Dynamic management of tracking ressources for hyper-manoeuvring targets

Marion Pilté

THÈSE DE DOCTORAT

de l'Université de recherche Paris Sciences et Lettres
PSL Research University

Préparée à MINES ParisTech

Dynamic management of tracking resources for hyper-manoeuvring targets
Gestion dynamique des ressources de poursuite pour cibles hyper-manœuvrantes

École doctorale n°432

SCIENCE DES MÉTIERS DE L'INGÉNIEUR

Spécialité MATHÉMATIQUES, INFORMATIQUE TEMPS-RÉEL, ROBOTIQUE

COMPOSITION DU JURY :

M Stéphane PUECHMOREL
ENAC, PRÉSIDENT DU JURY

M NICOLAS LE BIHAN
CNRS/GIPSA-LAB, RAPPORTEUR

M DANIEL CLARK
TELECOM SUD-PARIS, EXAMINATEUR

MME AUDREY GIREMUS
IMS BORDEAUX, EXAMINATEUR

M SILVÈRE BONNABEL
MINES PARISTECH, EXAMINATEUR

M FRÉDÉRIC BARBARESCO
THALES, EXAMINATEUR

Soutenue par **Marion PILTÉ**
le 14 Novembre 2018

Dirigée par **Silvère BONNABEL**

MINES ParisTech | PSL RESEARCH UNIVERSITY PARIS

Abstract

The new generation of radars is facing increasingly threatening targets. These radars are asked to perform several tasks in parallel, including surveillance and tracking. To this aim, they can be equipped with staring antennas, so they overcome the constraints induced by the rotation of the antenna. The tracking function of the radar has thus to be upgraded to respond to the double issue of tracking highly manoeuvring targets and managing the resources to balance time between tasks.

In this context, this thesis investigates new means of tracking highly manoeuvring targets. A new target model based on intrinsic coordinates to perform target tracking is proposed. This new target model is expressed in the frame of the target itself, and uses the Frenet-Serret frame, which is well suited to the description of highly dynamic manoeuvres involving normal accelerations that are much larger than earth gravity. A filtering algorithm using the special intrinsic formulation of the target model is developed. This filtering algorithm is very similar in terms of implementation to an Extended Kalman filter, and was implemented using real data. The comparison with standard target models and filtering algorithms show improvements over simple models and algorithms on a large set of trajectories. A new estimation method, relying on the least squares formulation of the smoothing approach, and taking into account kinematic jumps in the trajectory is also developed. This method also shows improvements over a set of common algorithms based on standard manoeuvre detection. Independently, we investigate the issue of update rate adaptation for radar measurements. A very general update rate adaptation algorithm is derived to optimise the time of revisit of each target, allowing to preserve the radar time budget for other tasks simultaneously performed, such as surveillance.

Keywords: State estimation, Target tracking, Kalman filter, Lie groups, Update rate adaptation

Remerciements

Mener à bien une thèse n'est pas chose aisée. Cela n'aurait pu être possible sans l'aide et le soutien de nombreuses personnes, que je tiens à remercier ici.

Tout d'abord, je remercie particulièrement mon directeur de thèse, Silvère Bonnabel, sans qui les travaux présentés dans ce document n'auraient pas eu la même allure. Merci d'avoir relu et minutieusement corrigé les différents articles, journaux, et autres rapports. Merci aussi pour tous les conseils avisés et l'aide précieuse apportée pendant ces trois années. Je remercie également mon encadrant chez Thales, Frédéric Barbaresco, pour m'avoir proposé ce sujet de thèse passionnant, et pour m'avoir donné la possibilité de tester mes méthodes dans diverses unités de Thales avec des problématiques à chaque fois différentes.

Mes remerciements vont également aux ingénieurs de Thales Guillaume Foliard et François Gosselin, assidus dans leurs suivi, avec des remarques toujours pertinentes, et qui m'ont permis d'apporter des améliorations à mes résultats. Merci aussi d'avoir courageusement et patiemment tenté de suivre des explications parfois un peu compliquées au début. Merci aussi à Mathieu Klein, Roch Settineri, ainsi que Loïc Zimmer pour avoir proposé de nouvelles pistes de réflexion pendant la thèse.

Je suis très reconnaissante aux membres du jury d'avoir accepté d'évaluer mon travail de thèse. En particulier, je remercie les rapporteurs Stéphane Puechmorel et Nicolas le Bihan qui ont consacré du temps à la lecture de mon manuscrit, ainsi que Audrey Giremus et Daniel Clark pour la participation au jury de thèse. Enfin, merci à Frédéric Livernet, avec qui j'ai eu l'occasion d'échanger, et de confronter nos points de vue pendant ma thèse.

Que serait une thèse sans collègues doctorants ? Je tiens à remercier tous les doctorants du CAOR, qui m'ont permis de passer trois années de thèse très agréables. Tout d'abord, les doctorants de mon année, Philip, Florent, Xavier, Daniele, et Mathieu, avec qui j'ai partagé les mêmes doutes au cours de ces années. Je n'oublie pas les autres doctorants : Guillaume, Aubrey, Grégoire, Eva, Paul, Martin, Marin, Hassan, Salwa, Houssem, Olivier. Je remercie également les doctorants de Thales, Hour, Fabien, Yanis et Alice pour les pauses passées à argumenter sur la supériorité de nos villes respectives.

Enfin, un mot pour mes parents et mon frère, qui ont commencé à me soutenir bien avant le début de cette thèse, et qui ont continué durant ces années.

Contents

Chapter 1

Introduction

Sommaire

1.1 Résumé en français

Ce travail est le résultat de trois années de thèse CIFRE-Défense menée avec Thales Land and Air Systems, et l'école des Mines ParisTech et en partie financée par la DGA (Direction Générale de l'Armement).

Le mot radar est l'acronyme anglais de "RAdio Detection And Ranging", et est apparu en 1940. L'histoire du radar remonte pourtant au début du vingtième siècle, avec les expériences de Nikola Tesla et de Christian Hülsmeyer. Cependant, les principales avancées des systèmes radars ont eu lieu avec la demande militaire dans les années 1930. Les radars modernes n'ont cessé de se développer depuis lors. Ces radars modernes sont des radars à compression d'impulsion, et ont mené aux radars à balayage électronique actuels. La théorie radar comprend de multiples champs de recherche, comme par exemple la théorie des antennes, le traitement du signal, le traitement numérique. Le travail présenté dans ce document traite principalement du traitement numérique, et notamment de la conception d'algorithmes dédiés aux besoins actuels en termes de performances de suivi de cibles, et de capacité de charge des radars.

Un radar envoie des ondes électromagnétiques dans l'espace et analyse les ondes réfléchies par les objets présents dans son champ de vision. Notamment, la position de la cible peut être connues en coordonnées (distance, azimut, élévation), correspondant aux mesures effectuées par le radar. Ces mesures n'étant pas infiniment précises, il est nécessaire de tenir compte du bruit de mesure, dont on connaît les caractéristiques dans ce système de coordonnées. Le pistage de cible consiste alors à former des "pistes" à partir de ces mesures de position bruitées. Une piste correspond à la trajectoire d'un objet d'intérêt, accompagnée de caractéristiques calculées par le radar, comme par exemple la cinématique de la cible considérée (vitesse, accélération, ...). Dans cette thèse, on s'intéresse à la phase d'estimation de l'état cinématique des cibles. On cherche donc à estimer des paramètres cinématiques choisis à partir de mesures de position bruitées, dans le but de prédire la position de la cible pour la poursuite active, qui nécessite d'orienter le faisceau du radar spécifiquement vers la cible que l'on veut suivre.

Les motivations de ce travail sont multiples. D'une part, de nouveaux types de radars voient le jour, notamment les radars multifonctions à panneaux fixes. Ces radars doivent faire face à de multiples tâches en parallèle. Un gestionnaire de ressources est donc utilisé afin d'ordonner les différentes tâches. Mais chacune des tâches doit également être optimisée en temps de façon à permettre au radar d'effectuer un maximum de tâches. Une partie du travail a donc été consacrée à la recherche d'une méthode pour optimiser le temps utilisé par l'antenne radar pour effectuer la poursuite active. D'autre part, les menaces ont également évolué. En effet, les cibles sont de plus en plus manœuvrantes, et peuvent avoir des accélérations, notamment normales, supérieures à $15g$. Du côté des applications civiles, les drones représentent une menace, avec des comportements différents des cibles usuelles traitées dans le cadre du contrôle de trafic aérien. Les solutions de pistage utilisées actuellement ne sont pas entièrement satisfaisantes pour ces nouvelles menaces, il s'agit donc dans la thèse de proposer d'autres méthodes pouvant répondre de façon plus adaptée aux problèmes de pistage des cibles hyper-manœuvrantes.

Les contributions de la thèse peuvent donc se résumer aux points suivants :

- Modèle de cible : un nouveau modèle de cible est proposé dans ce document. Ce modèle est exprimé en coordonnées intrinsèques, et est conçu pour représenter tous les mouvements possibles d'un objet en 2D ou en 3D.

- Algorithme de filtrage : l'espace mathématique utilisé pour décrire le modèle de cible étant différent des espaces vectoriels habituels, un algorithme de filtrage tirant parti de la formulation du modèle est développé. La formulation de cet algorithme est proche de celle du filtre de Kalman, et le filtre peut donc être implémenté assez facilement sur des simulateurs de pistage existants.

- Algorithme de lissage : afin de prendre en compte des sauts importants dans la dynamique des cibles, un algorithme reposant sur l'optimisation et spécialement conçu pour les mod-

èles à sauts est proposé. L'algorithme de lissage ainsi obtenu détecte les sauts très rapidement, et permet une convergence rapide de l'estimation après un saut.

- Cadence adaptative : cette contribution est indépendante des précédentes, et propose une méthode d'optimisation de la cadence de mesure de pistage, dans le but de réduire le temps du radar consacré à cette tâche.

- Expériences industrielles : des expérimentations de l'algorithme de filtrage présenté ont été réalisées dans différentes équipes de Thales. Ces résultats sont partiellement présentés dans ce document, et conduisent également à une discussion sur l'enjeu du réglage de paramètres des différents algorithmes.

1.2 Foreword

This work has been done in collaboration between Thales Land and Air Systems, the Ecole des Mines ParisTech PSL University, and has been partly supported by the DGA (Direction Générale de l'Armement), part of the French defence ministry. The main application of the work is ground and naval radars. In this introduction, the general principles of radar systems are presented, to have an overview of the main issues addressed in this work. The objectives and the main contributions of the thesis are presented. Finally, the organisation of the document is outlined.

1.3 Radar systems

1.3.1 History

The word Radar is an acronym for *Radio Detection and Ranging*. For a full introduction to the history of radar, please refer to [41] or to [17]. The word appeared in 1940 in the US army, it was in fact a code name. However, radar systems have begun to emerge much earlier. The earliest radar systems were developed to detect objects, without their participation, to avoid collisions for navigation purpose in the early 20th century. First experiments have been held by Nikola Tesla in 1900 and Christian Hülsmeyer in 1904. Yet, the main advances in radar systems are due to the military requirements of the 1930s, and of the second world war. Several countries developed their own radar systems simultaneously.

The modern radars have been developed in the second part of the 20th century. These modern radars are pulse compression radars. Their development have led to nowadays radars, such as scanned array radars. There are now very different types of radars, bistatic or monostatic radars, active or passive radars, primary or secondary radars. These radars have different use, including both military and civilian (for air traffic control) applications.

Radar theory deals with different scientific fields. Antenna theory, signal processing, digital processing, including algorithm design, statistical analysis are the main scientific domains that are used to design a complete radar system. This work is mostly concerned with digital processing, however, some fundamental notions about radar systems are necessary to understand the ins and outs of the applications developed in this document. They are briefly presented thereafter.

1.3.2 General description of radar systems

Numerous books are dedicated to radar, such as [41] or [105], and provide a thorough description of radar systems. A radar sends electromagnetic waves in the air. The wave is reflected when it encounters an object, as represented by figure 1.1. The radar can then analyse the reflected signal. The radar is thus composed of a transmitter and a receptor that are responsible for the creation and the analysis of the electromagnetic wave.

The design of the antenna and the transmitted signal have to be carefully devised, since the signal is polluted with noise in the atmosphere, and the reflected signal has a very lower amplitude

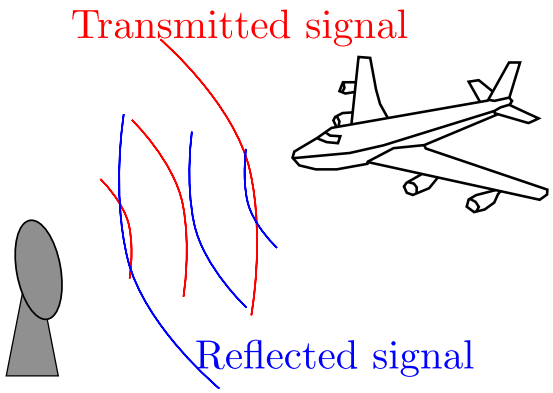


Figure 1.1 – Radar fundamental principle

than the transmitted one, depending on the distance of the object. The energy of the received signal is proportional to the energy of the transmitted signal divided by the distance to the power 4. The receptor is thus very sensitive, and amplifies the received signal, but also adds some noise to it, due to the processing applied.

The primary radars measure the position of the target without any cooperation of the target. The position is usually given as the range (distance from the target to the radar), the azimuth of the target with respect to the radar, and the elevation of the target. Azimuth and elevation are angles. The range, azimuth and elevation coordinates are represented on figure 1.2.

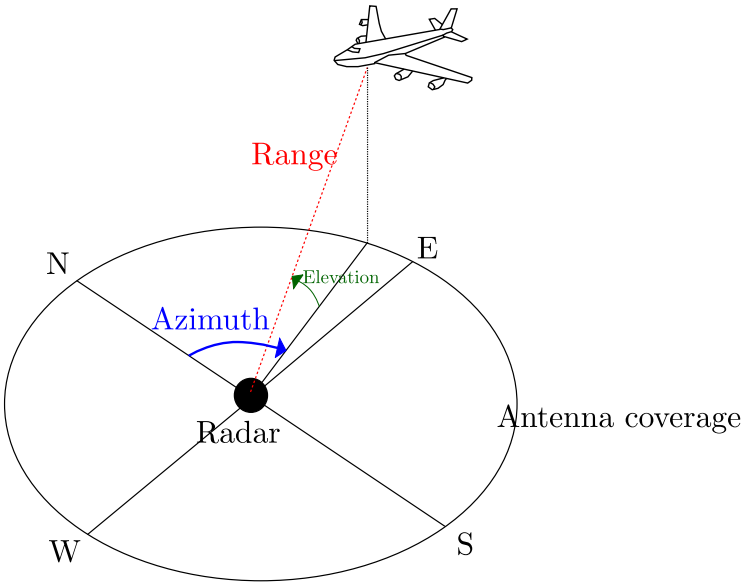


Figure 1.2 – Radar measurements

The range measurement is done by measuring the time the wave takes to go back and forth. This time is then multiplied by the velocity of the wave propagation, assumed to be the light velocity, and divided by two (to take into account the round trip). The angle measurements (azimuth and elevation) are possible because of the directivity of the beam of the radar. The radar sends energy in a given direction, and the measured azimuth and elevation thus lie in a limited angle. This measurement is results in a lesser precision in the final position measurement than the range measurement, and the precision is degraded with the distance of the target to the radar, as will be developed in the radar measurement models in section 2.5.

The secondary radars do not measure directly the position of the target. They ask for the presence of a target, and for its position, and wait for the target answer. The target itself measures its own position. Secondary radars information relies on cooperation. This is the case (most often) for civilian traffic control, but not for military surveillance. Even for civilian applications, a non-

cooperative handling must be performed just in case. The advantage of using secondary radars is the greater precision of the measurements, especially for the altitude information. Secondary radars are also able to receive ADS-B (Automatic Dependent Surveillance - Broadcast) data from an aircraft that provides extensive information about its state, among which its position.

1.3.3 Digital processing

The digital processing unit receives raw information from the receptor. The digital processor is in charge of creating the plots, which are objects that represent a detection, with a radar observation associated to it. A plot represents one object, with diverse attributes, which may include its position, identification, the uncertainty of the measurement, the Doppler measurement, and possibly other useful information.

The atmosphere estimation is used to evaluate the level of noise present in a scene (for example, the presence of clouds in the sky can bring some noise to the signal). The detection test is a logical test of Neyman-Pearson, which is done thanks to a threshold. The Neyman-Pearson test consists of setting the false alarm probability and maximising the detection probability (this is possible because statistical noise models exist). The elementary hits correspond to zeros or ones that indicates the presence or not of an object (or part of an object).

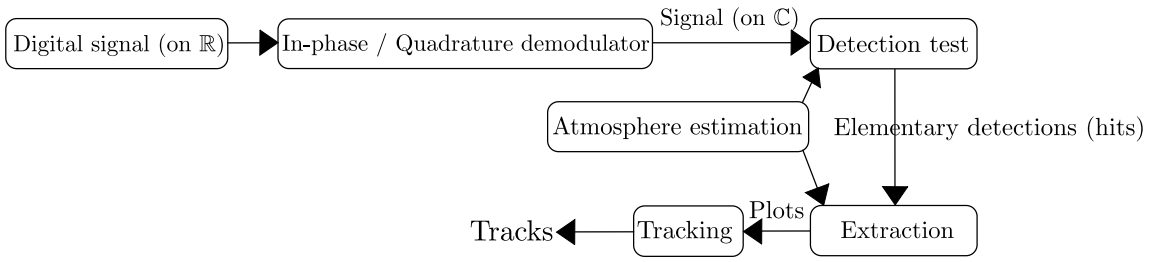


Figure 1.3 – Digital processing

The signal received is digitised on an intermediate carrier (on $\mathbb{R}$). Then, several treatments are applied to process the data and convert it into the description of one detection of one physical object (indeed, a single object is illuminated by several elementary signals, that have to be agglomerated). The elementary treatments are summarised on figure 1.3.

1.3.4 Target tracking

The radar can operate two different tasks: search and tracking. Search consists of detecting the presence of objects and reporting the detections. Tracking has to follow a given object, which is labelled, and infer its kinematic parameters.

The tracker then has to generate tracks from the plots. A track corresponds to the evolution of one given detected object in time. The observations are polluted by noise. There are two types of uncertainties: first the presence or not of an object of interest, when a detection does not correspond to such an object, it is called false alarm; and second, the uncertainties on the measurements, created by the radar itself. Indeed, the observations are not infinitely precise, and the values are polluted by some noise, that must be estimated (it depends on the radar used), and this noise has to be taken into account in the algorithms. The noise comes from different factors, as explained in [40]. There is a part of the error dependent on the signal-to-noise ratio, a random part due to the noise in the final steps of the receiver (but it leads to relatively small errors, and gives mostly a limit on the achievable precision), some bias can be due to the calibration and measurement steps, some errors are due to propagation or uncertainties in the correction of propagation errors, and finally errors due to interferences or clutter are also present. Usually, only the first source of error (dependent on the signal-to-noise ratio) is modelled in the observation equation.

The objective of tracking is to analyse the plots, and output the tracks. A track contains the label of the object, some kinematic description (position, velocity ...), possibly the identification of

the object (aircraft, missile, boat for instance), and other useful information for the radar operator. Some of this information is directly output for the operator, and the rest is only used to feed the tracker. The chain for tracking is represented on figure 1.4.

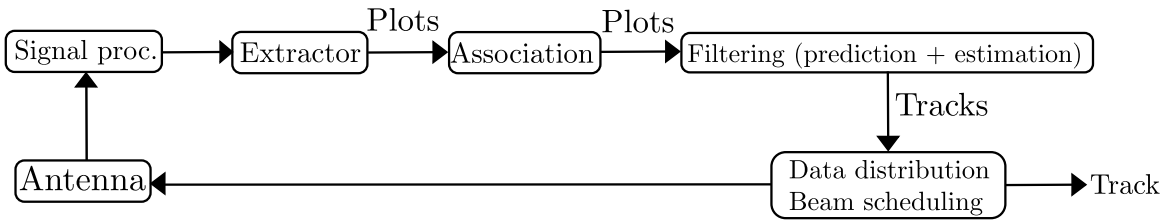


Figure 1.4 – Tracking chain

A track has first to be initialised. This is done during the initialisation phase, and the initial values of the kinematic parameters are computed, with their covariance. The association is the step where the plots are associated to the existing tracks. Indeed, assume there are at one point 6 identified tracks. The extractor gives a plot to the tracker. This plot has to be either associated to one of the 6 tracks, or considered as false alarm. This is the role of the association. The filtering part corresponds to the estimation of the different useful kinematic parameters of one track (for instance position, velocity, or acceleration ...). Finally the beam scheduler manages the order of the tasks that must be done, as detailed in the following. When the track is lost for several time steps, which means the radar cannot recover sight of the target, the track is killed.

There are two modes in which the tracking can be done. The first mode is the track while scan mode (TWS). It is used to perform tracking during the search task. The second mode is Active Tracking (AT). It consists of dedicating the illuminations for one target. This mode can be used when the targets are menacing. The radar beam is oriented specifically in the direction of the target. The beam is thinner, so that the position measurement is more precise. This means that the radar has to know in which direction it should send its beam to find the target. The role of tracking is then both to tell the radar this direction, as shown on figure 1.5, and estimate the kinematics of the target.

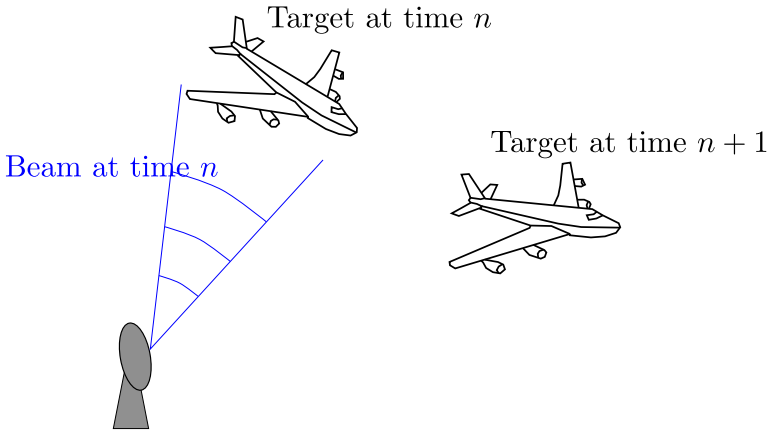


Figure 1.5 – Active track

If the radar antenna is non-rotating, the rate of the measurements can be made fully adaptive, in this case, the update rate adaptation for active tracking is part of the beam scheduling. The update rate is computed by the filtering algorithm, which tells the beam scheduler at what time the target should be next illuminated. The scheduler has to decide which task to perform in which order. There might be several targets in the active track mode, and the track while scan mode has to have enough time budget to detect any target that enters the sight of the radar. Moreover, in case of losses in detection, reacquisition beams or reacquisition patterns are scheduled.

1.4 Motivations and objectives

New types of radars are arising, and specifically fixed antenna, multifunction radars. This new generation of radars require new algorithms, for two reasons. The first one is a necessity, since these radars are designed to perform several tasks in parallel. As concerns the tracking, the surveillance task and the active tracking tasks have to be done simultaneously (almost). This requires a careful management of the tasks the radar has to achieve. To do that, a resource manager (or a scheduler) is used. But each task has to be optimised in itself, in order to save the radar's time budget. The second reason is the new possibilities they offer. Indeed, the observations of the radar are no longer conditioned by the rotation of the antenna, so the observations update rate can be more easily adapted.

Another change in the environment of both military and civilian radars is the evolution of the threats. For the civilian application, drones are becoming more and more present in the air, and the airports are already faced with a drone problem. Indeed, their kinematics are different from the other targets, and there will also be the challenge to track a lot of targets at the same time. In the military world, the targets are becoming always more threatening, adopting unpredictable motions and manoeuvres to make the tracking fail and the radar lose the track.

In this context, new hyper-manoeuvring targets have emerged. Those targets can have accelerations higher than 15g, that cannot be supported by pilots in an aircraft, even during a few tenth of seconds. This concerns mostly missiles that can have really any kind of motion. The tracking solutions used in the radars nowadays are not always satisfying against these targets. In particular, the state estimation task has to be thought over again.

1.4.1 State estimation

State estimation consists of estimating the main kinematic parameters of a target, thanks to partial observations. The radar observes the position of the target, the state estimator will have to predict and estimate the position, but also the velocity, or the acceleration if needed. For active tracking, the state estimation has to be fairly precise, because the challenge is to guide the radar beam for the next observation, to make sure not to lose the target.

Usually, in radar applications, the state estimation is performed using a filtering algorithm, such as the Kalman filter. One of the main objective of the thesis is to provide a state estimation method that is robust, precise, and that is able to keep track of highly-manoeuvring targets.

We identified several drawbacks of the target models usually used in industry, or developed in the literature to tackle the specific problem of estimating the state of highly-manoeuvring targets. These elements will be developed in detail within the document. Briefly, the models tend to be very simplistic, and do not allow enough degrees of freedom for the motion of the target, the challenge was to build a target model which is also very simple, but that is more general and able to consider more class of targets and manoeuvres.

1.4.2 Update rate adaptation

The other task was to provide an update rate adaptation method that is efficient with the proposed estimation algorithm. The role of update rate adaptation is to minimise the radar load for the Active Tracking task. Indeed, to let the radar have enough time for all the tracks in AT, and for the surveillance, the duration for each task has to be optimised.

An adaptation algorithm is already used, but it is in fact not really adapted to the state estimation algorithms that are commonly adopted. The challenge is then to build an algorithm that is general, and that is suitable for all state estimation algorithms.

1.5 Contributions of the thesis

The work has led to contributions both for the state estimation task in itself, and for the adaptation of the update rate.

Target model The present work proposes a novel target model, expressed in intrinsic coordinates. This target model is designed to represent the possible motions of the target directly in its local frame. It uses the Frenet-Serret frame, which can describe any curve in the 2D or the 3D space. Indeed, it appears that the targets very often follow piecewise constant motions, which means that the commands that are applied to it are piecewise constant. The model is meant to represent these commands in the best possible way.

Filtering algorithm The state estimation methods usually applied to other target models are not appropriate for the one we propose. A new filtering algorithm is created to perform filtering for our target model. This algorithm is designed as a geometric version of the extended Kalman filter, and easily implementable for industrial applications.

Smoothing algorithm To account for jumps in the dynamic features governing the target motion such as acceleration or jerk, and to consider accurately the fact that the command parameters may often be modelled as piecewise constant, an optimisation based smoothing algorithm has been constructed for the 2D target model. This smoothing algorithm has the ability to detect jumps in the motion and to react very rapidly when a jump occurs, so that the estimation converges faster after the jump.

Update rate adaptation Besides the preceding contributions related to filtering, this work proposes a new update rate adaptation algorithm that is compatible with any target model. The new algorithm is designed to be very general, and can cooperate with all the estimation algorithms that are presented in this document (the algorithms developed in this work, but also the ones presented as bibliographic references).

Industrial experiments Experiments have been done with Thales, in several teams coordinated by a working group on tracking, both for civilian and military applications on real tracking simulators, a result on a real trajectory is presented in this document. Some other extensive experiments with other algorithms on several different simulated trajectories have been done and the results are also presented in a separate chapter. A discussion on the tuning issue is presented, since it has been one of the major subject of discussion when the algorithm was presented and tested in the different teams.

1.6 Papers published during the thesis

Several publications have been made during this thesis. They are listed here, with the following corresponding chapters:

Conference papers:

- Pilté, M., Bonnabel, S., Barbaresco, F. (2017, June). An innovative nonlinear filter for radar kinematic estimation of maneuvering targets in 2D. In Radar Symposium (IRS), 2017 18th International (pp. 1-10). IEEE, [94]. This paper presents the 2D Frenet-Serret target model, with the 2D Invariant Extended Kalman Filter (IEKF) applied to target tracking, results can be found in chapter 2 and 3.

- Pilté, M., Bonnabel, S., Barbaresco, F. (2017, November). Drone Tracking Using an Innovative UKF. In International Conference on Geometric Science of Information (pp. 301-309). Springer, Cham, [93]. This paper exposes an extension of the IEKF with the 2D Frenet-Serret target model, with the use of the update step of an Unscented Kalman Filter adapted to the Lie group space. Results can be found in chapter 3.

- Pilté, M., Bonnabel, S., Barbaresco, F. (2017, December). Tracking the Frenet-Serret frame associated to a highly maneuvering target in 3D. In Decision and Control (CDC), 2017 IEEE 56th Annual Conference on (pp. 1969-1974). IEEE, [91]. This paper presents a 3D target model based on the Frenet-Serret frame, with the IEKF algorithm to perform estimation. Results can be found in chapters 2 and 3.

- Pilté, M., Bonnabel, S., Barbaresco, F. (2018, June). Maneuver Detector for Active Tracking Update Rate Adaptation. In 2018 19th International Radar Symposium (IRS) (pp. 1-10). IEEE, [95]. This paper proposes a manoeuvre detector based on a particle filtering algorithm designed to track kinematic jumps in the trajectory. This paper was not reproduced in this document.

Journal papers:

- Pilté, M., Bonnabel, S., Barbaresco, F. (2018). Fully-Adaptive Update Rate for Nonlinear Trackers. IET Radar, Sonar & Navigation, [90]. This paper explores the update rate adaptation problem to optimise the radar load for active tracking. The results can be found in chapters 6 and 7.

- Pilté, M., Bonnabel, S., Livernet, F. A novel nonlinear least squares approach to highly maneuvering target target tracking. Submitted upon invitation to Comptes Rendus Physique (Elsevier-Académie des Sciences), 2018. This paper exposes a new method for target tracking, based on a modified smoother that tracks jumps. The results are explained in chapter 5.

1.7 Organisation of the document

This document is organised in three parts:

- **Part I** focuses on state estimation. The construction of a new target model in intrinsic coordinates, adapted to nowadays manoeuvring targets, is presented in chapter 2. This target model is expressed in 2D and in 3D. The two models designed in this chapter will be used throughout the entire document. Then, a filtering algorithm that is adapted to this target model is derived in chapter 3. Firsts results on toy examples are presented. Some extensive tests and comparison with other commonly used target models and filtering algorithms are also presented in chapter 4.

- **Part II** introduces an estimation method that has not been much used in the radar community: smoothing. We take inspiration from the use of optimisation based smoothing methods that have become prevalent in the robotics community for robot localisation and mapping since the beginning of the 2010s. In this work, we have applied it to radar target tracking, and modified the algorithm to suit the problem of tracking manoeuvring and unpredictable targets. A smoothing algorithm is presented, first for a linear target model, and then applied to the 2D target model derived in Part I. This smoothing algorithm is more specifically constructed to track kinematic and/or dynamic jumps in the motion of the targets, to be able to provide accurate estimations right after the jumps.

- **Part III** is dedicated to the update rate adaptation problem. The algorithm of Blackman and Van Keuk, used in industry, is first pedagogically and thoroughly explained in chapter 6. We

thus point out the approximations that were made in their paper [108]. Then a more general algorithm is derived in chapter 7. This algorithm is compatible with any filtering algorithm or target model, and experiments show that the radar load is lower using this new algorithm.

Finally, a synthesis of the work and a discussion on possible future leads are presented in the conclusion of the document, in chapter 8.

Part I

State estimation: target models and filtering algorithms

Chapter 2

Target model in intrinsic coordinates

Sommaire

2.1 Résumé en français : Modèle de cible en coordonnées intrinsèques

L'estimation d'état nécessite deux composants principaux : un modèle d'évolution de cible d'une part, et un algorithme de filtrage (ou d'estimation en général) d'autre part. Ce chapitre est dédié à l'élaboration d'un modèle adapté aux cibles hyper-manœuvrantes. D'une façon générale, on peut identifier trois grands types de modèles de cible :

- Les modèles linéaires, qui sont aussi les plus simples, peuvent être en 2D, ou en 3D, et de dimension variée dans l'état (modèle linéaire de vitesse, ou d'accélération, ou de jerk ...).
- Les modèles non-linéaires plus complexes, qui décrivent des mouvements plus complexes, comme des virages coordonnés par exemple.
- Enfin, les modèles à sauts, qui permettent de décrire des trajectoires qui ont une composante discontinue (par exemple avec des sauts de vitesse ou d'accélération), ou des trajectoires avec des sauts entre modèles.

Il est aussi nécessaire de définir le modèle de mesure, permettant de relier les mesures radar à l'état du système. Souvent, il se résume à donner les équations de passage du repère des mesures radars (distance, azimut, élévation) au repère cartésien fixe dans lequel est exprimé la position de la cible dans l'état. On y ajoute un bruit Gaussien pour modéliser l'imprécision des mesures radar.

Le modèle de cible proposé dans cette thèse utilise le repère de Frenet-Serret, qui permet de décrire n'importe quelle trajectoire en 2D ou en 3D. Le modèle de cible a donc été développé en 2D et en 3D. Deux hypothèses communément admises sont également appliquées pour établir le modèle : d'une part, la cible ne glisse pas (on appelle cette hypothèse la contrainte non-holonome), c'est-à-dire que le vecteur vitesse de la cible est toujours colinéaire au vecteur tangent du repère de Frenet-Serret. D'autre part, on fait une hypothèse sur la cinématique de la cible, dans notre cas, on suppose que la norme de la vitesse, la courbure et la torsion de la trajectoires sont (presque) constantes. Cette seconde hypothèse peut être légèrement relâchée et remplacée par une hypothèse d'évolution linéaire. Cependant, il n'est pas toujours intéressant de complexifier les équations du modèle car cela rend le problème d'estimation plus difficile. Le modèle ainsi obtenu est particulièrement adapté aux virages, contenus dans un plan ou non. C'est une des particularités de ce modèle par rapport à d'autres modèles de virages : il peut décrire un virage avec de la torsion (dans le cas 3D). Du bruit blanc peut être rajouté au modèle afin de tenir compte des écarts entre la réalité et le modèle construit. Ce modèle a une forme particulière, due à la présence de la matrice du repère de Frenet-Serret dans la formulation de l'état. L'état est donc partiellement à valeurs dans le groupe de Lie $SE(2)$ pour le cas 2D, ou $SE(3)$ pour le cas 3D, et partiellement dans l'espace vectoriel $\mathbb{R}^2$ ou $\mathbb{R}^3$. Le modèle nécessite donc une adaptation des algorithmes de filtrage traditionnels, c'est l'objet du chapitre suivant.

2.2 Introduction

To perform state estimation, two main components are needed: a target evolution model, and an estimation algorithm, based on this model an on the radar observations. In this chapter, we concentrate on the target model. The model describes the possible motions of a target, considered as a point object, in space. For example, an overview of missile motion equations is given in [104]. The most basic motion comes from Newton's laws, which can lead to complex equations, specific to one category of targets. However, to perform state estimation, we seek the simplest models as possible, that take into account a large class of targets. The model has to be simple enough, so that the estimation problem is kept tractable. Indeed, very high derivation orders of the position are very difficult to estimate because the radar only measures the position of the target, polluted by noise (and sometimes the Doppler velocity), so variables coming from complex equations are difficult to estimate. However, if the model is too rigid because of its simplicity, and does not allow enough degrees of freedom in the motion of the target, the estimation algorithm might not be able

to keep the track, because of the lack of adequacy between the model and the real trajectory. This can lead to divergence or very poor precision. The difficult part when designing a target model is thus to find a balance between the simplicity and the universality of the model.

The quality of the target model and its adequacy to the real movement are decisive for the tracking. The target models possibilities are virtually infinite. In this chapter, we list the ones that seem most pertinent, and we group them into different categories:

- The simplest models are the linear models, that can be formulated in 2D, in 3D, and of increasing dimension (linear model of velocity, of acceleration, of jerk, ...).
- There are more complex non-linear models, that account for more complex motions, such as coordinated turns.
- Finally, the jumping models can account for trajectories that have a discontinuous component (for instance that have jumps in the velocity or in the acceleration).

To describe the target evolution and derive the so-called target and measurement models, we use a state space representation. The targets are considered as point objects, and target models describe the motion of a point moving in space. This is a general setting in which the state $\mathrm{X}_t \in \mathbb{R}^n$ is the solution of a stochastic differential equation, and can be defined as one realisation of the underlying stochastic process, so it is an element of a vector space (or an element of a more general space, as we will see later), and the observation is a vector $y_t \in \mathbb{R}^m$, and the equations write

$$\dot{\mathrm{X}}_t = f(\mathrm{X}_t, u_t, w_t) \tag{2.1}$$

$$y_t = h(\mathrm{X}_t, u_t, v_t) \tag{2.2}$$

where u_t is an input vector, f is the evolution function, h is the measurement function, w_t, v_t are independent white Gaussian noises. w_t is called the process (or model) noise, and v_t is called the observation (or measurement) noise. The process noise is used to compensate for the differences between the model and the real trajectory of the target. The observation noise is used to model the imprecisions of the radar measurement process. Usually, the noises are chosen to be additive, because it is more convenient for the state estimation algorithm.

Nowadays, the air defence radar industry is facing new challenges with ever increasingly maneuvering targets. Some targets can reach Mach 8 velocities with $30g$ accelerations, and this will increase even more in the future years. The target models that are currently used are not always entirely satisfactory when it comes to track these targets, and new models have to be designed. A way to inject some structure through a motion model into a trajectory that is deliberately trying to make the radar lose track of it, is to resort to physical considerations: the changes in aerodynamic lift and thrust-drag accelerations are limited, and those accelerations can in fact be expected to be piecewise nearly constant.

In the present work, we propose, as a very simple geometric model, to use the Frenet-Serret frame to describe the motion and to assume nearly constant curvature and torsion. This model includes helical motions that are particularly challenging to track. Our model can be related to [31], [16], or more recently [32].

Target models can be expressed either in continuous or in discrete time. In this chapter, we will use one or the other. The discretisation of continuous models is usually quite simple and straightforward, or the estimation algorithms can accommodate a continuous target model. It is also quite widespread to describe the target evolution in continuous time. The radar measurement model is always expressed in discrete time. In this case, we have $\mathrm{X}_n = \mathrm{X}_{t_n}$ in the measurement equation.

This chapter first presents the most well-known target models in literature in section 2.3, and class them into categories, and the target models widely used in industry are detailed in section 2.4. Then, the radar measurement model is described in section 2.5, and finally the target model created during this thesis is developed in section 2.6.

In this work, the state will be denoted by $X_t \in \mathbb{R}^n$ for the continuous time or $X_k \in \mathbb{R}^n$ for the discrete time, where n is the dimension of the state. The measurements will be denoted $y_k \in \mathbb{R}^p$, where $p = 2$ or 3, depending on the dimension of the measurements (2D or 3D). The cartesian position is denoted by $x = \begin{pmatrix} x^1 & x^2 & x^3 \end{pmatrix}^T$.

2.3 State of the art

2.3.1 Model without manoeuvres

A point in space can be described by its position and velocity vectors. For example the vector $X_t = \begin{pmatrix} x_t^1 & \dot{x}_t^1 & x_t^2 & \dot{x}_t^2 & x_t^3 \end{pmatrix}^T$ can be used as a state vector in a Cartesian coordinate system. When the target is considered as a punctual object, the non-manoeuvring motion is described by the fact that the velocities along the first two coordinates x_1 and x_2 are constant, and that the velocity is null for the x_3 coordinate. Indeed, in target tracking, we consider there is no manoeuvre when the target stays in a horizontal plane, see [80]. So this means:

$$\ddot{x}_t^1 = 0$$

$$\ddot{x}_t^2 = 0$$

$$\dot{x}_t^3 = 0$$

Usually, in practice this ideal equation is modified to add a white noise $w(t)$, that accounts for small unpredictable errors, such as turbulences. The equations then become

$$\ddot{x}_t^1 = 0 + w_t^1$$

$$\ddot{x}_t^2 = 0 + w_t^2$$

$$\dot{x}_t^3 = 0 + w_t^3$$

The corresponding state space representation is given by:

$$\dot{X}_t = AX_t + Bw_t \tag{2.3}$$

where $w_t = [w_t^1, w_t^2, w_t^3]^T$ is a white continuous Gaussian noise, with

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \; B = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

The discrete time equivalent for this model is, see for instance [6]

$$X_{k+1} = FX_k + Gw_k \tag{2.4}$$

where

$$F = \begin{pmatrix} 1 & T & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & T & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \; G = \begin{pmatrix} T^2/2 & 0 & 0 \\ T & 0 & 0 \\ 0 & T^2/2 & 0 \\ 0 & T & 0 \\ 0 & 0 & T \end{pmatrix} \tag{2.5}$$

and $w_k = [w_k^1, w_k^2, w_k^3]^T$ is a white discrete Gaussian noise and T is the sampling time. w_k^1 and w_k^2 correspond to noisy accelerations, whereas w_k^3 corresponds to a noisy velocity along the x^3-axis.

If the coefficients of w are uncorrelated, the covariance matrix Q associated to Gw is

$$\mathrm{Q} = \begin{pmatrix} q_1\mathrm{T}^4/4 & q_1\mathrm{T}^3/2 & 0 & 0 & 0 \\ q_1\mathrm{T}^3/2 & q_1\mathrm{T}^2 & 0 & 0 & 0 \\ 0 & 0 & q_2\mathrm{T}^4/4 & q_2\mathrm{T}^3/2 & 0 \\ 0 & 0 & q_2\mathrm{T}^3/2 & q_2\mathrm{T}^2 & 0 \\ 0 & 0 & 0 & 0 & q_3 \end{pmatrix} \tag{2.6}$$

with q_1, q_2, q_3 the variances of w^1, w^2, w^3 respectively.

These models are known to be models with (almost) constant velocity. Adding non-essential components in the state vector (for instance the acceleration or the jerk) would add complexity in the model and decrease the performances for constant velocity trjectories.

2.3.2 Manoeuvre models with decoupled coordinates

The manoeuvres of a target are triggered by the control input u, which is most often unknown to the user. There are then two main solutions to tackle this problem:

- Input estimation: this consists in modelling the input as an unknown but deterministic process, that will be estimated along with the state during the estimation. Such methods are called input estimation methods, see for instance [20]. However, it is hard to model an unknown process, and it often amounts to estimating jumps and values of the input.
- Stochastic process: the other solution is to model the input as a stochastic process. This is more often used in practice, and it comes to using noise models. In the literature, there are three groups of methods:
  1. White noise models: the control input added to the state evolution equation is modelled as a white noise. This includes constant velocity or acceleration models and polynomial models.
  2. Markov process models: the control input is modelled as a Markov process, autocorrelated in time. This includes the Singer model.
  3. Semi-Markovian jumping models: the control input is modelled as a semi-Markovian process with jumps.

A lot of target models assume the coupling between the coordinates is low and can be neglected. It is the case for those for which the control input u is modelled as a random process. In the reminder of this section, the models will be presented in one dimension. The generalisation in 2D or 3D consists only in concatenating the directions x^1, x^2, x^3 to form one state vector.

Let x, $\dot{x}$ and $\ddot{x}$ be the position, velocity and acceleration of a target in one dimension. We have:

$$\dddot{x}_t = a_t \tag{2.7}$$

The possible definitions of a_t will lead to different target models, listed in this section. In the following paragraphs, the state vector will always be $\mathrm{X} = [x, \dot{x}, \ddot{x}]^\mathrm{T}$.

Before going further, let us first give the definition of a stochastic process. This notion will be used several times in the sequel.

Definition 2.1. Stochastic process: it is a parametrised collection of random variables $\{\mathrm{X}_t\}_{t\in\mathrm{T}}$ defined on a probability space. A complete definition and examples are provided in [89]. An intuitive interpretation is to see a stochastic process as a mathematical object that represents the evolution of a random variable. A family of random variables $(\mathrm{X}_t)_{t\in\mathbb{R}^+}$ is a continuous stochastic process, $(\mathrm{X}_k)_{k\in\mathbb{N}}$ is a discrete stochastic process. A basic example is the random walk.

The target state is the solution of a stochastic differential equation, and it is defined as one realisation of the underlying stochastic process.

White noise acceleration model

The simplest model for a target manoeuvre is to consider a model with white noise acceleration. This model assumes that the acceleration of the target $\ddot{x}(t)$ is only white noise, see [6]. This means that

$$a(t) = 0 + w(t)$$

or in other words

$$\ddot{x}(t) = 0 + w(t)$$

The acceleration is thus constant up to a white noise. The difference with the non-manoeuvring model is the level of the noise added: the white noise process w is used to model the effects of a manoeuvre (a switch from a given acceleration to another one for instance). A manoeuvre has the aim to achieve a task and is rarely independent of the state variables in time. The major appeal for this model is its simplicity. It is nonetheless used quite often in practice. This is also referred to as the (almost) constant velocity model.

