-23.

[52] E. Snelson and Z. Ghahramani. Sparse gaussian processes using pseudo-inputs. *Advances in neural information processing systems*, 18, 2005.

[53] H. W. Sorenson. Kalman filtering: theory and application. *(No Title)*, 1985.

[54] J. Sun. The 1090 megahertz riddle: a guide to decoding mode s and ads-b signals. *TU Delft OPEN Publishing*, 2021.

[55] U. S. Federal Aviation Administration. Code of federal regulations by federal aviation administration. https://www.ecfr.gov/current/title-14/ chapter-I/subchapter-F/part-91/subpart-C/section-91.225, 2020. Accessed: 2023-07-12.

[56] E. A. Wan and R. Van Der Merwe. The unscented kalman filter for nonlinear estimation. In *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No. 00EX373)*, pages 153–158. Ieee, 2000.

[57] Z. Wang, M. Liang, and D. Delahaye. Short-term 4d trajectory prediction using machine learning methods. In *SID 2017, 7th SESAR Innovation Days*, 2017.

[58] A. Wilson and H. Nickisch. Kernel interpolation for scalable structured gaussian processes (kiss-gp). In *International conference on machine learning*, pages 1775–1784. PMLR, 2015.

[59] A. G. Wilson. *Covariance kernels for fast automatic pattern discovery and extrapolation with Gaussian processes.* PhD thesis, University of Cambridge Cambridge, UK, 2014.

[60] Z. Wu, J. Li, J. Zuo, and S. Li. Path planning of uavs based on collision probability and kalman filter. *IEEE Access*, 6:34237–34245, 2018.

[61] K. Xiao, J. Zhao, Y. He, and S. Yu. Trajectory prediction of uav in smart city using recurrent neural networks. In *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE, 2019.

[62] G. Xie and X. Chen. Efficient and robust online trajectory prediction for non-cooperative unmanned aerial vehicles. *Journal of Aerospace Information Systems*, 19(2):143–153, 2022.

[63] Z. Xu, W. Zeng, X. Chu, and P. Cao. Multi-aircraft trajectory collaborative prediction based on social long short-term memory network. *Aerospace*, 8(4):115, 2021.

[64] X. Yang, J. Sun, and R. T. Rajan. Aircraft trajectory prediction using ads-b data. In *Pre-Proceedings of the 2022 Symposium on Information Theory and Signal Processing in the Benelux*, page 113, 2022.

[65] Y. Zhou, Y. Wan, S. Roy, C. Taylor, C. Wanke, D. Ramamurthy, and J. Xie. Multivariate probabilistic collocation method for effective uncertainty evaluation with application to air traffic flow management. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 44(10):1347–1363, 2014.

[66] J. A. F. Zuluaga, J. F. V. Bonilla, J. D. O. Pabon, and C. M. S. Rios. Radar error calculation and correction system based on ads-b and business intelligent tools. In *2018 International Carnahan Conference on Security Technology (ICCST)*, pages 1–5. IEEE, 2018.