Almost constant acceleration

The second simplest model is the acceleration model along a Wiener process, see [6]. It assumes the acceleration is a process with independent increments. It is also simply called the (nearly) constant acceleration model. This model has too widely used versions. The first one, the jerk model with white noise, assumes the derivative of the acceleration (the jerk) $\dot{a}_t$ is an independent process: $\dot{a}_t = w_t$. The evolution equation is $\dot{\mathrm{X}}_t = \mathrm{A}\mathrm{X}_t + \mathrm{B}\,w_t$, where

$$\mathrm{A} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}, \ \mathrm{B} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

The discrete equivalent for this model is

$$\mathrm{X}_{k+1} = \mathrm{F}_3\mathrm{X}_k + w_k, \ \mathrm{F}_3 = \begin{pmatrix} 1 & \mathrm{T} & \mathrm{T}^2/2 \\ 0 & 1 & \mathrm{T} \\ 0 & 0 & 1 \end{pmatrix} \tag{2.8}$$

The second version is the acceleration model of the Wiener sequence. The acceleration increment is supposed to be an independent process. An acceleration increment on a time range is the integral of the jerk on this interval. This model is directly expressed in discrete time by

$$\mathrm{X}_{k+1} = \mathrm{F}_3\mathrm{X}_k + \mathrm{G}_3\,w_k, \ \mathrm{G}_3 = \begin{pmatrix} \mathrm{T}^2/2 \\ \mathrm{T} \\ 1 \end{pmatrix} \tag{2.9}$$

The models above are simple but rough. Real maneuvers have almost never almost constant accelerations that are decoupled along the different directions.

Polynomial models

Any target trajectory can be approximated by a polynomial with a given and known precision. It is thus possible to model the movement of the target by a polynomial of degree n in Cartesian coordinates:

$$\begin{pmatrix} x_t^1 \\ x_t^2 \\ x_t^3 \end{pmatrix} = \begin{pmatrix} a_0 & a_1 & \dots & a_n \\ b_0 & b_1 & \dots & b_n \\ c_0 & c_1 & \dots & c_n \end{pmatrix} \begin{pmatrix} 1 \\ t \\ \vdots \\ t^n/n! \end{pmatrix} + \begin{pmatrix} w_t^1 \\ w_t^2 \\ w_t^3 \end{pmatrix} \tag{2.10}$$

with a specific choice of coefficients a_i, b_i, c_i, where (x^1, x^2, x^3) are the position coordinates and (w^1, w^2, w^3) are the corresponding noises.

These polynomials models, of degree n means that the n-th time derivative of the position is (almost) constant. Constant velocity or acceleration models described in the previous sections are particular cases of this model (for $n = 1, 2$ respectively).

In the general case, this model does not seem very satisfying for the tracking application. This type of method is of better use when confronted to smoothing problems, when one needs to fit a smooth curve to a set of data. It is hard to design an efficient method to determine the coefficients a_i, b_i and c_i in the general case.

The Singer model

The white noise models presented earlier are the simplest class of random processes in time. Another class contains the Markov processes, including Wiener processes and the white noises as particular cases.

The Singer model, fully explained in [103], assumes the acceleration of the target a_t is a stationary Markov process of zero mean. The Singer model is based on the assumption that the acceleration is an Ornstein-Uhlenbeck process. See for example [57] for a definition of the Ornstein-Uhlenbeck process. Indeed, this means that each coordinate of the acceleration a^1, a^2, a^3 is an Ornstein-Uhlenbeck process and that the coordinates are mutually independent. We have:

$$\dot{a}_t = -\alpha a_t + w_t, \alpha > 0 \tag{2.11}$$

with w_t a continuous white Gaussian noise, and the autocorrelation of each acceleration coordinate thus writes:

$$\mathrm{E}[a_t^i a_{t+\tau}^j] = \delta_{ij}\Sigma^2 e^{-\alpha\tau}$$

where Σ is the acceleration noise standard deviation and $1/\alpha$ is a manoeuvre time constant. Such a model accounts for the fact that accelerations in one direction tend to last for some time $1/\alpha$, but on average the acceleration of the target is 0. This gives the following discrete evolution equation (2.12) for each position coordinate, and between two time instants k and $k+\mathrm{T}$:

$$x_{k+\mathrm{T}} = \mathrm{F}x_k + w_k \tag{2.12}$$

with

$$\mathrm{F} = \begin{pmatrix} 1 & \mathrm{T} & \frac{\alpha\mathrm{T}-1+e^{-\alpha\mathrm{T}}}{\alpha^2} \\ 0 & 1 & \frac{1-e^{-\alpha\mathrm{T}}}{\alpha} \\ 0 & 0 & e^{-\alpha\mathrm{T}} \end{pmatrix}$$

The process noise covariance matrix Q (from which the Gaussian white noise w_k is drawn) writes $\mathrm{Q}_k = \mathrm{E}[w_k w_k^{\mathrm{T}}]$. The expectation can be explicitly computed as a time integral, as shown in the Appendix I of [103], and finally the matrix writes

$$\mathrm{Q}_k = 2\alpha\Sigma^2 \begin{pmatrix} \frac{\mathrm{T}^5}{20} & \frac{\mathrm{T}^4}{8} & \frac{\mathrm{T}^3}{6} \\ \frac{\mathrm{T}^4}{8} & \frac{\mathrm{T}^3}{3} & \frac{\mathrm{T}^2}{2} \\ \frac{\mathrm{T}^3}{6} & \frac{\mathrm{T}^2}{2} & \mathrm{T} \end{pmatrix}$$

From equation (2.12), we see that:

1. When the manoeuvre duration constant $1/\alpha$ increases to the infinity (which means that $\alpha\mathrm{T}$ decreases), the Singer model becomes a constant acceleration model. This is normal since the deterministic part of the acceleration in the Singer model becomes constant when τ goes to the infinity.

2. When $1/\alpha$ decreases (which means that $\alpha\mathrm{T}$ increases), the Singer model goes to the constant velocity model. In this case, the acceleration becomes a white noise.

The Singer model is thus a mix of a (almost) constant velocity and (almost) constant acceleration model. It is thus more general than one or the other, and is more suited to track manoeuvring targets.

This is why this acceleration Singer model has become popular for target manoeuvre models, and has led to the development of update rate adaptation method, specifically designed for this model, see chapter 6. It also led to the development of other target manoeuvre models.

The Singer target model is by construction an *a priori* model. Indeed, real time information about the target behaviour is not used to tune the model. The parameter Σ can be made adaptive. The adaptation is however limited because in the model, the acceleration has a zero-mean at each time instant. Indeed, for an *a priori* model, it is the natural approximation.

2.3.3 Non-linear models, intrinsic models

Coordinated turn model

The constant turn model assumes the target undergoes a turn manoeuvre, with constant angular velocity, and constant speed, in a 2D plane, see for instance [6], [87]. The state is defined to be $\mathrm{X}_t = \left(x_t^1, \dot{x}_t^1, x_t^2, \dot{x}_t^2, \omega_t\right)$, where ω is the (constant) angular rate. The equations in continuous representation are

$$\dot{\mathrm{X}}_t = \mathrm{A}\mathrm{X}_t$$

with

$$\mathrm{A} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\omega & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & \omega & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

The discrete equation writes

$$\mathrm{X}_{k+1} = \mathrm{F}_k \mathrm{X}_k \tag{2.13}$$

with (if ω is not too small, otherwise a Taylor development can be performed):

$$\mathrm{F}_k = \begin{pmatrix} 1 & \frac{\sin(\omega t)}{\omega} & 0 & -\frac{1-\cos(\omega t)}{\omega} & 0 \\ 0 & \cos(\omega t) & 0 & -\sin(\omega t) & 0 \\ 0 & \frac{1-\cos(\omega t)}{\omega} & 1 & \frac{\sin(\omega t)}{\omega} & 0 \\ 0 & \sin(\omega t) & 0 & \cos(\omega t) & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Of course as always, white Gaussian noise can be added to the evolution equation, to account for differences between the real motion and the model. This model is very useful in practice, as we will see for multiple models. When it is used as a single model, some significant noise should be added to ω and to the velocity, to allow manoeuvres.

Intrinsic coordinate model

The simplest 2D intrinsic coordinate model considers the target as a point mass subject to two accelerations, one tangential, a_{T,t_k} and one normal, a_{N,t_k}, and to the current velocity, see [59]. The accelerations are known. The continuous kinematic state of the target is described by its direction, θ_t (in the trigonometric way from the x^1-axis), its velocity $\dot{s}_t$, and its cartesian coordinates x_t^1 and x_t^2. The state and parameter vectors thus are

$$\mathrm{X}_t = \left(x_t^1, x_t^2, \theta_t, \dot{s}_t\right)^{\mathrm{T}}$$
$$\mathbf{u}_{t_k} = \left[a_{\mathrm{T},t_k}, a_{\mathrm{N},t_k}\right]^{\mathrm{T}}$$

The dynamics of the target are described by the usual differential equations for a curvilinear motion:

$$\begin{aligned} \ddot{s}_t &= a_{\mathrm{T},t_k} \\ \dot{s}_t\dot{\theta}_t &= a_{\mathrm{N},t_k} \\ \dot{x}_t &= \dot{s}_t\cos(\theta_t) \\ \dot{y}_t &= \dot{s}_t\sin(\theta_t) \end{aligned}$$

These equations can then be integrated to get the corresponding discrete time target model. This model can of course be extended to lie in the 3D space, but it can also be augmented to avoid a degeneration problem due to the fact that all values of X_k cannot be reached. The accelerations a_{T,t_k}, a_{N,t_k} are here supposed to be known, or measured, we will see in section 2.3.4 how they can enter the state (or the estimation in general).

2.3.4 Models with jumps

Jump Markov linear systems

Jump Markov linear systems are linear systems with parameters that evolve in time according to a Markov chain, with a finite state space, as explained in [50]. Let $r_k, k = 1,2,\ldots$ be a discrete time Markov chain with known transition probabilities. A linear system with markovian jumps can be modelled in the following way:

$$\mathrm{X}_{k+1} = \mathrm{F}(r_{k+1})\mathrm{X}_k + \mathrm{B}(r_{k+1})u_{k+1} + \mathrm{G}(r_{k+1})w_{k+1} \tag{2.14}$$

$$y_k = \mathrm{C}(r_k)\mathrm{X}_k + \mathrm{A}(r_k)u_k + \mathrm{D}(r_k)v_k \tag{2.15}$$

where u_k is a known input, and w_k and v_k are sequences of white Gaussian independent noises. A jump Markov linear system can be seen as a linear system whose matrices $(\mathrm{A}(r_k),\mathrm{B}(r_k),\mathrm{C}(r_k),$ $\mathrm{D}(r_k),\mathrm{F}(r_k),\mathrm{G}(r_k))$ evolve in time according to a Markov chain with finite state space r_k. Neither the process X_k nor the process r_k are observed, only the noisy measurements y_k are observed. r_k represents in fact pieces of trajectory, on which one model is true. So changes in the Markov chain (when $r_k \neq r_{k-1}$) implies jumps in the trajectory from one model to another.

Variable rate models

In [32], [58] and [59], the authors have designed some models that take into account the temporal structure of a manoeuvring target trajectory. The variable rate models consider that the target motion is deterministic when it is conditioned by a sequence of change-points and manoeuvre parameters. It is similar to Jump Markov Linear Systems, except that we have non-linear equations (also possible for the latter), and more importantly, continuous time changes.

Tracking is the step where the kinematic state of a target is estimated (its position, velocity, ...) from a set of noisy or incomplete observations. The state of the target is continuous. However, as we saw in the last section, a lot of tracking systems model the dynamics of a target as a discrete Markov process. This hypothesis is indeed quite simple: when the state is discretised at the instants of the observations, we come to consider a Hidden Markov Model (HMM), which can then be used to design a standard Kalman filter or a particle filter. HMM methods are explained in [34]. The drawback of such an hypothesis is that the dynamics of the system may not be as accurate as using a continuous-time model (the equations of motion of a continuous system being continuous in time).

The principle of variable rate models is the following: consider a general model from 0 to T, time between which some observations $\{y_1,\ldots,y_\mathrm{N}\}$ are made at time instants $\{t_1,\ldots,t_\mathrm{N}=\mathrm{T}\}$. During this time period, an unknown number of changes, K, occur at time instants $\{\tau_0=0,\tau_1,\ldots,\tau_\mathrm{K}\}$, each change is associated to change parameters, $\{u_0,u_1,\ldots,u_\mathrm{K}\}$. We assume the trajectory can be entirely recovered, knowing the initial conditions at τ_k and the parameters u_k. Pairs $\{\tau_k,u_k\}$ are

elements of a Marked Point Process (MPP). The hidden state is a continuous process called X_t. Discrete sets containing multiple values will be called $y_{1:n} = \{y_1, \dots, y_n\}$. The variable rate model is an example of a hybrid dynamic system, used to model at the same time continuous and discrete behaviours.

The objective for inference will be to estimate the sequence of change-points until the current time t_n: $\theta_n = \{\tau_j, u_j, \forall j : 0 \leq \tau_j < t_n\}$. It will also be useful to define a variable for the changes that occur in the time interval $[t_{n-1}, t_n) : \theta_{n\backslash n-1} = \{\tau_j, u_j, \forall j : t_{n-1} \leq \tau_j < t_n\}$.

To simplify the notations, let us introduce the following notations, to keep in mind the most recent changes:

$$K_t = \max(k : \tau_k < t)$$
$$K_n = K(t_n)$$

The sequence of changes is supposed to be a Markov process:

$$\{\tau_k, u_k\} \sim p(u_k|\tau_k, \tau_{k-1}, u_{k-1})p(\tau_k|\tau_{k-1}, u_{k-1})$$

Now, it is possible to write the *a priori* density for the sequence of changes, $p(\theta_n)$ and the extent of the sequences $p(\theta_{n\backslash n-1}|\theta_{n-1})$. The dynamics of the state is governed by a differential equation which depends on the most recent change-point.

$$\dot{X}_t = f(X_t, \tau_{K_t}, u_{K_t}) \tag{2.16}$$

With a new sequence $\{X_0, X_1, \dots, X_K\}$, which is the value of the state at each change-point (*i.e.* x_{τ_k}) and assuming that an analytical solution exists, a transition function can be derived:

$$X_t = f(X_{K_n}, u_{K_n}, \tau_{K_n}, t),\ \tau_{K_n} \leq t \leq \tau_{K_{n+1}} \tag{2.17}$$

2.3.5 Lie group based models

These late years, work has been done to model the motion of objects with rigid transformations and the associated appropriate spaces. Especially in the robotic field, Lie groups have appeared to be the right spaces to describe the possible moves of a robot. Indeed, one of the most well-known Lie groups, SE(2), represents all possible translations and rotations in 2D. The definition of Lie groups and the main results used in this document are provided in appendix A. This thus appears the right frame to work in. The works of [10], or [36] are examples of Lie group models developed in robotics.

More recently, target models on Lie groups have emerged in the radar target tracking community. This includes the very recent (2018) work of [42]. In this paper, the authors use a coordinated turn target model, and show it can be embedded in a Lie group setting. The state is set to be $X_k = \begin{pmatrix} x_k & \theta_k & \omega_k & u_k \end{pmatrix}^T$, with x_k the Cartesian position, θ_k the orientation of the target, ω_k the angular rate, and u_k the translational speed. It can be represented in $G = SE(2) \times \mathbb{R}^2$, with the following 6×6 matrix:

$$g_k = \begin{pmatrix} \cos\theta_k & -\sin\theta_k & x_k^1 & 0 & 0 & 0 \\ \sin\theta_k & \cos\theta_k & x_k^2 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & u_k \\ 0 & 0 & 0 & 0 & 1 & \omega_k \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

The authors model then the dynamics with the following equation on Lie groups

$$g_{k+1} = g_k \exp_G\left(\mathscr{L}_G(\Omega(g_k) + q_k)\right) \tag{2.18}$$

where q_k is a white Gaussian noise in the Lie algebra, and $\Omega : G \to \mathbb{R}^p$ is a non-linear function. The Lie group and Lie algebra definitions and notations are explained in appendix A. In the coordinated turn case, it writes

$$\Omega(g_k) = \begin{pmatrix} u_k T \\ 0 \\ \omega_k T \\ 0 \\ 0 \end{pmatrix}$$

with T the time between two predictions. It is easy to verify that (2.18) is equivalent for the discrete case to the coordinated turn equation (2.13). As we will see in section 2.6, this is closely related to our model, which was introduced before, see [91].

2.4 Radar industrial tracking models

In industry, the target models tend to be the simplest ones, notably for simplicity of use and implementation, robustness, and safety reasons.

2.4.1 3D target model

One of the simplest, but one of the most used target model in industry is the 3D constant velocity model with white Gaussian noise. The state is composed of the cartesian position and velocity, as in (2.19). The evolution of the state is linear, with constant velocity. A white Gaussian noise is added independently on the three components of the velocity, as in (2.20).

$$X_t = \left(x_t^1, x_t^2, x_t^3, \dot{x}_t^1, \dot{x}_t^2, \dot{x}_t^3\right)^T \tag{2.19}$$

$$\dot{X}_t = FX_t + w_t \tag{2.20}$$

where w_t is a white Gaussian noise. w_t has independent components, and it is used as in the constant acceleration model of section 2.3.2: to account for the possible manoeuvres of the target, and F is defined as

$$F = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

This model is the simplest 3D model that can be designed, it is in fact a white noise acceleration model. However, it is not the best model to track simple manoeuvres, such as constant turns for instance, because the model is not appropriate, and somehow very reductive. One trick so to say, is to use a process noise adaptation heuristic, known as Castella's noise adaptation, described in [35], and further developed in section 4.3.2. The idea is that the process noise is increased during manoeuvres, and decreased during constant velocity motions.

2.4.2 Multiple target models

Another widely used model in industry is to use a Jump Markov Linear model, as in [18]. In practice, several models are designed, and the estimation algorithm is then in charge to tell in which model the target most probably evolves. This is usually called a Multiple Model.

For example one frequently used association of models can be:

- Constant velocity model

- Constant horizontal turn model (a 2D motion)
- Constant vertical turn model (a 2D motion)
- Constant acceleration model

The equations for each of the four models can be found in [102]: for each model, the state is denoted X_k. The discrete time equation is

$$X_{k+1} = F_k X_k + G_k w_k \tag{2.21}$$

where F_k and G_k are explicitly given for each model below, and $w_k \in \mathbb{R}^3$ is a white Gaussian noise. We will also denote dt the duration between two predictions.

Constant acceleration model

The state X_k is of dimension 9 and contains the Cartesian position x, the Cartesian velocity v and the Cartesian acceleration a. The evolution matrix F_k of (2.21) is given by

$$F_k = \begin{pmatrix} 1 & 0 & 0 & dt & 0 & 0 & dt^2/2 & 0 & 0 \\ 0 & 1 & 0 & 0 & dt & 0 & 0 & dt^2/2 & 0 \\ 0 & 0 & 1 & 0 & 0 & dt & 0 & 0 & dt^2/2 \\ 0 & 0 & 0 & 1 & 0 & 0 & dt & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & dt & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & dt \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \; G_k = \begin{pmatrix} dt^2/2 & 0 & 0 \\ 0 & dt^2/2 & 0 \\ 0 & 0 & dt^2/2 \\ dt & 0 & 0 \\ 0 & dt & 0 \\ 0 & 0 & dt \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Constant Horizontal Turn

In this case, the target is supposed to perform a constant turn in the horizontal Cartesian (x, y)-plane. The state X_k is composed of the Cartesian position, the Cartesian velocity, and the angular velocity ω_k.

If the angular velocity ω_k is not too small, and letting $c = \cos(\omega_k dt)$ and $s = \sin(\omega_k dt)$, the evolution matrices write

$$F_k = \begin{pmatrix} 1 & 0 & 0 & s/\omega_k & (c-1)/\omega_k & 0 & 0 \\ 0 & 1 & 0 & (1-c)/\omega_k & s/\omega_k & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & dt & 0 \\ 0 & 0 & 0 & c & -s & 0 & 0 \\ 0 & 0 & 0 & s & c & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \; T_k = \begin{pmatrix} dt^2/2 & 0 & 0 \\ 0 & dt^2/2 & 0 \\ 0 & 0 & dt^2/2 \\ dt & 0 & 0 \\ 0 & dt & 0 \\ 0 & 0 & dt \\ 0 & 0 & 0 \end{pmatrix} \tag{2.22}$$

To define properly G_k, we need the following rotation matrix, and the heading and the speed in the horizontal plane $h = \text{atan2}(v_y, v_x)$, $sp = \sqrt{v_x^2 + v_y^2}$.

$$R_k = \begin{pmatrix} \cos(h) & -\sin(h) & 0 & 0 & 0 & 0 & 0 \\ \sin(h) & \cos(h) & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \cos(h) & -\sin(h) & 0 & 0 \\ 0 & 0 & 0 & \sin(h) & \cos(h) & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1/sp \end{pmatrix} \tag{2.23}$$

and G_k can now be expressed as

$$G_k = R_k T_k$$

Constant Vertical Turn

The target has a constant turn in the vertical Cartesian plane, it is again a 2D motion. The state X_k is composed of the Cartesian position, the Cartesian velocity, and the angular velocity ω_k. We suppose again that the angular velocity is not too small. Let R_z be the change of basis from (X_{long}, Z, Y_{lat}) to (X_{long}, Y_{lat}, Z):

$$R_z = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Let us call $\tilde{F}_k$ and $\tilde{T}_k$ the evolution matrices already defined for the constant horizontal turn in (2.22), and R_k the rotation matrix defined by (2.23). Then the full matrices for the constant vertical turn write:

$$F_k = R_k R_z \tilde{F}_k R_z^T R_k^T$$

$$G_k = R_k \tilde{T}_k$$

Constant Velocity Model

The state is composed of the Cartesian position and velocity and the angular velocity. The evolution matrices are defined by

$$F_k = \begin{pmatrix} 1 & 0 & 0 & dt & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & dt & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & dt & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \; G_k = \begin{pmatrix} dt^2/2 & 0 & 0 \\ 0 & dt^2/2 & 0 \\ 0 & 0 & dt^2/2 \\ dt & 0 & 0 \\ 0 & dt & 0 \\ 0 & 0 & dt \\ 0 & 0 & 0 \end{pmatrix}$$

Then the state is a weighted sum of these models, and the weights are computed thanks to the estimation algorithm. It is not strictly a Jump Markov Linear model as in section 2.3.4 since we do not completely switch between models, but it is inspired by this type of models.

The equations for each of the four models can be found in [102], and they have all been described in 2.3. Usually, the process noise added on the constant acceleration model is tuned quite high. Indeed, this model is used to consider all motions that do not enter the three other models.

2.5 Radar measurement models

After having presented the different classes of possible target models, we can now present the radar measurement (also called observation) models. The measurement equation also plays an important role in the quality of the tracking. The measurement noise, which depends on the radar and on the processing that is done until the information received is converted into plots, has to be modelled carefully.

The simplest model is to consider Cartesian position measurements y_n. The information transmitted by an aircraft, via the ADS-B, to the air traffic control is the Cartesian coordinates for instance. The ADS-B (for Automatic Dependent Surveillance - Broadcast) is a transmitter that broadcasts position information to the receiver. White Gaussian noise is added independently

on each observation coordinate, in order to model the measurement noise. So the observation equation writes

$$y_n = \begin{pmatrix} x_n^1 \\ x_n^2 \\ x_n^3 \end{pmatrix} + \begin{pmatrix} v_n^1 \\ v_n^2 \\ v_n^3 \end{pmatrix} \tag{2.24}$$

with $v_n = [v_n^1, v_n^2, v_n^3]^{\mathrm{T}}$ is a white Gaussian noise, with independent coordinates. Usually, ADS-B informations are fairly precise (in any case, much more precise than any primary radar observations), so the measurement noise is quite low. Unfortunately, this requires a cooperative aircraft, so another method has to be used for military applications. Moreover, this does not truly correspond to radar observations.

For military applications, only primary radars are used, due to the presence of non-cooperative targets, and they do not measure the Cartesian position of the target, but rather its position in range r, azimuth az, and elevation el coordinates. Sometimes, the radial velocity (also called range rate, or Doppler) is added to the observation equation. The coordinates are represented on figure 2.1. The transformation that converts Cartesian position to range, azimuth and elevation is:

$$\begin{aligned} r &= \sqrt{x_1^2 + x_2^2 + x_3^3} \\ az &= \arctan(x_1 / x_2) \\ el &= \arcsin(x_3 / r) \end{aligned}$$

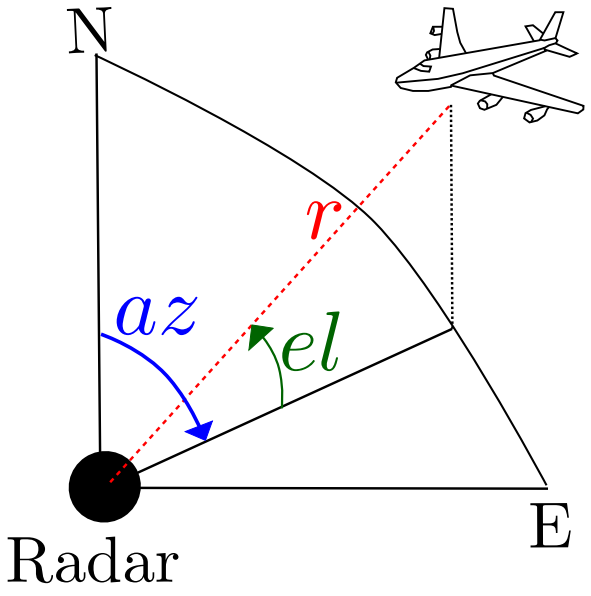


Figure 2.1 – Range, azimuth and elevation coordinates

One must then have an appropriate noise model, which has to be realistic. In the range, azimuth, elevation and Doppler measurements case, [40] gives approximations to model the noise for the different coordinates. These approximations can be refined, when knowing the radar's characteristics, but it represents well the order of magnitude of the amplitude of the noise in a first approximation. Let S/N be the signal-to-noise ratio. S/N can be approximated by $\mathrm{S/N} \approx k \times \mathrm{RCS}/r^4$ at each instant, with k a constant depending on the radar, RCS, the radar cross-section depending on the target, and r the range.

The noise for the range measurement is Gaussian, with a standard deviation given by

$$\sigma_{\mathrm{Range}} = \frac{\Delta \mathrm{R}}{\sqrt{2(\mathrm{S/N})}} \tag{2.25}$$

where ΔR is the range resolution of the radar.

The noise for the azimuth measurement is also assumed Gaussian and dependent on the ratio S/N, its standard deviation can be modelled by

$$\sigma_{az} = \frac{\theta}{k_M \sqrt{2(S/N)}} \tag{2.26}$$

where θ is the beam-width and k_M is a dimensionless parameter. This formula is also used for the elevation measurement noise, with a different θ. An angular noise cannot be Gaussian strictly speaking, but it seems a good approximation at first.

Finally, the standard deviation for the Doppler measurement noise is

$$\sigma_{Dop} = \frac{\Delta V}{\sqrt{2(S/N)}} \tag{2.27}$$

where ΔV is the velocity resolution. See for example [103] or [53] for tracking problems using the Doppler measurement. In this document however, the Doppler measurement will not be used in the measurement equations.

The noise increases with the distance of the target to the radar. Indeed, for angle measurements, the width of the beam is greater when the target is far away, and for the range measurement, the precision of the velocity and the time measurement lead to lesser precision when the distance increases.

2.6 New target model in intrinsic coordinates

In order to track manoeuvring targets, a model representing 3D motions is needed. Indeed, during a manoeuvre, the target does not necessarily stay in a 2D plane. This has to be made possible directly in a single model, instead of using several models in parallel representing 2D motions. Moreover, we want to express the manoeuvres directly in the frame of the target, to represent them in a more accurate way, but in a simple manner at the same time. To this aim, intrinsic coordinates have been preferred. Two assumptions have been made to design the target models:

1. There is no sliding during the turns, which means that the velocity vector is always co-linear to the vector tangent to the trajectory.

2. The commands are piecewise constant. This idea comes from the example of an aircraft, where the pilot orders piecewise constant commands to the aircraft. To some extent, this also applies to highly manoeuvring targets, such as missiles, which tend to concatenate manoeuvres, each being based on constant control inputs. Thus we will have a simple model, yet realistic.

This leads to the use of the Frenet-Serret frame to represent the target motions. More precisely, the speed and the turn rate of the target are assumed to be piecewise constant. All these parameters are the parameters described by the evolution of the Frenet-Serret frame, as will be shown in the next section. In fact, in the equations they are considered to be constant, and a white noise is added to account for the jumps.

We first present a 2D model (some tracking applications use only 2D models), and then we will present the 3D model, that is a generalisation of the 2D model.

2.6.1 2D target model

In the 2D case, the motion is similar to the one of a non-holonomous car. This model represents in fact a coordinated turn model, with a slightly different state vector, this model is also used in [42], as explained in section 2.3.4. The state vector for our model is $X_t = \begin{pmatrix} \theta_t & x_t^T & \omega_t & u_t \end{pmatrix}^T$ with θ_t the orientation of the target, *i.e.* the angle of the velocity vector with respect to the first coordinate

axis, $x_t = \begin{pmatrix} x_t^1 & x_t^2 \end{pmatrix}^{\mathrm{T}}$ its Cartesian position, ω_t its angular velocity and u_t the norm of its velocity. Let us call $w_t = \begin{pmatrix} w_t^\theta & w_t^x & w_t^\omega & w_t^u \end{pmatrix}^{\mathrm{T}}$ a white Gaussian noise. The vectorial equations are

$$\begin{cases} \dfrac{d}{dt}\theta_t = \omega_t + w_t^\theta \\ \dfrac{d}{dt}x_t = \begin{pmatrix} \cos(\theta_t) \\ \sin(\theta_t) \end{pmatrix}(u_t + w_t^x) \\ \dfrac{d}{dt}\omega_t = 0 + w_t^\omega \\ \dfrac{d}{dt}u_t = 0 + w_t^u \end{cases} \tag{2.28}$$

In this 2D case, the (almost) constant parameters are the angular velocity ω_t and the norm of the velocity u_t.

The model seems quite simple, but the equation giving the derivative of x_t is not linear, owing to the presence of the transcendental functions sine and cosine. The cosine and sine functions can be avoided by introducing the rotation matrix

$$\mathrm{R}_t = \begin{pmatrix} \cos\theta_t & -\sin\theta_t \\ \sin\theta_t & \cos\theta_t \end{pmatrix} \in \mathrm{SO}(2)$$

The state evolution will still be non-linear, but the formulation is more friendly, and it will lead to a special type of estimation algorithm that is generalisable in 3D, as we will see in section 3.5.

The evolution of this rotation matrix writes

$$\frac{d}{dt}\mathrm{R}_t = \mathrm{R}_t(\Omega_t + (w_t^{\mathrm{R}})_\times)$$

where

$$\Omega_t = \begin{pmatrix} 0 & -\omega_t \\ \omega_t & 0 \end{pmatrix}$$

$w_t^{\mathrm{R}} \in \mathbb{R}$ is a white Gaussian noise. The notation $(.)_\times$ is explained extensively in the appendix A. It is linked with the matrix Lie group SO(2), it represents a skew-symmetric matrix belonging to the Lie algebra $\mathfrak{so}(2)$. Here,

$$(w_t^{\mathrm{R}})_\times = \begin{pmatrix} 0 & -w_t^{\mathrm{R}} \\ w_t^{\mathrm{R}} & 0 \end{pmatrix}$$

The evolution of the position henceforth writes

$$\frac{d}{dt}x_t = \mathrm{R}_t\left(\begin{pmatrix} u_t \\ 0 \end{pmatrix} + \begin{pmatrix} w_t^x \\ 0 \end{pmatrix}\right)$$

A more appropriate form to represent these equations is to use the matrix Lie group of 2D rotations and translations, SE(2), which can be alternatively seen as $\mathrm{SE}(2) = \mathrm{SO}(2) \ltimes \mathbb{R}^2$. The definition and the principle results for SE(2) and matrix Lie groups in general can be found in appendix A, and a more thorough description of Lie groups can be found in [3] and [54]. Let us introduce the matrix χ_t, which represents the rotation and translation part of the state, and the vector ζ_t which represents the angular velocity and the norm of the velocity:

$$\chi_t = \begin{pmatrix} \cos\theta_t & -\sin\theta_t & x_t^1 \\ \sin\theta_t & \cos\theta_t & x_t^2 \\ 0 & 0 & 1 \end{pmatrix}, \quad \zeta_t = \begin{pmatrix} \omega_t \\ u_t \end{pmatrix} \tag{2.29}$$

The matrix χ_t lies in the matrix Lie group SE(2), and ζ_t is in $\mathbb{R}^2$.

The equations can then be written using this state formulation:

$$\frac{d}{dt}\chi_t = \chi_t\left(\nu_t + \mathscr{L}_{\mathfrak{se}(2)}(w_t^\chi)\right), \quad \frac{d}{dt}\zeta_t = 0 + w_t^\zeta \tag{2.30}$$

where the notation $\mathscr{L}_{\mathfrak{se}(2)}(.)$ is fully explained in appendix A, and $w_t^\chi = \begin{pmatrix} w_t^\theta & w_t^x & 0 \end{pmatrix}^{\mathrm{T}}$ and $w_t^\zeta = \begin{pmatrix} w_t^\omega & w_t^u \end{pmatrix}$, and ν_t is the evolution matrix defined by

$$\nu_t = \begin{pmatrix} 0 & -\omega_t & u_t \\ \omega_t & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \in \mathfrak{se}(2)$$

$\mathfrak{se}(2)$ is called the Lie algebra associated with the Lie group SE(2), it is also defined in appendix A. $\mathscr{L}_{\mathfrak{se}(2)}(w_t^\chi) \in \mathfrak{se}(2)$ writes

$$\mathscr{L}_{\mathfrak{se}(2)}(w_t^\chi) = \begin{pmatrix} 0 & -w_t^{\mathrm{R}} & w_t^x \\ w_t^{\mathrm{R}} & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

To sum up, the state is

$$\chi_t = \begin{pmatrix} \cos\theta_t & -\sin\theta_t & x_t^1 \\ \sin\theta_t & \cos\theta_t & x_t^2 \\ 0 & 0 & 1 \end{pmatrix}, \ \zeta_t = \begin{pmatrix} \omega_t \\ u_t \end{pmatrix}$$

and the evolution equation is

$$\frac{d}{dt}\chi_t = \chi_t\left(\nu_t + \mathscr{L}_{\mathfrak{se}(2)}(w_t^\chi)\right), \ \frac{d}{dt}\zeta_t = 0 + w_t^\zeta$$

The Lie group part of equation (2.30) can be written alternatively as

$$\chi_t^{-1}\frac{d}{dt}\chi_t = \left(\nu_t + \mathscr{L}_{\mathfrak{se}(2)}(w_t^\chi)\right)$$

This shows the right hand side of the equation is in the Lie algebra because it is equal to the derivative application of the left translation on the Lie group $\mathrm{L}_{\chi_t^{-1}} : \mathrm{A} \to \chi_t^{-1}\mathrm{A}$.

The 2D target model based on (piecewise -) constant control commands, and in intrinsic coordinates is now fully established. However, we have seen that we need to have a target model that allows motions that do not lie in a plane. So the 3D model is also derived. The Frenet-Serret frame seems most appropriate to transpose the 2D model into the 3D one.

2.6.2 3D target model

The 3D Frenet-Serret frame is represented on figure 2.2. The evolution equations of the Frenet-Serret frame are known, they are reminded in (2.31). The three vectors of the frame are T, the tangent vector, N the normal vector and B the binormal vector. In equations (2.31), u denotes the norm of the velocity of the centre of the frame, κ the curvature of the trajectory, and $\tilde{\tau}$ the torsion of the trajectory.

Remark 2.1. : The parameters κ and $\tilde{\tau}$ are attached to the trajectory, and not to the target. This means that if the target (an aircraft for instance) has a non-zero roll, this is not observable in the Frenet-Serret frame. That is convenient for us, because the roll of the target is anyway not observable with radar position measurements.

$$\frac{d}{dt}\mathrm{T} = u\kappa\mathrm{N}, \ \frac{d}{dt}\mathrm{N} = u(-\kappa\mathrm{T} + \tilde{\tau}\mathrm{B}), \ \frac{d}{dt}\mathrm{B} = -u\tilde{\tau}\mathrm{N} \tag{2.31}$$

Using a concise matrix notation, equations (2.31) can be re-written

$$\frac{d}{dt}\begin{pmatrix} \mathrm{T} & \mathrm{N} & \mathrm{B} \end{pmatrix} = u\begin{pmatrix} \mathrm{T} & \mathrm{N} & \mathrm{B} \end{pmatrix}\begin{pmatrix} 0 & -\kappa & 0 \\ \kappa & 0 & -\tilde{\tau} \\ 0 & \tilde{\tau} & 0 \end{pmatrix}$$

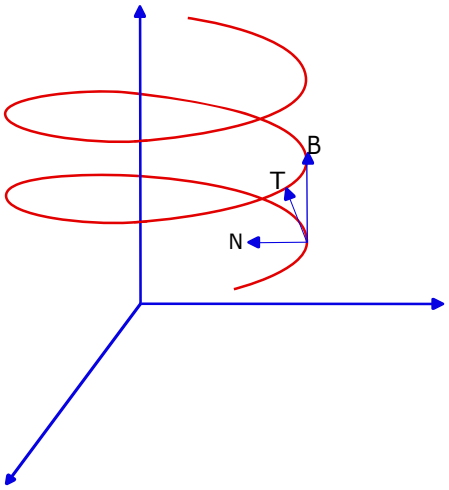


Figure 2.2 – Frenet-Serret frame in 3D

Defining γ and τ as $\gamma = u\kappa$ and $\tau = u\tilde{\tau}$, we finally get

$$\frac{d}{dt}\begin{pmatrix}\mathrm{T} & \mathrm{N} & \mathrm{B}\end{pmatrix} = \begin{pmatrix}\mathrm{T} & \mathrm{N} & \mathrm{B}\end{pmatrix}\begin{pmatrix}0 & -\gamma & 0\\ \gamma & 0 & -\tau\\ 0 & \tau & 0\end{pmatrix} \tag{2.32}$$

In the sequel, the parameters γ and τ will still be referred to as curvature and torsion with a slight abuse of language.

Using the Frenet-Serret formulas, we let $\mathrm{R}_t \in \mathrm{SO}_3$ be the matrix $\mathrm{R}_t = \begin{pmatrix}\mathrm{T}_t & \mathrm{N}_t & \mathrm{B}_t\end{pmatrix}$. We then assume nearly constant tangential velocity, curvature and torsion, leading to the following model for the dynamics (the notation $(.)_\times$ is explained in appendix A):

$$\begin{cases}\frac{d}{dt}x_t = \mathrm{R}_t(v_t + w_t^x)\\ \frac{d}{dt}\mathrm{R}_t = \mathrm{R}_t(\omega_t + w_t^\omega)_\times\\ \frac{d}{dt}\gamma_t = 0 + w_t^\gamma\\ \frac{d}{dt}\tau_t = 0 + w_t^\tau\\ \frac{d}{dt}u_t = 0 + w_t^u\end{cases} \tag{2.33}$$

where the curvature γ_t, the torsion τ_t and the velocity norm u_t are unknown parameters, $\omega_t = \begin{pmatrix}\tau_t & 0 & \gamma_t\end{pmatrix}^\mathrm{T}$, $v_t = \begin{pmatrix}u_t & 0 & 0\end{pmatrix}^\mathrm{T}$, and $w_t^\omega \in \mathbb{R}^3$, $w_t^x = \begin{pmatrix}w_t^x & 0 & 0\end{pmatrix}^\mathrm{T} \in \mathbb{R}^3$, $w_t^\gamma \in \mathbb{R}$, $w_t^\tau \in \mathbb{R}$, $w_t^u \in \mathbb{R}$ are white noises that account for changes over time in the motion parameters. Moreover, we let $(a)_\times \in \mathbb{R}^{3\times 3}$ denote the skew-symmetric matrix associated with cross product with the vector $a \in \mathbb{R}^3$. This notation is again explained extensively in appendix A, along with the definitions of the Lie groups.

As for the 2D model, we can embed the rotation and position of this model into a Lie group, SE(3). Indeed, SE(3) is the group of 3D rotations and translations, so it seems perfectly appropriate for this model. Let us call χ_t and ζ_t the matrix and vector parts of the state, as for the 2D model.

$$\chi_t = \begin{pmatrix}\mathrm{R}_t & x_t\\ 0_{1,3} & 1\end{pmatrix},\ \zeta_t = \begin{pmatrix}\gamma_t\\ \tau_t\\ u_t\end{pmatrix}$$

From (2.33), we can derive more compact equations:

$$\frac{d}{dt}\chi_t = \chi_t(\nu_t + \mathscr{L}_{\mathfrak{se}(3)}(w_t^\chi)),\ \frac{d}{dt}\zeta_t = 0 + w_t^\zeta \tag{2.34}$$

with

$$\nu_t = \begin{pmatrix} 0 & -\gamma_t & 0 & u_t \\ \gamma_t & 0 & -\tau_t & 0 \\ 0 & \tau_t & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \in \mathfrak{se}_3$$

and the noises are

$$w_t^\chi = \begin{pmatrix} w_t^\omega \\ w_t^x \\ 0 \\ 0 \end{pmatrix} \in \mathbb{R}^6, \; w_t^\zeta = \begin{pmatrix} w_t^\gamma \\ w_t^\tau \\ w_t^u \end{pmatrix} \in \mathbb{R}^3$$

Remark 2.2. The model (2.33) can be easily extended to accommodate nearly constant acceleration or acceleration of the Singer model type [103], by replacing $\frac{d}{dt}u_t = 0 + w_t^u$ with $\frac{d}{dt}u_t = a_t$ and $\frac{d}{dt}a_t = -\alpha a_t + w_t^a$. The general form of the model then stays the same. The model with acceleration will be more thoroughly described in section 2.6.3. In fact, the equations giving the evolution of γ_t, τ_t, u_t can be replaced by any linear evolution. One must however be careful, since it is preferable to avoid introducing too difficult parameters to track, as explained in section 3.7.

Associated difficulties and links with prior literature

There have been various prior attempts to use intrinsic coordinates to describe the target motion. The most prominent work in this direction is, to our best knowledge, the pioneering work of Antoulas and Bishop, see *e.g.* [16] and [88]. If we assume the acceleration of the target is large with respect to the gravity vector field g, which is the case for highly maneuvering targets, the model of [16] writes

$$\dddot{x}_t = \frac{\dot{x}_t \times \ddot{x}_t}{\|\dot{x}_t\|^2} \times \ddot{x}_t \tag{2.35}$$

This equation is obtained assuming the kinematic acceleration $\ddot{x}_t$, when projected onto the body frame, is constant. Although the motivations for this model is akin to ours, the obtained equations are slightly different, with different parameters. Note indeed that, first the velocity of the target $\dot{x}_t$ must always be different from zero for the model to be valid, which is not the case with equations (2.33). Moreover, the model (2.35) is based on a zero torsion assumption. As a result quoting [88] "necessary condition for the model to be an accurate representation of actual trajectories is for the real trajectories to have small torsions". It is easily seen that the retained model (2.35) does not lend itself very well to extended Kalman filtering due to its strong non-linearities. Finally, the authors thus propose a geometric filter, which is in fact a deterministic observer (of the Luenberger type [82]), but for a non-linear model, see [14].

2.6.3 Generalisations

Until now, we have supposed the parameters representing control commands were constant, *i.e.* that the torsion, the curvature and the norm of the velocity were constant. It is possible to relax this assumption, and replace the constant evolution by any linear evolution. This is specifically useful for the norm of the velocity, indeed, the norm of the velocity is rarely constant in most applications. To have a more general model, we add the norm of the acceleration to the state, and assume that the norm of the acceleration is almost constant, which seems a more consensual model.

However, this should not be pushed too far, and it seems inefficient to add more terms in the state that are high derivatives of the position. Indeed, the torsion, which comes from a third derivative of the position is already weakly observable.

The definition of the state and the evolution equations are given thereafter. The state is thus composed of the same rotation matrix R_t, the target's position x_t, the curvature γ_t, the torsion τ_t, the norm of the velocity u_t and the norm of the acceleration a_t.

$$\begin{cases} \frac{d}{dt}x_t = \mathrm{R}_t v_t + w_t^x \\ \frac{d}{dt}\mathrm{R}_t = \mathrm{R}_t(\omega_t + w_t^\omega)_\times \\ \frac{d}{dt}\gamma_t = 0 + w_t^\gamma \\ \frac{d}{dt}\tau_t = 0 + w_t^\tau \\ \frac{d}{dt}v_t = a_t + w_t^u \\ \frac{d}{dt}a_t = 0 + w_t^a \end{cases} \tag{2.36}$$

Or, if we prefer the more concise version:

$$\chi_t = \begin{pmatrix} \mathrm{R}_t & x_t \\ 0_{1,3} & 1 \end{pmatrix},\ \zeta_t = \begin{pmatrix} \gamma_t \\ \tau_t \\ u_t \\ a_t \end{pmatrix}$$

$$\nu_t = \begin{pmatrix} 0 & -\gamma_t & 0 & u_t \\ \gamma_t & 0 & -\tau_t & 0 \\ 0 & \tau_t & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix},\ w_t^\chi = \begin{pmatrix} w_t^{\mathrm{R}} \\ w_t^x \end{pmatrix},\ w_t^\zeta = \begin{pmatrix} w_t^\gamma \\ w_t^\tau \\ w_t^u \\ w_t^a \end{pmatrix}$$

$$\frac{d}{dt}\chi_t = \chi_t(\nu_t + (w_t^\chi)_\times),\ \frac{d}{dt}\zeta_t = 0 + w_t^\zeta \tag{2.37}$$

2.7 Conclusion

In this chapter, new target models have been designed. The first one is built for 2D trajectories. Sometimes, radar measurements are expressed in 2D, so this is not a fruitless model. Moreover, as we will see in the next chapter, we can easily build a standard non-linear Kalman filter on this model, using the vectorial formulation (2.28). It has also served as a basis in our work to build the 3D model, namely the idea to use the Frenet-Serret frame in 3D has arisen from the 2D formulation.

The 3D target model, albeit quite simple, seems to have not appeared and been used in the tracking literature so far. It has several advantages. First, the model is fairly elementary. Indeed, it uses the Frenet-Serret frame to express any possible motion in the 3D space. Then, it is expressed in intrinsic coordinates, which means that the motions of the object are represented directly in the frame of the target. This will be proved to yield better accuracy, namely for the orientation angle of the velocity vector during turns.

We have made several assumptions to build our model. First, the assumption that the target cannot slide is common, and all other target models also make the same assumption. Moreover, the process noise included into the model allows modelling a certain extend of sliding. It would add too much complexity to add in the kinematic model a sliding factor, and tracking too many parameters may lead to inefficient state estimation. The other assumption concerns the fact that the norm of the velocity, the curvature and the torsion are (almost) constant. This is less realistic, because it is often not the case in practise, but in chapters 3, and 4, we will discuss on the implications of this assumption on the tuning of the estimation algorithm.

Despite these assumptions, the new target model seems more appropriate to track manoeuvring targets than the ones presented in sections 2.3 and 2.4. Indeed, the models used in industry are also very simple, but they are more rigid in terms of degrees of freedom than the Frenet-Serret target model. Models with jumps are very appealing, but our target model can also be adjusted to take into account jumps in the piecewise constant parameters, as we will see in chapter 5.

The target model is also very versatile, as some new parameters can be added to it. Indeed, we can replace any of the linear equations by other linear equations without changing the general form and framework of the model.

Chapter 3

Filtering algorithms

Sommaire

3.1 Résumé en français : Algorithmes de filtrage

L'objectif d'un algorithme de filtrage est de filtrer la position bruitée de la cible mesurée par le radar, et d'estimer également d'autres caractéristiques cinématiques de la cible, qui sont présentes dans le vecteur d'état. Le filtrage est une méthode probabiliste qui permet d'estimer les paramètres d'un système physique évoluant au cours du temps à partir de mesures bruitées. Considérons l'état $(X_k)_{k\geq 0}$ qui évolue de façon probabiliste, et les mesures $(y_k)_{k\geq 0}$ qui dépendent de façon probabiliste de l'état. On cherche la distribution de probabilité $p(X_k|y_0,\dots,y_k)$. C'est ce que donne exactement le filtre optimal. Malheureusement, ce filtre ne peut être calculé que dans deux cas : quand l'espace d'état est fini et discret, ou quand les modèles de cible et de mesure sont linéaires, avec des bruits blancs Gaussiens. Dans le second cas, le filtre de Kalman linéaire peut être appliqué, et il est alors optimal. Dans tous les autres cas, on ne peut espérer qu'une approximation de la distribution $p(X_k|y_0,\dots,y_k)$. On peut identifier deux familles de méthodes permettant d'obtenir cette approximation :

- Les méthodes qui approximent la distribution par une Gaussienne, et qui donnent une estimation des deux premiers moments (la moyenne et la covariance). Ce sont toutes les variantes du filtre de Kalman. L'EKF (Extended Kalman Filter) est le filtre non-linéaire le plus répandu, et il consiste à calculer des linéarisations à l'ordre un de l'erreur, et d'appliquer un filtre de Kalman sur le système d'erreur approximativement linéaire.
- Les méthodes qui approximent la distribution par des particules, ce sont les filtres particulaires.

Les filtres fonctionnent récursivement et en deux étapes : dans un premier temps, le modèle d'évolution de cible retenu est propagé dans le temps, on appelle cela l'étape de prédiction. Puis lorsqu'une mesure est disponible, la prédiction est corrigée avec la mesure, c'est l'étape de mise à jour.

Afin de développer un algorithme de filtrage adapté à la formulation de notre modèle, il faut revoir la définition de l'erreur habituellement utilisée. En effet, pour un filtre de Kalman non-linéaire usuel (de type EKF par exemple), l'erreur est définie par $e_t = \hat{X}_t - X_t$, avec X_t l'état réel au temps t, et $\hat{X}_t$ l'état estimé au temps t. L'état étant composé d'une matrice de rotation, il n'est pas possible de définir l'erreur de cette façon, en effet, la différence entre deux matrices de rotation ne représente pas une valeur ayant du sens. Il est nécessaire d'utiliser l'opération adaptée sur le groupe de Lie. L'erreur effectuée sur la partie de l'état dans le groupe de Lie, appelée χ_t est donc définie par $\eta_t = \chi_t^{-1}\hat{\chi}_t$. L'erreur étant alors bien définie, il suffit ensuite de calculer son évolution, et de linéariser pour pouvoir obtenir un filtre de type Kalman, appelé l'IEKF (Invariant Extended Kalman Filter). Le filtre est construit exactement de la même façon que l'EKF.

Les premières expérimentations sur des trajectoires simulées montrent que ce filtre est adapté aux trajectoires de cibles manœuvrantes, même en présence de sauts dans la trajectoire. En 2D, il est en particulier possible de comparer les résultats de l'EKF et de l'IEKF sur le même modèle (car il peut être écrit sous forme vectorielle aussi bien que sous forme groupe de Lie), et on montre que l'IEKF donne de meilleurs résultats que l'EKF, en ce qui concerne le cap ainsi que la vitesse. Des résultats plus approfondis et détaillés sont présentés dans le chapitre suivant.

3.2 Introduction

In radar applications, track maintenance is one essential component of the process. Mathematically, it boils down to a filtering problem, where one must filter the current position of the target as well as its velocity and possibly higher order derivatives, from noisy position measurements. We will refer to this problem simply as "target tracking". When the target is manoeuvring, the problem is difficult due to the unpredictable nature of the motion. This area has been the object of extensive research over the four past decades, see [8]. The main degrees of freedom for tracking are 1- the dynamical model describing the motion of the target, and 2- the (statistical) filter used.

In this document, only the single target tracking problem is addressed, and we assume that the plot to track association problem has been solved.

As concerns the models, an extensive presentation has been made in chapter 2. As concerns the filters, the subject of this chapter, a straightforward solution is to use a Kalman filter (or extended Kalman filter for the non-linear models). More modern approaches include particle filters [50] and the reference filter for tracking which is the Interacting Multiple Model (IMM), see *e.g.* [6]. As we will see, the latter filter runs banks of (extended) Kalman filters in parallel based on various models and assess weights to each model by evaluating likelihood of the measured outputs. This allows accommodating the various types of targets and degrees of manoeuvrability a single radar can be confronted with. The academic community has now largely turned to the challenges of multi-target tracking, with joint applications in video, see [83] for the description of the Probability Hypothesis Density filter. Filters performing association along with tracking for multi-target scenarios are becoming widely studied, see for example [38], or [98], which uses the theory of random finite sets, very popular for multi-target tracking.

Our target model described in chapter 2 being partially an element of a matrix Lie group, it is then also possible to cast (partially) the tracking problem into a filtering problem on Lie groups. Slightly extending the Invariant Kalman Filter (IEKF), introduced in [24] and [22], we obtain a novel tracking algorithm. The IEKF is indeed a recent methodology that accounts for the geometric nature of the state space, and comes with convergence properties [11]. It can also be related to the discrete EKF on Lie groups of [26], [27], or to the generalised multiplicative EKF of [84], see also [56].

This chapter will thus be organised in seven different sections. In section 3.3, the general estimation problem is introduced. In section 3.4, the most popular filtering algorithms are presented, along with less used ones, but which have been investigated more in depth in this thesis. In section 3.5, the filter used for our 2D target model is presented. In section 3.6, the filter for the 3D target model is derived. The theory to obtain the estimation filters is explained with different viewpoints for the 2D and the 3D target models. In section 3.7, we present some toy simulations to show the behaviour that can be expected from the model and the filter. Finally, in section 3.8, we present a slight modification of the filter previously presented to avoid computing the Jacobian when using range and bearing observations in 2D.

3.3 The estimation problem for single target tracking

Filtering is a probabilistic method that estimates parameters of a physical system that evolves in time from partial and noisy measurements. For the radar application for instance, important parameters to estimate can be the position, the velocity and the heading angle of a target. Like for target models, there are numerous algorithms to perform filtering, we present in the beginning of this chapter the principle solutions adapted to the filtering problem.

3.3.1 Optimal filter

The setting is the following: consider a state $(\mathrm{X}_k)_{k\geq 0}$ that evolves in a probabilistic way. On an other side, there are some measurements $(y_k)_{k\geq 0}$ that depend probabilistically on the state. The objective is to find the probability distribution $p(\mathrm{X}_k|y_0,...,y_k)$. The optimal filter solves the mean-square estimation problem, see [89], and it gives exactly this probability. Unfortunately, in practice, the optimal filter can be computed for only two cases:

- the discrete finite case,
- the linear Gaussian case.

The Kalman filter used with a linear target model and a linear observation model, with additive Gaussian white noises is optimal, as we will see in section 3.4.1.

3.3.2 Suboptimal filters

In all other cases, we can only obtain an approximation of $p(X_k|y_0,...,y_k)$. We can identify two families of methods to obtain this approximation:

- the first way to perform filtering is to compute the first two moments of the distribution (*i.e.* the mean $\hat{X}$ and the covariance P), assuming that the distribution is Gaussian. This gives all the filters derived from the Kalman filter,
- the second way to perform filtering is to approximate the distribution with particles. This leads to the particle filters.

We will thus present the general description of filters from this two families, and we will present some specific filters more extensively, because they have retained our attention for this work.

3.4 State of the art

3.4.1 Linear Kalman Filter

Suppose we have the following discrete linear target model, with X_k the state at time instant k, of dimension n:

$$X_{k+1} = F_k X_k + B_k u_k + w_k$$
$$y_k = H_k X_k + v_k$$

F_k is the evolution matrix, u_k is a known command input, B_k is the matrix that links the input to the future state, H_k is the observation matrix, w_k and v_k are white Gaussian noises. If the state is of dimension n and the observation of dimension p, then H_k is of dimension $p \times n$.

The principle of the filter is recalled below, a complete description can be found in [71]. The idea is to find an iterative algorithm to compute the least mean square error between the estimation and the observations.

The filter proceeds in two steps. First, the prediction step propagates the model, then when an observation is available, the update step corrects the prediction thanks to the observation. The equations are:

1. Prediction step:
$$\hat{X}_{k|k-1} = F_{k-1}\hat{X}_{k-1|k-1} + B_k u_{k-1}$$
$$P_{k|k-1} = F_{k-1}P_{k-1|k-1}F_{k-1}^T + Q_k$$
2. Update step:
$$\nu_k = y_k - H_k\hat{X}_{k|k-1}$$
$$S_k = H_k P_{k|k-1} H_k^T + N_k$$
$$K_k = P_{k|k-1}H_k^T S_k^{-1}$$
$$\hat{X}_{k|k} = \hat{X}_{k|k-1} + K_k\nu_k$$
$$P_{k|k} = (I_n - K_k H_k)P_{k|k-1}$$

Where I_n is the identity matrix of size $n \times n$, $P_{k|k-1}$ is the predicted covariance, $\hat{X}_{k|k-1}$ the predicted state, Q_k the process noise covariance matrix, $P_{k|k}$ the estimated covariance, $\hat{X}_{k|k}$ the estimated state, N_k the measurement noise covariance matrix, ν_k the innovation, S_k the covariance of the innovation, and K_k the Kalman gain.

Proposition 3.1. *For linear Gaussian systems, the Kalman filter is optimal.*

Proof. This is a consequence of the theory of Gaussian vectors and Gaussian conditioning in the field of probability. In appendix B, we provide the proof partially. □

3.4.2 Interacting Multiple Model Filter (IMM)

It is one of the reference filters in industry. Although it was developped decades ago, it is considered as state of the art for industrial implementations. The IMM filter, developed by Bar-Shalom in the end of the 1980s, and presented in [6] and [5], is designed to take into account several models ni parallel. Indeed, the standard Kalman filter assumes the target follows a "best" model. The IMM allows to overcome this constraint. However, one needs to know the transition probabilities between the models. As for the standard Kalman filter, the choice of models is decisive. This filter implies to let several Kalman filters run in parallel, as many as the number of models: the state and covariance estimations are done for each model, and then a weight computation allows to know which one is the best model. The final state is the weighted sum of the different estimates. A brief description of the IMM is provided in figure 3.1. And the complete cycle of the IMM is described below. For this, let us assume there are r different models $\mathrm{M}_1, \ldots, \mathrm{M}_r$.

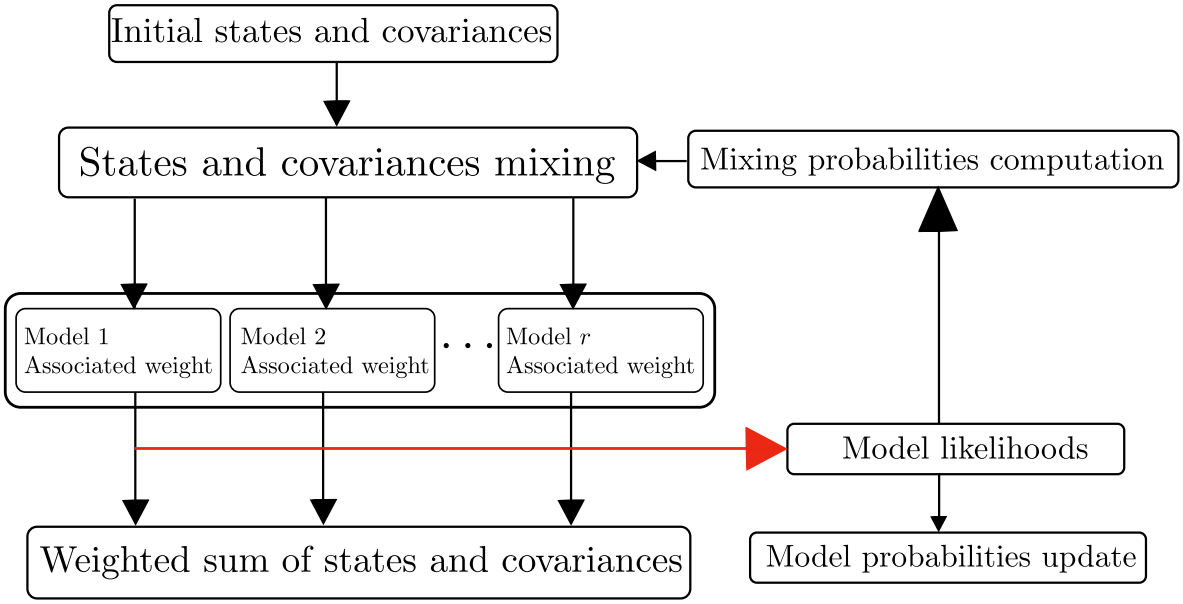


Figure 3.1 – Interacting Multiple Model

1. Mixing probabilities computation (for $i, j = 1, \ldots, r$): the probability that the model M_i is used at time $k-1$, knowing that M_j is used at time k, conditioned by $y_{1:k-1}$ is:

$$\begin{aligned}\mu_{i|j}(k-1|k-1) &\triangleq \mathrm{P}(\mathrm{M}_i(k-1)|\mathrm{M}_j(k), y_{1:k-1}) \\ &= \frac{1}{c_j}\mathrm{P}(\mathrm{M}_j(k)|\mathrm{M}_i(k-1), y_{1:k-1})\mathrm{P}(\mathrm{M}_i(k-1)|y_{1:k-1})\end{aligned}$$

   The mixing probabilities can be written

$$\mu_{i|j}(k-1|k-1) = \frac{1}{c_j} p_{ij}\mu_i(k-1) \text{ for } i, j = 1, \ldots, r$$

   where the normalizing constants are

$$c_j = \sum_{i=1}^{r} p_{ij}\mu_i(k-1) \text{ for } j = 1, \ldots, r$$

2. Mixing ($j = 1, \ldots, r$): from $\hat{\mathrm{X}}^i(k-1|k-1)$, the mixed initial condition is computed for the filter for $\mathrm{M}_j(k)$ with:

$$\hat{\mathrm{X}}^{0j}_{k-1|k-1} = \sum_{i=1}^{r} \hat{\mathrm{X}}^{i}_{k-1|k-1}\mu_{i|j}(k-1|k-1) \text{ for } j = 1, \ldots, r$$

The associated covariance is

$$\mathrm{P}^{0j}_{k-1|k-1} = \sum_{i=1}^{r} \mu_{i|j}(k-1|k-1) \times \left(\mathrm{P}^{i}(k-1|k-1) + [\hat{\mathrm{X}}^{i}_{k-1|k-1} - \hat{\mathrm{X}}^{0j}_{k-1|k-1}][\hat{\mathrm{X}}^{i}_{k-1|k-1} - \hat{\mathrm{X}}^{0j}_{k-1|k-1}]^{\mathrm{T}}\right)$$

3. Filtering associated to a model $j = 1, \ldots, r$: the estimate $\hat{\mathrm{X}}^{0j}_{k-1|k-1}$ and its covariance $\mathrm{P}^{0j}_{k-1|k-1}$ are used as an input for the filter for model $\mathrm{M}_j(k)$. The filter uses $y(k)$ to output $\hat{\mathrm{X}}^{j}_{k|k}$ and $\mathrm{P}^{j}_{k|k}$. The likelihood functions corresponding to the r filters defined by $\Delta_j(k) = \mathrm{P}(y_k|\mathrm{M}_j(k), y_{1:k-1})$ are computed with the formula:

$$\Delta_j(k) = \mathrm{P}(y_k|\mathrm{M}_j(k), \hat{\mathrm{X}}^{0j}_{k-1|k-1}, \mathrm{P}^{0j}_{k-1|k-1})$$

with P corresponding to the normal distribution.

4. Update of the models probabilities for $j = 1, \ldots, r$:

$$\mu_j(k) = \frac{1}{c}\Delta_j(k)c_j \text{ for } j = 1, \ldots, r$$

where $c = \sum_{i=1}^{r} \Delta_j(k)c_j$.

5. The output state and covariance estimations are then given by:

$$\hat{\mathrm{X}}_{k|k} = \sum_{i=1}^{r} \hat{\mathrm{X}}^{i}_{k|k}\mu_i(k)$$

$$\mathrm{P}_{k|k} = \sum_{i=1}^{r} \mu_j(k)\mathrm{P}^{i}_{k|k} + [\hat{\mathrm{X}}^{i}_{k|k} - \hat{\mathrm{X}}_{k|k}][\hat{\mathrm{X}}^{i}_{k|k} - \hat{\mathrm{X}}_{k|k}]^{\mathrm{T}}$$

This step serves only for the output of the filter, and is not part of the recursions of the algorithm.

Remark 3.1. : Here, a reduction of hypothesis is made. Indeed, we normally have a Gaussian mixture: at each time instant, there are r densities on $\mathbb{R}^n$. The densities on $\mathbb{R}^n$ are rarely Gaussian. Even if $p(\mathrm{X}_0|y_0)$ is Gaussian, $p(\mathrm{X}_k|y_{1:k})$ is in general a sum of r^{k-1} weighted Gaussians. However, we approximate this mixture with a single Gaussian.

This algorithm is one of the most used in radars: it is a fairly simple improvement of the Kalman filter, and enables to take into account several models at once, without introducing too much extra complexity, and without changing the paradigm, since it consists of running a few Kalman filters in parallel, and of computing weights quite easily. However, the models must be elaborately chosen, and the possible transitions between the models must be modelled. And since the target has behaviours that always deviate from one of the chosen model, it is often necessary to introduce a model with a high process noise, that deals with all motions that do not correspond to the other models. Particle filters presented in section 3.4.3 permit to overcome these kind of constraints. Moreover, authors of [81] present motion models coupled with other multi-model filters.

3.4.3 Non-linear filters

When the model equations are non-linear, other algorithms are used. The general form of the model is

$$\begin{cases} \mathrm{X}_k = f(\mathrm{X}_{k-1}, u_k) + w_k \\ y_k = h(\mathrm{X}_k) + v_k \end{cases} \tag{3.1}$$

In the non-linear case, w_k and v_k are not necessarily Gaussian, but we assume they are mutually independent and with known probability density functions, and f and h are not necessarily linear. As explained in section 3.3, non-linear filters can be divided in two categories:

- Gaussian filters;
- Particle filters.

We will first describe two of the most popular Gaussian filters, the Extended Kalman Filter (EKF), the Unscented Kalman Filter (UKF), and the more specific Invariant Extended Kalman Filter (IEKF), then we will present the general particle filter, and the more specific Rao-Blackwell Particle Filter, and Variable Rate Particle Filter.

Extended Kalman Filter

To derive the Extended Kalman Filter (EKF), we assume again the noises are Gaussian and additive. The EKF is based on linearisations of the non-linear functions. This includes computing the Jacobian of these functions. It is one of the most used filters in practice (and especially in industry). The equations are very similar to that of the linear Kalman filter, except that the matrices F_k and H_k of the Kalman filter are replaced by the Jacobians of the non-linear functions intervening in the equations. Let us call F_{k-1} and H_k the Jacobians defined as:

$$\mathrm{F}_{k-1} = \left.\frac{\partial f}{\partial x}\right|_{\hat{\mathrm{X}}_{k-1|k-1}, u_k}$$

$$\mathrm{H}_k = \left.\frac{\partial h}{\partial x}\right|_{\hat{\mathrm{X}}_{k|k-1}}$$

The equations of the filter are as follows:

1. Prediction step:

$$\hat{\mathrm{X}}_{k|k-1} = f(\hat{\mathrm{X}}_{k-1|k-1}, u_k)$$
$$\mathrm{P}_{k|k-1} = \mathrm{F}_{k-1}\mathrm{P}_{k-1|k-1}\mathrm{F}_{k-1}^{\mathrm{T}} + \mathrm{Q}_k$$

2. Update step:

$$\nu_k = y_k - h(\hat{\mathrm{X}}_{k|k-1})$$
$$\mathrm{S}_k = \mathrm{H}_k\mathrm{P}_{k|k-1}\mathrm{H}_k^{\mathrm{T}} + \mathrm{N}_k$$
$$\mathrm{K}_k = \mathrm{P}_{k|k-1}\mathrm{H}_k^{\mathrm{T}}\mathrm{S}_k^{-1}$$
$$\hat{\mathrm{X}}_{k|k} = \hat{\mathrm{X}}_{k|k-1} + \mathrm{K}_k\nu_k$$
$$\mathrm{P}_{k|k} = (\mathrm{I}_n - \mathrm{K}_k\mathrm{H}_k)\mathrm{P}_{k|k-1}$$

The EKF is not a very stable filter, especially when the non-linearities are strong. Indeed,when looking with more depth into the computations leading to the algorithm, we see that the error is defined as

$$e_{k|k-1} = \mathrm{X}_k - \hat{\mathrm{X}}_{k|k-1} \tag{3.2}$$

and

$$e_{k-1|k-1} = \mathrm{X}_{k-1} - \hat{\mathrm{X}}_{k-1|k-1} \tag{3.3}$$

This error has to be linearised to derive the covariance evolution. This linearisation is:

$$e_{k|k-1} = \frac{\partial}{\partial x} f(\hat{\mathrm{X}}_{k-1|k-1}, u_k) e_{k-1|k-1} + w_k$$

The error evolution depends on the Jacobian $\mathrm{F}_{k-1} = \frac{\partial}{\partial x} f(\hat{\mathrm{X}}_{k-1|k-1}, u_k)$ of the evolution function f. The problem is that this Jacobian can only be computed at the point $\hat{\mathrm{X}}_{k-1|k-1}$, that is an estimate. If the estimate is wrong, then the error becomes wrong also, and the covariance no longer represents the actual incertitude over the estimate. The problem is of course the same with the linearisation of the measurement function h. This can lead to the filter's divergence.

To avoid the linearisation problem, a lot of filters have been developed, including the Unscented Kalman Filter (UKF), see [66] or [67], that we present in the next paragraph, the Geometric Non-linear Filter (GNF), see [15] or [16], or the Invariant Extended Kalman Filter (IEKF), see [11] that we will develop later.

Remark 3.2. : It is perfectly possible to use an IMM with non-linear target models. Instead of running Kalman filters in parallel, Extended Kalman filters have to be used. The other equations, giving the mixing of the state and covariance do not change.

Unscented Kalman Filter

Developed by Julier and Uhlmann in [66] and [67] to avoid the problems of stability of the EKF, and to avoid the problem of bias when computing the association window, the UKF is becoming increasingly used. This filter is also well described in [109], or [106]. The noises do not necessarily need to be Gaussian.

Let us consider again the target model (3.1). The idea of the UKF is to sample the distribution deterministically, thanks to the use of so-called *sigma points*. In this aim, an augmented state, including the process and observation noises is defined:

$$\mathrm{X}_k^a = \begin{pmatrix} \mathrm{X}_k \\ w_k \\ v_k \end{pmatrix} \tag{3.4}$$

We then define a set of $2n+1$ sigma points α_k (n is the dimension of the augmented state), and their weights W_k:

$$\alpha_k^0 = \mathrm{X}_k$$
$$\alpha_k^i = \mathrm{X}_k + (\sqrt{(n+\lambda)\mathrm{P}_k})_i, \ i = 1,\dots,n$$
$$\alpha_k^i = \mathrm{X}_k - (\sqrt{(n+\lambda)\mathrm{P}_k})_{i-n}, \ i = n+1,\dots,2n$$
$$\mathrm{W}_0^m = \frac{\lambda}{n+\lambda}$$
$$\mathrm{W}_0^c = \frac{\lambda}{n+\lambda} + (1-\beta^2+\gamma)$$
$$\mathrm{W}_i^m = \mathrm{W}_i^c = 1/(2(n+\lambda)), \ i = 1,\dots,2n$$

where $\lambda = \beta^2(n+\kappa) - n$ is a scaling parameter. The parameters β, γ, κ are used to set the dispersion of the sigma points.

The filter then operates in two steps, as any filter. During the prediction step, the sigma points are propagated through the non-linear evolution function:

$$\alpha_k^i = f(\alpha_k^{i-1})$$

The mean and covariance for the prediction can be deduced from these propagated sigma points:

$$\mathrm{X}_k = \sum_{i=0}^{2n} \mathrm{W}_i^m \alpha_k^i$$

$$\mathrm{P}_k = \sum_{i=0}^{2n} \mathrm{W}_i^c (\alpha_k^i - \mathrm{X}_k)(\alpha_k^i - \mathrm{X}_k)^{\mathrm{T}}$$

Then the sigma points are propagated through the measurement function h:

$$z_k^i = h(\alpha_k^i)$$

$$z_k = \sum_{i=0}^{2n} \mathrm{W}_i^m z_k^i$$

And the update equations thus are:

$$\mathrm{P}_z = \sum_{i=0}^{2n} \mathrm{W}_i^c (z_k^i - z_k)(z_k^i - z_k)^{\mathrm{T}}$$

$$P_{x,z} = \sum_{i=0}^{2n} W_i^c (\alpha_k^i - X_k)(z_k^i - z_k)^T$$

$$K_k = P_{x,z} P_z^{-1}$$

$$X_k^+ = X_k + K_k(y_k - z_k)$$

$$P_k^+ = P_k - K_k P_z K_k^T$$

Remark 3.3. Another advantage of the UKF over the EKF is that there is no need to compute Jacobians. The Jacobians can indeed be tricky to compute, depending on the evolution or observation functions.

Invariant Extended Kalman Filter

The general setting of the Invariant Extended Kalman Filter (IEKF) is presented in [11], and in [9]. The idea is to develop a filter adapted to models that have an invariant property. We summarise here the results obtained for specific invariant models of these papers. All the necessary mathematical tools to understand what a Lie group is are given in appendix A.

Let us consider the following target model, expressed on a Lie group:

$$\frac{d}{dt}\chi_t = f_{u_t}(\chi_t) + \chi_t \mathscr{L}_{\mathfrak{g}}(w_t) \tag{3.5}$$

with the state $\chi_t \in G \subset \mathbb{R}^{n\times n}$, w_t a white noise belonging to the Lie algebra $\mathfrak{g}$ with covariance Q_t, u_t is a known control input, and f a function that verifies $f_u(ab) = af_u(b) + f_u(a)b - af_u(Id)b$. The latter property of f is called the affine invariance property of the model, and was shown in [11] to be the proper generalisation of linear systems on Lie groups. Let us consider left-invariant observations (Cartesian position observations are left-invariant observations if the state contains the Cartesian position):

$$y_{t_n} = \chi_{t_n} d + v_n \tag{3.6}$$

where d is a known vector, and v_n a white Gaussian noise with covariance N_n. The left-IEKF is defined by the state propagation and update:

$$\frac{d}{dt}\hat{\chi}_t = f_{u_t}(\hat{\chi}_t) \tag{3.7}$$

$$\hat{\chi}_{t_n}^+ = \hat{\chi}_{t_n} \exp(L_n(\hat{\chi}_{t_n}^{-1} y_{t_n} - d)) \tag{3.8}$$

where L_n is the Kalman gain. The left-invariant error is defined by

$$\eta_t^L = \chi_t^{-1}\hat{\chi}_t \tag{3.9}$$

The error evolution can be computed, and the result is:

$$\frac{d}{dt}\eta_t^L = f_{u_t}(\eta_t^L) - f_{u_t}(Id)\eta_t^L - w_t\eta_t^L$$

So the error is independent from the state trajectory. The natural way to represent the error as a vector, is to linearise the error η_t on the Lie algebra $\mathfrak{g}$. This gives a vector $\xi_t \in \mathbb{R}^{\dim \mathfrak{g}}$ such that

$$\eta_t^L = \exp(\xi_t) = \exp_m(\mathscr{L}_{\mathfrak{g}}(\xi_t))$$

This linearised error propagates as

$$\frac{d}{dt}\xi_t = A_t\xi_t + w_t \tag{3.10}$$

where A_t is defined by $g_{u_t}(\exp(\xi_t)) = \mathscr{L}_{\mathfrak{g}}(A_t\xi_t) + O(||\xi_t||^2)$, where the terms of order $O(||\xi_t||^2)$ and $O(||w_t||||\xi_t||)$ have been neglected. The error update can be computed as follows:

$$(\eta_{t_n}^L)^+ = \chi_{t_n}^{-1}\hat{\chi}_{t_n}^+ = \eta_{t_n}^L \exp\left(L_n\left((\eta_{t_n}^L)^{-1}d - d + \hat{\chi}_{t_n}^{-1}v_n\right)\right)$$

which can be linearised in

$$\xi_{t_n}^+ = \xi_{t_n} + \mathrm{L}_n(\mathrm{H}\xi_{t_n} + \hat{\chi}_{t_n}^{-1}\upsilon_n)$$

with H the observation function, defined such that $\mathrm{H}\xi = -\mathscr{L}_{\mathfrak{g}}(\xi)d$. The Kalman gain L_n can be computed with the following equations (similar to the standard Kalman filter):

$$\frac{d}{dt}\mathrm{P}_t = \mathrm{A}_t\mathrm{P}_t + \mathrm{P}_t\mathrm{A}_t^{\mathrm{T}} + \mathrm{Q}_t$$

$$\mathrm{P}_{t_n}^+ = (\mathrm{I} - \mathrm{L}_n\mathrm{H})\mathrm{P}_{t_n}$$

$$\mathrm{S}_n = \mathrm{H}\mathrm{P}_{t_n}\mathrm{H}^{\mathrm{T}} + \mathrm{N}_n$$

$$\mathrm{L}_n = \mathrm{P}_{t_n}\mathrm{H}^{\mathrm{T}}\mathrm{S}_n^{-1}$$

The IEKF has stability properties that an EKF does not have. In particular, it is asymptotically stable under quite loose conditions. This means that the filter will not diverge if the true trajectory is close to the model. This is a quite strong property, since there is no analogous property for the EKF.

Moreover, this filter has the advantage of being close to a Kalman filter, and in practice, it is quite easy to implement. The theory that underlies it is powerful, and allows to prove some stability results, but the method is the same as the Kalman filter: it is a recursive Bayesian filter, and we compute the first two moments of the probability distribution of the state, assuming it is Gaussian.

Other filters have been developed on Lie groups. This includes the work of [42]. The authors estimate the state of their Lie group model (based on Coordinated turns), with a filter called a Discrete Lie Group EKF. They modify the distribution of the noises to take into account the Lie group formalism of their state. Another approach, yet very close to the one presented in this section, is presented in [26] and [27]. One other method proposed is to model the observations on a manifold (when the target evolves in a Lie group $\mathrm{SO}(n)$), and use a sampling filtering method to perform state estimation, as explained in [25].

Particle filters

Another way to cope with the non-linearities is to use random sampling. The particle filter is described in [50], [61] and [4]. This sampling method appeared at the end of the nineties. We sum up here the principal contribution of this type of filtering. Let us call $y_{1:k} = \{y_0, \dots, y_k\}$ the set of the measurements from time 0 to time k, and $\mathrm{X}_{1:k} = \{\mathrm{X}_0, \dots, \mathrm{X}_k\}$ the set of states from time 0 to time k. The particle filter approximates $p(\mathrm{X}_k|y_{1:k})$. The X_k are the hidden parameters, and the y_k are the observed data. Let us assume the following hypotheses are valid:

1. The states $\mathrm{X}_0, \mathrm{X}_1, \dots$, form a Markov chain such that $\mathrm{X}_k|\mathrm{X}_{k-1} \sim p(\mathrm{X}_k|\mathrm{X}_{k-1})$ of known initial distribution;

2. The observations y_k depend only on X_k.

Equation (3.1) does satisfy these hypotheses. The particles, *i.e.* a sum of Dirac measures are used to sequentially approximate the distribution $p(\mathrm{X}_k|y_{1:k})$. The final state is a weighted sum of particles, after a re-sampling step for which there are plenty of solutions, listed for instance in [48]. An illustration explaining how the filter works is provided in figure 3.2, with an importance distribution $q(\mathrm{X}_{k+1}|\mathrm{X}_{1:k}, y_{k+1}) = p(\mathrm{X}_{k+1}^i|\mathrm{X}_k^i)$.

A tutorial on particle filters can be found in [51], or in [62]. We propose here to describe the basic particle filter, which belongs to a larger class of algorithms, namely the Sequential Monte-Carlo (SMC) algorithms. Before running the algorithm, we must define an importance distribution called $q(\mathrm{X}_{k+1}|\mathrm{X}_{1:k}, y_{k+1})$, from which we are able to sample, and a re-sampling strategy. The number of particles is N. To initialise the particle filter, we must first generate samples with $\mathrm{X}_1^i \sim p_{x_0}$ for $i = 1, \dots, \mathrm{N}$, and initialise the weights at $w_{1|0} = 1/\mathrm{N}$. Then the algorithm iterates as follows:

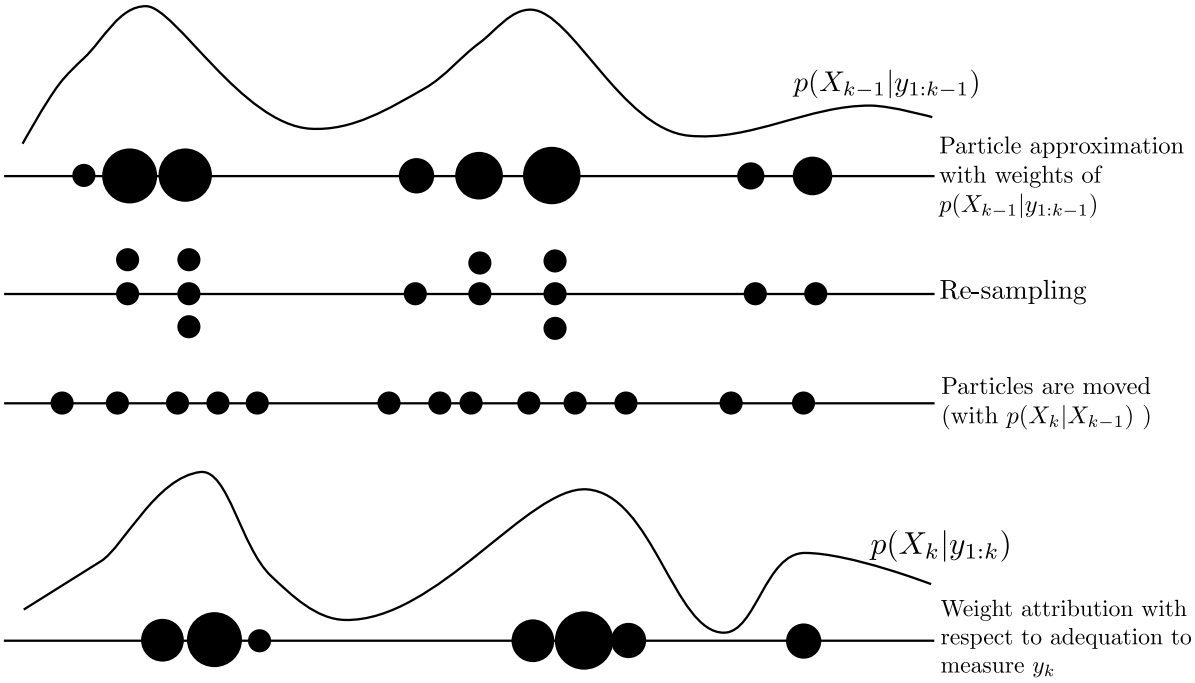


Figure 3.2 – Particle filter

1. Measurement update: For $i = 1, ..., \mathrm{N}$,

$$w^i_{k|k} = \frac{1}{c_k} w^i_{k|k-1} p(y_k|\mathrm{X}^i_k)$$

where c_k is a normalization constant

$$c_k = \sum_{i=1}^{\mathrm{N}} w^i_{k|k-1} p(y_k|\mathrm{X}^i_k)$$

2. Estimation of the distribution: $p(\mathrm{X}_{1:k}|y_{1:k}) \approx \sum_{i=1}^{\mathrm{N}} w^i_{k|k} \delta(\mathrm{X}_{1:k} - \mathrm{X}^i_{1:k})$, and the mean of the state is $\hat{\mathrm{X}}_{1:k} = \sum_{i=1}^{\mathrm{N}} w^i_{k|k} \mathrm{X}^i_{1:k}$.

3. Re-sampling (optional, depends on the resampling strategy): delete the particles with the lowest weights and duplicate those with the highest weights. The number of particles should be kept constant. At the end of this step, the weights are $w^i_{k|k} = 1/\mathrm{N}$.

4. Prediction:

$$\mathrm{X}^i_{k+1} \sim q(\mathrm{X}_{k+1}|\mathrm{X}^i_k, y_{k+1})$$

5. Weight update:

$$w^i_{k+1|k} = w^i_{k|k} \frac{p(\mathrm{X}^i_{k+1}|\mathrm{X}^i_k)}{q(\mathrm{X}^i_{k+1}|\mathrm{X}^i_k, y_{k+1})}$$

Remark 3.4. Some convergence results can be obtained for a large class of particle filters, among which the basic result presented in [64], which proves that the solution to the estimation problem given by the particle filter converges towards the optimal solution when the number of particles increases to infinity, for a large class of unbounded functions.

Rao-Blackwellized Particle Filter

The Rao-Blackwellized particle filter is explained in detail in [49]. The idea is to separate the state in two parts, $\mathrm{X}_k = (r_k, \theta_k)$ such that $p(\mathrm{X}_k|\mathrm{X}_{k-1}) = p(\theta_k|r_{k-1:k}, \theta_{k-1}) p(r_k|r_{k-1})$. The two parts

are chosen also so that once $r_{1:k}$ is known, the distribution $p(\theta_{0:k}|y_{1:k}, r_{0:k})$ can be easily computed. The Bayes rule is then used: $p(r_{0:k}, \theta_{0:k}) = p(\theta_{0:k}|y_{1:k}, r_{0:k})p(r_{0:k}|y_{1:k})$, and the sampling is used only to estimate $p(r_{0:k}|y_{1:k})$, which is a reduced problem compared to the sampling of the probability for the whole state. In our case, it is convenient to use a Kalman filter to compute $p(\theta_{0:k}|y_{1:k}, r_{0:k})$.

The idea of this filter is quite close to that of the IMM, in the sense that several Kalman filters run in parallel. The IMM is based on a finite choice of models, and computes the probabilities of each model. The Rao-Blackwellized particle filter samples a continuous parameter and computes the likelihood of each particle thanks to a Kalman filter. The sampled parameter is chosen so that once this parameter is known, the state can be fully estimated with a Kalman filter. This filter is fully developed in the appendix D.1.

Variable Rate Particle Filter

[32] develops models based on the use of intrinsic coordinates that are akin to our 3D Frenet-Serret model presented in the preceding chapter. The target model proposed by the authors has been detailed in section 2.3.4. The proposed model is based on piecewise constant tangential and normal accelerations and constant-plane motions, and differs from [5] and [6] only in the way the noise enters the system. The authors assume indeed the system to be deterministic, and the acceleration to jump from time to time. It is thus a variable rate model. They then use a particle filter to select the most likely accelerations. The method is very close to the Rao-Blackwell Particle Filter, except that there is no process noise, and the jumps are allowed in continuous time resorting to the use of marked point processes (the time of jumps are also sampled). This filter is also detailed more explicitly in appendix D.2.

In this work, we have focused on the Frenet-Serret model, and we aim at developing an estimation filter suited to this model. The IEKF seems to be the most suited filter to achieve estimation. Indeed, the state of the Frenet-Serret model fits partly the requirements of the filter, and the presence of the rotation matrix in the state requires a special treatment. However, since the setting does not fully satisfy the conditions to apply directly the IEKF, we need to adapt and extend it to fit our model.

3.5 IEKF applied to the 2D Frenet-Serret model

Let us first concentrate on the 2D model derived in section 2.6.1. We recall the 2D Frenet-Serret model equations (2.30):

$$\frac{d}{dt}\chi_t = \chi_t\left(\nu_t + \mathscr{L}_{\mathfrak{se}(2)}(w_t^\chi)\right), \quad \frac{d}{dt}\zeta_t = 0 + w_t^\zeta \tag{3.11}$$

where

$$\chi_t = \begin{pmatrix} \cos\theta_t & -\sin\theta_t & x_t^1 \\ \sin\theta_t & \cos\theta_t & x_t^2 \\ 0 & 0 & 1 \end{pmatrix}, \ \zeta_t = \begin{pmatrix} \omega_t \\ u_t \end{pmatrix} \tag{3.12}$$

and

$$\nu_t = \begin{pmatrix} 0 & -\omega_t & u_t \\ \omega_t & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \in \mathfrak{se}(2)$$

The predicted state equation in continuous time writes:

$$\frac{d}{dt}\hat{\chi}_t = \hat{\chi}_t\hat{\nu}_t, \quad \frac{d}{dt}\hat{\zeta}_t = 0 \tag{3.13}$$

with

$$\hat{\nu}_t = \begin{pmatrix} 0 & -\hat{\omega}_t & \hat{u}_t \\ \hat{\omega}_t & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

In order to derive the equations of the IEKF for our model, let us first define the error $\eta_t = \begin{pmatrix} \eta_t^\chi & \eta_t^\zeta \end{pmatrix}$ on the state $\begin{pmatrix} \chi_t & \zeta_t \end{pmatrix}$:

$$\begin{cases} \eta_t^\chi = \chi_t^{-1}\hat{\chi}_t \in \mathrm{SE}(2) \\ \eta_t^\zeta = \hat{\zeta}_t - \zeta_t \in \mathbb{R}^2 \end{cases} \tag{3.14}$$

The reason why the error is defined in two parts is that the χ-part of the state follows a left-invariant evolution equation assuming that ζ is known, as defined in [11], and the ζ-part of the state follows a linear vectorial equation.

Let us compute the evolution of the error η_t: the evolution of η_t^ζ is quite easy to derive:

$$\frac{d}{dt}\eta_t^\zeta = \frac{d}{dt}\hat{\zeta}_t - \frac{d}{dt}\zeta_t = -w_t^\zeta \tag{3.15}$$

The evolution of η_t^χ is a little more subtle, it writes:

$$\begin{aligned} \frac{d}{dt}\eta_t^\chi &= \frac{d}{dt}(\chi_t^{-1}\hat{\chi}_t) \\ &= -\chi_t^{-1}\frac{d}{dt}\chi_t\chi_t^{-1}\hat{\chi}_t + \chi_t^{-1}\frac{d}{dt}\hat{\chi}_t \\ &= -\chi_t^{-1}\chi_t(\nu_t + \mathscr{L}(w_t^\chi))\chi_t^{-1}\hat{\chi}_t + \chi_t^{-1}\hat{\chi}_t\hat{\nu}_t \end{aligned}$$

So finally, we obtain the following equation for the evolution of η_t^χ:

$$\frac{d}{dt}\eta_t^\chi = -\nu_t\eta_t^\chi - \mathscr{L}_{\mathfrak{se}(2)}(w_t^\chi)\eta_t^\chi + \eta_t^\chi\hat{\nu}_t \tag{3.16}$$

Equation (3.16) can be written differently, to make appear a perturbation term:

$$\frac{d}{dt}\eta_t^\chi = [\eta_t^\chi, \nu_t] - \mathscr{L}_{\mathfrak{se}(2)}(w_t^\chi)\eta_t^\chi + \eta_t^\chi(\hat{\nu}_t - \nu_t)$$

Equations (3.15) and (3.16) have to be linearised to compute the evolution of the covariance.

To perform linearisation, we introduce the linearised error $\xi_t = \begin{pmatrix} \xi_t^\chi & \xi_t^\zeta \end{pmatrix}^\mathrm{T} \in \mathbb{R}^{3+2}$, because $\dim \mathfrak{se}(2) = 3$. $\mathscr{L}_{\mathfrak{se}(2)}(\xi_t^\chi)$ lies in the Lie algebra $\mathfrak{se}(2)$. As for the other variables, the notation means that ξ_t^χ is the linearisation of the error made on χ_t which is η_t^χ, and that ξ_t^ζ is the linearisation of the error made on ζ_t, which is η_t^ζ. The ζ-part of the state follows a linear evolution equation, so in fact we have $\xi_t^\zeta = \eta_t^\zeta$. ξ_t^χ is defined such that:

$$\eta_t^\chi \approx \mathrm{I}_3 + \mathscr{L}_{\mathfrak{se}(2)}(\xi_t^\chi)$$

Let us first assume that the norm of the velocity and the angular velocity are known. The computations in [11] show that the linearised error evolution is

$$\frac{d}{dt}\xi_t^\chi = \mathrm{A}_t\xi_t^\chi - w_t^\chi \tag{3.17}$$

where

$$\mathrm{A}_t = -\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & \omega_t \\ u_t & -\omega_t & 0 \end{pmatrix}$$

Thanks to this formula we can derive the global linearised error evolution equation for the 2D Frenet-Serret model. The computations are detailed below:

$$\mathscr{L}_{\mathfrak{se}(2)}\begin{pmatrix}\xi_t^\zeta\\0\end{pmatrix}=\begin{pmatrix}0&-(\hat{\omega}_t-\omega_t)&\hat{u}_t-u_t\\\hat{\omega}_t-\omega_t&0&0\\0&0&0\end{pmatrix}=\hat{\nu}_t-\nu_t$$

So we obtain:

$$\nu_t=\hat{\nu}_t-\mathscr{L}_{\mathfrak{se}(2)}\begin{pmatrix}\xi_t^\zeta\\0\end{pmatrix}$$

It is possible to re-write equation (3.16) using this formulation:

$$\frac{d}{dt}\eta_t^\chi=\eta_t^\chi\hat{\nu}_t-\left(\hat{\nu}_t-\mathscr{L}_{\mathfrak{se}(2)}\begin{pmatrix}\xi_t^\zeta\\0\end{pmatrix}\right)\eta_t^\chi-\mathscr{L}_{\mathfrak{se}(2)}(w_t^\chi)\eta_t^\chi \tag{3.18}$$

A mere analogy with the results of equation (3.17) shows that the linearised equation writes:

$$\frac{d}{dt}\xi_t^\chi=\mathrm{A}_t\xi_t^\chi+\begin{pmatrix}\xi_t^\zeta\\0\end{pmatrix}-w_t^\chi \tag{3.19}$$

This gives

$$\frac{d}{dt}\xi_t=\tilde{\mathrm{A}}_t\xi_t-\begin{pmatrix}w_t^\chi\\w_t^\zeta\end{pmatrix} \tag{3.20}$$

with

$$\tilde{\mathrm{A}}_t=-\begin{pmatrix}0&0&0&-1&0\\0&0&\hat{\omega}_t&0&-1\\\hat{u}_t&-\hat{\omega}_t&0&0&0\\0&0&0&0&0\\0&0&0&0&0\end{pmatrix}$$

The matrix $\tilde{\mathrm{A}}_t$ is independent of $\hat{x}_t$ and $\hat{\theta}_t$. This is a property inherited from the Invariant Theory and the Invariant Extended Kalman Filter of [11]. However, for the tracking problem, u_t and ω_t are unknown parameters, and they enter the estimation problem. So the filter is not completely invariant. But the presence of these parameters are of a lesser concern, because they are time derivatives of the position, and not directly observed, so they have a lesser impact on the filter.

3.5.1 Position observations in Cartesian coordinates

In this section, the observations are Cartesian positions:

$$y_n=\begin{pmatrix}x_{t_n}^1&x_{t_n}^2\end{pmatrix}^{\mathrm{T}}+v_n$$

with v_n a white Gaussian noise of covariance N_n with independent coordinates in 2D. To mimic the description of the IEKF in section 3.4.3, they can also be written:

$$\begin{pmatrix}y_n^1\\y_n^2\\0\end{pmatrix}=\chi_{t_n}\begin{pmatrix}0\\0\\1\end{pmatrix}+\begin{pmatrix}v_n^1\\v_n^2\\0\end{pmatrix}$$

In this case, the observation matrix is

$$\mathrm{H}=\begin{pmatrix}0&1&0&0&0\\0&0&1&0&0\end{pmatrix}$$

Applying standard Kalman equations to the obtained linearised model yields the following algorithm:

1. Propagation:
$$\frac{d}{dt}\hat{\theta}_t = \hat{\omega}_t,\ \frac{d}{dt}\hat{x}_t = \begin{pmatrix}\cos\hat{\theta}_t \\ \sin\hat{\theta}_t\end{pmatrix}\hat{u}_t,\ \frac{d}{dt}\hat{\omega}_t = 0,\ \frac{d}{dt}\hat{u}_t = 0 \tag{3.21}$$
$$\frac{d}{dt}\mathrm{P}_t = \tilde{\mathrm{A}}_t\mathrm{P}_t + \mathrm{P}_t\tilde{\mathrm{A}}_t + \mathrm{Q}_t \tag{3.22}$$
where Q_t is the covariance matrix of the Gaussian process noise w_t.

2. Update:
$$\mathrm{K}_n = \mathrm{P}_{t_n}\mathrm{H}(\mathrm{H}\mathrm{P}_{t_n}\mathrm{H}^{\mathrm{T}} + \mathrm{R}(\hat{\theta}_{t_n})\mathrm{N}_n\mathrm{R}(\hat{\theta}_{t_n})^{\mathrm{T}})^{-1} \tag{3.23}$$
$$z_n = \mathrm{R}(\hat{\theta}_{t_n})^{\mathrm{T}}(y_n - \hat{x}_t t_n) \tag{3.24}$$
$$e = \mathrm{K}_n z_n \tag{3.25}$$
$$\hat{\chi}^+_{t_n} = \chi_{t_n}\exp_m((e_{1:3})_\times),\ \hat{\omega}^+_{t_n} = \hat{\omega}_{t_n} + e_4,\ \hat{u}^+_{t_n} = \hat{u}_{t_n} + e_5 \tag{3.26}$$
$$\mathrm{P}^+_{t_n} = (\mathrm{I}_5 - \mathrm{K}_n\mathrm{H})\mathrm{P}_{t_n} \tag{3.27}$$
In these equations, K_n is the Kalman gain, z_n is the innovation, $\exp_m$ is the matrix exponential, $e = \begin{pmatrix}e_1 & e_2 & e_3 & e_4 & e_5\end{pmatrix}^{\mathrm{T}}$, $e_{1:3} = \begin{pmatrix}e_1 & e_2 & e_3\end{pmatrix}^{\mathrm{T}}$, and $\mathrm{R}(\hat{\theta}_{t_n}) = \begin{pmatrix}\cos(\hat{\theta}_{t_n}) & -\sin(\hat{\theta}_{t_n}) \\ \sin(\hat{\theta}_{t_n}) & \cos(\hat{\theta}_{t_n})\end{pmatrix}$.

3.5.2 Range and bearing observations

In general, 2D radar observations are provided in range and bearing (also called azimuth) coordinates. This means that the noises on the observations are Gaussian in the range and bearing coordinates, and the transformation into Cartesian coordinates is non-linear, so the noise does not stay Gaussian if we perform coordinate transformation. We thus need to adapt the filter's equations to these new non-linear observations.

Let us call r_n and α_n respectively the range and bearing observations provided at time n. The measurement expression is then:
$$y_n = \begin{pmatrix}r_n \\ \alpha_n\end{pmatrix} + v_n$$
with v_n a Gaussian white noise, of covariance matrix N_n. r_n and α_n can be obtained from the Cartesian position thanks to the following transformation:
$$\begin{pmatrix}r_n \\ \alpha_n\end{pmatrix} = h\begin{pmatrix}x^1_{t_n} \\ x^2_{t_n}\end{pmatrix} = \begin{pmatrix}\sqrt{(x^1_{t_n})^2 + (x^2_{t_n})^2} \\ \arctan\left(\frac{x^2_{t_n}}{x^1_{t_n}}\right)\end{pmatrix} \tag{3.28}$$
In order to compute the Kalman gain, we perform the linearisation of the function h. The computations are slightly different than for an EKF, but here the idea is the same, we linearise the measurement function to derive the Kalman equations.

Let us call $\tilde{\mathrm{H}}_n$ the linearised measurement matrix. Simple computations show that
$$y_n - h(\hat{x}^1_{t_n}, \hat{x}^2_{t_n}) = \tilde{\mathrm{H}}_n\xi_{t_n} + \mathrm{O}\left(||\xi_{t_n}||^2\right)$$
and
$$\tilde{\mathrm{H}}_n = \nabla h_{\hat{x}_{t_n}}\mathrm{R}(\hat{\theta}_{t_n})\begin{pmatrix}0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0\end{pmatrix} \tag{3.29}$$
with $\nabla h_{\hat{x}_{t_n}}$ the Jacobian of h at the point $\hat{x}_{t_n}$ defined by
$$\nabla h_{\hat{x}_{t_n}} = \begin{pmatrix}-\hat{x}^1_{t_n}/r_n & \hat{x}^2_{t_n}/r_n \\ -\hat{x}^2_{t_n}/r_n^2 & \hat{x}^1_{t_n}/r_n^2\end{pmatrix} \tag{3.30}$$

The propagation step of the IEKF (equations (3.21) and (3.22)) is not modified, however the update step is slightly modified as follows:

$$\mathrm{K}_n = \mathrm{P}_{t_n}\tilde{\mathrm{H}}_n(\tilde{\mathrm{H}}_n\mathrm{P}_{t_n}\tilde{\mathrm{H}}_n^{\mathrm{T}} + \mathrm{N}_n)^{-1} \tag{3.31}$$

$$z_n = y_n - h(\hat{x}_{t_n}) \tag{3.32}$$

$$e = \mathrm{K}_n z_n \tag{3.33}$$

$$\hat{\chi}_{t_n}^+ = \chi_{t_n}\exp_m((e_{1:3})_\times),\ \hat{\omega}_{t_n}^+ = \hat{\omega}_{t_n} + e_4,\ \hat{u}_{t_n}^+ = \hat{u}_{t_n} + e_5 \tag{3.34}$$

$$\mathrm{P}_{t_n}^+ = (\mathrm{I}_5 - \mathrm{K}_n\tilde{\mathrm{H}}_n)\mathrm{P}_{t_n} \tag{3.35}$$

Note that only the equations of the Kalman gain (3.23) and the innovation (3.24) have changed into (3.31) and (3.32), and that we replaced H by $\tilde{\mathrm{H}}_n$. The innovation has changed because we transform the Cartesian position estimate into range and bearing estimates. The Kalman gain equation has changed because we do not transpose the observation noise covariance matrix into the Frenet-Serret frame. Indeed, this matrix is now expressed in the range and bearing coordinates, and the rotation to the Frenet frame is already taken into account in the linearised observation matrix $\tilde{\mathrm{H}}_n$. In fact, instead of performing the transformation on the observations (that are now non-linear), we perform it on the state to express it in the range and bearing coordinates.

3.5.3 Comparison with an EKF derived from the same target model

In 2D, the equations can be expressed in a vectorial form as well as in the matricial form. This allows us to derive an EKF for this same model. We shall thus see the differences between the two filters, both theoretically and experimentally.

Let us recall the vectorial 2D equations:

$$\begin{cases} \dfrac{d}{dt}\theta_t = \omega_t + w_t^\theta \\ \dfrac{d}{dt}x_t = \begin{pmatrix}\cos(\theta_t)\\ \sin(\theta_t)\end{pmatrix}(u_t + w_t^x) \\ \dfrac{d}{dt}\omega_t = 0 + w_t^\omega \\ \dfrac{d}{dt}u_t = 0 + w_t^u \end{cases} \tag{3.36}$$

Let us call

$$f : \begin{cases} \mathbb{R}^5 \to \mathbb{R}^5 \\ x \mapsto \begin{pmatrix} x_4 \\ x_5\cos x_1 \\ x_5\sin x_1 \\ 0 \\ 0 \end{pmatrix} \end{cases}$$

The Jacobian for this evolution function can be easily computed:

$$\nabla f_{\hat{\mathrm{X}}_t} = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 \\ -\sin(\hat{\theta}_t)\hat{u}_t & 1 & 0 & 0 & \cos\hat{\theta}_t \\ \cos(\hat{\theta}_t)\hat{u}_t & 0 & 1 & 0 & \sin\hat{\theta}_t \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Let us call r_n the range as in (3.28). The Jacobian of the measurement function h for the range and bearing measurements is given by (3.30). As the Jacobians can be computed, an EKF can then be derived to perform state estimation.

To sum up, we can draw a chart, presented in table 3.1, that compares the main characteristics of the EKF and the IEKF. To simplify, we assume here that the state for the IEKF can be fully written

as a matrix χ_t. It is possible to add the angular velocity ω_t and the norm of the velocity u_t in the matrix, and to compute the associated evolution matrix ν_t. This gives

$$\chi_t = \begin{pmatrix} \cos\theta_t & -\sin\theta_t & x_t^1 & 0 & u_t \\ \sin\theta_t & \cos\theta_t & x_t^2 & \omega_t & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \ \nu_t = \begin{pmatrix} 0 & -\omega_t & u_t & 0 & 0 \\ \omega_t & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

However, this formulation is only used to draw a comparison between the EKF and the IEKF, and it is preferable to use $\chi \in \mathrm{SE}(2)$ to derive the algorithm, even if the two formulations are strictly equivalent in terms of computations.

	EKF (vectorial formulation)	**IEKF (matricial formulation)**
Kinematic model	$\frac{d}{dt}\mathrm{X}_t = f(\mathrm{X}_t, w_t)$	$\frac{d}{dt}\chi_t = \chi_t(\nu_t + (w_t)_\times)$
State prediction	$\frac{d}{dt}\hat{\mathrm{X}}_t = f(\hat{\mathrm{X}}_t)$	$\frac{d}{dt}\hat{\chi}_t = \hat{\chi}_t\hat{\nu}_t$
Error definition	$\eta_t = \hat{\mathrm{X}}_t - \mathrm{X}_t$	$\eta_t = \chi_t^{-1}\hat{\chi}_t$
Error evolution	$\frac{d}{dt}\eta_t = f(\hat{\mathrm{X}}_t) - f(\mathrm{X}_t)$	$\frac{d}{dt}\eta_t = \eta_t\hat{\nu}_t - \nu_t\eta_t - (w_t)_\times\eta_t$
Linearised error	$\frac{d}{dt}\xi_t = \mathrm{F}_t\xi_t + \mathrm{Q}_t$	$\frac{d}{dt}\xi_t = \mathrm{A}_t\xi_t + \mathrm{Q}_t$, A_t indep. of $\hat{x}_t$, $\hat{\theta}_t$
Covariance prediction	$\frac{d}{dt}\mathrm{P}_t = \mathrm{F}_t\mathrm{P}_t + \mathrm{P}_t\mathrm{F}_t + \mathrm{Q}_t$	$\frac{d}{dt}\mathrm{P}_t = \mathrm{A}_t\mathrm{P}_t + \mathrm{P}_t\mathrm{A}_t + \mathrm{Q}_t$

Table 3.1 – Comparison between the EKF and the IEKF equations

The main difference lies in the definition of the error η_t and in its linearisation. the other equations are only transposition of the vectorial equations in the Lie group/algebra setting.

3.5.4 Discussion

For the IEKF, the linearised error evolution does not depend on the predicted position nor on the predicted orientation. In the target tracking problem, it depends on the predicted angular and tangential velocities $\hat{\omega}_t$ and $\hat{u}_t$. It is thus not possible to derive any convergence properties for the IEKF for this application, contrary to the application presented in [11]. This will be discussed in more details in section 3.6, with the algorithm for the 3D target model.

The EKF performs a linearisation around the predicted state, assuming the error made by the prediction (and the previous estimation) is sufficiently low. However, this can lead to divergence. Indeed, if the prediction is in fact far from the real state, then the linearisation is no longer valid, and can induce divergence, because the covariance matrix becomes inaccurate. Indeed, it might not reflect the uncertainty of the filter, since it is based on an approximation.

For the update part, in the range and bearing case, the IEKF works as the EKF, meaning that we need to linearise the observation function, and compute its Jacobian. This cannot be avoided, since the range and bearing observations cannot be embedded in the Lie group setting. However, we shall see in section 3.8 that we can replace the update step by the one of an UKF, for both the EKF and the IEKF.

In the next section, we consider the 3D Frenet-Serret target model, and derive the equations. We add more arguments for the use of the IEKF with such a model, and present the derivations with a different viewpoint.

3.6 IEKF applied to the 3D Frenet model

The proposed model (2.33) seems well suited to the tracking of highly maneuvering targets. Indeed it is closely related to constant commands in the body frame of the target, albeit different. Devising a filter to track equations (2.32), or more precisely the retained model (2.33), from position measurements is difficult for various reasons. First, because there are many quantities to be estimated only from position measurements and which appear non-linearly in the equations. Then, because of the constraint of T, N, B forming an orthonormal base of the 3D space. It is not trivial to encode such non-linear constraints in an extended Kalman filter. Using the model (2.33), we see that the pair (x_t, R_t) defines an element of the Lie group SE(3). This indicates (part of) the state could be estimated by bringing to bear the Lie group structure underlying the state space. There has been extensive work on observers on Lie groups over the past years, see [21], [23], [74] or [73]. However, for target tracking applications it is important that the filter outputs a covariance matrix, preventing the use of deterministic observers. Indeed, radar tracking involves the non-trivial task of associating radar reports (the plots) with actual target tracks. This task (which is not considered in the present thesis) is critical and requires an acceptance gate based on the estimated error dispersion (covariance). This is why we will resort to a Lie group based extended Kalman filter - the invariant EKF of [11] - for the tracking task.

The model considered is (2.34):

$$\frac{d}{dt}\chi_t = \chi_t(\nu_t + \mathscr{L}_{\mathfrak{se}(3)}(w_t^\chi)),\ \frac{d}{dt}\zeta_t = 0 + w_t^\zeta \tag{3.37}$$

with

$$\chi_t = \begin{pmatrix} \mathrm{R}_t & x_t \\ 0_{1,3} & 1 \end{pmatrix},\ \zeta_t = \begin{pmatrix} \gamma_t \\ \tau_t \\ u_t \end{pmatrix}$$

and

$$\nu_t = \begin{pmatrix} 0 & -\gamma_t & 0 & u_t \\ \gamma_t & 0 & -\tau_t & 0 \\ 0 & \tau_t & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

3.6.1 Similarities with the Invariant theory

χ_t verifies an equation of the type

$$\frac{d}{dt}\chi_t = f_{i_t}(\chi_t) + \chi_t w_t \tag{3.38}$$

where $i_t = \zeta_t \in \mathbb{R}^3$ is a known input (let us assume ζ_t is known for the moment). w_t is a continuous white Gaussian noise, and $f : \chi_t \to \chi_t \nu_t$ satisfies the condition:

$$f_i(ab) = a f_i(b) + f_i(a) b - a f_i(\mathrm{Id}) b \tag{3.39}$$

for all $(i, a, b) \in \mathbb{R}^3 \times \mathrm{SE}(3) \times \mathrm{SE}(3)$. An IEKF can thus be designed to estimate χ_t.

In the case where the input ζ_t is not known (for the tracking problem for instance), the algorithm can be adapted to treat χ_t as a Lie group part, and ζ_t as a standard vectorial part, as was done for the 2D case. The system thus satisfies (3.40).

$$\begin{cases} \frac{d}{dt}\chi_t = f_\zeta(\chi_t, \zeta_t) + \chi_t w_t^\chi \\ \frac{d}{dt}\zeta_t = g(\zeta_t) + w_t^\zeta \end{cases} \tag{3.40}$$

where in our case

$$f_\zeta : \begin{pmatrix} \mathrm{R} & x \\ 0_{1,3} & 1 \end{pmatrix} \to \begin{pmatrix} \mathrm{R}(\omega)_\times & \mathrm{R}v \\ 0_{1,3} & 0 \end{pmatrix},\ g(\zeta) = 0_{3,1}$$

We can also write the Cartesian observations (2.24) with the help of the Lie group setting, and with $v_n \in \mathbb{R}^3$ being a white Gaussian noise:

$$y_n = \chi_{t_n} \begin{pmatrix} 0_{3,1} \\ 1 \end{pmatrix} + \begin{pmatrix} v_n \\ 0 \end{pmatrix}$$

The condition (3.39) is easily verified.

3.6.2 Derivation of the algorithm

Error definition

The classical definition of the error for a vector state X_t, $\eta_t = \hat{X}_t - X_t$ does not hold here. Indeed, if χ_1 and χ_2 belong to the Lie group SE(3), there is no reason why $\chi_1 - \chi_2$ should also belong to this same Lie group. We define the error differently depending on which part of the state we are considering:

$$\eta_t^\chi = \chi_t^{-1} \hat{\chi}_t \in \mathrm{SE}(3)$$

$$\eta_t^\zeta = \hat{\zeta}_t - \zeta_t \in \mathbb{R}^3$$

More explicitly, the global error $\eta_t = (\eta_t^\chi, \eta_t^\zeta)$ is defined as:

$$\eta_t = \begin{pmatrix} \eta_t^R \\ \eta_t^x \\ \eta_t^\gamma \\ \eta_t^\tau \\ \eta_t^u \end{pmatrix} = \begin{pmatrix} R_t^T \hat{R}_t \\ R_t^T(\hat{x}_t - x_t) \\ \hat{\gamma}_t - \gamma_t \\ \hat{\tau}_t - \tau_t \\ \hat{u}_t - u_t \end{pmatrix} \tag{3.41}$$

where $\eta_t^R \in \mathrm{SO}(3)$, $\eta_t^x \in \mathbb{R}^3$, $\eta_t^\gamma \in \mathbb{R}$, $\eta_t^\tau \in \mathbb{R}$ and $\eta_t^u \in \mathbb{R}$.

Linearisation of the error and propagation step

The propagation equations are:

$$\frac{d}{dt}\hat{\chi}_t = f_{\zeta_t}(\hat{\chi}_t) = \hat{\chi}_t \hat{\nu}_t, \quad \frac{d}{dt}\hat{\zeta}_t = g(\zeta_t) = 0 \tag{3.42}$$

Now let us assume once again (for the last time) that ζ_t is known. We can compute the error η_t^χ evolution. This gives:

$$\frac{d}{dt}\eta_t^\chi = \eta_t^\chi \nu_t - \nu_t \eta_t^\chi - \mathscr{L}_{\mathfrak{se}(3)}(w_t^\chi)\eta_t^\chi$$

This equation has the particular property that it does not depend on the predicted state $\hat{\chi}_t$ at all. This is due to the property (3.39) of the evolution function f.

As for our radar tracking application ζ_t is not known, the evolution of η_t^χ is slightly modified and it writes

$$\frac{d}{dt}\eta_t^\chi = -\nu_t \eta_t^\chi - \mathscr{L}_{\mathfrak{se}(3)}(w_t^\chi)\eta_t^\chi + \eta_t^\chi \hat{\nu}_t \tag{3.43}$$

indeed, the matrix ν_t depends on the vector ζ_t, so it has to be estimated as well. The evolution of η_t^ζ is more conventional:

$$\frac{d}{dt}\eta_t^\zeta = 0 + w_t^\zeta \tag{3.44}$$

To linearise equations (3.43) and (3.44), see [11], or refer to section 3.4.3, we let $\eta_t^R = I_3 + (\xi_t^R)_\times$. This means that $\xi_t^R \in \mathbb{R}^3$ is a small instantaneous rotation vector. We also let $\xi_t^x = \eta_t^x$, $\xi_t^\gamma = \eta_t^\gamma$, $\xi_t^\tau = \eta_t^\tau$ and $\xi_t^u = \eta_t^u$. Then we mimic the EKF methodology, and perform a first order linearisation in the components of ξ, and we also neglect terms of order $||\xi||.||w||$. To do this, we use a property

of Lie groups: $(\xi_t^{\mathrm{R}})_\times(\hat{\omega}_t)_\times - (\hat{\omega}_t)_\times(\xi_t^{\mathrm{R}})_\times = (\xi_t^{\mathrm{R}} \times \hat{\omega}_t)_\times$. This allows to identify the term $\frac{d}{dt}\xi_t^{\mathrm{R}}$ using $(a)_\times = (b)_\times \implies a = b$. We use the same denomination for ξ as for η:

$$\xi_t = \begin{pmatrix} \xi_t^{\mathrm{R}} \\ \xi_t^{x} \\ \xi_t^{\mathrm{v}} \\ \xi_t^{\tau} \\ \xi_t^{u} \end{pmatrix} \in \mathbb{R}^9$$

During the propagation step, the error evolves as:

$$\frac{d}{dt}\xi_t = \mathrm{A}_t\xi_t + w_t$$

with

$$\mathrm{A}_t = -\begin{pmatrix} 0 & -\hat{\mathrm{v}}_t & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ \hat{\mathrm{v}}_t & 0 & -\hat{\tau}_t & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \hat{\tau}_t & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -\hat{\mathrm{v}}_t & 0 & 0 & 0 & -1 \\ 0 & 0 & -\hat{u}_t & \hat{\mathrm{v}}_t & 0 & -\hat{\tau}_t & 0 & 0 & 0 \\ 0 & \hat{u}_t & 0 & 0 & \hat{\tau}_t & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

This permits to write the covariance evolution during the propagation step: the covariance matrix evolves with the Riccati equation (3.45).

$$\frac{d}{dt}\mathrm{P}_t = \mathrm{A}_t\mathrm{P}_t + \mathrm{P}_t\mathrm{A}_t^{\mathrm{T}} + \mathrm{Q}_t \tag{3.45}$$

Remark 3.5. A_t is a sparse matrix. Although the model is non-linear, it only depends weakly on the trajectory, because it depends only on intrinsic quantities, and not on the position or the velocity vectors that are linked to the choice of a terrestrial reference frame.

Gain computation and update step

An observation $y_n = x_{t_n} + v_n$ is available at time t_n. The update of the state writes:

$$\hat{\chi}_{t_n}^{+} = \hat{\chi}_{t_n}\exp(\mathrm{L}_n^{\chi}(\hat{\chi}_{t_n}^{-1}y_n)),\ \hat{\zeta}_{t_n}^{+} = \hat{\zeta}_{t_n} + \mathrm{L}_n^{\zeta}(\hat{\chi}_{t_n}^{-1}y_n) \tag{3.46}$$

more explicitly, this can be developed as:

$$\begin{pmatrix} \hat{\mathrm{R}}_{t_n}^{+} \\ \hat{x}_{t_n}^{+} \\ \hat{\mathrm{v}}_{t_n}^{+} \\ \hat{\tau}_{t_n}^{+} \\ \hat{u}_{t_n}^{+} \end{pmatrix} = \begin{pmatrix} \hat{\mathrm{R}}_{t_n}\exp_m[(\delta_\omega)_\times] \\ \hat{x}_{t_n} + \hat{\mathrm{R}}_{t_n}\mathrm{B}(\delta_\omega)\delta_x \\ \hat{\mathrm{v}}_{t_n} + \delta_{\mathrm{v}} \\ \hat{\tau}_{t_n} + \delta_\tau \\ \hat{u}_{t_n} + \delta_u \end{pmatrix} \tag{3.47}$$

where $\begin{pmatrix}\delta_\omega & \delta_x & \delta_{\mathrm{v}} & \delta_\tau & \delta_u\end{pmatrix}^{\mathrm{T}} = \mathrm{L}_n(\hat{\mathrm{R}}_{t_n}(y_n - \hat{x}_{t_n}))$, $\exp_m$ denotes the matrix exponential map in $\mathscr{M}_3$, and

$$\mathrm{B}(\delta_\omega) = \mathrm{I}_3 + \frac{1-\cos||\delta_\omega||}{||\delta_\omega||^2}(\delta_\omega)_\times + \frac{\delta_\omega - \sin||\delta_\omega||}{||\delta_\omega||^3}[(\delta_\omega)_\times]^2$$

The gain matrix $\mathrm{L}_n \in \mathbb{R}^{9\times 3}$ derives from with the Riccati equation (3.45), as will be explained in the following. The innovation is defined as $\hat{\chi}_{t_n}^{-1}y_n = \hat{\mathrm{R}}_{t_n}^{\mathrm{T}}(y_n - \hat{x}_{t_n})$, it verifies:

$$\hat{\mathrm{R}}_{t_n}^{\mathrm{T}}(y_n - \hat{x}_{t_n}) = \hat{\mathrm{R}}_{t_n}^{\mathrm{T}}(x_{t_n} - \hat{x}_{t_n}) + \hat{\mathrm{R}}_{t_n}^{\mathrm{T}}v_n = -(\eta_{t_n}^{\mathrm{R}})^{-1}\eta_{t_n}^{x} + \hat{\mathrm{R}}_{t_n}^{\mathrm{T}}v_n$$

thus, as $\eta^x_t = \xi^x_t$ and $(\xi^R_t)_\times \xi^x_t$ is of order two, the linearisation gives $\hat{R}^T_t(y_n - \hat{x}_{t_n}) \approx -H\xi + \hat{R}^T_{t_n} \upsilon_n$, where $H \in \mathbb{R}^{3\times 9}$ is defined as $H = \begin{pmatrix} 0_{3,3} & I_3 & 0_{3,3} \end{pmatrix}$. We are now able to derive the update ξ^+_t of ξ_t from (3.47) and (3.41), and using the fact that the first order approximation of B be the identity, which gives:

$$\xi^+_{t_n} = \xi_{t_n} - L_n \left[H\xi_{t_n} - \hat{R}^T_{t_n} \upsilon_n \right]$$

Finally , the Kalman gain can be computed with the full Riccati equations, with Q_t and N_n the process and measurement noise covariances respectively:

$$\begin{aligned} \frac{d}{dt} P_t &= A_t P_t + P_t A^T_t + Q_t \\ S_n &= H P_{t_n} H^T + \hat{R}^T_{t_n} N_n \hat{R}_{t_n} \\ L_n &= P_{t_n} H^T S^{-1}_n \\ P^+_{t_n} &= (I_9 - L_n H) P_{t_n} \end{aligned}$$

Summary of the filter's equations

To sum up the results obtained before, we can write extensively the filter's equations for cartesian coordinates observations.

1. Propagation step:
   - Solve $\frac{d}{dt}\hat{\chi}_t = \hat{\chi}_t \hat{\nu}_t$ and $\frac{d}{dt}\hat{\zeta}_t = 0$
   - Solve the Riccati equation $\frac{d}{dt} P_t = A_t P_t + P_t A^T_t + Q_t$
2. Update step:
   - Compute the innovation $z_n = \hat{R}_{t_n}(y_n - \hat{x}_{t_n})$
   - Compute the Kalman gain $L_n = P_{t_n} H (H P_{t_n} H^T + \hat{R}^T_{t_n} N_n \hat{R}_{t_n})$
   - Update the state $\hat{\chi}^+_{t_n} = \hat{\chi}_{t_n} \exp(L_n z_n)_{1:6}$ and $\hat{\zeta}^+_{t_n} = \hat{\zeta}_{t_n} + (L_n z_n)_{7:9}$
   - Update the covariance $P^+_{t_n} = (I_9 - L_n H) P_{t_n}$

3.6.3 Discussion on the filter's expected stability

The following proposition show our filter possesses convergence properties conditionally on the Frenet parameters u_t, γ_t and τ_t, which is an indication of stability for the full state estimation.

Proposition 3.2. *Consider model* (2.33) *with noise turned off, measurements with noise turned off too, and assume that the quantities* u_t, γ_t, τ_t *are known at all times, and may be freely varying inside an interval* (α, β) *with* $\alpha, \beta > 0$. *Assume also that the lowest eigenvalues of* Q_t *and* N_n *are lower bounded by some* $\epsilon > 0$. *Then the IEKF proposed above when reduced to the task of estimating* x_t *and* R_t, *locally converges around* any *trajectory of the (non-linear) system.*

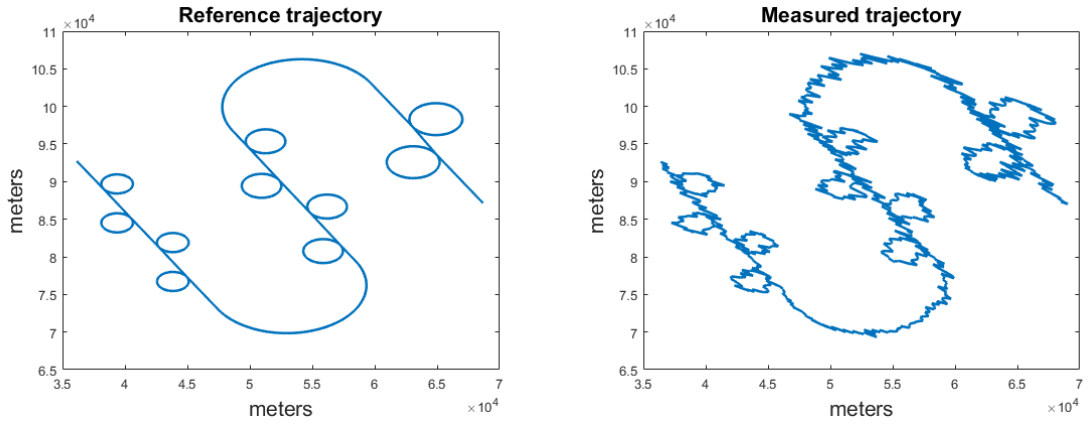
Proof. The reduced system can be viewed as a non-holonomic car in 3D, that is, evolving in SE(3) instead of SE(2). The result appears then to be a simple extension of the simplified car example of [11]. □

3.7 Simulations

In the previous sections, we derived the filter equations for the tracking problem. In the present section, we test the IEKF algorithm on trajectories with motions close to real challenging tracking problems, simulated using the Frenet-Serret 2D and 3D models, with jumps in the parameters. Other simulations with more trajectories are presented in chapter 4. The 2D model is compared here with the EKF on the same target model.

3.7.1 2D simulations and comparison with the EKF on the same target model

In this section, we compare the IEKF on the 2D Frenet-Serret model (2.30), with an EKF built on the same model, as proposed in section 3.5.3.



(a) (b)

Figure 3.3 – Reference trajectory fig. 3.3a and trajectory with measurement noise added in polar coordinates fig. 3.3b. The simulated trajectory is strongly inspired by real data from a fighter jet.

The trajectory presented in this section is synthetic, and inspired by a real fighter trajectory. The trajectory was made using a trajectory simulator, with its own kinematic model. The measurements are in meters and give the Cartesian coordinates, and the radar is centred at the origin. The measures were manually transformed in range and bearing coordinates with the function h of equation (3.28). Some realistic measurement noise in range and bearing was also added by hand, as an additive noise.

The trajectory of the target is presented on figure 3.3. The amplitude of the measurement noise is visible on figure 3.3b. The results are presented for a measurement noise close to real measurement noises, and we see indeed that the noise is greater when the target is far from the radar (located in $(0,0)$), and that it is mostly spread in the angular direction. The orientation angle θ is modulo 2π. On figure 3.4, the results are presented for an EKF with the 2D Frenet-Serret target model, and on figure 3.5, the results are presented for the IEKF. The process noises are optimised for this particular trajectory by maximising the measurement likelihood, with the method proposed in [1]. Indeed, in the model, ω_t and u_t are supposed to be constant, so we need to add some sufficiently high process noise to these variables to account for their variations. The Root Mean Square Errors (RMSE) for each parameter are presented in table 3.2. The IEKF performs globally better, especially for the estimation of the norm of the velocity, the orientation and the angular velocity. This is corroborated by the results on figures 3.4 and 3.5, where we see that the velocity estimation is more precise for the IEKF, and the angular velocity converges faster after a jump. The position is less critical to compare the filters accuracy. More complete experiments will be presented in chapter 4.

This experiment shows also that the target model expressed in intrinsic coordinates is suitable for manoeuvring targets. Indeed, the filtering works quite well on this trajectory that is close to the one of a real fighter. This trajectory has accelerations up to $5g$. The filters are moreover compatible with realistic measurement noise.

3.7.2 3D simulations

Helical trajectory

To simulate a 3D trajectory, we describe the evolution of a manoeuvring target by (2.33). We add some measurement Gaussian noise, on Cartesian measurements. The noise added is greater along the third coordinate, x^3 (as it is the case in practice), the standard deviations for the measurement

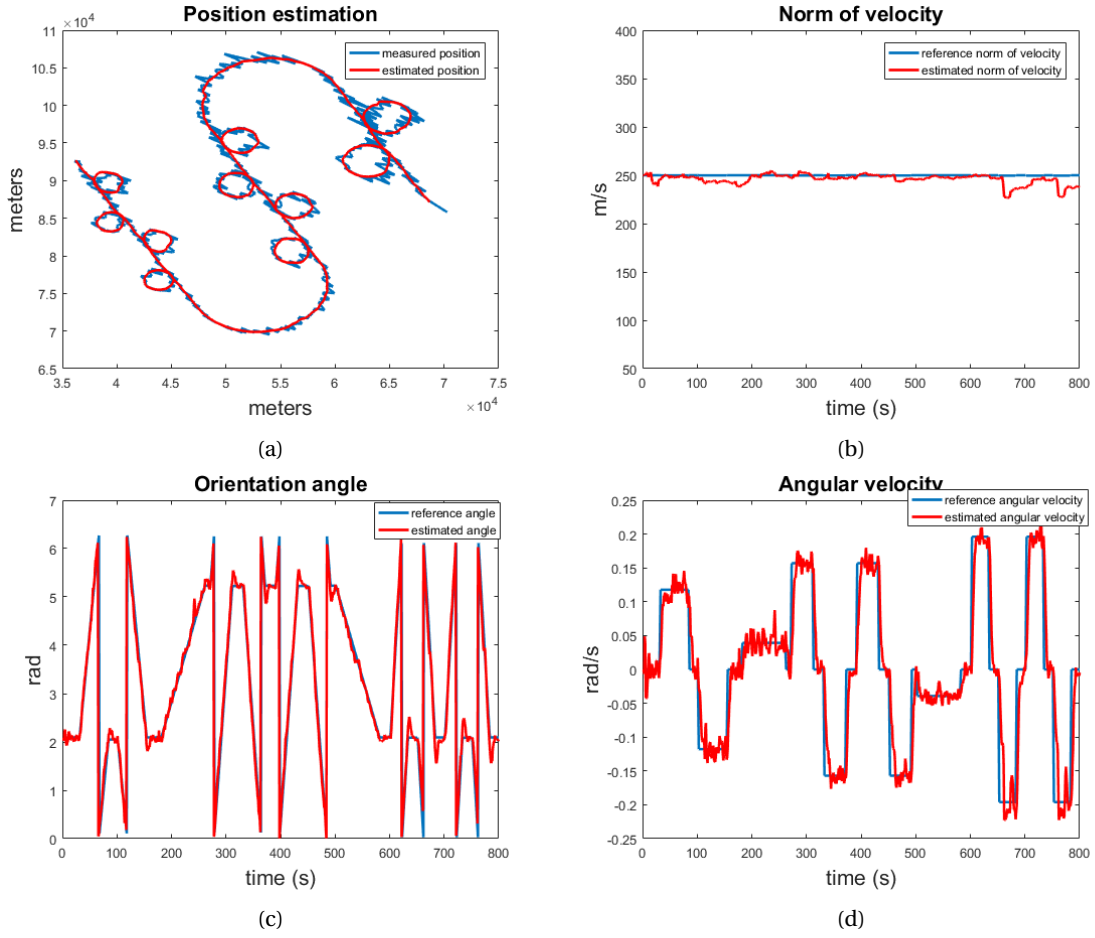


Figure 3.4 – EKF tests. Estimations (in red) of the position fig. 3.4a, of the norm of the velocity fig. 3.4b, of the orientation angle fig. 3.4c, and of the angular velocity fig. 3.4d. The true values for the parameters are in blue.

Parameter	EKF	IEKF
x (m)	143	193
θ (RMSE for $1-\cos\theta$)	0.0243	0.0193
ω (rad/s)	0.044	0.038
u (m/s)	7.2	2.5

Table 3.2 – RMSE for each parameter on 100 Monte-Carlo simulations and on the whole trajectory for an EKF and an IEKF with optimised process noise. We see the IEKF recovers with much more accuracy the difficult to track parameters θ, ω and u which are derivatives of the position x, which is the measured quantity. The EKF has a better estimate for x, but this is because it "sticks" more to measured data. As in modern radars x is accurately measured, and may be directly used for target localisation, the interest of a filter is above all to provide the radar with accurate estimates of the velocity and if possible curvature. This allows the radar to anticipate the motion of the target.

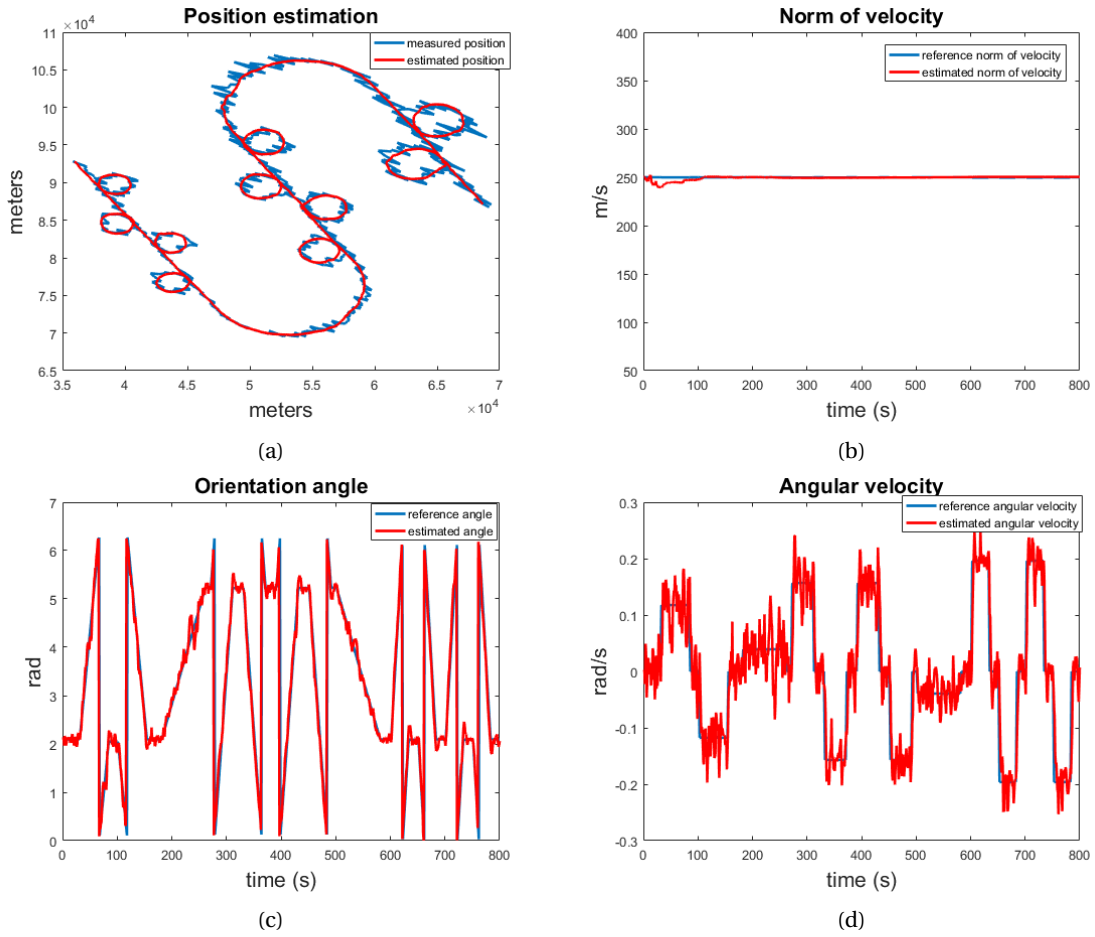
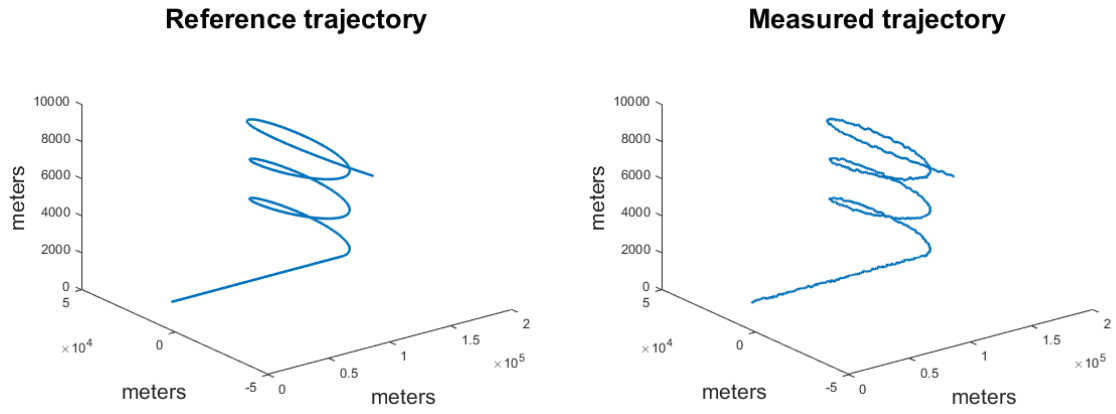


Figure 3.5 – IEKF tests. Estimations (in red) of the position fig. 3.5a, of the norm of the velocity fig. 3.5b, of the orientation angle fig. 3.5c, and of the angular velocity fig. 3.5d. The true values for the parameters are in blue.

noise are $20, 20, 40$ for the coordinates x^1, x^2, x^3 respectively. The reference trajectory and the noisy measured trajectory are presented on figure 3.6. We have simulated two consecutive manoeuvres. The first one switches from a constant velocity straight line motion to a helical motion with non-zero (but nevertheless constant) curvature and torsion, along with a jump in the norm of the velocity. The second manoeuvre switches from this last motion to a motion with another constant curvature. This last motion lies in a plane (the torsion is zero). Such trajectories can be truly encountered in current air defence applications, and are considered as very challenging to track, as the acceleration may go up to $16g$.



(a) (b)

Figure 3.6 – Reference trajectory fig. 3.6a and trajectory with measurement noise added in Cartesian coordinates fig. 3.6b

We then perform filtering with the IEKF algorithm presented in section 3.6. the estimations of the position x, of the norm of the velocity u, of the curvature κ and of the torsion $\tilde{\tau}$ are displayed on figure 3.7. Note that we plot here the curvature and the torsion that correspond to the usual definitions, and that appear in (2.31).

The norm of the velocity converges quite fast after initialisation, or after a jump. The curvature is also quite well estimated, even though it is a little less precise than the norm of the velocity. The torsion is the most difficult parameter to estimate, and although it eventually converges, it is not well estimated by the filter. This stems from the fact that it is weakly observable.

To understand why the convergence of the curvature and even more of the torsion are slower, let us look at how they are derived from the position. The equations are given in (3.48) and (3.49) for the curvature and the torsion respectively, with $. \times .$ the cross product, and $[.,.,.]$ the scalar triple product. These equations underline that the curvature comes from a second derivative of the position, and the torsion from a third derivative of the position. This shows that the torsion is hardly observable when it is estimated from noisy position measurements.

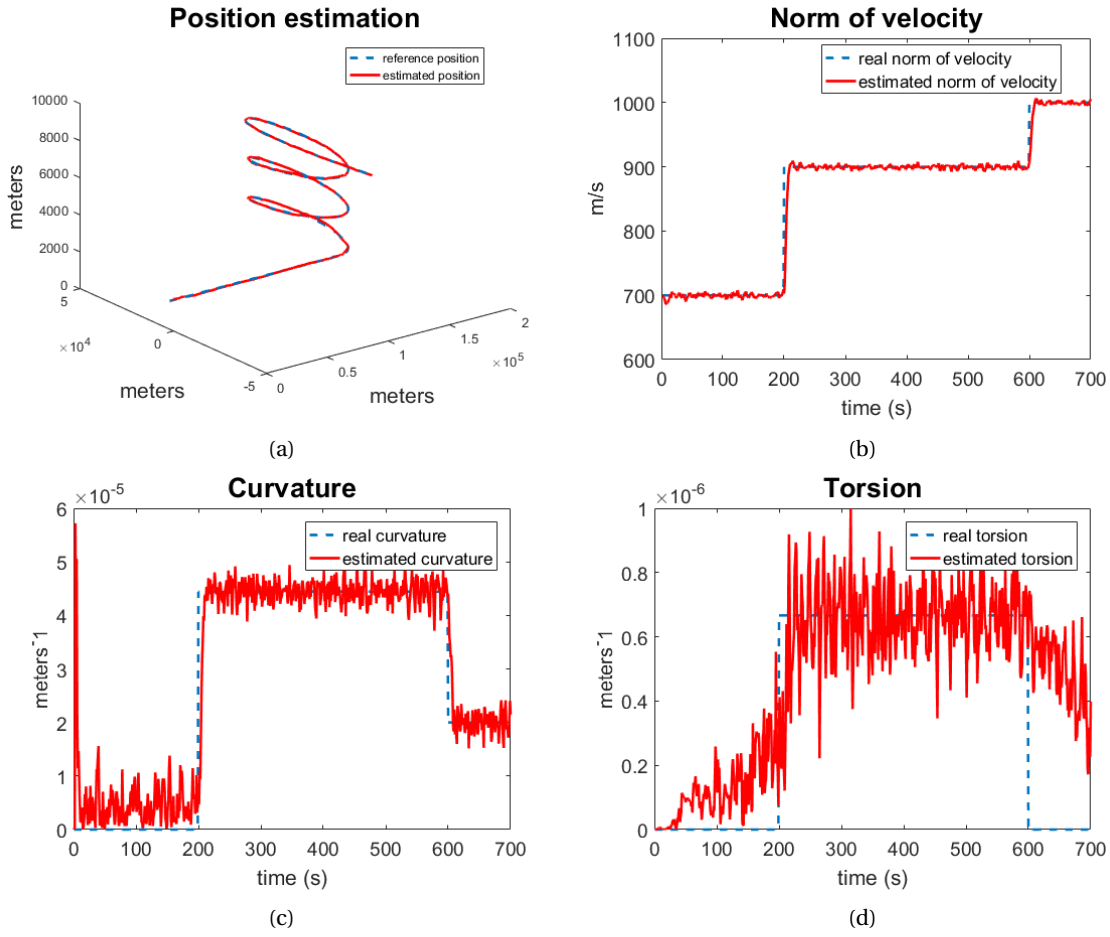
$$\kappa = \frac{||x'(t) \times x''(t)||}{||x'(t)||^3} \tag{3.48}$$

$$\tilde{\tau} = \frac{[x'(t), x''(t), x'''(t)]}{||x'(t) \times x''(t)||^2} \tag{3.49}$$

The process noise has to be adjusted when confronted to this kind of trajectory with jumps. A discussion on the tuning of this process noise is provided in section 4.3. Indeed, a balance has to be found between the accuracy wanted for the estimations and the ability to converge after a jump.

Comparison with other filtering algorithms

Few filtering algorithms are designed to estimate this kind of target model. Indeed, most of the time the models are expressed in two dimensions, or are linear in three dimensions. The well-



(a) (b) (c) (d)

Figure 3.7 – Estimations (in red) of the position fig. 3.7a, of the norm of the velocity fig. 3.7b, of the curvature fig. 3.7c, and of the torsion fig. 3.7d. The true values for the parameters are in blue.

Parameter	IEKF	Castella
x^1	15.8	18.4
x^2	15.3	18.0
x^3	26.0	24.4
$\dot{x}^1$	7.5	13.4
$\dot{x}^2$	8.5	12.6
$\dot{x}^3$	8.9	7.1

Table 3.3 – Comparison of the RMSE for the position and the velocity for the IEKF and Castella's Kalman filter

known IMM of [6] uses 2D turn models, and so is the one presented in [19]. It is also difficult to build an Extended Kalman filter in the 3D case from the Frenet-Serret equations above, because there are orthogonality constraints for the Frenet-Serret frame that cannot be easily dealt with the standard EKF formulation. In chapter 4, we nevertheless build an EKF from a model derived from the 3D Frenet-Serret one. But here, we show only a comparison with an algorithm using a well-known trick in industrial radars.

This trick is called the process noise adaptation of Castella, from the author of [35]. Just to have an idea of what it might yield, we have compared the RMSE of both filters in table 3.3. The model used with the Castella noise adaptation is a constant acceleration model, with a linear Kalman filter. The noises have been tuned manually to get the best possible results with both filters. The Castella seems to slightly perform better for the third coordinate x^3. It is due to the fact that the noise along this coordinate is higher, and the estimations of Castella's Kalman filter tend to follow the measurements. However, for the velocity and the other position coordinates, the IEKF gives better results.

3.8 Left-invariant UKF on a 2D model

Let us go back to the 2D target model (2.30). We consider in this section range and bearing measurements. These are non-linear measurements, and their form does not allow us to use the invariant theory, so the update step is close to the one of an EKF, with the same stability problems. To avoid that, we can resort to the Unscented Kalman Filter (UKF), presented in the bibliographic part of this chapter. The filter has to be adapted to the Lie group formulation. We present in this section how to use the update step of a left-invariant UKF along with the propagation step of our IEKF algorithm on the 2D Frenet-Serret target model.

Indeed, the strength of the IEKF is that in a perfect theoretical setting, the linearisations (they intervene in the equations as A_t for the propagation step and as H_n for the update step) do not depend on the predicted state $(\hat{\chi}_t, \hat{\omega}_t, \hat{u}_t)$. In the equations of the IEKF with polar measurements however, we see that with our model, the matrices A_t and H_n depend on the predicted state. For A_t, this does not seem too preoccupying, since it only depends on $\hat{\omega}_t$ and $\hat{u}_t$, and not directly on the position. However, for H_n, the problem is different, since it depends directly on $(\hat{x}_t^1, \hat{x}_t^2)$, and we have the same approximation and stability problems as for the EKF. We thus designed another method to avoid computing the Jacobian of h. The UKF update step seems most appropriate for this.

3.8.1 Derivation of the filter

The Unscented Kalman Filter allows to approximate the posterior (Gaussian) distribution $p(\mathrm{X}|\mathrm{Y})$ thanks to the use of so-called sigma points. This UKF is adapted here as in [30] to suit the Lie group formulation of the model, this adaptation is called the left-UKF (l-UKF). We combine the prediction step of the IEKF with the update step of the left-UKF.

Instead of performing a linearisation of the non-linear model, the unscented transform is used to pick a minimal set of sigma points around the mean state. These sigma points are updated through the non-linear function h, and a new mean and covariance are derived from this update, as in the regular UKF.

Let us call $\bar{\chi}$ the mean of the whole state put in matrix form, that is:

$$\bar{\chi} = \begin{pmatrix} \cos\bar{\theta} & -\sin\bar{\theta} & \bar{x}^1 & 0 & \bar{u} \\ \sin\bar{\theta} & \cos\bar{\theta} & \bar{x}^2 & \bar{\omega} & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.50}$$

We now define the augmented covariance as $\mathrm{P}_n^a = \mathrm{diag}(\mathrm{P}_n, \mathrm{N}_n)$, with N_n the covariance of the measurement noise at time n.

We then construct a set of $2\mathrm{L}+1$ sigma points (in our model $\mathrm{L}=7$, it is the dimension of the state augmented by the measurement noise), as in (3.51), and where λ is a scaling factor.

$$\begin{cases} \bar{\alpha} = [0^{\mathrm{T}}, \nu^{\mathrm{T}}] \\ \alpha_n^0 = \bar{\alpha} \\ \alpha_n^i = \bar{\alpha} + \left(\sqrt{(\mathrm{L}+\lambda)\mathrm{P}_n^a}\right), \ i = 1, \dots, \mathrm{L} \\ \alpha_n^i = \bar{\alpha} - \left(\sqrt{(\mathrm{L}+\lambda)\mathrm{P}_n^a}\right), \ i = \mathrm{L}+1, \dots, 2\mathrm{L} \end{cases} \tag{3.51}$$

Let us introduce ξ_n^i and ν_n^i such that $[\xi_n^i, \nu_n^i] = \alpha_n^i$, and our state at time n is $\bar{\chi}_n$. Then these sigma points go through the measurement function h:

$$y_n^i = h(\bar{\chi}_n \exp \xi_n^i) + \nu_n^i, i = 0, ..., 2\mathrm{L}$$

The expected measure is thus $\bar{y}_n = \sum_{i=0}^{2\mathrm{L}} \mathrm{W}_s^i y_n^i$. The values for the weights W_s^i can be found explicitly in [30], they are:

$$\mathrm{W}_s^0 = \frac{\lambda}{\lambda+\mathrm{L}}, \ \mathrm{W}_c^0 = \frac{\lambda}{\lambda+\mathrm{L}} + (2 - \frac{\lambda}{\mathrm{L}})$$

$$\mathrm{W}_s^i = \mathrm{W}_c^i = \frac{1/2}{\lambda+\mathrm{L}}$$

The state and covariance are then updated as:

$$\mathrm{P}_{yy} = \sum_{i=0}^{2\mathrm{L}} \mathrm{W}_c^i (y_n^i - \bar{y}_n)(y_n^i - \bar{y}_n)^{\mathrm{T}}$$

$$\mathrm{P}_{\alpha y} = \sum_{i=0}^{2\mathrm{L}} \mathrm{W}_c^i (\alpha_n^i - \bar{\alpha}_n)(y_n^i - \bar{y}_n)^{\mathrm{T}}$$

$$[\bar{\xi}_n^{\mathrm{T}}, *]^{\mathrm{T}} = \mathrm{P}_{\alpha y} \mathrm{P}_{yy}^{-1} (y_n - \bar{y}_n)$$

$$\chi_n^+ = \bar{\chi}_n \exp \bar{\xi}_n$$

$$\mathrm{P}_n^+ = \mathrm{P}_n - \mathrm{P}_{\alpha y} (\mathrm{P}_{\alpha y} \mathrm{P}_{yy}^{-1})^{\mathrm{T}}$$

The final filter, that we call the l-UKF (left-Unscented Kalman Filter), is composed of the propagation step of the IEKF (equations (3.21) and (3.22)) and of the left-UKF update step. This does not interfere with the consistency properties of the IEKF in the optimal setting, but this allows to get around the approximations of the measurement function linearisation.

Parameter	EKF	IEKF	l-UKF
x	143	193	191
θ (RMSE for $1-\cos\theta$)	0.0243	0.0193	0.0242
ω	0.044	0.038	0.038
u	7.2	2.5	1.9

Table 3.4 – RMSE for each parameter on 100 Monte-Carlo simulations and on the whole trajectory for an EKF, an IEKF, and an l-UKF with optimised process noise

3.8.2 Results

We take again the trajectory from section 3.7.1. We test the l-UKF on this trajectory, the RMSE for the EKF and the IEKF are recalled, along with the results for the l-UKF in table 3.4. The results are slightly better for the l-UKF, but not with a great order of magnitude.

This shows that the use of sigma points instead of the linearisation of the measurement function does not enhance the performances of the estimation a lot. The IEKF alone being simpler to implement, closer to the EKF which engineers are used to, and quite faster to run, we will thus use the IEKF in the sequel.

3.9 Conclusion

This chapter describes the construction of Kalman type filtering algorithms to perform state estimation with the new target models of chapter 2. Some discussion on the stability of the filters have been presented. Basic simulations have also been presented to show how the filters perform, and how they react to jumps in the trajectory. Indeed, as mentioned in chapter 2, the target model was built on assumptions of constant norm of velocity and constant turns. However, in reality, this is not the case, and we have shown that the filter can adapt to these differences between the target model and the real trajectories, thanks to the tuning of the process noise.

Comparing the filters on the same target model is not easy, due to the particular form of the model, especially in 3D. In the next chapter, we compare other filtering algorithms built on other models, but we also show how to derive an EKF from the 3D target model (allowing a slight change in the formulation of the model).

A more thorough discussion on the tuning of process noises will be provided in the next chapter. The process noises will no longer be tuned on the trajectories used to test the filters, but on a bank of other trajectories, that are used only for noise tuning. The implications of the process noise on the performances of the filters will also be presented.

Chapter 4

Comparison with other existing algorithms and models

Sommaire

4.1 Résumé en français : Comparaison avec d'autres modèles et algorithmes existants

Dans ce chapitre, la question du réglage des paramètres des filtres est évoquée, en particulier pour le réglage des bruits de modèle. En effet, si le bruit de modèle est très faible, l'estimation est très précise sur les parties correspondant parfaitement au modèle, mais les écarts entre la trajectoire réelle et le modèle théorique provoquent de mauvaises estimations. Dans le cas inverse où le bruit de modèle est très élevé, le filtre aura tendance à trop faire confiance aux mesures, et à ne pas filtrer le bruit de mesure, ce qui conduit également à une dégradation de l'estimation de tous les paramètres de l'état. Il est donc nécessaire de trouver un juste milieu, qui peut dépendre aussi de l'application envisagée : est-il nécessaire d'avoir une estimation très précise pour la position, ou au contraire pour la vitesse, le cap ? Une méthode assez couramment utilisée, appelée adaptation de Castella, consiste à adapter le bruit en fonction des performances du filtre, et donc à augmenter le bruit de modèle pendant les écarts entre le modèle et la trajectoire, et à le diminuer dans les phases où le modèle correspond à la réalité.

Une comparaison entre le filtre proposé dans le chapitre précédent, appliqué au modèle de Frenet-Serret en 3D et d'autres modèles et filtres souvent utilisés en pratique est présentée. Le modèle en 3D exprimé dans le repère de Frenet-Serret assorti du filtre IEKF développé dans le troisième chapitre est comparé avec trois autres associations de modèles et filtres :

- un modèle vecteur vitesse constant, avec un filtre EKF,
- un modèle à sauts markoviens (modèle multiple) à trois modèles : un modèle vitesse constante, un modèle virage coordonné dans le plan horizontal, un modèle virage coordonné dans le plan vertical, avec un filtre multi-modèles IMM (Interacting Multiple Model).
- Le modèle de Frenet-Serret en 3D, mais exprimé avec des quaternions plutôt qu'avec une matrice de rotation, ce qui permet d'utiliser un EKF comme filtre.

La comparaison est menée sur un ensemble de trajectoires provenant de trois simulateurs différents. Le premier simulateur permet de générer des trajectoires déterministes par morceaux, avec entre les sauts, un modèle avec accélérations normales et tangentielles constantes. Le deuxième simulateur fournit des trajectoires rectilignes ou circulaires par morceaux, avec possibilité de rajouter de l'accélération tangentielle. Enfin, le dernier simulateur génère des trajectoires de cibles balistiques, comprenant une phase de *boost*, pendant laquelle la cible a une accélération tangentielle forte, puis une phase purement balistique. Le modèle de Frenet-Serret couplé à l'IEKF est meilleur, notamment en cap, sur les trajectoires des deux premiers simulateurs. Ces résultats sur la précision en cap ont été observés sur la plupart des expériences menées, et en particulier dans le cas du contrôle de trafic aérien civil, sur des trajectoires réelles. En revanche, pour les cibles balistiques, l'IEKF est moins bon que les autres algorithmes. Un autre point à noter est le fait que l'EKF avec le modèle à vitesse constante est assez facile à régler, et relativement robuste (même pour les cibles balistiques), au prix de résultats parfois un peu moins précis, et de bruit de modèle extrêmement élevé. L'IMM en revanche est le filtre le plus difficile à régler, notamment à cause de la présence de bruits de modèle pour chacun des trois modèles, mais aussi à cause du réglage de la matrice de transition entre les modèles. Enfin, la comparaison entre le modèle de Frenet-Serret avec l'IEKF et avec l'EKF montre que l'IEKF est un filtre plus adapté à ce type de modèle que l'EKF.

4.2 Introduction

In chapters 2 and 3, new target models and filtering algorithms have been presented. Some first results have been presented at the end of chapter 3. However, we need to lead more tests, to really compare the performances of the algorithms. To achieve this, we have selected two target models and filtering algorithms that are widely used in industry, and we have slightly modified our 3D Frenet-Serret target model to apply the EKF algorithm with it.

We have also selected different trajectories, coming from three different trajectory simulators, and which have different trajectory models. We are not able to present real trajectories, however, the simulators selected are used to test the algorithms in industry, so the trajectories produced correspond to real issues radar engineers are confronted to. We generated trajectories with different kinematic characteristics, to be as comprehensive as possible.

Tuning the filters is one of the major issues for all filtering algorithms. Indeed, a bad tuning can lead to the divergence of the algorithm, or to very poor precision. Moreover, the tuning has to be valid for a set of different trajectories. A discussion on tuning is thus provided in section 4.3. There are mainly two important issues that we have been confronted to during this work:

1. How to tune a filter, so that it is precise enough on all trajectories ?
2. How to make the tuning fair to compare different algorithms ?

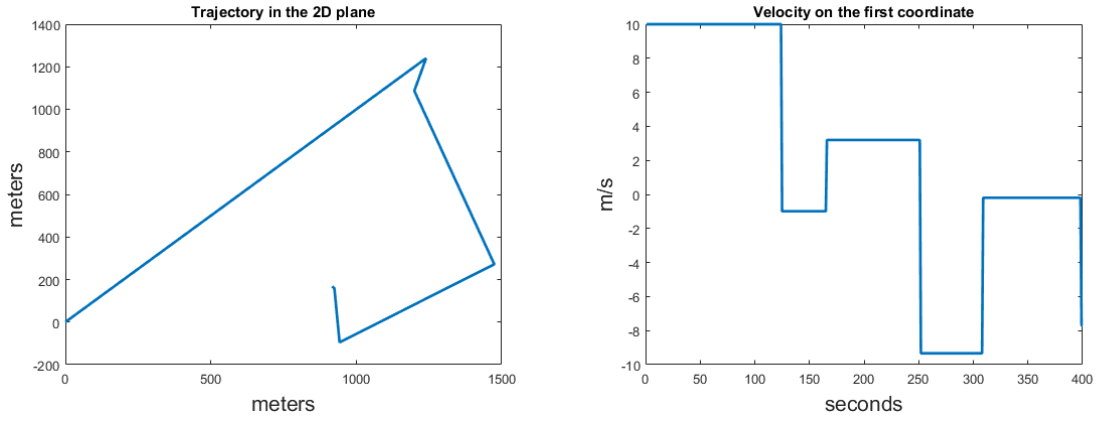
In particular in this chapter, we show that the tuning may depend on the target considered, and we tune our algorithms on a specific set of representative trajectories, different from the set used to compare the performances.

The chapter is organised as follows: first, the tuning issue is discussed in section 4.3. In section 4.4, some preliminary results obtained on a real air traffic control scenario are presented. Then, we present the models and algorithms selected to perform an extensive comparison in section 4.5. The set of trajectories used is also briefly described in section 4.6. Finally, the results are given in section 4.7 for spherical coordinates observations with realistic observation noise, on a training set (used to tune the algorithms) and a testing set to compare the performances of all algorithms.

4.3 Process noise tuning

4.3.1 Issues of noise tuning

To see the implications of noise tuning on the performances of the estimation, let us use a very simple linear model. We generate a trajectory with piecewise constant velocity, in two dimensions. The trajectory and the velocity for the first coordinate are represented on figure 4.1.



(a) (b)

Figure 4.1 – Reference trajectory 4.1a and velocity on the first coordinate 4.1b

Let us use a linear Kalman filter to perform state estimation on this 2D trajectory. The target model used is a constant velocity model, with white Gaussian noise on the velocity only, and Cartesian measurements:

$$X_n = \begin{pmatrix} x_n^1 \\ \dot{x}_n^1 \\ x_n^2 \\ \dot{x}_n^2 \end{pmatrix}$$

$$X_{n+1} = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} X_n + w_n$$

$$y_n = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} X_n + v_n$$

w_n and v_n are white Gaussian noises with covariances Q_n and N_n respectively. In the absence of jumps, the Kalman filter is optimal. We perform three experiments with Gaussian noise on the velocity only. The results are presented on figure 4.2. The standard deviation of the process noises are 0 for 4.2a, 0.01 for 4.2b, and 1 for 4.2c.

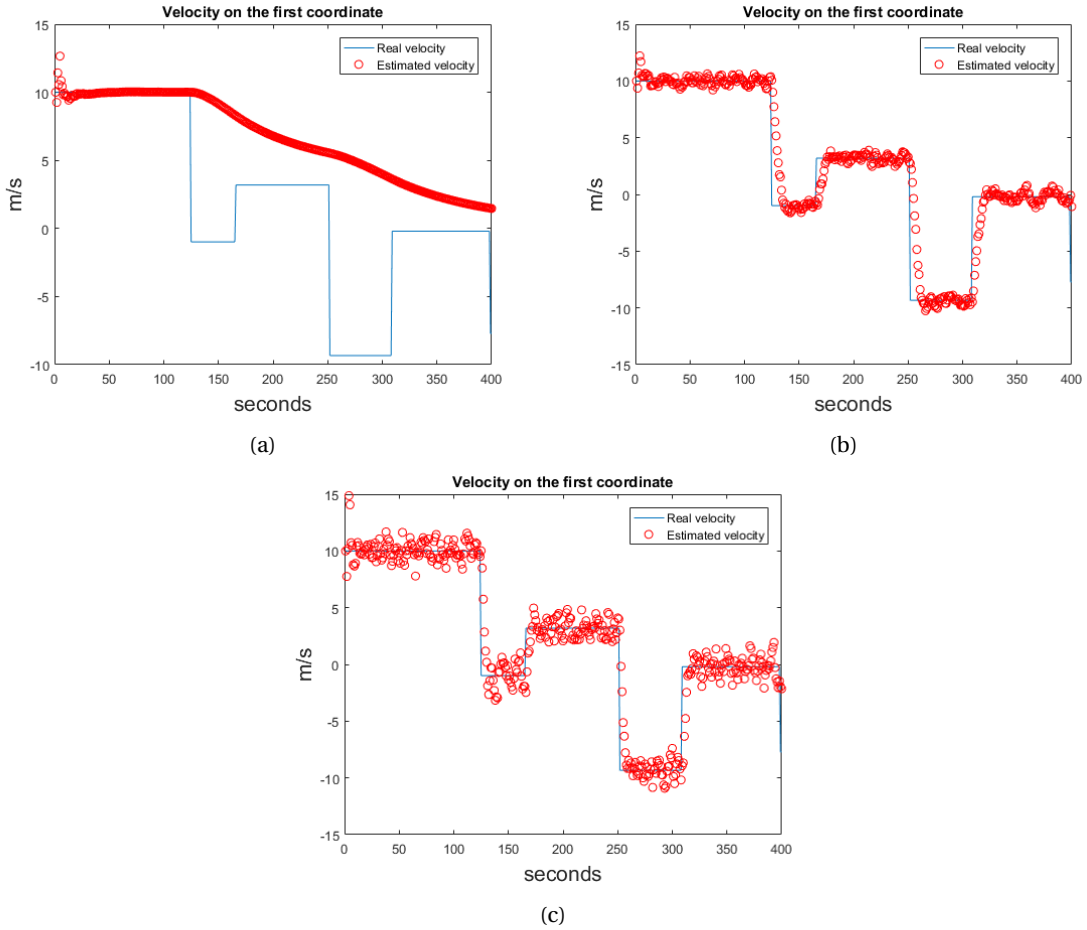


Figure 4.2 – Velcotiy estimation with a Kalman filter with no process noise fig. 4.2a, intermediate process noise fig. 4.2b, and high process noise fig. 4.2c.

The Kalman filter with no process noise does not adapt to the jumps. Indeed, the Kalman filter being optimal, when there is no process noise, its covariance matrix, and its gain go to zero, when the time n goes to infinity. This a problem here, as the trajectory has some jumps, that are not contained in the model. However, we see that after an initialisation phase the estimation becomes very precise on the first constant part.

We see that the jumps can be taken into account, by increasing the process noise. This is known as *robust tuning*, see for instance [97]. When the process noise is increased, the filter can adapt to the jumps, but the estimation for the linear motions becomes less precise. There is thus a compromise to find between the speed of convergence of the filter after a jump, and the required precision on the rest of the motion.

In fact, when the process noise is high, the precision of the position estimation on the constant parts cannot be greater than the one of the measurement noise. Indeed, a high model noise

implies that the filter has more confidence in the observations that in the target model provided, and the estimation tend to follow the measurements. This also leads in a reduced accuracy of the other state parameters (notably the derivatives of the position).

4.3.2 Castella noise tuning

The Castella noise tuning is used to adapt the process noise to the manoeuvres of the target. Indeed, as we have seen in the previous section, we seek to have good precision during non-manoeuvre phases, but at the same time, we want the model to be able to adapt when a manoeuvre occurs. This is not possible with one single tuning for the whole trajectory. In [35], the author proposes a way to adapt the process noise during the estimation.

The principle of the Castella noise tuning is presented on figure 4.3. The idea is to compute the normalised innovation NIS. When it is below a given threshold NIS_{low}, the process noise covariance σ_γ is tuned low $\sigma_\gamma = \sigma_{\gamma_{min}}$, when it exceeds another threshold NIS_{high}, the process noise is tuned high $\sigma_\gamma = \sigma_{\gamma_{max}}$. Between the two thresholds, the process noise increases linearly.

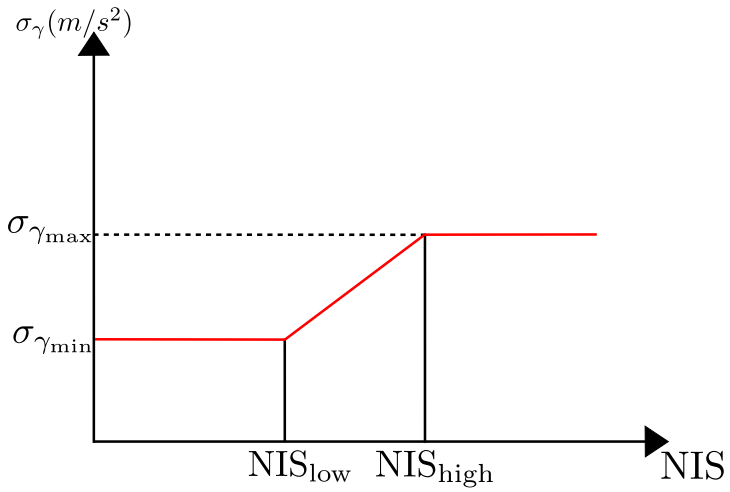


Figure 4.3 – Castella noise tuning

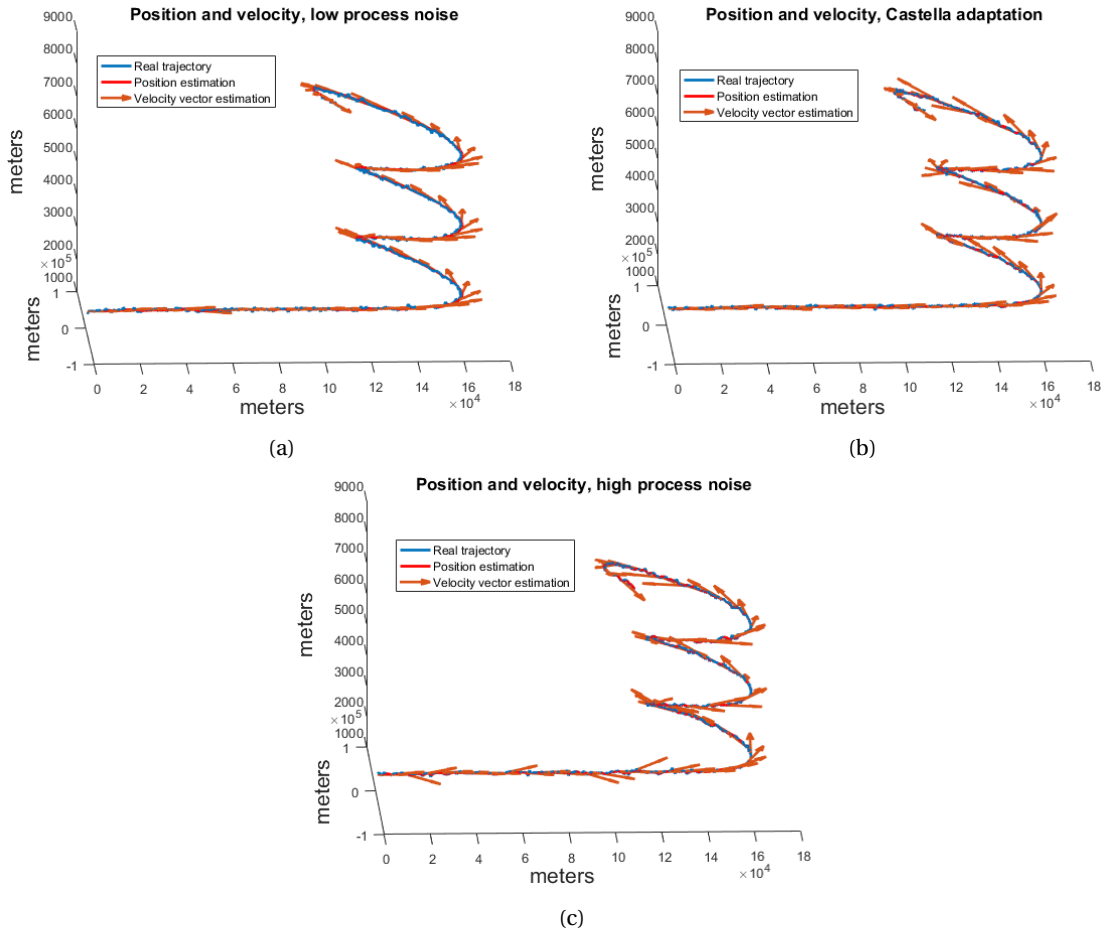
This method seems reasonable, as it gives more flexibility to the estimation filter in the case when the model does not correspond to the real trajectory. For example, when using a constant velocity model, this increases the process noise during turns. This allows the filter to adapt to the real trajectories, and deviations to the target model. However, the accuracy during manoeuvres is not optimal, since the estimations will tend to follow the measurements, and the measurement noise will not be smoothed, so the accuracy is degraded for the velocity vector, compared to a model adapted to the manoeuvre.

To understand the impact of the Castella noise adaptation, let us take an example. We use a linear Kalman filter, with Cartesian position measurements, and a constant velocity model in 3D. The trajectory used is the same as in section 3.7.2. We plot the RMSE for three different cases: the first one is a low process noise tuning during all the trajectory, the second one is the Castella process noise tuning, and the third one is a high process noise tuning. The process noise is added only on the velocity evolution, and is kept null for the position evolution. The RMSE are collected in table 4.1. We see that the Castella tuning does not give the best RMSE values. For the position, this is natural, since the high process noise filter will tend to be closer to the measurements, and far from the model. For the velocity vector, we can see on figure 4.4, that the Castella filter is more accurate than the high process noise on the straight line, and then the process noise adapts to the helix, contrary to the low process noise filter. The velocity vector and the position for the low process noise filter are late during the helix motion (this can be seen with the RMSE values). For the high process noise tuning, the estimations are less smoothed on the first motion, but they accommodate well during the manoeuvre.

Parameter	Low process noise	Castella process noise	High process noise
$x_1(m)$	262	32	26
$x_2(m)$	251	29	25
$x_3(m)$	19	22	26
$v_1(m/s)$	83	23	21
$v_2(m/s)$	80	21	20
$v_3(m/s)$	3	6	8

Table 4.1 – RMSE for the position and velocity for 100 Monte-Carlo simulations

The Castella noise tuning permits to adapt to the manoeuvres and takes the best of low and high process noise, however, during the turns, a turn model is still better suited to perform accurate estimations, than simply adding more process noise.



(a) (b) (c)

Figure 4.4 – Velocity vector (and position) estimation for low process noise tuning fig. 4.4a, Castella process noise tuning fig. 4.4b, and high process noise tuning fig. 4.4c.

In the sequel, we do not use the Castella noise adaptation, even though it is commonly used in radars. Indeed, this implies even more tuning issues (to tune the thresholds and the different values of the process noise), and it can be applied to any of the filtering algorithms used in this chapter.

4.4 Test on a real scenario

The IEKF has been inserted in a simulator in the Air Traffic Management Unit of Thales. The simulator implements all the tracking chain of the radar, including the initialisation, the association and the end of the tracks. The code is in C. Some preliminary results have been obtained, and show improvements compared to an EKF, as shown on figure 4.5. This trajectory is taken from a real flight scenario in the neighbourhood of an airport, and it corresponds to the trajectory of a small private jet. The positions of the aircraft given to the tracker come from the ADS-B, they are thus in Cartesian coordinates.

The 3D Frenet-Serret model with the IEKF algorithm is compared to the constant velocity with the EKF algorithm, which will be also used for comparisons on a wider set of trajectories in section 4.7. These two filters are tuned with the Castella noise adaptation algorithm described in section 4.3.2. This permits to tune the process noise accordingly to the performances of the filter. These preliminary results are encouraging, and further tests are in progress. The IEKF is more accurate during the turns than the EKF, even when considering all the tracking chain. The accuracy of the IEKF during manoeuvres will be confirmed in section 4.7. A closer look on the last turn is given on figure 4.6.

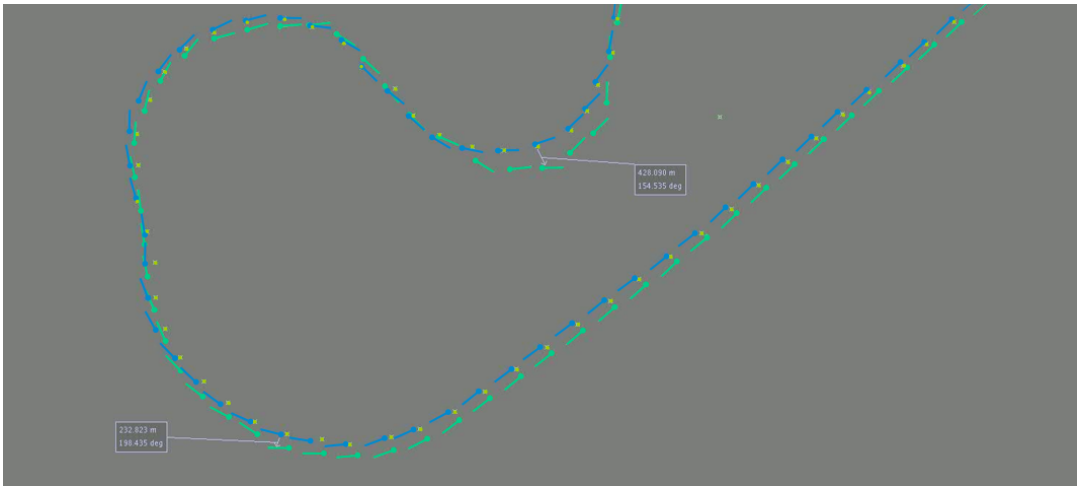


Figure 4.5 – Results of the IEKF with a Frenet-Serret model, and the EKF with a constant velocity model. The IEKF results are in blue, the EKF results are in green. We see that the IEKF tends to be more accurate during the turns.

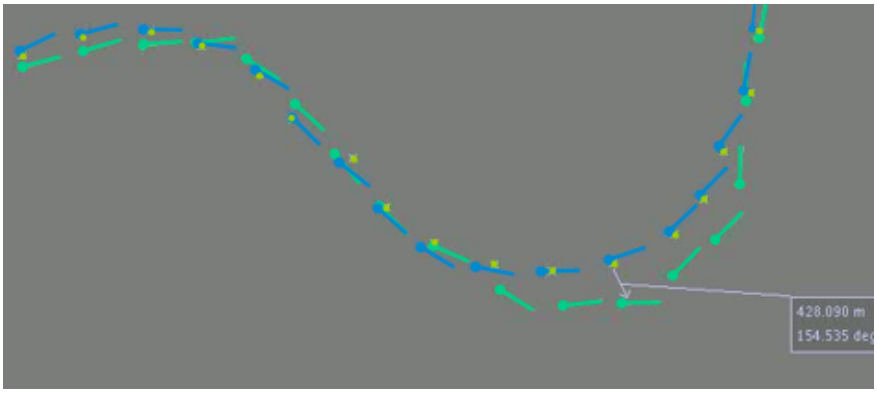


Figure 4.6 – Results of the IEKF with a Frenet-Serret model, and the EKF with a constant velocity model. The IEKF results are in blue, the EKF results are in green. This is a closer look of the last turn in figure 4.5. The IEKF in this case provides much better estimations than the EKF, even with a Castella adaptation noise tuning.

4.5 The models and algorithms used for comparison

In this section, we present the four models and algorithms that we will use to perform a comparison of the filters. All models used are expressed in 3D. The first model is the constant velocity model, and the associated filter is the EKF. The second model is a multiple model, with three models, along with an IMM. The third one is our 3D Frenet-Serret model, with the IEKF. And the fourth one is a slightly modified 3D Frenet-Serret model, with an EKF.

The tests will be performed with noisy range, azimuth and elevation observations. The description of the observation equations and the associated observation noises are provided in section 2.5.

4.5.1 Constant velocity and the EKF

This first model is still today one of the most used in industry. Its simplicity is appealing and makes it quite easy to tune. Moreover, it performs fairly well on non-challenging scenarios. We use a 3D constant velocity model, see section 2.3. The state is $\mathrm{X}_t = \begin{pmatrix} x_t^1 & \dot{x}_t^1 & x_t^2 & \dot{x}_t^2 & x_t^3 & \dot{x}_t^3 \end{pmatrix}^{\mathrm{T}}$. The evolution equation is:

$$\dot{\mathrm{X}}_t = \mathrm{A}\mathrm{X}_t + w_t \tag{4.1}$$

with

$$\mathrm{A} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

When considering spherical observations, the measurement equation becomes non-linear, so an EKF has to be used, even though the model is linear. This will be referred to as the constant velocity (CV) EKF model.

4.5.2 Multiple model and the IMM

We use a multiple model, with three models:

- a 3D constant velocity model,
- a coordinated turn in the horizontal plane,
- a coordinated turn in the vertical plane.

This second type of model we consider is also widely used. Usually, it is used with four models, these three and a constant acceleration model with a high process noise, which is used when none of the other models corresponds to the real trajectory. However, here we seek to compare the IEKF with an equivalent filter (with the same kinematic assumptions), so we do not use the constant acceleration model.

The natural algorithm to perform the estimation is the IMM. We will have to tune the process noises differently for each model. Indeed, one of the models will have to take into account the motions that do not correspond to any of the models, and thus have more process noise than the other two. The IMM has three models, so there are three process noise covariance matrices that have to be tuned. Moreover, the transition matrix between the models has also to be tuned for the IMM.

4.5.3 The Frenet-Serret target model with an IEKF

The new target model expressed in the 3D Frenet-Serret frame of section 2.6.2 is used. The estimation is performed by the IEKF of section 3.6. Indeed, we have seen in chapter 3 that it is appropriate to track maneuvering targets.

4.5.4 The Frenet-Serret target model with an EKF

Quaternions are often used to represent rotations. We take advantage of them here to derive an EKF algorithm from our 3D Frenet-Serret target model, this is also a contribution of this thesis. Quaternions as random variables have been studied, for example in [78].

It is possible to express the 3D Frenet-Serret target model using quaternions. This work has been done during an internship within the team ADW of Thales. This formulation is useful to compare the same target model with two algorithms, the IEKF on the one hand, and the EKF on the other hand, that is applied to the quaternion formulation. We explain how to have a vectorial formulation here:

Let us start from equations (2.33) reminded here:

$$\begin{cases} \frac{d}{dt}x_t = \mathrm{R}_t(v_t + w_t^x) \\ \frac{d}{dt}\mathrm{R}_t = \mathrm{R}_t(\omega_t + w_t^\omega)_\times \\ \frac{d}{dt}\gamma_t = 0 + w_t^\gamma \\ \frac{d}{dt}\tau_t = 0 + w_t^\tau \\ \frac{d}{dt}u_t = 0 + w_t^u \end{cases}$$

The quaternions are used to replace the rotation matrix, to avoid having to work in the Lie group and Lie algebra spaces. The state is $\mathrm{X}_t = \begin{pmatrix} x_t & \dot{x}_t & \mathrm{Q}_t & p_t & r_t & u_t \end{pmatrix}^\mathrm{T}$, with x_t the Cartesian position, $\dot{x}_t$ the Cartesian velocity, Q_t the quaternion, p_t the first component of the angular velocity vector, r_t its second component, and u_t the norm of the velocity. The evolution of the position is thus modified into:

$$\frac{d}{dt}x_t = \Theta_t \begin{pmatrix} u \\ 0 \\ 0 \end{pmatrix}$$

The matrix Θ_t is expressed as:

$$\Theta = \begin{pmatrix} q_0^2+q_1^2-q_2^2-q_3^2 & 2(q_2q_3+q_0q_3) & 2(q_1q_3-q_0q_2) \\ 2(q_1q_2-q_0q_3) & q_0^2-q_1^2+q_2^2-q_3^2 & 2(q_2q_3+q_0q_1) \\ 2(q_1q_3+q_0q_2) & 2(q_2q_3-q_0q_1) & q_0^2-q_1^2-q_2^2+q_3^2 \end{pmatrix}$$

The evolution of the quaternions is given by

$$\frac{d}{dt}\mathrm{Q}_t = \frac{1}{2}\Omega_t\mathrm{Q}_t$$

with $q = \begin{pmatrix} q_0 & q_1 & q_2 & q_3 \end{pmatrix}^\mathrm{T}$ and

$$\Omega = \begin{pmatrix} 0 & -p & -q & -r \\ p & 0 & r & -q \\ q & -r & 0 & p \\ r & q & -p & 0 \end{pmatrix}$$

For the Frenet-Serret model, the rotation around the roll is not considered, so here $q = 0$. The evolution of p_t and r_t are considered (almost) constant, as well as the one of u_t, to mimic equations (2.33). This model will be referred to as the Frenet-EKF model.

4.6 Set of trajectories

The trajectories used in this chapter to compare the algorithms described previously have been simulated with three different simulators. The models used in the simulators, and the kinematic characteristics of the trajectories are detailed in this section.

4.6.1 Simulators

The first simulator uses the variable rate model of [59], also presented in section 2.3.4. This model represents piecewise deterministic trajectories with piecewise constant normal and tangential accelerations and orientation angle, with occasional jumps. To generate a trajectory with this simulator, there are several degrees of freedom: the initialisation of the kinematic parameters, the jump times, and the values of the jumping parameters. One example of trajectory simulated with this generator is provided on figure 4.7. This simulator will be called simulator 1.

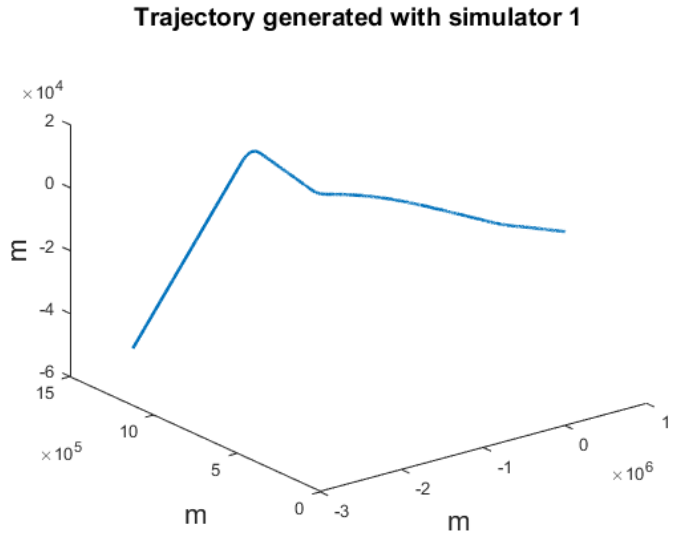


Figure 4.7 – Trajectory generated with the first simulator

The second simulator is also based on a jumping model. The portions of trajectories between the jumps are either straight line motions with constant velocities or accelerations, or portions of circles with constant radius. An example of trajectory is given on figure 4.8. It will be referred to as simulator 2.

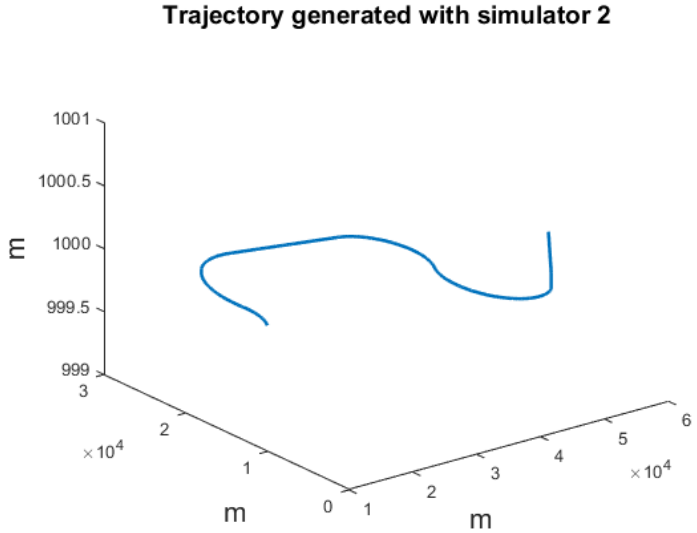


Figure 4.8 – Trajectory generated with the second simulator

Finally, we use a generator of ballistic trajectories. This is simulator 3. The ballistic trajectories are characterised by a boost phase very far from the radar with a high tangential acceleration, and then a ballistic motion, when they obey the equations given by Newton's laws. An example is given on figure 4.9.

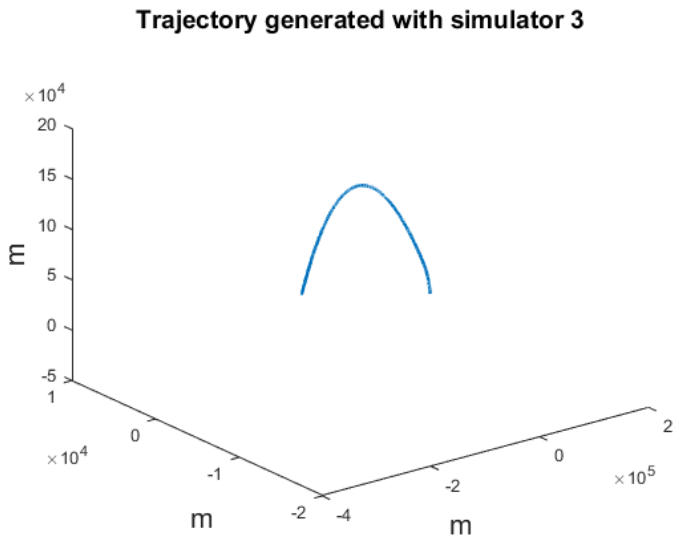


Figure 4.9 – Trajectory generated with the third simulator

4.6.2 Kinematic characteristics

The trajectories have different kinematic characteristics, and mainly different accelerations, which characterize the most manoeuvring targets. As we have briefly mentioned in the introduction, the highly-manoeuvring targets are the ones which have normal accelerations beyond $15g$. We have thus collected all the accelerations of the different trajectories simulated, to be sure there are both highly and slowly manoeuvring targets (even though our solution is designed for highly-manoeuvring targets, we do not want the tracking to be degraded for targets that do not manoeuvre too much). We have selected approximately 30 different trajectories to perform the tests. The maximal accelerations for each trajectory are given in table 4.2. We have also played on the duration of each portion of the trajectory, meaning that the jumps are more or less close to one another, depending on the trajectory considered. Finally, the acceleration of ballistic trajectories are mainly tangential.

The sampling time is constant for each trajectory. We thus perform simulations with two different observation update rates: $1s$ and $0.5s$. This is to represent the different possibilities of radars. Indeed, the radars with rotating antennas cannot have observation periods below $1s$ for instance.

Trajectory	Acceleration	Trajectory	Acceleration
Trajectory 1 from simulator 1	6.8g	Trajectory 1 from simulator 2	10.4g
Trajectory 2 from simulator 1	9.7g	Trajectory 2 from simulator 2	0.39g
Trajectory 3 from simulator 1	18.2g	Trajectory 3 from simulator 2	0.23g
Trajectory 4 from simulator 1	9.7g	Trajectory 4 from simulator 2	0.19g
Trajectory 5 from simulator 1	9.8g	Trajectory 5 from simulator 2	1.3g
Trajectory 6 from simulator 1	20.0g	Trajectory 6 from simulator 2	9.4g
Trajectory 7 from simulator 1	20.0g	Trajectory 7 from simulator 2	21.0g
Trajectory 8 from simulator 1	26.7g	Trajectory 1 from simulator 3	18.6g
Trajectory 9 from simulator 1	17.1g	Trajectory 2 from simulator 3	22.4g
Trajectory 10 from simulator 1	16.1g	Trajectory 3 from simulator 3	13.3g
Trajectory 11 from simulator 1	17.1g	Trajectory 4 from simulator 3	19.3g
Trajectory 12 from simulator 1	10.3g	Trajectory 5 from simulator 3	14.0g
Trajectory 13 from simulator 1	14.8g	Trajectory 6 from simulator 3	19.0g
Trajectory 14 from simulator 1	5.2	Trajectory 7 from simulator 3	6.3g
Trajectory 15 from simulator 1	7.3g	Trajectory 8 from simulator 3	1.1g
Trajectory 16 from simulator 1	10.4g		

Table 4.2 – Accelerations for the trajectories used in this chapter

4.7 Results

To tune the process noises for the different algorithms used, we have made two distinct groups of trajectories. The first group is used for tuning (we call it the "train set'), and the other group is used to test the algorithms with the tuning obtained with the train set. This second set of trajectories is thus called the test set.

We compute the Root Mean Square Errors (RMSE) of three different parameters: the position estimation, the norm of the velocity vector estimation, and the heading estimation.

4.7.1 Set of trajectories to tune the process noise

The trajectories used to tune the process noise are the following:

- Trajectories from simulator 1: trajectories 1, 2, 5, 7, 10, 12, 13.
- Trajectories from simulator 2: trajectories 1, 2, 6.
- Trajectories from simulator 3: trajectories 1, 2, 3, 5.

For the train set, we selected the process noise tuning for each trajectory which minimises the overall RMSE on all the trajectories.

First, the tests performed showed that we have to separate the trajectories from simulators 1 and 2 on the one hand, and trajectories from simulator 3 on the other hand. Indeed, simulator 3 generates ballistic trajectories with a boost phase at the beginning, which results in a high longitudinal acceleration. The filters thus need to be tuned differently for this phase, and for the ballistic trajectories.

The results for the best tuning for each algorithm are collected in table 4.3 for trajectories from simulators 1 and 2. The train tests were made for 10 Monte-Carlo on each trajectory, with an update rate of 1s. The first important observations is that the IEKF has the lowest RMSE for the orientation. As we will see in the next section, this is one of the major strengths of the IEKF. Indeed, during turns, the orientation is kept very accurate thanks to the Frenet-Serret frame being included in the target model. Finally, we notice that the quarternion model, close to the Frenet-Serret one is a little less precise than the IEKF, which confirms the fact that the IEKF is the natural algorithm to use with the Frenet-Serret model.

Parameter	CV + EKF	IMM	IEKF	Frenet + EKF
Position (m)	324	55	57	53
Norm of velocity (m/s)	63	106	80	203
Orientation (rad)	0.54	0.49	0.46	0.51

Table 4.3 – RMSE for the training set for each algorithm, on trajectories from simulators 1 and 2. The algorithm CV + EKF corresponds to the constant velocity model with an EKF, the IMM is composed of three models (constant velocity, horizontal turn, vertical turn), the IEKF is the 3D Frenet-Serret model with an IEKF, and the term Frenet + EKF refers to the model with quaternions, tracked with an EKF, and explained in section 4.5.4.

Parameter	CV + EKF	IMM	IEKF	Frenet + EKF
Position (m)	421	382	586	900
Norm of velocity (m/s)	56	62	614	866
Orientation (rad)	0.13	0.17	0.48	0.38

Table 4.4 – RMSE for the training set for each algorithm, on trajectories from simulator 3.

The particular case of ballistic targets

Ballistic targets have a boost phase, very far from the radar (around $100km$ for the closest ones), then they follow a purely ballistic trajectory. This impacts the tuning of the algorithms, and it appeared very awkward to use the same tunings for these trajectories and the ones from the other simulators for all the algorithms tested. In this case, we have to say that the IEKF is very sensitive to the noise tuning, and is more difficult to tune than the EKF with a constant velocity model. Indeed, it is not sufficient to add a great noise on one variable to have an appropriate tuning. The RMSE computed for each algorithm are given in table 4.4.

The results for the IEKF are not as good as the ones for the other algorithms. Indeed, the underlying target model was not designed to track ballistic missiles. The boost phase, with high tangential acceleration, at a great distance is not really compatible with the target model formulation. This is confirmed by the fact that the same model with an EKF does not show very good performances too. Indeed, the Frenet-Serret model is designed to be accurate for turn motions. The "turn phase" of ballistic trajectories is not very challenging for a constant velocity model such as the ones used in the EKF of the IMM, so these filters perform well on this type of trajectory.

For simulators 1 and 2, some tangential acceleration is also present, but at a lower level. Despite this tangential acceleration, the Frenet-Serret model-based algorithms preformed well. So up to a certain point, the model is still valid for low tangential accelerations.

4.7.2 Set of trajectories to test the tunings

The trajectories used to test the performances of the algorithms with the process noise obtained on the training set are:

- Trajectories from simulator 1: trajectories 3, 4, 6, 8, 9, 11, 14, 15, 16.
- Trajectories from simulator 2: trajectories 3, 4, 5, 7.
- Trajectories from simulator 3: trajectories 4, 6, 7, 8.

The results for the best tuning for each algorithm are collected in table 4.5 for trajectories from simulators 1 and 2, and an update rate of $1s$.

One first observation concerns the RMSE for the norm of the velocity of the IMM. Since there is no acceleration model in the IMM, we had to tune one of the models with high process noise. This explains this poor result, but it can be avoided by adding a constant acceleration model that

Parameter	CV + EKF	IMM	IEKF	Frenet + EKF
Position (m)	218	56	158	63
Norm of velocity (m/s)	41	902	51	210
Orientation (rad)	0.38	0.36	0.32	0.37

Table 4.5 – RMSE for the testing set for each algorithm, on trajectories from simulators 1 and 2, with an update rate of $1s$.

manages the change of models in the trajectory. The second observation is that once again, the orientation accuracy is best for the IEKF. The norm of the velocity precision is a little less precise for the IEKF than for the EKF, but the lack of precision is minor compared to the gain obtained for the heading.

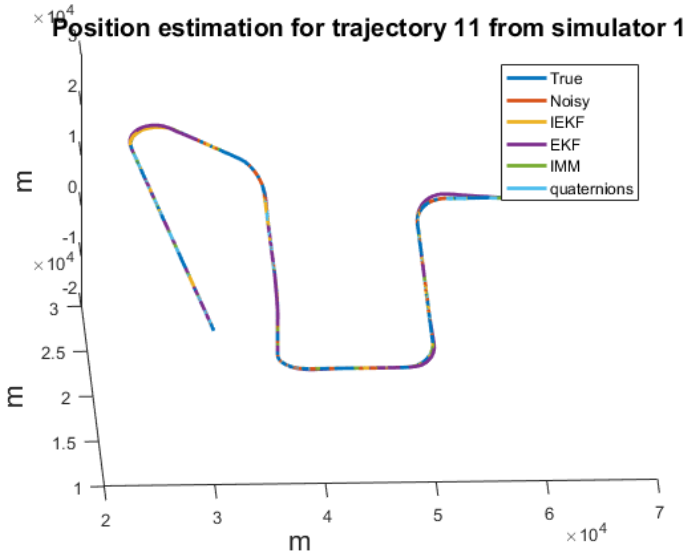


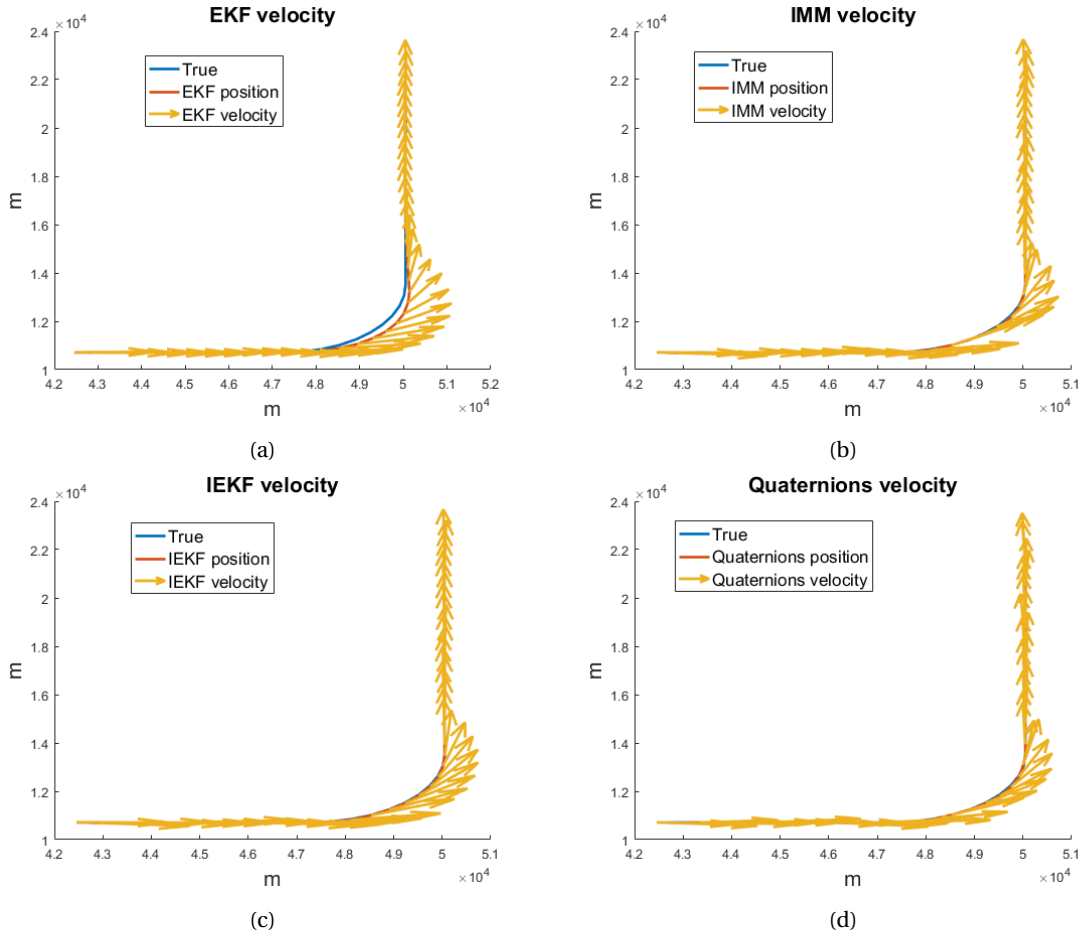
Figure 4.10 – Trajectory 11 from simulator 1, with the position estimations of all four algorithms.

Let us look closer to what happens during a manoeuvre. We use trajectory 11 from simulator 1, and have a closer look at the velocity vectors of each algorithm during a turn. The trajectory and position estimations for the four algorithms are presented on figure 4.10. The velocity vectors for all algorithms during one particular turn of the trajectory is provided on figure 4.11. We have focused on one turn, in the 2D plane. This figure shows that the EKF algorithm with the constant velocity model has some delay during the turn motion. Indeed, its target model is not at all adapted to turns, and the algorithm adjusts its estimation thanks to the observations and the process noise. The estimations for the IMM and the quaternions model with an EKF seem more accurate, namely concerning the position estimation, however, their velocity vector are less precise during the turn. This is corroborated by the errors in heading and norm of velocity, shown on figure 4.12.

We have also tested the algorithms with a different update rate $t = 0.5s$. We use the same tunings obtained with the training set for $1s$. The results are provided in table 4.6. These tests confirm the fact the the IMM can provide reliable velocity estimations, in particular for the norm, when its tuning correspond more to the manoeuvres of the trajectory (less visible for a reduced update rate). The IEKF is still the best algorithm for the orientation precision.

Ballistic trajectories

Once again the results for the ballistic trajectories of simulator 3 are reported in table 4.7. As for the training set, the results for the IEKF are not as good as for the EKF or the IMM. One solution

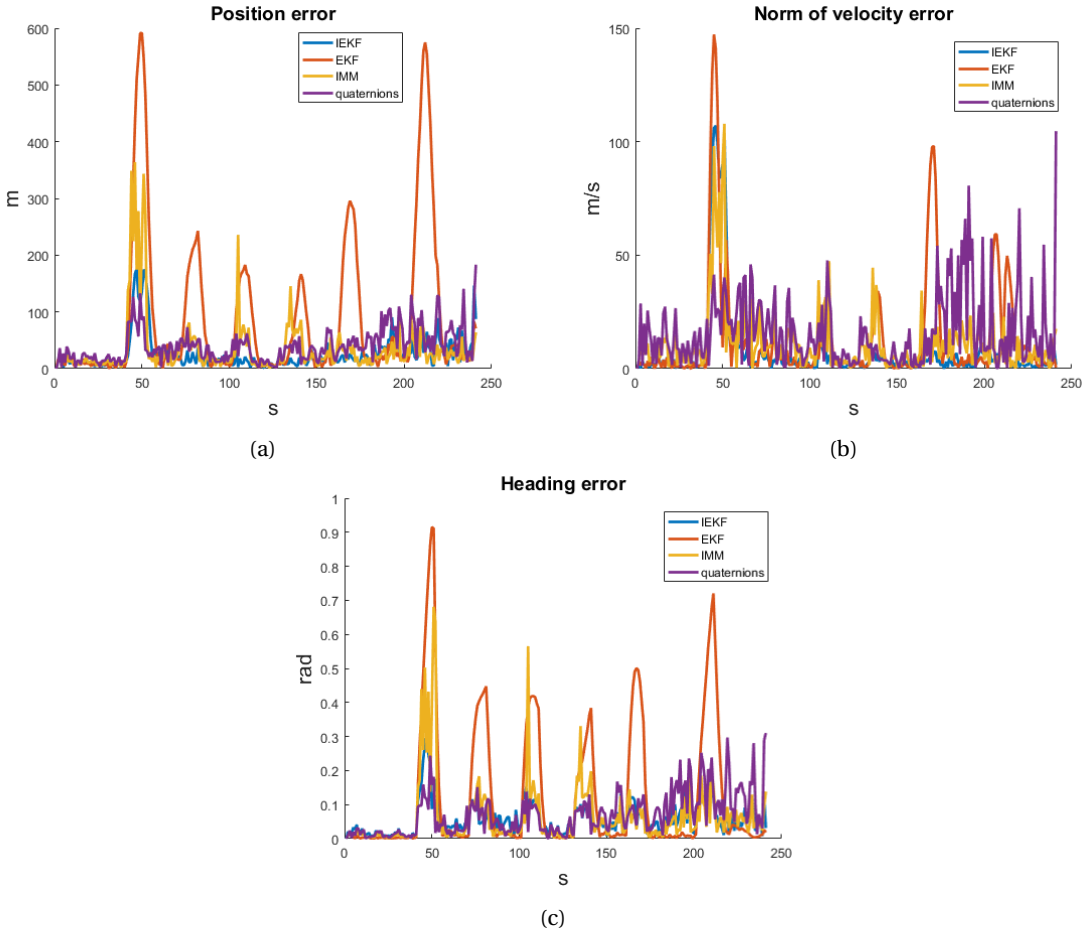


(a) (b) (c) (d)

Figure 4.11 – Velocity vector represented for the four algorithms, for one of the turns of trajectory 11 from simulator 1. Fig. 4.11a is the EKF with the constant velocity model, fig. 4.11b is the IMM, fig. 4.11c is the IEKF and fig. 4.11d is the EKF with the quaternions model. The IEKF is the filter whose velocity vector is the most accurate during the turn.

Parameter	CV + EKF	IMM	IEKF	Frenet + EKF
Position (m)	134	45	102	56
Norm of velocity (m/s)	33	31	43	206
Orientation (rad)	0.36	0.35	0.31	0.39

Table 4.6 – RMSE for the testing set for each algorithm, on trajectories from simulators 1 and 2, with an update rate of 0.5s. Once again, we notice the IEKF is appropriate to estimate the heading of the velocity vector.



(a) (b)

(c)

Figure 4.12 – Errors in position fig. 4.12a, in norm of velocity fig. 4.12b and in heading fig. 4.12c. This corroborates the IEKF with the Frenet-Serret model perform better than the other filters.

Parameter	CV + EKF	IMM	IEKF	Frenet + EKF
Position (m)	589	463	696	424
Norm of velocity (m/s)	105	149	820	654
Orientation (rad)	0.21	0.24	0.50	0.30

Table 4.7 – RMSE for the testing set for each algorithm, on trajectories from simulator 3.

would be to couple the IEKF with a constant acceleration model in an IMM, to ensure the boost phase is compatible with the models proposed. The fact that the results are less accurate for the testing set confirm that the IEKF is hard to tune on this kind of trajectories. In industry however, a specific treatment is applied to ballistic trajectories.

4.8 Conclusion

In this chapter, we have discussed the tuning issues for the filtering algorithms. Target models are usually designed with assumptions that some kinematic parameters are constant. However, the real trajectories have non-constant motions. Some process noise thus has to be added to the model to take into account the variations of the true trajectories.

The Frenet-Serret target model and the IEKF have been conceived for manoeuvring targets, but we have also seen that they are well suited to a civilian application as well. This field of civilian air traffic management addresses also other problems than the military application, including trajectory prediction, which might seem a similar problem, but that must be achieved for a longer time scale. Solutions to the trajectory prediction problem differ from the ones of target tracking, and can be based on regression, as in [45], and more generally to air traffic optimisation in [52].

We have simulated trajectories with different simulators to perform tests on four different algorithms to compare them. The first result is that tunings can differ from one simulator to the other. Indeed, when targets have very different behaviours, a single tuning is generally not satisfying. For this reason, tunings for ballistic missiles and other trajectories are different, since the acceleration is in one case tangential, and in the other cases lateral. For this same reason, the results of the IEKF on ballistic targets are not entirely satisfying.

When tested on trajectories with high manoeuvres (turns), the IEKF proved to be more accurate than the other algorithms. In particular, the heading precision is very high. This is of crucial importance, since it is required for several needs. The first need is to anticipate the location of the next observation, *i.e.* to direct the radar beam to be sure to find the target again during active tracking. This is decided by the heading of the target. Another important aspect is the display for an operator. An operator always prefer an accurate velocity vector that does not change directions from one moment to the other. Finally, heading is one crucial factor for fire-control radars, so that the target can be intercepted.

Part II

Alternative state estimation: Smoothing

Chapter 5

Smoothing applied to target state estimation

Sommaire

5.1 Résumé en français : Lissage appliqué à l'estimation d'état

Le filtre présenté dans le troisième chapitre permet de réaliser l'estimation de trajectoires où la norme de la vitesse, la courbure et la torsion sont presque constantes. On a vu que le filtre pouvait gérer les sauts ou écarts entre modèle et réalité grâce au bruit de modèle. Cependant, dans le cas de sauts très importants, on peut envisager une autre méthode pour permettre une convergence plus rapide de l'estimation après un saut, en détectant directement la présence du saut. Pour cela, on reprend le modèle (en 2D) du deuxième chapitre, et on adapte la formulation de façon à inclure directement la possibilité de sauts dans les équations du modèle, à l'aide de modèles stochastiques par processus ponctuels. Ce type de modèle est appelé "modèle à taux variable", et le modèle est déterministe par morceaux, c'est-à-dire que le bruit de modèle est supprimé dans les équations. L'adaptation aux légères déviations de modèle peut s'effectuer grâce à l'utilisation d'une fenêtre glissante d'horizon plus ou moins important.

L'algorithme de lissage classique, qui consiste à trouver le meilleur état possible en minimisant directement l'équation aux moindres carrés est adapté afin de détecter la présence de sauts. On utilise l'algorithme de lissage classique comme base. Pour détecter les sauts, on calcule la distance de Mahalanobis à chaque instant entre la dernière mesure, et la prédiction faite grâce à la propagation du modèle d'évolution. Cette distance suit une loi du χ^2 dans ce cas, et lorsque le seuil de la table du χ^2 est dépassé, un saut est détecté, et la fenêtre de calcul pour le lissage est réinitialisée. Dans un premier temps, on conserve une solution sans saut en parallèle de la solution avec saut, afin d'éviter des sauts intempestifs dus à des mesures aberrantes. Puis au bout d'un temps donné, seule la meilleure solution des deux est conservée.

L'algorithme ainsi obtenu permet de réaliser l'estimation de trajectoires avec des sauts abrupts et de forte amplitude. La précision de l'algorithme de lissage dépasse celle de l'IEKF avec le modèle de Frenet-Serret en 2D avec bruit de modèle, ainsi que celle d'un IMM avec trois modèles (vitesse constante, virage coordonné et accélération constante). De plus, il y a moins de paramètres à régler pour l'algorithme de lissage que pour les filtres, étant donné qu'il n'y a pas de bruit de modèle. Il faut régler la taille de l'horizon maximal de la fenêtre glissante, ainsi que la fréquence des sauts, et la durée pendant laquelle deux solutions peuvent cohabiter, mais ces réglages nécessitent moins de précision que les réglages des bruits de modèle des filtres.

5.2 Introduction

In chapter 2, we have proposed a target model that tackles trajectories that are designed with almost constant control commands. The underlying idea is to handle piecewise constant control commands. In chapter 3, we have presented a filtering algorithm to perform state estimation. We have seen in this chapter, and in chapter 4 that this filter can accommodate the jumps of the real trajectories. However, we would like to detect when jumps occur, and have a filter that reacts instantaneously to the jumps, which leads to faster convergence after the jumps.

One first method to perform this is to use a Rao-Blackwell particle filter, see [49]. This filtering method is based on sampling, and samples the time of the jumps. It is presented in more details in appendix D of this document. However, this filter requires a lot of particles. Moreover, the noise, necessary for a nonlinear target model makes the detection of the jumps sometimes difficult, when used with a non-linear target model.

The authors of [32] have used Piecewise Deterministic Markov Models (PDMM) to model the target's behaviour, resulting in Variable Rate models. Between jumps, trajectories are modelled by ordinary differential equations driven by constant inputs. This kind of trajectories have long been a key model in tracking: see for example the constant velocity model of [7], the coordinated turn model in [87], and the 2D Frenet-Serret model presented in chapter 2. Adapting particle filter's techniques to the continuous-time setting of the PDMM, the authors of [33] have proposed the Variable Rate Particle Filter (VRPF). However, such filters are also computationally demanding, as many particles are needed to fully cover the space of possible jumps and parameters.

In this chapter, we consider PDMM that are akin to those considered in the VRPF literature. Instead of using a particle filtering approach, we opt for a smoothing optimisation-based approach. The use of such techniques for filtering and tracking have long been known, but only recent advances in computers have allowed them to be fully implementable. We thus use smoothing methods to track the state of a PDMM driven by unknown constant inputs, and we use a probabilistic approach for jump detection. In the stationary phase, the state is well tracked as our deterministic-based model provides smooth trajectories that are not fluctuating, and in turn, the accuracy of the state estimates helps to rapidly detect jumps.

This chapter is structured as follows. We first present state of the art smoothing techniques in the field of robotics in section 5.3. Then in sections 5.4 and 5.5, we introduce a slightly different version of smoothing that we have imagined during this thesis, *i.e.* with a handling of jumps appropriately added in the model. An application for a linear target model is presented in section 5.6. Finally, in section 5.7, we use the vectorial formulation of the 2D Frenet-Serret model developed in section 2.6.1 to apply the jumping smoother developed.

5.3 Smoothing as an estimation procedure for target tracking

5.3.1 Smoothing as an alternative to filtering algorithms

The first works introducing smoothing in robotics were concerned with trajectory smoothing only, see for instance the works of [37], [55] or [75]. But smoothing can also be used as a solution to full estimation problems, such as the SLAM (Simultaneous localisation and Mapping) problem, see [46], [107], [69], [68]. The SLAM problem is the problem of computing a map of an unknown environment and of a robot position inside the map, given some known landmarks observed by the robot. This is a trending topic, since it is used for robot navigation, drone navigation, autonomous vehicles... Optimisation based smoothing [69] is considered as state of the art for SLAM and this field has boosted algorithmic developments in the realm of smoothing based techniques. The optimisation-based smoothing approaches have virtually replaced the once extremely popular particle filter, see [86].

Smoothing consists of solving a least squares problem on (a portion of) a trajectory to estimate both measured and hidden parameters. The formulation of the least squares problem comes from the model and measurement equations. It is equivalent to the Kalman filter for a linear and Gaussian problem (as we have seen in appendix B, the Kalman filter is a least squares error minimiser). In the case where it solves an estimation problem, smoothing can be used as an alternative to the Kalman filter. The main difference is that the Kalman filter is recursive and requires only the use of the last measurement point and estimation, whereas the smoother needs to have in memory the whole trajectory to perform its estimation. However, some linear algebra techniques, such as the Cholesky factorisation, the use of the sparsity of the matrices, and the square root form of the matrices can provide very fast computations. The authors of [46] thus claim that the square root smoothing and mapping is a more efficient and precise approach to the SLAM problem than the Extended Kalman Filter.

Some refinements for computational efficiency have been proposed in [69] and [68]. The idea is to derive an incremental algorithm based on matrix factorisation, to avoid computing the entire least squares problem at each time step. This solution includes both the estimation and the association problem, which is not dealt with in this work.

We present the general formulation and resolution of the smoothing problem used for state estimation in the remainder of this section.

5.3.2 Classical smoothing approach

Consider a target that one must track. Assume a discrete time model, and let the target's state at time i be denoted $\mathrm{X}_i \in \mathbb{R}^p$. Consider a non-linear evolution model for the target of the form

$$\mathrm{X}_{i+1} = f_i(\mathrm{X}_i) + w_i, \tag{5.1}$$

with noisy radar (range and bearing) measurements of the form

$$y_i = h(\mathrm{X}_i) + v_i \tag{5.2}$$

The goal of any filter, such as the EKF or the IMM, is to compute the distribution $p(\mathrm{X}_n|y_{0:n})$ of the present state X_n conditionally on past and present measurements $y_{0:n} := \{y_1, \ldots, y_n\}$. In contrast, a smoother (sometimes referred to as Kalman smoothing) computes the distribution $p(\mathrm{X}_{0:n}|y_{0:n})$ of the entire past trajectory $\mathrm{X}_{0:n} := \{\mathrm{X}_0, \ldots, \mathrm{X}_n\}$, conditionally on past measurements $y_{0:n} := \{y_1, \ldots, y_n\}$. Both a filter and a smoother allow us to find the best estimate of the state, that is, the most likely state X_n in the light of the information $y_{0:n}$ we have collected so far. This is referred to as the *maximum a posteriori* (MAP) estimate X_n. The MAP estimate of the entire past trajectory $\mathrm{X}_{0:n}$ is thus defined as $\mathrm{argmax}_{\mathrm{X}_0,\ldots,\mathrm{X}_n} \mathrm{P}(\mathrm{X}_{0:n}|y_{0:n})$, *i.e.*:

$$\mathrm{X}^*_{0:n} = \underset{\mathrm{X}_{0:n}}{\mathrm{argmin}} - \log \mathrm{P}(\mathrm{X}_{0:n}|y_{0:n}) \tag{5.3}$$

Under standard assumptions of independence of noises w_i, v_i we get

$$\mathrm{P}(\mathrm{X}_{0:n}|y_{0:n}) = \mathrm{P}(\mathrm{X}_0) \prod_{i=1}^{n} \mathrm{P}(\mathrm{X}_i|\mathrm{X}_{i-1}) \prod_{k=1}^{n} \mathrm{P}(y_k|\mathrm{X}_k)$$

In this equation $\mathrm{P}(\mathrm{X}_0)$ is a prior knowledge that we have on the initial state.

Under the assumption of Gaussian noises $w_i \sim \mathcal{N}(0, \mathrm{Q}_i)$ and $v_i \sim \mathcal{N}(0, \mathrm{N}_i)$ to represent respectively model and measurements uncertainties, from equation (5.1) we have that $\mathrm{P}(\mathrm{X}_i|\mathrm{X}_{i-1}) = \bar{\mathrm{C}} \exp\left(||f_i(\mathrm{X}_{i-1}) - \mathrm{X}_i||^2_{\mathrm{Q}_i}\right)$ and from (5.2), we have $\mathrm{P}(y_k|\mathrm{X}_k) = \bar{\mathrm{C}} \exp\left(||h(\mathrm{X}_{t_k}) - y_k||^2_{\mathrm{N}_k}\right)$. Thus we end up with the following non-linear least squares problem

$$\mathrm{X}^*_{0:n} = \underset{\mathrm{X}_{0:n}}{\mathrm{argmin}} \left\{ ||\mathrm{X}_0 - \bar{\mathrm{X}}_0||^2_{\mathrm{P}_0} + \sum_{i=1}^{n} ||f_i(\mathrm{X}_{i-1}) - \mathrm{X}_i||^2_{\mathrm{Q}_i} + \sum_{k=1}^{n} ||h(\mathrm{X}_k) - y_k||^2_{\mathrm{N}_k} \right\} \tag{5.4}$$

where the norm is the Mahalanobis distance defined by $||e||_\Sigma = e^{\mathrm{T}} \Sigma^{-1} e$ for Σ a covariance matrix, where the initial distribution $\mathrm{P}(\mathrm{X}_0)$ is assumed to be Gaussian with mean $\bar{\mathrm{X}}_0$ and covariance matrix P_0.

If the dynamical model f_i and measurement function h are non-linear, and a linearisation point is not available, one must resort to non-linear optimisation methods such as Gauss-Newton or Levenberg-Marquardt algorithm. The algorithm is based on successive linear approximations to (5.4), that iteratively improve the estimate $\mathrm{X}_{0:n}$. Indeed, at each iteration, by denoting $\hat{\mathrm{X}}_{0:n} = \{\hat{\mathrm{X}}_0, \cdots, \hat{\mathrm{X}}_n\}$ the current estimate, the problem may be linearised around $\hat{\mathrm{X}}_{0:n}$ as follows. We let:

$$\mathrm{F}_i = \left.\frac{\partial f_i(\mathrm{X})}{\partial \mathrm{X}}\right|_{\hat{\mathrm{X}}_{i-1}}, \; \mathrm{H}_k = \left.\frac{\partial h(\mathrm{X})}{\partial \mathrm{X}}\right|_{\hat{\mathrm{X}}_k}$$

Letting $a_i = \hat{\mathrm{X}}_i - f_i(\hat{\mathrm{X}}_{i-1})$, $c_k = y_k - h(\hat{\mathrm{X}}_k)$, and $p_0 = \hat{\mathrm{X}}_0 - \bar{\mathrm{X}}_0$, we have first orders linearisations of the terms in (5.4):

$$f_i(\mathrm{X}_{i-1}) - \mathrm{X}_i \approx (f_i(\hat{\mathrm{X}}_{i-1}) + \mathrm{F}_i \delta\mathrm{X}_{i-1}) - (\hat{\mathrm{X}}_i + \delta\mathrm{X}_i) = \mathrm{F}_i \delta\mathrm{X}_{i-1} - \delta\mathrm{X}_i - a_i$$

$$h(\mathrm{X}_k) - y_k \approx (h(\hat{\mathrm{X}}_k) + \mathrm{H}_k \delta\mathrm{X}_k) - y_k = \mathrm{H}_k \delta\mathrm{X}_k - c_k$$

So the optimisation problem can be approximated as

$$\delta\mathrm{X}^* = \underset{\delta\mathrm{X}}{\mathrm{argmin}} \left\{ ||p_0||^2_{\mathrm{P}_0} + \sum_{i=1}^{n} ||\mathrm{F}_i \delta\mathrm{X}_{i-1} - \delta\mathrm{X}_i - a_i||^2_{\mathrm{Q}_i} + \sum_{k=1}^{n} ||\mathrm{H}_k \delta\mathrm{X}_k - c_k||^2_{\mathrm{N}_k} \right\} \tag{5.5}$$

yielding at each iteration a linear least squares problem to solve. Noting that we can re-write norms as follows

$$||e||^2_\Sigma = e^{\mathrm{T}} \Sigma^{-1} e = (\Sigma^{-\mathrm{T}/2} e)^{\mathrm{T}} (\Sigma^{-\mathrm{T}/2} e) = ||\Sigma^{-\mathrm{T}/2} e||^2,$$

and stacking the matrices F_i and H_k in a large matrix A and the vectors p_0, a_i and c_k in a large vector b, (5.5) may be re-written as

$$\delta X^* = \underset{\delta X}{\operatorname{argmin}} ||A\delta X - b||^2 \tag{5.6}$$

The solution of this linear least squares problem is then notoriously obtained by equating the gradient of $||A\delta X - b||^2$ to 0, which yields

$$\delta X^* = (A^T A)^{-1} A^T b \tag{5.7}$$

A is a large but sparse matrix, and linear algebra methods can be used to compute efficiently this solution, as explained in e.g., [69], the Cholesky decomposition or the QR matrix factorisation allow us to efficiently compute $(A^T A)^{-1}$. The obtained solution δX^* of (5.7) depends on a particular realisation of random noises w_i, v_i, and varies due to fluctuations in the data y_i which are stacked in vector b. Its variability over a large number of noise realisations is encoded in the covariance matrix $\mathrm{Cov}(\delta X^*) = (A^T A)^{-1}$. The standard smoothing algorithm is summarised in algorithm 1.

Algorithm 1 Smoothing algorithm to perform state estimation

Input: Observations $y_1, \dots, y_m$, initial state X_0, model (f,h,Q,N)

Set $X_0^* = X_0$
for $k = 1, \dots, m$ **do**
  Perform prediction $\hat{X}_k = f_k(X_{k-1}^*)$
  Set $i = 0$ and $\hat{X}_j^i = \hat{X}_j$ for $j = 0, \dots, k$
  while The linearisation has not converged **do**
    $i := i + 1$
    Linearise f and h around $\hat{X}_k^i$
    Solve $\delta X^* = \mathrm{argmin}_{\delta X} ||A\delta X - b||^2$
    Update $X_j^i = X_j^{i-1} + \delta X^*$ for $j = 0, \dots, k$
  end while
  Set $X_j^* = \hat{X}_j^i$ for $j = 0, \dots, k$
end for
Output: $X_0^*, \dots, X_m^*$

As already mentioned, the objective of a filter is to return the state that maximises the posterior distribution $P(X_n|y_{0:n})$, whereas a smoother returns the maximum argument (argmax) of $P(X_{0:n}|y_{0:n})$. As time passes, n grows boundlessly and re-estimating the entire trajectory may become intractable. Typically, the matrix A that appears in (5.6) at each iteration is of dimension $O(n^2)$, yielding a $O(n^3)$ complexity to evaluate (5.7). As a result there have been various attempts to compute incrementally the MAP estimate for the smoothing problem. Notably, in robotics, the well-studied problem of SLAM has a structure that lends itself to such incremental methods, as proved in [69].

Another popular solution is to use a fixed-lag smoother, which aims to compute $P(X_{n-k:n}|y_{0:n})$ for some fixed lag $k \in \mathbb{N}$. Such smoothers are obtained by marginalising the old states $X_{0:n-k-1}$ out, see e.g. [47], see also [101].

5.3.3 Restriction to a deterministic evolution model over a sliding window as a tuning strategy

Actual motion of objects such as aircrafts and marine vehicles typically consist of a succession of distinct manoeuvres commanded by an operator. As a result, the trajectories of objects look like a succession of smooth trajectories that are well described by continuous time ordinary differential equations (ODE). In section 5.4.1, we will take into account the possibility of abrupt changes in the trajectory, but for now let us consider only the phase in between manoeuvres where the trajectory

is governed by deterministic equations. Prosaically, this means that the covariance matrix Q_i of process noise w_i in (5.1) is null. Thus (5.4) becomes:

$$\begin{aligned} \underset{\mathrm{X}_{0:n}}{\text{minimize}} \quad & \left\{ ||\mathrm{X}_0 - \bar{\mathrm{X}}_0||^2_{\mathrm{P}_0} + \sum_{k=1}^{n} ||h(\mathrm{X}_k) - y_k||^2_{\mathrm{N}_k} \right\} \\ \text{subject to} \quad & \mathrm{X}_i = f_i(\mathrm{X}_{i-1}), \quad i = 1, \dots, n. \end{aligned} \tag{5.8}$$

Of course, such a model is too rigid in practice, as there are always fluctuations in the target behaviour with respect to a model specified in advance. A boat or a plane may deviate slightly from its planned trajectory due to perturbations, or to slight motion adaptations from the pilot. This is why in the target tracking literature, the covariance Q_i of noise w_i is always positive, and serves as a tuning parameter.

Let us temporarily assume we are dealing with problem (5.8). To simplify the exposure, assume f_i, h_k are linear and let $\mathrm{F}_i, \mathrm{H}_k$ denote the corresponding matrices. This means $\mathrm{X}_k = \mathrm{F}_k \cdots \mathrm{F}_1 \mathrm{X}_0$, and thus $h(\mathrm{X}_k) = \mathrm{H}_k \mathrm{F}_k \cdots \mathrm{F}_1 \mathrm{X}_0$. Equivalently f_i, h may be considered as non-linear and F_i and H_k then represent their first-order expansion at convergence of the Gauss-Newton algorithm. As a result, solving problem (5.8) is equivalent to minimising $||\mathrm{X}_0 - \bar{\mathrm{X}}_0||^2_{\mathrm{P}_0} + \sum_{k=1}^{n} ||\mathrm{H}_k \mathrm{F}_k \cdots \mathrm{F}_1 \mathrm{X}_0 - y_k||^2_{\mathrm{N}_k}$ with respect to X_0. Let $\tilde{\mathrm{H}}_0 = \mathrm{I}d, \tilde{\mathrm{H}}_1 = \mathrm{H}_1 \mathrm{F}_1, \cdots \tilde{\mathrm{H}}_k = \mathrm{H}_k \mathrm{F}_k \cdots \mathrm{F}_1$. We see by applying the results of Section 5.3.2 that $\mathrm{Cov}(\mathrm{X}^*_{0:n}) = (\sum_{i=0}^{n} \tilde{\mathrm{H}}_i^{\mathrm{T}} \tilde{\mathrm{H}}_i)^{-1}$, and as X^*_n is obtained deterministically from X^*_0, it has a similar covariance. As a result, when there is no process noise, the confidence about the current state X^*_n obtained by solving problem (5.8) grows as $1/n$ where n is the number of measurements. Indeed as n grows new observations y_i do not modify much the value of X^*_0: the algorithm has gathered sufficient data to be totally confident about its estimate. However, as the model cannot be completely accurate due to unpredictability of the target's behaviour, new observations need to constantly impact the estimate for accurate tracking.

On the other hand, if process noise is considered, the covariance of X^*_n obtained by solving problem (5.4) is lower bounded by some matrix C^* that depends on the magnitude of the Q_i's. Matrix C^* is known as the Cramér-Rao bound. As a result, we see there are two different routes for the practitioner to tune its estimator. Either, one can attempt to solve (5.4) and tune the process noise Q_i, leading to an asymptotic confidence C^* about the estimate. Or we can consider the estimation problem with no process noise $\mathrm{Q}_i = 0$, leading to (5.8) , but only on a sliding window of size $\bar{k}$, that is,

$$\begin{aligned} \underset{\mathrm{X}_{n-\bar{k}:n}}{\text{minimize}} \quad & \left\{ \sum_{j=n-\bar{k}}^{n} ||h(\mathrm{X}_j) - y_j||^2_{\mathrm{N}_j} \right\} \\ \text{subject to} \quad & \mathrm{X}_i = f_i(\mathrm{X}_{i-1}), \quad i = n - \bar{k}, \dots, n. \end{aligned} \tag{5.9}$$

Of course those two routes are not strictly equivalent mathematically, but they may be viewed as different and equally valid ways to tune the smoother. The second route that consists in solving (5.9) at each time step n, is the one that we advocate in the present paper. In this case, the depth of the window $\bar{k}$ is the tuning parameter which appears as an alternative to process noise Q_i: one should bear in mind the resulting uncertainty about the current state X_n is of magnitude $1/\bar{k}$, and this should be tuned in accordance with the fit between evolution model $\mathrm{X}_i = f_i(\mathrm{X}_{i-1})$ and the true motion of the target (in the extreme case where the motion of the target is exactly modelled by this deterministic approach, one may set $\bar{k} = n$, on the other hand if the evolution is uncertain $\bar{k}$ should be set small).

In the literature devoted to systems identification, the use of recursive least squares is pivotal. For real-time implementation, where one must track a parameter that varies (slowly) over time, it is common to use a "forgetting factor" that gives less weight to old observations. Our approach may be related to this practice, by setting a forgetting factor as 1 for the $\bar{k}$ latest observations and 0 for the preceding ones.

5.4 Smoothing applied to deterministic systems with random jumps

5.4.1 Considered systems and simplifying assumptions

As already explained, actual motions of objects such as aircrafts and marine vehicles typically consist of a succession of distinct manoeuvres commanded by an operator. As a result, the trajectories of objects are in fact smooth and well described by continuous time ordinary differential equations (ODE) driven by constant inputs between change-points. This was advocated in particular by S. Godsill with various co-authors who proposed the variable rate particle filter [58], [32], [33], a sequential Monte-Carlo (SMC) method, well suited to piecewise deterministic models, and explained also in appendix D. Following [58], the target behaviour is modelled by the following piecewise deterministic Markov model:

$$\frac{d}{dt}x_t = f(x_t, u_{\mathrm{K}(t)}) \tag{5.10}$$

where $x_t \in \mathbb{R}^p$ is the continuous time state, $\mathrm{K}(t) \in \mathbb{N}$ is a stochastic point process that counts the number of random jumps up to time t, and $u_0, u_1, \cdots$ is a sequence of random inputs that drive the ODE (5.10). Moreover, at discrete time instants $t_0, t_1, \cdots$, we get (range and bearing) measurements of the form

$$y_n = h(x_{t_n}) + v_n \tag{5.11}$$

The goal is to estimate the most likely state value, that is $\operatorname{argmax} p(x_t \mid y_{0:n})$ for $t_n \le t < t_{n+1}$. To simplify the estimation task, we will assume jumps can only occur at pre-specified discrete times. This may look like a harmful approximation, but we will see in the sequel it is easy to modify the least squares problem to mitigate its impact on the estimation. Furthermore, to keep the notation simple we will assume jump times coincide with observation times $t_1, t_2, \cdots$. We let r_k be the random variable indicating jump at time k ($r_k = 1$ if there is a jump). $\mathrm{K}(t)$ is the number of jumps between times 0 and t. We also let $\tilde{\mathrm{K}}_n$ to be the number of jumps between 0 and t_n. We obviously have $\mathrm{K}(t_n) = \tilde{\mathrm{K}}_n$. Note that, $\tilde{\mathrm{K}}_n$ is a function of $r_{0:n}$. For simplicity, and as done in the variable rate particle filter literature [58], we assume the initial state x_0 to be known. Finally, we let $\theta_n = (r_{0:n}, u_{0:\tilde{\mathrm{K}}_n}, x_0)$ be the parameters we seek to estimate. Indeed, for known x_0, x_t is a (deterministic) function of θ_n for $t \le t_n$, so to recover x_t, we only need to integrate (5.10) based on the knowledge of θ_n for $t < t_{n+1}$.

5.4.2 Corresponding smoothing problem

For the problem described at the preceding paragraph, our goal is to find the most likely state x_t at present time, and for $t_n \le t < t_{n+1}$ this amounts to finding the parameter θ_n. We have

$$\log p(y_{0:n}|\theta_n) = \log p(y_{0:n}|r_{0:n}, u_{0:\tilde{\mathrm{K}}_n}, x_0) = -\sum_{k=1}^{n} ||y_k - h(x_{t_k})||^2_{\mathrm{N}_k} + \mathrm{Cste} \tag{5.12}$$

where N_k is the covariance matrix of the Gaussian measurement noise v_k, and the x_{t_k} are obtained by integrating (5.10), since $\mathrm{K}(t_k)$ is a function of $r_{0:k}$. Trying to estimate θ_n by maximising the likelihood (5.12) is not suitable. Indeed, the optimal solution will keep jumping to stick to the observations. Obviously, we need a prior on the average time between successive manoeuvres. Letting $0 < p < 1$ be the probability of a jump at each observation time t_i, we have the following prior

$$p(\tilde{\mathrm{K}}_n = j) = p(\{r_{0:n} \text{ contains } j \text{ ones and } n-j \text{ zeros}\}) = \mathrm{P}(\mathrm{bin}(n,p) = j) = \binom{n}{j} p^j (1-p)^{n-j} \tag{5.13}$$

Let us assume a prior on the initial state: $x_0 \sim \mathcal{N}(\bar{x}_0, \mathrm{P}_0)$. Estimating the most likely state for the piecewise deterministic model of section 5.4.1 amounts to the following optimisation problem:

$$\begin{aligned}
\theta_n^* &= \underset{r_{0:n}, u_{0:\tilde{\mathrm{K}}_n}, x_0}{\operatorname{argmin}} \; -\log p(r_{0:n}, u_{0:\tilde{\mathrm{K}}_n}, x_0 | y_{0:n}) \\
&= \underset{r_{0:n}, u_{0:\tilde{\mathrm{K}}_n}, x_0}{\operatorname{argmin}} \; \left[-\log p(y_{0:n} | r_{0:n}, u_{0:\tilde{\mathrm{K}}_n}, x_0) - \log p(u_{0:\tilde{\mathrm{K}}_n} | r_{0:n}) - \log p(r_{0:n}) - \log p(x_0) \right] \\
&= \underset{r_{0:n}, u_{0:\tilde{\mathrm{K}}_n}, x_0}{\operatorname{argmin}} \left[\sum_{k=1}^{n} ||y_k - h(x_{t_k})||^2_{\mathrm{N}_k} - \log \left(\binom{n}{\tilde{\mathrm{K}}_n} p^{\tilde{\mathrm{K}}_n} (1-p)^{n-\tilde{\mathrm{K}}_n} \right) - ||x_0 - \bar{x}_0||^2_{\mathrm{P}_0} \right]
\end{aligned} \tag{5.14}$$

with the x_{t_k} are obtained by integrating (5.10) with parameters $r_{0:n}, u_{0:\tilde{\mathrm{K}}_n}$. The justification for removing the term $\log p(u_{0:\tilde{\mathrm{K}}_n} | r_{0:n})$ from the optimisation problem is that we assume a flat prior on the parameters $u_0, u_1, \cdots$, as we assume at each jump they can completely shift. Of course alternative priors might be considered depending on the application.

5.5 Proposed algorithm

The optimisation problem (5.14) is not tractable, owing to the combinatorics in the jump times. Unfortunately, this remains true even if the optimisation is restricted to a sliding window along the lines of section 5.3.3. Over a window of size k, there are 2^k possibilities for the discrete variable $r_{n-k:n}$, each leading to a continuous optimisation problem with respect to $u_{\mathrm{K}_{n-k}:\tilde{\mathrm{K}}_n}$. To efficiently approximate the optimisation problem, we propose the following tractable strategy.

Setting a horizon We first choose a size $\bar{k}$ for a sliding window for the reasons explained in section 5.3.3, corresponding to the forgetting horizon of the smoother in the absence of jumps: even if there are no jumps this allows the state to deviate from deterministic model (5.10) while continuing to be efficiently tracked.

Continuity assumption of x_t at jumps In our model described in section 5.4.1 we made the simplifying assumptions that jumps only occurred at discrete time instants $t_0, t_1, \cdots$ while they actually can occur at any time. We also said there was a way around the issue this poses. Indeed, the "true" considered model (5.10) implies continuity in x_t, since its derivative is bounded. If a jump actually occurs between t_{n-1} and t_n, for instance at time $(t_{n-1} + t_n)/2$, assuming it has occurred at time t_n and the trajectory is continuous may result in degraded accuracy. However, by relaxing the continuity assumption and assuming small jumps in the state x_t may occur as well at jumping times, i.e. when u is jumping, may compensate for the assumption that jumps may only occur at pre-specified instants. See figure 5.1 for a graphical illustration.

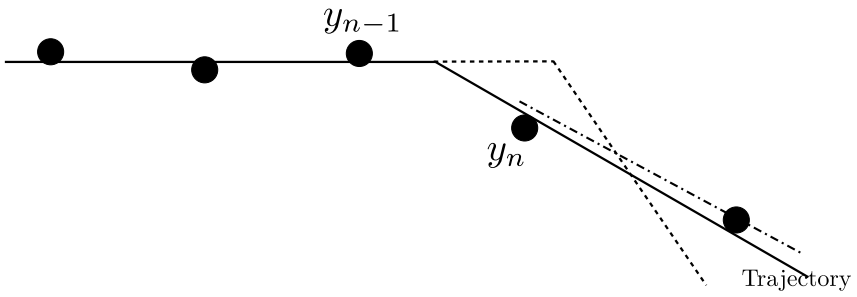


Figure 5.1 – Trajectory with velocity u jumping strictly between the observations at times t_{n-1} and t_n. Under the assumption that a jump may only occur at t_n and the trajectory x_t is continuous we obtain the dotted line which is a poor trajectory estimate. However if we assume u jumps at time t_n but we relax the assumption that the trajectory x_t must be continuous and allow it to jump - see (5.15) - we obtain a much better estimate (dashed line).

Assuming a jump has occurred at a time *between* t_{l-1} and t_l, and given that no other jump has occurred until current time n, to find the most likely state x_{t_n} we relax the continuity assumption and solve the optimisation problem

$$\underset{u,x_{t_l}}{\operatorname{argmin}} \left[||x_{t_l} - \bar{x}_{t_l}||^2_{\mathrm{P}_{jump}} + \sum_{j=l}^{n} ||y_j - h(x_{t_j})||^2_{\mathrm{N}_j} \right], \tag{5.15}$$

where $\bar{x}_{t_l}$ is the value obtained at instant index l by integrating (5.10) until time t_l based on the former value of u (i.e. extension by continuity), and where P_{jump} is a covariance matrix that must be tuned to represent the typical squared distance x_t may achieve between successive observations.

Jump detection Assume n denotes the current time step, and the last jump occurred at time $l > n - \bar{k}$. This means $r_{l:n}$ contains only zeros, and the state is since driven by equation (5.10) with input $u_{\tilde{\mathrm{K}}_n}$. Using (5.12) we may solve the problem

$$\underset{u,x_{t_l}}{\operatorname{argmin}} -\log p(u, x_{t_l} | y_{l:n}, r_{l:n}) = \underset{u,x_{t_l}}{\operatorname{argmin}} \left[||x_{t_l} - \bar{x}_{t_l}||^2_{\mathrm{P}_{jump}} + \sum_{k=l}^{n} ||y_k - h(x_{t_k})||^2_{\mathrm{N}_k} \right] \tag{5.16}$$

where the x_{t_k} are obtained by integrating $\dot{x}_t = f(x_t, u)$ with initial condition x_{t_l}, and $\bar{x}_{t_l}$ corresponds to the estimate obtained by continuity with the model before the jump. As explained in section 5.3.2, it is possible to assess a covariance matrix $\tilde{\mathrm{P}}$ to the couple (u, x_{t_l}). As in the absence of jumps $x_{t_n} = \phi(u, x_{t_l})$ is a deterministic function of the parameters (u, x_{t_l}), the corresponding covariance matrix is given by $\mathrm{P}_n = \mathrm{D}\phi\tilde{\mathrm{P}}\mathrm{D}\phi^{\mathrm{T}}$ where $\mathrm{D}\phi$ denotes the differential of ϕ at the optimal values $(u^*, x^*_{t_l})$. This allows us to compute the associated Mahalanobis distance

$$\Delta = \sqrt{(y_k - h(x^*_{t_n}))^{\mathrm{T}} (\mathrm{D}h^{\mathrm{T}} \mathrm{P}_n \mathrm{D}h)^{-1} (y_k - h(x^*_{t_n}))} \tag{5.17}$$

where $\mathrm{D}h$ denotes the differential of h at $x^*_{t_n}$. We may then apply the χ^2-test with a given threshold to determine if there is a jump in the parameters u that are considered piecewise deterministic in the model.

Proposed strategy We approximate the solution to the true optimisation problem (5.14) by first restricting it to a sliding window of horizon $\bar{k}$. Then, we assume jumps are scarce (that is, the jump probability p is small) and we let $\mathrm{T}_n \in \mathbb{N}$ denote the time index at which the last jump before current (observation) time t_n occurred. At each jump time T_n, the window is re-initialised, since u jumps to an unknown arbitrary value. As a result, at time t_n, the current optimal parameter $u^*_{\tilde{\mathrm{K}}_n}$ is obtained as a solution to problem (5.16) with $l = \max(n - \bar{k}, \mathrm{T}_n)$. Assume according to the χ^2 test a possible jump is detected at time index n as (5.17) goes above some predefined quantile q, corresponding to, say 95%. As this does not mean a jump has necessarily occurred, we initialise a second smoother based on jump at time n. The former smoother is assumed not to have jumped and u continues to be re-evaluated given new observations y_i, $i \geq n$.

Suppose we are at time $n + k$, and let $l = \max(\mathrm{T}_{n-1}, n + k - \bar{k})$. The smoother assuming "no jump has occurred at n" solves the problem:

$$\underset{u,x_{t_l}}{\operatorname{argmin}} \left(||x_{t_l} - \bar{x}_{t_l}||^2_{\mathrm{P}_{jump}} + \sum_{j=l}^{n+k} ||y_j - h(x_{t_j})||^2_{\mathrm{N}_j} \right)$$

where x_{t_j} are obtained by integrating $\dot{x}_t = f(x_t, u)$. Recalling computations (5.14) the associated log likelihood writes

$$\mathrm{L}_{\text{no jump}}(n + k) = \log p(y_{l:n+k} | u^*, x^*_{t_l}) + (n + k - l) \log(1 - p) + \log p(x^*_{t_l}) \tag{5.18}$$

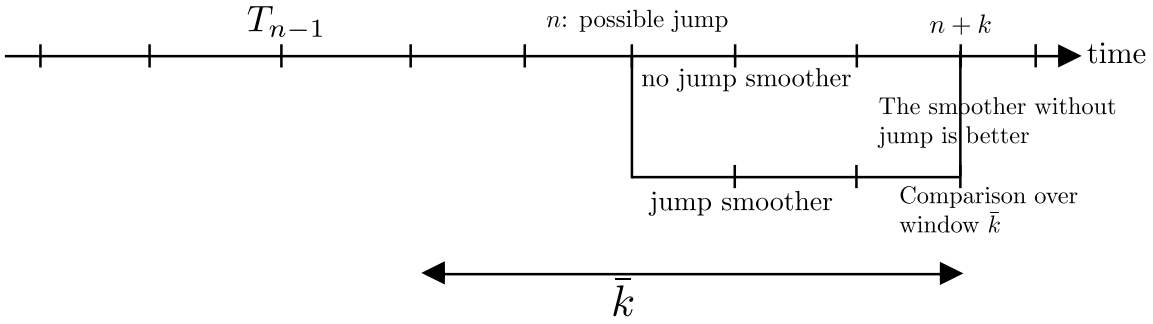


Figure 5.2 – Smoothing with jumps and underlying non-jumping smoother.

On the other hand, the candidate "jumping" smoother which assumes one jump occurred at time index n solves the optimisation problem

$$\underset{u,x_{t_n}}{\operatorname{argmin}} \left(||x_{t_n} - \bar{x}_{t_n}||^2_{\mathrm{P}_{jump}} + \sum_{j=n}^{n+k} ||y_j - h(x_{t_j})||^2_{\mathrm{N}_j} \right) \tag{5.19}$$

where x_{t_j} are obtained by integrating $\dot{x}_t = f(x_t, u)$. The associated log likelihood writes

$$\mathrm{L}_{\mathrm{jump}}(n+k) = \log p(y_{n:n+k}|u^*, x^*_{t_n}) + \log\left(kp(1-p)^{k-1}\right) + \log p(x^*_{t_n}) \tag{5.20}$$

We clearly see the benefit of the prior: as the size of window for the jumping smoother is smaller, the residual of the least squares will be smaller, as it is easier to find a u^* fitting with less data. But its likelihood will be penalised by the jump assumption as p is assumed small, that is, as long as $kp < 1-p$. This will favor the non jumping smoother, and prevent the estimation from constantly jumping, which may end up in yielding meaningless estimates. As p is assumed small, the likelihood of smoothers corresponding to two or more jumps is considered as negligible.

Note that there must be a minimum time elapsed after the candidate jump at n for comparing the smoothers. We will denote by index k.

After a possible jump detection at time n, our strategy is to let j increase until either $\mathrm{L}_{\mathrm{jump}}(n+j) > \mathrm{L}_{\mathrm{no\,jump}}(n+j)$ and then a jump at n is validated, leading to $\mathrm{T}_n = \mathrm{T}_{n+j} = n$, or until $j = k$ in which case both smoothers yield the same estimates and the jumping smoother is suppressed. The strategy is illustrated by figure 5.2.

Algorithm The pseudo code is displayed in algorithm 2. After a jump, during the phase when two smoothers run in parallel, we can either decide to output the estimations of the jumping or the non jumping smoother. The complexity of this algorithm corresponds to the complexity of the smoothing problem on a sliding window, multiplied by two, because two smoothers at most can be kept in parallel for a few time steps.

Algorithm 2 Smoothing algorithm with jumps

Input: Observations $y_1, y_2, \cdots, \bar{x}_0, \mathrm{P}_0$

1: Set $\mathrm{P}_{jump} = \mathrm{P}_0$, $\mathrm{T}_0 = 0$, $n = 0$
2: Solve $(u^*, x^*_{t_l}) = \mathrm{argmin}_{u,x_{t_l}} \left[||x_{t_l} - \bar{x}_{t_l}||^2_{\mathrm{P}_{jump}} + \sum_{j=l}^{n} ||y_j - h(x_{t_j})||^2_{\mathrm{N}_j} \right]$ with $l = \max(0, n - \bar{k})$ and where $\bar{x}_{t_l}$ is either $\bar{x}_0$ or obtained by continuity through model (5.10) if $l > 0$.
3: **while** $\Delta^2 < q$ with Δ defined by (5.17) and q the quantile for the χ^2-test **do**
4: Set $\mathrm{T}_n = \mathrm{T}_{n-1}$
5: $n = n + 1$
6: **if** $\Delta^2 > q$, with q the quantile for the χ^2-test, *i.e.* a candidate jump is detected at time index n **then**
7: **for** $j = n : n + k$ **do**
8: Compute estimations for a smoother with jump and with no jump.
9: **end for**
10: **if** for some j we have $\mathrm{L}_{\text{jump}}(j) > \mathrm{L}_{\text{no jump}}(j)$ for one jump (see (5.18), (5.20) for a definition) **then**
11: Set $\mathrm{T}_j = n$ and select the jumping smoother by selecting $(u^*, x^*_{t_l}) = \mathrm{argmin}_{u,x_{t_n}} \left[||x_{t_n} - \bar{x}_{t_n}||^2_{\mathrm{P}_{jump}} + \sum_{j=n}^{n} ||y_j - h(x_{t_j})||^2_{\mathrm{N}_j} \right]$ where $\bar{x}_{t_n}$ is obtained by continuity through model (5.10) with optimal u^*.
12: **end if**
13: Set $n = j$
14: **end if**
15: **end while**

Output: x_t is obtained for $t_l \leq t < t_{n+1}$ with $l = \min(\mathrm{T}_n, n - \bar{k})$ by integrating (5.10) with parameters $(u^*, x^*_{t_l})$.

5.6 Application to a linear target model

We propose in this section to apply this least squares formulation to the target tracking problem. We first present results on a linear target model, for visual understanding. For simplicity, we assume 2D Cartesian observations.

5.6.1 Target model

The target model is supposed to be linear, and deterministic, *i.e.* without process noise. The model and observation equations are:

$$\mathrm{X}_{k+1} = \mathrm{F}_k \mathrm{X}_k \tag{5.21}$$

$$y_k = \mathrm{H}_k \mathrm{X}_k + v_k \tag{5.22}$$

v_k is a white Gaussian noise with covariance N. The corresponding least squares problem is

$$\mathrm{L(N)} = ||\mathrm{X}_0 - \hat{\mathrm{X}}_0||^2_{\mathrm{P}_0} + \sum_{k=1}^{\mathrm{N}} ||y_k - \mathrm{H}_k \mathrm{X}_k||^2_{\mathrm{N}} \tag{5.23}$$

5.6.2 Full resolution of the deterministic problem

This problem is relatively easy to solve. Indeed, we can express the state at each time instant as a function of the state at the first instant, like this:

$$\mathrm{X}_k = \mathrm{F}_k \mathrm{F}_{k-1} \ldots \mathrm{F}_0 \mathrm{X}_0$$

So the function to optimise L(N) can be re-written in the following way, assuming the initial state X_0 is known:

$$\mathrm{L(N)} = \sum_{k=1}^{\mathrm{N}} ||y_k - \mathrm{H}_k \mathrm{F}_k \mathrm{F}_{k-1} \ldots \mathrm{F}_0 \mathrm{X}_0||^2_{\mathrm{N}}$$

Let us call

$$\tilde{H}_k = H_k F_k F_{k-1} \dots F_0$$

$$y = \begin{pmatrix} y_1 \\ \vdots \\ y_N \end{pmatrix}$$

$$A = \begin{pmatrix} \tilde{H}_1 \\ \vdots \\ \tilde{H}_N \end{pmatrix}$$

The function L(N) writes

$$L(N) = ||y - AX_0||_N^2$$

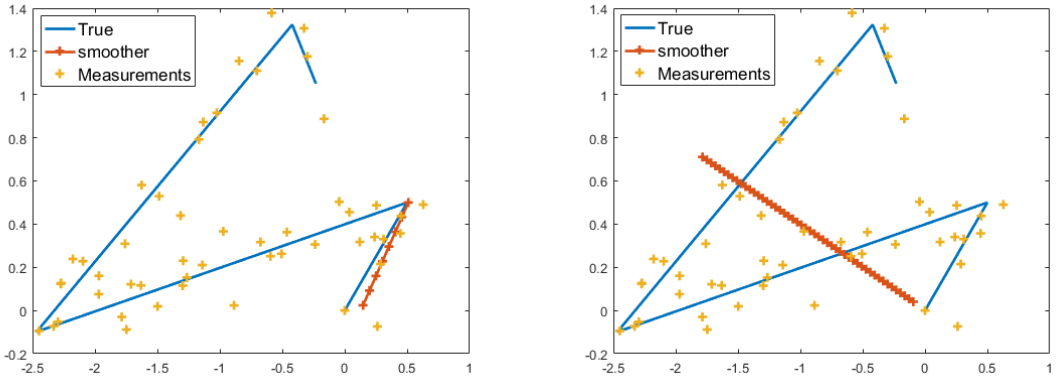
The solution is given by

$$\hat{X}_0 = (\tilde{A}^T \tilde{A})^{-1} \tilde{A} \tilde{y} \tag{5.24}$$

with $\tilde{A} = \begin{pmatrix} \tilde{A}_1 & \dots & \tilde{A}_N \end{pmatrix}^T$, $\tilde{y} = \begin{pmatrix} \tilde{y}_1 & \dots & \tilde{y}_N \end{pmatrix}^T$, $\tilde{A}_k = N^{-T}/2\tilde{H}_k$ and $\tilde{y}_k = N^{-T/2} y_k$.

In this particular case, it is possible to propagate the state, knowing only the initial state X_0. This is due to the fact that the target model equation is deterministic (no process noise is added). So the smoothing algorithm only has to estimate the best X_0 that minimises the error between the estimated trajectory and the observations.

An example of an estimation coming from a linear smoothing algorithm without process noise is given on figure 5.3. The model chosen is a constant velocity model. The trajectory is generated with a piecewise constant velocity model. We see that the algorithm performs well until a jump occurs. This is similar to what a linear Kalman filter without process noise would do, as presented in section 4.3.



(a) (b)

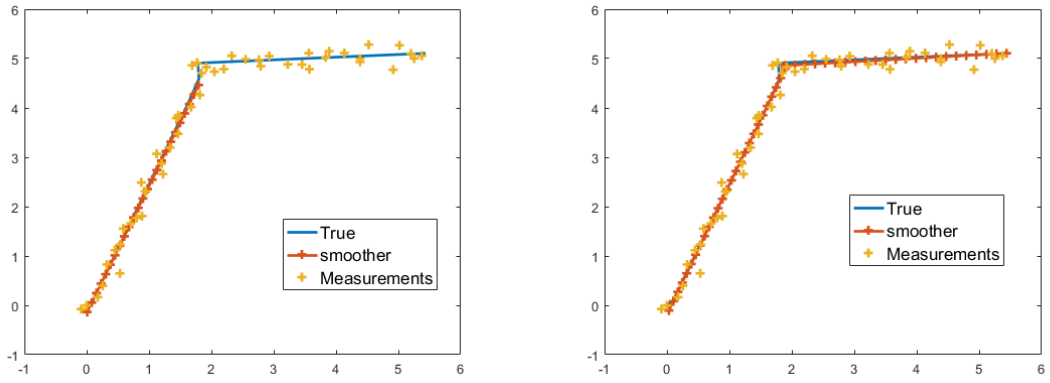
Figure 5.3 – Estimations of the position for the smoothing algorithm at different time steps. The true trajectory is in blue, the measurements are in yellow, and the estimation in red. On fig. 5.3a the smoother works well because the model is still valid, however, with time increasing and jumps in the trajectory, we see the smoothing algorithm is very wrong. The final estimation is on fig. 5.3b.

This solution is a very rigid one, since no process noise is added, and the sliding window is very large. Indeed, in the case of a constant velocity model for instance, the smoothing algorithm will select the best straight line that goes through the trajectory. One way to avoid this rigidity problem is to allow jumps in the model, and use algorithm 2 developed in section 5.5.

5.6.3 Linear target model with jumps

We generate a trajectory with piecewise constant velocity. We apply algorithm 2 to perform estimation on this trajectory, with a target model containing the target position and piecewise constant

velocity. The results at two different moments in the estimation process are presented on figure 5.4. This shows the algorithm performs quite well. However, we notice that in the trajectory, two jumps occur, whereas our algorithm detect only one. It is due to the fact that the amplitude of the first jump is quite low, and is hidden in the measurement noise. It would be hard for any change detection algorithm to detect a jump like this.



(a) (b)

Figure 5.4 – Estimations of the position for the smoothing algorithm at different time steps. The true trajectory is in blue, the measurements are in yellow, and the estimation in red. On fig. 5.4a the smoothing estimation is presented when no jump has yet occurred. On fig. 5.4b, the final estimation is presented. The estimation of the first part has not changed after the jump. Indeed, the window on which smoothing is performed has be re-initialised at the moment of the jump.

The algorithm thus performs almost optimally on piecewise linear trajectories. In the next section, we use the jumping smoothing algorithm for a non-linear target model, and compare it with two filtering algorithms, the IEKF of section 3.6, and the IMM.

5.7 Application to the 2D Frenet-Serret target model

Let us apply the smoothing algorithm to the 2D non-linear Frenet-Serret model, expressed with a vectorial state, and vectorial equations (2.28), but without process noise. The model is recalled here:

$$\frac{d}{dt}\theta_t = \omega_t,\ \frac{d}{dt}x_t = u_t\begin{pmatrix}\cos\theta_t\\ \sin\theta_t\end{pmatrix},\ \frac{d}{dt}\omega_t = 0,\ \frac{d}{dt}u_t = 0$$

The measurements are supposed to be Cartesian position, with additive Gaussian noise, as in (2.24), but in 2D.

5.7.1 Solving the smoothing problem without jumps

Let us first consider the problem without jumps, to explain the linearisations performed. We use only discrete time, so the state is

$$\mathrm{X}_k = \begin{pmatrix}\theta_k\\ x_k\\ \omega_k\\ u_k\end{pmatrix}$$

The corresponding least squares problem is

$$||\mathrm{X}_0 + \hat{\mathrm{X}}_0||^2_{\mathrm{P}_0} + \sum_{k=0}^{\mathrm{N}} ||y_k - h(\mathrm{X}_k)||^2_{\mathrm{N}} \tag{5.25}$$

In order to solve the least squares problem (5.25), we have to linearise $h(\mathrm{X}_k)$. In our case, since the model is deterministic, we can even approximate $h(\mathrm{X}_k)$ as a linear function of X_0. For Cartesian measurements, $h(\mathrm{X}_k) = x_k$. We have

$$x_n = x_0 + u_0 \Delta t \sum_{k=0}^{n-1} \begin{pmatrix} \cos(\theta_0 + k\omega_0\Delta t)\frac{\sin(\omega_0\Delta t)}{\omega_0} - \sin(\theta_0 + k\omega_0\Delta t)\frac{1-\cos(\omega_0\Delta t)}{\omega_0} \\ \sin(\theta_0 + k\omega_0\Delta t)\frac{\sin(\omega_0\Delta t)}{\omega_0} + \cos(\theta_0 + k\omega_0\Delta t)\frac{1-\cos(\omega_0\Delta t)}{\omega_0} \end{pmatrix} \tag{5.26}$$

Then we can perform linearisation. Let us suppose for simplification that ω_0 is small, which means that

$$x_n = x_0 + u_0 \Delta t \sum_{k=0}^{n-1} \begin{pmatrix} \cos(\theta_0 + k\omega_0\Delta t) \\ \sin(\theta_0 + k\omega_0\Delta t) \end{pmatrix}$$

The full computations for $\omega_0 \neq 0$ are provided in appendix C. Let us set

$$\begin{aligned} x_0 &= \tilde{x}_0 + \delta x_0 \\ \theta_0 &= \tilde{\theta}_0 + \delta\theta_0 \\ \omega_0 &= \tilde{\omega}_0 + \delta\omega_0 \\ u_0 &= \tilde{u}_0 + \delta u_0 \end{aligned}$$

We inject this in the formula giving x_n, and we get:

$$x_n = \tilde{x}_0 + \delta x_0 + (\tilde{u}_0 + \delta u_0)\Delta t \sum_{k=0}^{n-1} \begin{pmatrix} \cos(\tilde{\theta}_0 + \delta\theta_0 + k(\tilde{\omega}_0 + \delta\omega_0)\Delta t) \\ \sin(\tilde{\theta}_0 + \delta\theta_0 + k(\tilde{\omega}_0 + \delta\omega_0)\Delta t) \end{pmatrix}$$

With order 1 linearisation, we finally get:

$$\begin{aligned} x_n = \tilde{x}_0 &+ \tilde{u}_0 \Delta t \sum_{k=0}^{n-1} \begin{pmatrix} \cos(\tilde{\theta}_0 + k\tilde{\omega}_0\Delta t) \\ \sin(\tilde{\theta}_0 + k\tilde{\omega}_0\Delta t) \end{pmatrix} \\ &+ \tilde{u}_0 \Delta t \sum_{k=0}^{n-1} (\delta\theta_0 + k\delta\omega_0\Delta t) \begin{pmatrix} -\sin(\tilde{\theta}_0 + k\tilde{\omega}_0\Delta t) \\ \cos(\tilde{\theta}_0 + k\tilde{\omega}_0\Delta t) \end{pmatrix} + \delta u_0 \Delta t \sum_{k=0}^{n-1} \begin{pmatrix} \cos(\tilde{\theta}_0 + k\tilde{\omega}_0\Delta t) \\ \sin(\tilde{\theta}_0 + k\tilde{\omega}_0\Delta t) \end{pmatrix} \end{aligned}$$

Let us call

$$\begin{aligned} \mathrm{C1}(n) &= \sum_{k=0}^{n-1} \cos(\tilde{\theta}_0 + k\tilde{\omega}_0\Delta t) \\ \mathrm{S1}(n) &= \sum_{k=0}^{n-1} \sin(\tilde{\theta}_0 + k\tilde{\omega}_0\Delta t) \\ \mathrm{C2}(n) &= \sum_{k=0}^{n-1} k\cos(\tilde{\theta}_0 + k\tilde{\omega}_0\Delta t) \\ \mathrm{S2}(n) &= \sum_{k=0}^{n-1} k\sin(\tilde{\theta}_0 + k\tilde{\omega}_0\Delta t) \end{aligned}$$

Assuming the initial position is known, the problem we seek to minimise thus writes:

$$\sum_{k=0}^{\mathrm{N}} ||y_k - \tilde{x}_0 - \tilde{u}_0\Delta t \begin{pmatrix} \mathrm{C1}(k) \\ \mathrm{S1}(k) \end{pmatrix} - \tilde{u}_0\Delta t\delta\theta_0 \begin{pmatrix} -\mathrm{S1}(k) \\ \mathrm{C1}(k) \end{pmatrix} - \tilde{u}_0\delta\omega_0(\Delta t)^2 \begin{pmatrix} -\mathrm{S2}(k) \\ \mathrm{C2}(k) \end{pmatrix} - \delta u_0\Delta t \begin{pmatrix} \mathrm{C1}(k) \\ \mathrm{S1}(k) \end{pmatrix} ||^2_{\mathrm{N}} \tag{5.27}$$

This can be simplified into

$$||\mathrm{A}X - b||^2_{\mathrm{N}} \tag{5.28}$$

with

$$\mathrm{A} = \begin{pmatrix} \mathrm{A}_1 \\ \vdots \\ \mathrm{A}_{\mathrm{N}} \end{pmatrix}, \; b = \begin{pmatrix} b_1 \\ \vdots \\ b_{\mathrm{N}} \end{pmatrix}$$

$$\mathrm{A}_n = \mathrm{N}^{-\mathrm{T}/2}\mathrm{G}_n$$

and

$$b_n = \mathrm{N}^{-\mathrm{T}/2}\left(y_n - x_0 - u_0\Delta t\begin{pmatrix}\mathrm{C1}\\ \mathrm{S1}\end{pmatrix}\right)$$

with

$$\mathrm{G}_n = \begin{pmatrix}-u_0\Delta t\mathrm{S1}(n) & 1 & 0 & -u_0(\Delta t)^2\mathrm{S2}(n) & \Delta t\mathrm{C1}(n)\\ u_0\Delta t\mathrm{C1}(n) & 0 & 1 & u_0(\Delta t)^2\mathrm{C2}(n) & \Delta t\mathrm{C1}(n)\end{pmatrix}$$

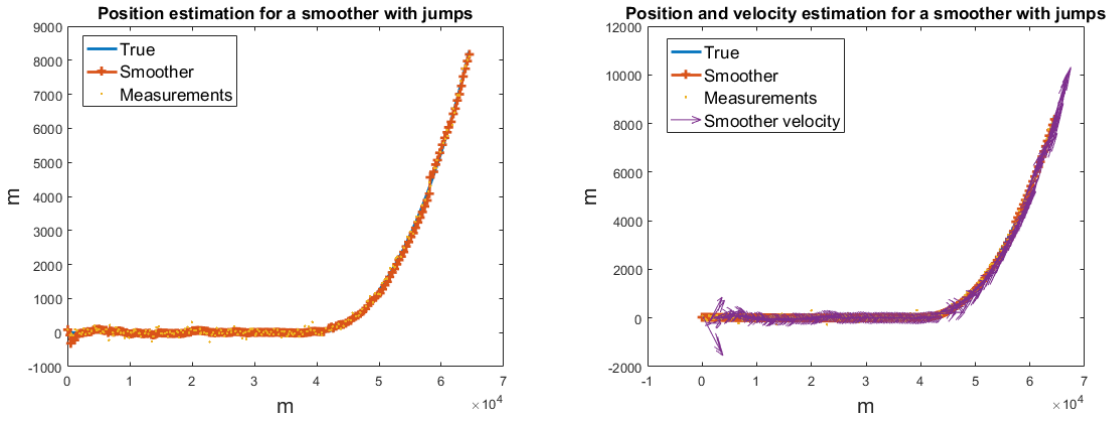
It is of course possible to add the initial state in this problem by integrating $||\mathrm{X}_0 - \hat{\mathrm{X}}_0||^2_{\mathrm{P}_0}$ in these matrices.

At time instant n, we can then estimate $\delta\mathrm{X}_0 = \begin{pmatrix}\delta\theta_0 & \delta x_0 & \delta\omega_0 & \delta u_0\end{pmatrix}^{\mathrm{T}}$ by finding the argument that minimises (5.28) around the linearisation point $\tilde{\mathrm{X}}_0 = \begin{pmatrix}\tilde{\theta}_0 & \tilde{x}_0 & \tilde{\omega}_0 & \tilde{u}_0\end{pmatrix}^{\mathrm{T}}$. A Gauss-Newton algorithm iterates the resolution to finally give X_0^*. Algorithm 2 can be applied to this linearised problem.

5.7.2 Accounting for jumps

For the 2D Frenet-Serret target model, with Cartesian observations, we have formulated the problem in the same way as the linear problem, meaning that we can recover the full trajectory from the knowledge of the state at the origin, X_0. This means that algorithm 2 can be applied to this model. The jumping parameters are (ω, u), as they are considered constant in the deterministic target model.

An example is provided on figure 5.5. The trajectory presented has been created with a 2D Frenet-Serret target model, with random jumps for the angular velocity ω and the norm of the velocity u. The algorithm applied for the estimation is algorithm 2, with the non-linear Frenet-Serret model. The sliding window horizon is tuned very long (more than the entire size of the trajectory).



(a) (b)

Figure 5.5 – The results for the precision estimation is given in fig. 5.5a, and for the velocity estimation in fig. 5.5b. The real trajectory has two jump. The smoothing algorithm detects also two.

5.8 Comparison with other algorithms

In this section, we lead two different comparisons: the first one is performed with an IEKF, a filter which is adapted to the Frenet-Serret target model, but which does not inherently adapt to the jumps. We of course add process noise to the model to run the IEKF. The second one is performed with an IMM. The IMM deals naturally with jumps, and it should perform well on trajectories with jumps.

Parameter	Smothing: smoothed	Smoothing: real time	IEKF
Position (m)	35	51	45
Norm of velocity (m/s)	15	29	31
Orientation (rad)	0.077	0.16	0.12

Table 5.1 – RMSE for the smoothing algorithm and the IEKF on 100 trajectories simulated with random jumps in the angular and tangential velocities

In this section, the smoothing algorithm 2 is tuned as follows: the sliding window horizon is taken to be the entire trajectories, and we tune the number of observations we need to wait after a candidate jump to accept or reject as $k = 7$. The jump probability is tuned very low: the average jump probability by unit step p is tuned such that the average number of jumps over the whole trajectory is 0.02, whereas the actual number of jumps in the trajectory is around 2.

5.8.1 Comparison with the IEKF

Let us first compare this estimation method with the filter adapted to the Frenet-Serret target model, the IEKF. The IEKF is *a priori* not adapted to trajectories with very abrupt jumps, although we have seen in chapters 3 and 4 that it can still converge quite rapidly after a jump if some large enough process noise is added to the model.

Trajectories with jumps in the norm of the velocity and the angular velocity

To draw a first comparison between the IEKF and the smoothing algorithm, let us generate trajectories as in section 5.7.2, *i.e.* with jumps on the norm of the velocity and on the angular velocity. We run 100 Monte-Carlo experiments with random jumps in the trajectory. The RMSE for the position, the norm of the velocity and the orientation angle are gathered in table 5.1. Since the smoothing algorithm has an impact on previously computed estimates (the entire estimations from the last jump are updated at each step), we present the results for the final estimation of the trajectory, the smoothed one, and also for the estimations made gradually as time advances, that are output by the algorithm in real time. The "real time" smoother is thus an implementation of algorithm 2, whereas the "smoothed" smoother is the one that returns the estimations at the end of the experiment. Thus, it may be viewed as nearly optimal. For the comparison to be fair, we need to compare the real-time smoother with the IEKF.

Unsurprisingly, the accuracy of the smoothing algorithm for the estimations considered after the entire smoothing process is much better than for the estimations being output in real time. Indeed, the estimations when a jump is detected and finally rejected or accepted can lead to (slightly) degraded accuracy for the real-time smoother. Moreover, the smoothing is more and more precise with the size of the window, so the first estimations output are less precise than the last ones.

The performances of both algorithms for this sort of trajectory are very similar. Indeed, we have seen in chapters 3 and 4 that the IEKF is accurate for trajectories with jumps in the angular velocity and the norm of the velocity (in 3D, jumps on the curvature and the torsion). It is thus not surprising that it can perform as well (even better) than the smoothing algorithm giving real time estimations. The smoothed estimations are nonetheless better than the IEKF estimations.

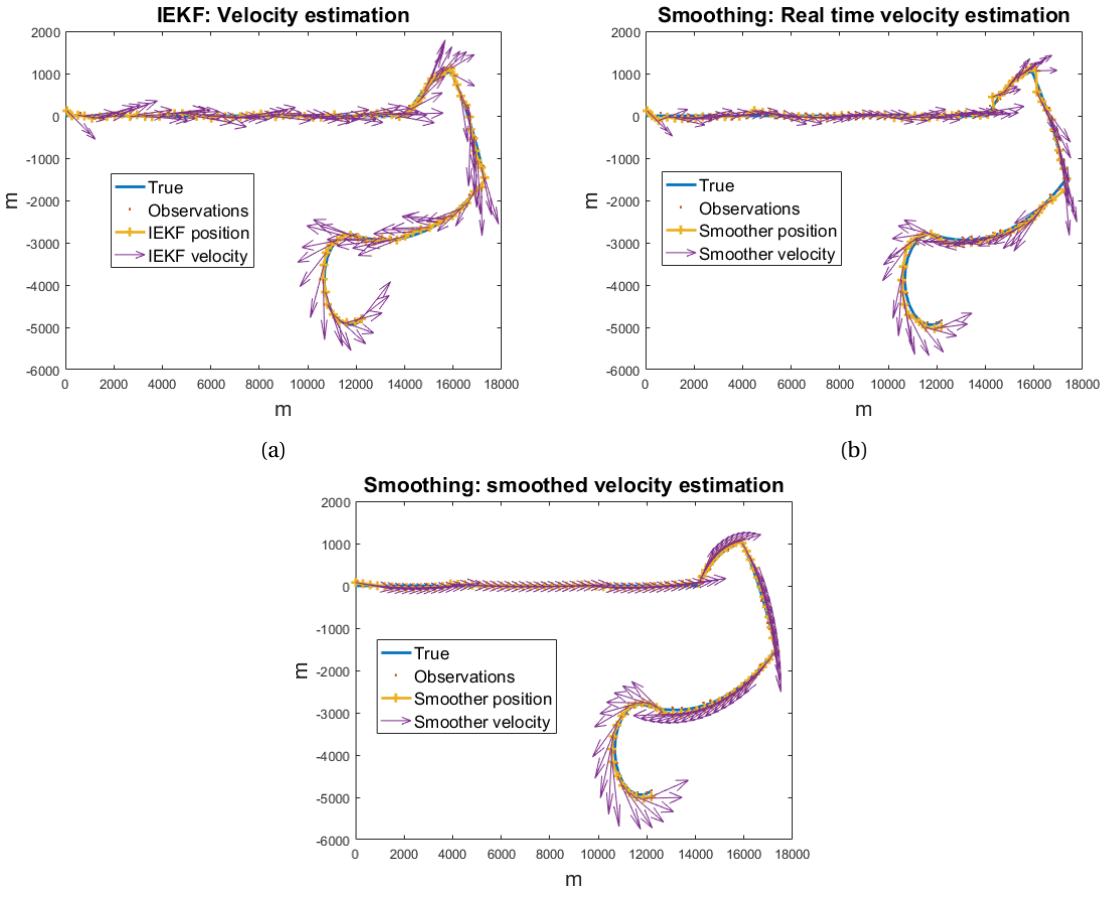
Trajectories with very abrupt jumps

To compare more deeply the two algorithms, we generate trajectories as in the previous paragraph, *i.e.*, we use the 2D Frenet-Serret target model to which we add random jumps of the norm of the velocity and of the angular velocity. In this section, we also add jumps on the orientation of the target, *i.e.* on the heading of the velocity vector directly to generate the trajectories. We use a set of 100 randomly generated trajectories, and we present the RMSE for the position, the orientation and the norm of the velocity in table 5.2.

Parameter	Smoothing: smoothed	Smoothing: real time	IEKF
Position (m)	46	70	142
Norm of velocity (m/s)	16	33	46
Orientation (rad)	0.16	0.31	0.34

Table 5.2 – RMSE for the smoothing algorithm and the IEKF on trajectories generated with 100 Monte-Carlo simulations of random jumps and random changes of heading

A visual result is presented on figure 5.6, in particular to show the velocity vector for both algorithms. The errors computed for this particular trajectory are gathered in table 5.3.



(a) (b) (c)

Figure 5.6 – Velocity vector along the trajectory for the IEKF fig. 5.6a, for the smoothing algorithm, with the results output in real time fig. 5.6b, and for the smoothing algorithm with the final result fig. 5.6c. This is one realisation of the Monte-Carlo experiments performed with random jumps. The results are more precise for the smoothing algorithm, even when considering the real time output.

The process noise added for the IEKF induces a lesser precision for the velocity vector on constant phases for the IEKF than for the smoothing algorithm. Moreover, the filter reacts less rapidly to the jumps.

5.8.2 Comparison with an IMM

The same sort of comparison is held with a filtering algorithm handling the jumps, namely the IMM. The IMM should be performing well on this type of trajectories. However, a model with high process noise is needed to accommodate the jumps. Indeed, at the moment of the jumps, none of the models is correct, so one of them has to be tuned with high process noise, so that the filter

Parameter	Smoothing: smoothed	Smoothing: real time	IEKF
Position (m)	33	55	48
Norm of velocity (m/s)	8	24	29
Orientation (rad)	0.10	0.31	0.34

Table 5.3 – RMSE for the smoothing algorithm and the IEKF for the trajectory of figure 5.6

Parameter	Smoothing: smoothed	Smoothing: real time	IMM
Position (m)	40	60	43
Norm of velocity (m/s)	15	32	75
Orientation (rad)	0.12	0.24	0.26

Table 5.4 – RMSE for the smoothing algorithm and the IMM on trajectories simulated with 100 Monte-Carlo simulations of random jumps and random changes of heading

converges. This model is usually a constant acceleration model.

We thus compare here an IMM with three models (constant velocity, constant turn and constant acceleration) with our smoothing algorithm. The transition probability matrix for the IMM is $\begin{pmatrix} 0.4 & 0.3 & 0.3 \\ 0.3 & 0.4 & 0.3 \\ 0.3 & 0.3 & 0.4 \end{pmatrix}$.

We again provide RMSE values computed for 100 different trajectories with random jumps as in the previous section, with the same amplitude and frequency, they are gathered in table 5.4. One example is given on figure 5.7, with the RMSE computed for this trajectory gathered in table 5.5. The smoothing algorithm here again is more accurate than the IMM. However, for the position precision, we observe that the estimations provided in real time of the smoothing algorithm are a little less accurate than the ones of the IMM. Indeed, the IMM has a quite large process noise, so the position sticks to the observations, whereas the smoothing position estimation can deviate a little.

To make this comparison complete, we also show one particular example where the trajectory is generated as follows: first, a constant velocity linear motion, then an abrupt 90° turn that occurs between two observations, followed by another straight line motion, and a slow turn, containing several observations. Results are on figure 5.8. There again, we see that the smoothing is better suited to this type of trajectories than an IMM, especially after a jump in the trajectory. The error values for this trajectory are collected in table 5.6. This sort of trajectories is used by the French department of defence (DGA) to challenge tracking algorithms.

For all the comparisons that were made, the IEKF and the IMM were tuned so that the convergence after the jumps is fairly fast. This implies high process noise, so it introduces less accuracy during constant motions. This is the role of the standard tuning process necessary for filtering algorithms, as already discussed in section 4.3. Moreover, the transition probabilities of the IMM have also to be tuned.

Parameter	Smoosthing: smoothed	Smoothing: real time	IMM
Position (m)	44	69	45
Norm of velocity (m/s)	19	52	92
Orientation (rad)	0.15	0.37	0.44

Table 5.5 – RMSE for the smoothing algorithm and the IMM for the trajectory of figure 5.7

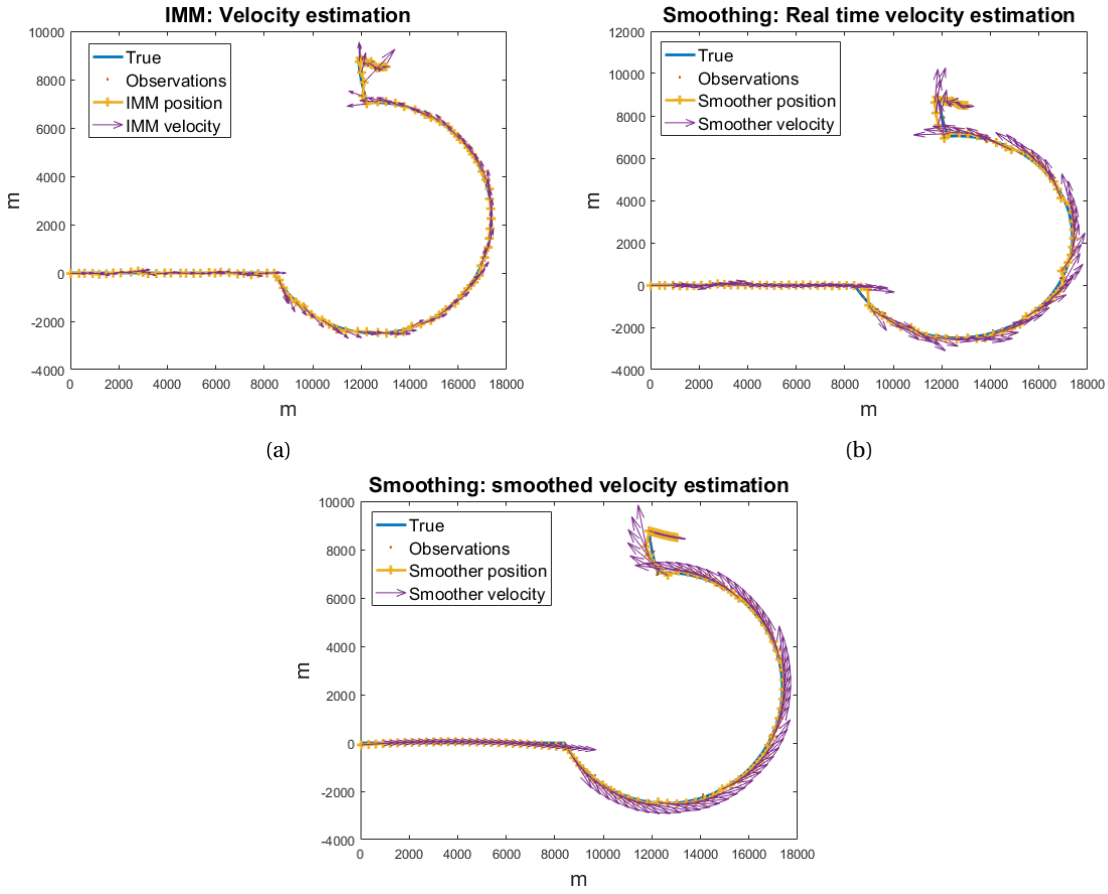
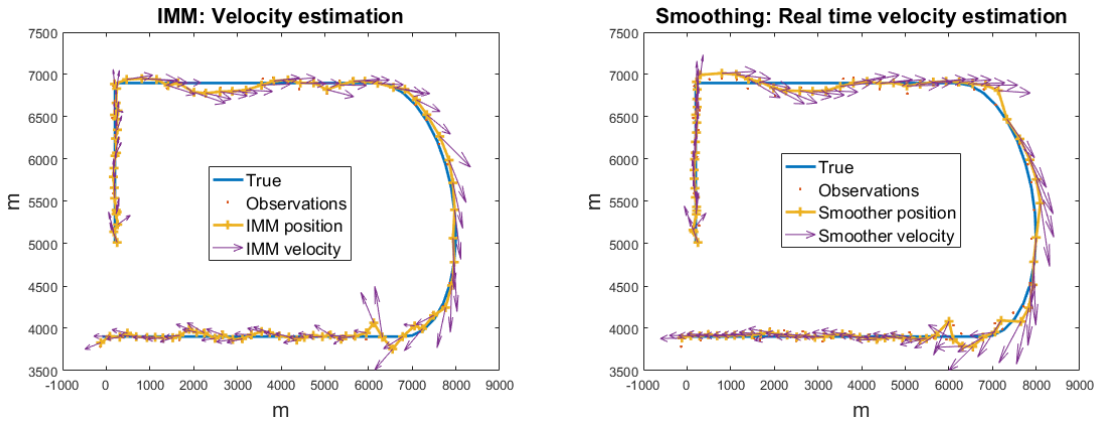


Figure 5.7 – Velocity vector along the trajectory for the IMM fig. 5.7a, for the smoothing algorithm, with the results output in real time fig. 5.7b, and for the smoothing algorithm with the final result fig. 5.7c. This one realisation of the Monte-Carlo experiments. As with an IEKF, the results are more precise for the smoothing algorithm, even when considering the real time output, and not the final smoothed estimation.

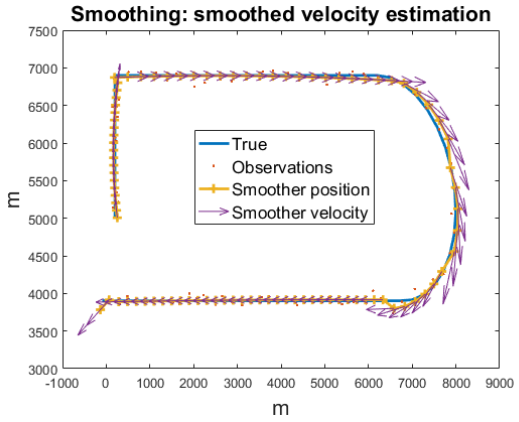
Parameter	**Smoothing: smoothed**	**Smoothing: real time**	**IMM**
Position (m)	33	49	46
Norm of velocity (m/s)	13	35	81
Orientation (rad)	0.18	0.45	0.67

Table 5.6 – RMSE for the smoothing algorithm and the IMM for the trajectory of figure 5.8



(a)

(b)



(c)

Figure 5.8 – Velocity vector along the trajectory for the IMM fig. 5.8a, for the smoothing algorithm, with the results output in real time fig. 5.8b, and for the smoothing algorithm with the final result fig. 5.8c. As with an IEKF, the results are more precise for the smoothing algorithm, even when considering the real time output, and not the final smoothed estimation.

5.9 Discussion

5.9.1 Comparison with other filters

During model changes, the sampling of the observations (that might be over one second, for rotating antenna radars) usually engender erroneous estimations. Indeed, during these transition phases, there exist an infinite number of likely solutions, when the exact time of the transition is not known. During these transition phases, recursive Kalman filters estimate a unique solution, which has a high probability to be erroneous. To overcome this problem, and avoid the divergence of the Kalman filters during transitions, IMM filters are commonly used. A model that accounts for transition is often used, with a high process noise, usually a constant acceleration model. This ensures there is at least one of the filters of the IMM that does not diverge. Engineers sometimes compare this filter with a "garbage collector" model, which is a security when none of the other models is giving proper estimations. The problem is that this model adds uncertainty on the estimate and usually delays the convergence of the filter after a transition during phases with constant kinematic model.

The jumping smoother with the Frenet-Serret model allows to optimise these transition phases. The conception of the algorithm is designed to be optimal during constant phases, and to handle the jumps optimally as well. The smoothing algorithm with jumps should thus work very well in the presence of very abrupt jumps. This has been confirmed by the comparisons between smoothing and filtering that we have performed, and presented in the preceding section.

Tuning

Another advantage of the smoothing algorithm proposed in this chapter is that there are very few parameters to tune. There are three scalar parameters to tune: the number of observations we wait after a possible jump to accept the jump or to reject it, called k in algorithm 2, the size of the sliding window $\bar{k}$, and the probability p of a jump, that intervene in equation (5.13), and which is used to compare the smoother that has just jumped with the older one. This parameter can be tuned quite low, even if it does not correspond exactly to the true probability used to generate the trajectory. Moreover, our experience is that the smoother is robust to small variations in the parameters. The IEKF and the IMM had to be tuned according to the amplitude of the jumps. Indeed, if the process noise is too low, then the filters cannot accommodate the jumps, whereas if it is too high, the accuracy elsewhere is degraded.

5.10 Conclusion

In this chapter, a new estimation method has been applied to the tracking problem based on real-time smoothing. The standard smoothing algorithm has been extended to take into account the possible jumps in the trajectory. The model is considered to be piecewise deterministic, *i.e.* there is no process noise on the constant parts. This allows to detect jumps in the trajectory, and to be very accurate on constant parts. Some types of modern targets, such as missiles, can have trajectories with sudden change of heading between two observations. This estimation algorithm is meant for this type of targets.

The smoothing algorithm with jumps has been compared with filtering algorithms. The comparison with the IEKF as well as with the IMM shows its superiority in the presence of very abrupt jumps. Although the two latter filters can also cope with softer jumps, when confronted to a jump of heading between two observations, they are less accurate, since it does not correspond to their target models. However, they have similar performances for more usual trajectories.

This smoothing algorithm has for the moment only been derived for a 2D target model, in a vector space. Switching to the Lie group setting requires being able to perform the linearisation on the target model, which is quite difficult, since it is a mix of a Lie group element and a vector space element. However, the method is generalisable as long as the linearisation is found. This

would open the way to a generalisation to the 3D target model proposed in this thesis, the 3D Frenet-Serret model.

Part III

Update rate adaptation

Chapter 6

Update rate real-time optimisation

Sommaire

6.1 Résumé en français : Optimisation en temps réel de la fréquence des mesures

Les deux derniers chapitres sont relativement indépendants des premiers chapitres et adressent un problème différent, celui de l'optimisation des ressources radar consacrées à la poursuite active.

Les radars multifonctions doivent accomplir plusieurs tâches en parallèle. La poursuite de cibles est seulement l'une de ces tâches. Blackman et Van Keuk ont proposé une méthode afin d'optimiser le temps du radar consacré à la poursuite. L'idée est d'exprimer la charge du radar comme un ratio entre le nombre d'illuminations nécessaire pour trouver une cible au moment d'une mesure et la durée entre deux mesures consécutives. Le raisonnement est le suivant : plus la durée entre deux mesures est élevée, plus la charge radar devrait être faible, mais plus il est difficile de retrouver la cible au moment voulu, et il est parfois nécessaire de faire plusieurs tentatives pour la détecter à nouveau. Ce raisonnement est valide pour des radars à faisceau focalisé, et à partir du moment où on autorise plusieurs illuminations au moment d'une mesure. De plus, la méthode de résolution proposée fait l'hypothèse que la cinématique critique est uniquement angulaire (on suppose donc qu'il n'est pas possible de perdre la cible sur l'axe distance), ce qui conduit à prendre en compte uniquement la difficulté de retrouver la cible dans une zone angulaire donnée, quelle que soit sa distance. Cette hypothèse est réaliste, car les faisceaux radars sont généralement moins focalisés en distance qu'en écartement angulaire.

Blackman et Van Keuk ont résolu le problème d'optimisation dans le cas d'un modèle de cible linéaire, à savoir le modèle de Singer. En effet, dans ce cas, il est possible d'exprimer à la fois le nombre d'illuminations et la durée entre deux mesures directement comme des fonctions de la covariance prédite de la position. En effet, on peut exprimer la covariance prédite en fonction du temps (elle grandit avec le temps, ce qui traduit l'incertitude qui augmente lorsque l'on repousse le moment de la mesure), et on peut inverser la fonction pour obtenir le temps en fonction de la covariance. Blackman et Van Keuk ont ainsi identifié la valeur optimale de la covariance prédite en position angulaire de la cible afin de minimiser la charge radar. Il est ensuite possible de calculer en temps réel le temps de revisite optimal à chaque étape, en faisant des projections de la covariance prédite grâce aux équations de l'algorithme de filtrage et du modèle de cible.

6.2 Introduction

The idea of cognitive radar has been defined in [63], and further developed in [60]. The ability to adjust the illuminations in an intelligent manner is one of the characteristics that distinguishes a cognitive radar from an adaptive one. Update rate adaptation falls into this category. Indeed, the beam can be controlled to illuminate the most interesting regions of the space in an intelligent manner, so that radar time budget is saved and tracks are all maintained. This is possible thanks to the emergence of phased-array fixed antennas that allow illuminating any region of the space at any time, as illuminations are no longer imposed by the rotation of the antenna.

Phased-array multifunction radars are designed to perform several tasks in parallel, such as surveillance, Track While Scan (TWS), and active tracking (AT). There is thus a competition between these different tasks regarding the resources of the radar. The role of the resource manager is then to study and organize the different requests. And notably as the radar beam is needed for surveillance, TWS and AT, its update frequency for active tracking has to be carefully controlled, as well as for the TWS task. The beam scheduler is in charge of ordering all the tasks asked, via the resource manager. Optimisation to cover the entire space during surveillance has already been studied, see [28], [29]. In the case of AT, the illuminations requested have a label containing a duration and an update period, and the adaptive resource manager uses a scheduling algorithm to determine in which order these illuminations should be performed. Moreover, priority requirements may be accounted for. This priority can for example be the result of a degradation in the tracking performances detected by an update rate adaptation algorithm or a change detector. Indeed, the

radar must definitely not drop a track, especially when performing active tracking. The drop probability, or equivalently the detection probability, are determined in function of the performances of the filtering algorithm used to estimate the target's state, and other fixed characteristics of the radar, such as the Signal to Noise Ratio (SNR) for a beam pointing in the direction of the target.

The work of Blackman and Van Keuk in [108] provide a method to perform update rate adaptation. The authors assume a Singer tracking model with a Kalman filter to perform estimation. They derive a criterion to optimise the radar load. This criterion gives the optimal time period to refresh the observations. In this chapter, we propose to analyse pedagogically the method to derive this criterion, and explain how it is applicable in practice. We also point out the approximations that were made to derive the criterion.

We thus derive the fixed update rate criterion of Blackman and Van Keuk, using the Singer model, and Cartesian observations. We have described the Singer model in section 2.3.2. This method is meant to be applied to pencil beams radars, where it is possible to have several illuminations to detect a target at a given moment.

6.3 Fixed update optimisation criterion

The problem of estimating the state of a Singer model using a Kalman filter is linear, Gaussian and time-invariant, the Kalman covariance matrix converges to a fixed value, called P_∞ (see [70]). It turns out that the latter convergence property allows deriving a fixed criterion for the optimisation of the update rate.

The rationale of Blackman and Van Keuk in [108] is to find a balance between the time during two observations and the number of illuminations necessary to find a target when ordering the observation. In target tracking, the limitations in term of radar load, and the number of illuminations are due to the angular motions of the target, indeed, the distance is usually not a problem, especially with a pencil beam. So the more time we wait between two observation, the more the incertitude over the angular position of the target is large. To express this, we need to use another coordinate system, close to the range, azimuth and elevation coordinate system, called the (r, u, v) coordinates. r still represents the distance from the radar to the target, and (u, v) are angles taken from the radar, and defined as follows:

$$u = \cos el \sin az$$

$$v = \sin el$$

where el and az are the elevation and azimuth angles respectively.

6.3.1 General formulation of the optimisation problem

In [108], Blackman and Van Keuk define the radar load L_c as

$$L_c = \frac{E(n)}{E(T)} \tag{6.1}$$

where $E(n)$ is the average number of illuminations to find a target and T is the duration between two measurements. The problem is feasible, as the more time T we wait between two measurements, the less energy is spent, but the more difficult it is to find the target again, and the number of illuminations has to be increased to recover sight of the target. It is possible to express $E(n)$ and $E(T)$ as a function of a common variable, V_0, explained thereafter. Blackman and Van Keuk have introduced the quantity V_0 to evaluate the angular precision of the relative prediction with respect to the beamwidth B such that $G(k) = V_0 B$, where $G(k)$ is the principle axis of the 1σ ellipse defined by the covariance matrix of the prediction error at time T_k given by the Kalman filter, and associated to coordinates in the angular (u, v) space. One can assume that the probability distributions of the measurement and the process noise are isotropic in the angular coordinate system (u, v), so the ellipse is in fact a circle. It is then possible to choose arbitrarily any coordinate axis as the

principal axis, say , u. $G(k)$ can thus be expressed from the filter's covariance $P(k)$ as follows. Recall that, $\tilde{P}(k) = HP(k)H^T$ is the position covariance matrix according to the filter, with H defined in (6.3). We have $G(k) = \sqrt{\tilde{H}_{uv}(k)\tilde{P}(k)\tilde{H}_{uv}^T}$, with $\tilde{H}_{uv}$ the Jacobian of the function that relates the Cartesian to the spherical (r,u,v) coordinates, *i.e.*,

$$\tilde{H}_{uv}(k) = \begin{pmatrix} \frac{x_1}{r} & \frac{x_2}{r} & \frac{x_3}{r} \\ \frac{x_2^2+x_3^2}{r^3} & -\frac{x_1x_2}{r^3} & -\frac{x_1x_3}{r^3} \\ -\frac{x_1x_3}{r^3} & -\frac{x_2x_3}{r^3} & \frac{x_1^2+x_2^2}{r^3} \end{pmatrix} \tag{6.2}$$

$$H = \begin{pmatrix} H_1 & 0_{1,3} & 0_{1,3} \\ 0_{1,3} & H_1 & 0_{1,3} \\ 0_{1,3} & 0_{1,3} & H_1 \end{pmatrix}, H_1 = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix} \tag{6.3}$$

6.3.2 Resolution

Estimation of $E(n)$

[108] proposes a way to estimate $E(n)$. It may depend on the radar and on the repointing strategy. The strategy used in [108] is to search the target at the maximum of the probability density function (pdf) of its predicted position in (u,v). If the target is not found at the first illumination, then a slightly modified pdf is computed, and we try again until the target is found, or a maximum number of illuminations is reached.

The search strategy is based on the pdf of the predicted position in (u,v) coordinates. We call this pdf $pdf(1)$. The radar beam points towards the maximum of this pdf in order to maximise the chance to detect the target. If we denote by (u,v) the angles of the beam of the radar and by $(\hat{u},\hat{v})$ the predicted position of the target, then following [108], the Signal-to-Noise Ratio (SNR) can be computed as in (6.4) to get the detection probability $P_D(u,v)$ in (6.5), with P_F the false alarm probability.

$$SNR(u,v) = SN_0 \exp\left(-2\frac{(u-\hat{u})^2+(v-\hat{v})^2}{B^2}\right) \tag{6.4}$$

$$P_D(u,v) = P_F^{\frac{1}{1+SNR(u,v)}} \tag{6.5}$$

If the target is not detected on the first illumination, then the pdf is slightly changed into

$$pdf(2)(u,v) := C(1-P_D(u,v))pdf(1)(u,v) \tag{6.6}$$

The constant is used to normalize the pdf, and of course P_D depends on the position because the SNR does. This is iterated until the target is found or a maximum number of authorised illuminations is reached.

Experiments allow to compute $E(n)$ for a given radar, and for a Swerling target model 0, that allows to define P_D by (6.5). In [108], the authors found that the expectation of the number of illuminations to find a target is given by (6.7), with $\tilde{\alpha} \approx 1 + 14(|\ln P_F|/SN_0)^{1/2}$, and $P_D = P_F^{1/(1+SN_0)}$ where P_F is the probability of false alarm, P_D the detection probability, and SN_0 the Signal to Noise Ratio at the centre of the beam.

$$E(n) = \frac{1}{P_D}(1+\tilde{\alpha}V_0^2)^{1/2} \tag{6.7}$$

Estimation of $E(t)$

It depends on the chosen target model. Blackman and Van Keuk use a Singer target model. For the Singer linear model with Cartesian observations, the covariance matrix of the full state converges towards a fixed matrix P_∞ so the variance of the predicted angular dispersion of the target's

position also converges towards a value $\tilde{P}_\infty$, which depends on T, the duration between two measurements (the larger T, the larger the eigenvalues of $\tilde{P}_\infty$). It means that at the measurement time, the position of the target can be modelled by a Gaussian random variable of variance $\tilde{P}_\infty$. The formula giving $\tilde{P}_\infty(T)$ can be inverted to get $T(\tilde{P}_\infty)$, which is also a non-trivial property due to the use of the Singer model. The inversion of the formula $\tilde{P}_\infty(T)$ will not be possible with other models indeed.

Using the two latter components, we can evaluate L_c of equation (6.1) as a function of $\tilde{P}_\infty$. As will be shown in the sequel, this is equivalent to express L_c as a function of V_0.

The time between two revisits, which depends on the average number of illuminations necessary to find the targets, also converges to a stable value when the error's variance filter becomes stationary. So let us focus on the stationary asymptotic phase. Define

$$P_\infty = \lim_{k\to\infty,\alpha T\to 0} P(k)$$

Using the asymptotic Riccati algebraic equation, the stationary position covariance $\tilde{P}_\infty$ is given by (6.8), where α and Σ are the parameters of the Singer model, and $(.)_{pos}$ extracts the coefficients corresponding to the position variable in the covariance matrix, *i.e.*, $(C)_{pos} := HCH^T$, and diag(a) is the matrix with a on its diagonal.

$$\tilde{P}_\infty = \left(F(P - PH^T(HPH^T + N)^{-1}HP)F^T\right)_{pos} + \text{diag}\left(2\alpha\Sigma^2\frac{T^5}{20}\right) \tag{6.8}$$

$N = \sigma^2 I_3$ is the measurement covariance matrix. Moreover, as we are only interested in the angular variability, we select the second diagonal element of $\tilde{H}_{uv}(k)\tilde{P}(k)\tilde{H}_{uv}(k)^T$ which is the variance of the asymptotic error in the u variable, hence:

$$G = \sqrt{[\tilde{H}_{uv}\tilde{P}_\infty\tilde{H}_{uv}^T]_{2,2}} = V_0 B \tag{6.9}$$

The idea is that $\tilde{P}_\infty$ is a good approximation for the position uncertainty of the radar. The uncertainty in (r,u,v) coordinates is $\tilde{H}_{uv}\tilde{P}_\infty\tilde{H}_{uv}^T$. since we are only interested in the angular uncertainty, we suppose that r is nearly constant (the same assumption is made in [108] where all formulas depend in fact of r).

Let also $\tilde{C} = \left(F(P - PH^T(HPH^T + N)^{-1}HP)F^T\right)_{pos}$. One can isolate T from equation (6.8), which gives (6.10).

$$T = (10)^{1/5}\left(\frac{\sqrt{1/\alpha}}{\Sigma}\right)^{2/5}(\tilde{P}_\infty - \tilde{C})^{1/5} \tag{6.10}$$

Moreover, from (6.9), and given that the range r is supposed to be approximately constant, we have

$$\tilde{P}_\infty \approx r^2 V_0^2 B^2$$

Finally, (6.10) becomes

$$T = (10)^{1/5}\left(\frac{\sigma\sqrt{1/\alpha}}{\Sigma/r}\right)^{2/5}\left(\frac{V_0^2B^2}{\sigma^2} - \frac{\tilde{C}}{\sigma^2 r^2}\right)^{1/5} \tag{6.11}$$

From now on, let $v_0 = \frac{V_0 B}{\sigma}$. Blackman and Van Keuk argue that over a large set of parameters, the following approximation is valid

$$T \approx 0.4 P_D\left(\frac{\sigma\sqrt{1/\alpha}}{\Sigma/r}\right)^{0.4}\frac{v_0^2}{1+\frac{v_0^2}{2}} \tag{6.12}$$

where P_D is the detection probability, r is the distance between the target and the radar (in m), ξ is the slope factor associated with the mono-pulse measurement process ($\xi = 1.37$). Moreover the measurement noise standard deviation in angular coordinates σ can be expressed as follows:

$$\sigma = \frac{B}{\xi\sqrt{2(SNR+1)}},\ SNR \approx \frac{SN_0 - \ln P_F}{1+2V_0^2}$$

6.3.3 Practical use of the criterion

Finally using (6.7) and (6.12), the expression of the radar load L_c is given by (6.13), with B the half-beamwidth.

$$L_c \approx \left(\frac{\Sigma}{Br\sqrt{1/\alpha}}\right)^{0.4} \frac{1}{P_D} f(V_0, SN_0, P_F) \tag{6.13}$$

The paper [108] states that for a quite large set of parameters ($SN_0 \in [10,160]$, $P_F \in [10^{-8}, 10^{-4}]$), the optimal V_0 is 0.3, which gives the optimal revisit period $T = 4.6s$ for $r = 60km$, $\Sigma = 10m/s^2$, $\alpha = 1/60s$, $P_F = 10^{-5}$, $SN_0 \approx 30(15dB)$, and $B = 1°$.

The Blackman Van Keuk-criterion thus gives the optimal track sharpness in the (u,v) space. Whatever the value of the beamwidth, the optimal track sharpness is one sixth of the beamwidth. The corresponding revisit period is also fixed ($T = 4.6s$). However, the criterion giving the optimal track sharpness can be used to adapt the revisit time. Indeed, the revisit time can then be computed on-line thanks to this criterion and to the predicted covariance matrix. The idea is to compute the covariance matrix at a thinner time step, and to order a new measurement just before the covariance exceeds the optimal track sharpness. This limits the use of approximations, especially those done to have formula (6.12).

This is the optimised revisit period for a Singer linear model, and the one sixth of the beamwidth sharpness provides the practitioner with a useful general rule for design purposes, but it should be refined when one wants to use alternative models and filters. In a non-linear context; the simplifications of the convergent covariance matrix and the inversion of the formula giving $\tilde{P}_\infty$ will no longer be possible, and we have to use numerical integration to derive the optimal revisit time at each time step. The advantage of this method will be to have a totally adaptive revisit time period, that will vary at each time step, and evolve with the measurements, instead of using a fixed criterion. Moreover, it is much more versatile, as it suits any model and filtering algorithm, contrary to the previous criterion.

6.4 Discussion

The method developed by Blackman and Van Keuk gives a fixed criterion to perform update rate adaptation. It can be implemented very easily. However it suffers from several disadvantages:

- The Singer model is used, and the computations to obtain the optimal track sharpness totally depend on the use and properties of this target model. So the method is not really generalisable to any target model, or at the price of even more approximations. For the moment, this criterion is applied to any target model directly, without considering the implications of such approximations, which can seem awkward.
- As we have pointed out in the development of the computations for $E(t)$, there have been some approximations made even with the Singer model. Indeed, the covariance is expressed in (u,v) coordinates, which depend non-linearly on the Cartesian position that appears in the Singer model. Because of this, to derive $E(t)$, we have made the approximation that the range is almost constant during all the trajectory, which is not always a reasonable assumption.
- Finally, the values for optimising L_c of equation (6.13) have been made by performing many experiments, but with a given radar and setting. If the maximum number of illuminations is changed, these tests have to be performed again.

6.5 Conclusion

Blackman and Van Keuk have introduced a strategy to optimise the update rate for a given target in active track. They propose to find a balance between the time between two observations and

the difficulty to find the target again, modelled by the number of illuminations required. However, their method is based on the use of the linear Singer target model, and at the cost of several approximations.

In this work, we need an update rate adaptation algorithm which is not dependant on the target model. So we need to derive another method to perform update rate adaptation, this is the subject of chapter 7.

Chapter 7

Adaptive update rate

Sommaire

7.1 Résumé en français : Cadence adaptative

Dans cette thèse, les travaux de Blackman et Van Keuk ont été repris, puis généralisés à tout type de modèle de cible et d'algorithme de filtrage. La seule condition requise sur l'algorithme d'estimation est qu'il fournisse la propagation de la matrice de covariance à l'étape de prédiction, ce qui est le cas pour tous les algorithmes de filtrage présentés dans ce document. La résolution du problème d'optimisation dans le cas de modèles non-linéaires ne peut pas se faire de façon directe. La méthode proposée dans ce chapitre utilise la probabilité de détection comme critère de renouvellement de la mesure. En effet, la probabilité de détection dépend de la covariance prédite, et du nombre d'illuminations autorisées pour retrouver la cible. On veut donc trouver la durée maximale entre deux mesures permettant d'assurer une probabilité de détection donnée minimale. Pour cela, on propage la covariance petit à petit dans le temps, on calcule la probabilité de détection associée pour une, deux, trois ... illuminations autorisées, on s'arrête lorsque la probabilité de détection devient en deçà d'un seuil fixé à l'avance quel que soit le nombre d'illuminations.

Le critère de renouvellement de mesure développé dans ce chapitre et le critère d'origine de Blackman et Van Keuk ont été comparé sur deux algorithmes : un algorithme de filtrage avec modèle linéaire de Singer, et un algorithme de filtrage avec modèle non-linéaire (celui de Frenet-Serret construit dans le chapitre 2). Les résultats obtenus montrent que les deux méthodes pour calculer la cadence de mesure donnent des résultats similaires sur le modèle linéaire (c'est-à-dire que la charge radar est équivalente). En revanche, sur le modèle non-linéaire, la charge radar obtenue avec le nouveau critère de cadence est réduite par rapport à la charge radar obtenue avec l'ancien critère du chapitre 6, ce qui montre que le nouveau critère est plus adapté aux modèles non-linéaires.

7.2 Introduction

In this chapter, we generalise the work of [108], presented in chapter 6, to optimise the update rate of the radar. In [108]; the update rate adaptation method is designed for one particular type of target model, namely the Singer model [103], or refer to section 2.3.2, combined with a Kalman tracking algorithm. Since, it has been used as an efficient (yet suboptimal) rule, even when using a different (possibly non-linear) target model. Owing to the progresses of computers over the past 25 years, we show it is now possible to extend the method to any target model and estimation algorithm. Moreover, one must bear in mind that the goal of update rate optimisation is rather to save radar budget, than to enhance the performances of the state estimation. Nevertheless, the tracking estimation precision has to stay within an acceptable range, as also ensured with the algorithm proposed in this chapter.

A Kalman-type filtering algorithm provides the estimated distribution of the state, assuming that this distribution is Gaussian. It thus provides the mean of the distribution, which is usually called the estimated state, and the covariance of the distribution. Although the method of Blackman and Van Keuk relies on a fixed criterion, because of the particular form of the state evolution model chosen, it is possible to compute the detection probability for one target with respect to the covariance output by any filter in real time. The update rate adaptation introduced in the present chapter, that allows for the criterion to adapt over time, builds upon the latter idea.

Target tracking typically relies on an accurate motion model and a robust filtering algorithm. In this chapter, the general methodology will be applied to the 3D target model in intrinsic coordinates based on the Frenet-Serret frame, described in section 2.6.2, with the use of the IEKF algorithm of section 3.6. However, it may in principle be applied to any model combined with any other filtering algorithm that provides a covariance to the user.

7.2.1 Links with prior literature

To optimise the update rate of observations during Active Tracking, one has to define an optimisation problem. In this chapter, we use the same criterion as in chapter 6, where the load of the radar is modelled as a ratio between the number of illuminations necessary to find a target (at each measurement instant) and the time elapsed between two measurements. The rationale is that scarce measurements lead to low radar load, but they also lead to an increased number of illuminations at each measurement epoch to find the target. As a result, finding the optimal update rate (*i.e.*, time between two consecutive measurements) is a feasible optimisation problem. Other optimisation schemes may be designed on the radar's performances and on the user's objectives, but in this work we focus on the optimisation problem as posed by Blackman and Van Keuk.

Other update rate adaptation algorithms have been developed in the literature. [39] uses an alpha-beta filter and an adaptation scheme so that the residual error of the filter remains constant. Another type of algorithm consists in using an IMM, as in [110], or later [13], where the idea is also to maintain a given level of filtering precision, thanks to the covariances computed by the IMM. the IMM is also used in [99] to control the size of the validation region, to address the association problem. Our problem is slightly different, since we are not only interested in maintaining a given precision, but also in reducing the overall radar time spent for each measurement. In [100], the IMM is used to compute an adaptive update rate related to the manoeuvres of the target, and based on the Blackman and Van Keuk approach. However, contrary to the latter, we propose a versatile algorithm that applies to virtually all kinds of filtering algorithms, and we rely on the filter to return necessary mathematical quantities.

Another method to adapt the update rate is to use a manoeuvre detection algorithm. Change detectors are a very wide class of algorithms, that are described in the book [12], or more precisely in [76] for the Generalized Likelihood Ratio (GLR) algorithm, or [2] for the CUSUM algorithm. Another change detection method based on the computation of appropriate distances between the outputs of Kalman filters can also be designed, as stated in [92]. All these detectors perform quite well when applied with an IMM algorithm. We will not discuss these methods in this chapter, since we opted for a wholly different route. Increasing the measurement rate when a change is detected is quite basic, and change detectors are rarely used as update adaptation means but rather as urgent pointing commands.

7.2.2 Organisation and contributions of the chapter

The update rate adaptation of Blackman and Van Keuk is generalised in section 7.3, and results in a versatile adaptive algorithm. More precisely our algorithm informs the radar resource manager what the (maximum) measure update should be in real time so that the detection probability of a given target stays above a given threshold. Then, in section 7.4 this novel algorithm is applied to the non-linear 3D Frenet-Serret model of chapter 2. The state estimation task is performed by a non-linear filtering algorithm, the IEKF of chapter 3. Finally, in section 7.5, the proposed algorithm is compared to the Blackman and Van Keuk criterion, using both the Singer model and the non-linear Frenet-Serret model.

7.3 Update rate adaptation with a non-linear model

The Singer model is a specific target model, that dates back to the seventies, and is not pervasively used nowadays. The aim of this section is to propose a generalisation of the previous method to any target model, given that the estimation algorithm can provide a predicted covariance matrix whenever it is required. The method of Blackman and Van Keuk to establish a fixed criterion is indeed too simplistic to be applied straightforwardly to any model. However, their initial idea to compute the optimal load with the covariance matrix of the filter can be used to derive another algorithm that can be applied to virtually any filtering algorithm. The computational power to

compute a criterion on-line and to perform the necessary numerical integrations might have been too demanding in the nineties, but can be considered as unproblematic with modern computers.

7.3.1 Method: an adaptive criterion for update rate adaptation

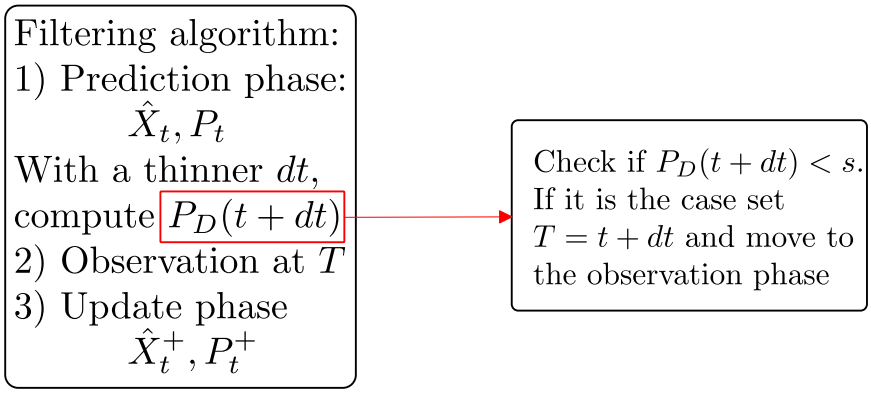


Figure 7.1 – Method used for the adaptive update rate adaptation algorithm

Let us call $P_D(t)$ the detection probability to find a target at a time t. Let us assume that there are a minimum and maximum authorised update rates. The rationale is as follows. We compute the predicted covariance associated to the minimum inter-measurement time, and then increase it gradually until the detection probability drops below a threshold given by client specifications or set by an engineer for example. The method is illustrated in figure 7.1. More precisely, we can compute $P_D(t)$ at each time step, and even more often than it is needed. Let us take a smaller time step than the minimum sampling rate. One can integrate the Riccati equation giving the predicted covariance at this short time step and compute the corresponding detection probability easily. A threshold can be used for the detection probability to request a new measurement when it is not satisfying. The new algorithm computes the maximum duration between two measurements, under the constraint that the detection probability, whose computation depends on the number of illuminations, stays high enough. In that sense, it mimics the criterion of Blackman and Van Keuk that minimises the radar load L_c.

Let us call dT the maximal revisit period between two measurements. We perform covariance prediction every $t + k.dt < d$T, with $k \in \mathbb{N}$, and dt the duration between two computations. The covariance prediction gives the detection probability. The algorithm to compute the time of the next target revisit is explained in Algorithm 3, where s is the acceptable detection probability threshold. Suppose we are at time t and we want to refresh the measurement to ensure a detection probability $P_D > s$. The next revisit time is called T.

We use the same optimisation criterion as in [108], so we allow several illuminations to find a target. We use the same search strategy as for the fixed criterion derivation. This includes expressing the covariance in (u, v) coordinates, the transformation matrix is given by (6.2) in section 6.3, with $r = \sqrt{x_1^2 + x_2^2 + x_3^2}$.

We also assume that we have an upper bound to the number of illuminations necessary to find a target if the first ones give no detection. This superior bound is called N_{max}. The detection probability also depends on the target search strategy.

7.3.2 Underlying search strategy

As detailed in section 6.3, the search strategy is based on the pdf of the predicted position in (u, v) coordinates. We use the same search strategy and notations as in section 6.3.

Now, we have to compute explicitly the constant C of equation (6.6). We can compute the normalising constant C of the pdf online, and find its mean μ_2 and covariance σ_2 and thus find the SNR with (7.1) or (6.4), that permits to compute P_D, with (6.5) again. This operation can be

Algorithm 3 Computation of the next revisit time T

Input: $\hat{X}_t, P_t$

while $P_D > s$ and $k < k_{max}$ **do**
  $k := k+1$
  Compute $\hat{X}_{t+k.dt}$ and $P_{t+k.dt}$ with the propagation equations of the filter applied to $\hat{X}_t, P_t$
  Compute $\tilde{H}_{uv}$, the measurement matrix in the (u,v) space, see (6.2)
  Compute $\sigma_2 = \tilde{H}_{uv} P_{t+k.dt}(\text{pos}) \tilde{H}_{uv}^T$
  for $n_i = 1$ to N_{max} **do**
    Compute the SNR and the detection probability, according to the number of the illumination n_i: Compute $pdf(n_i)$, find its covariance σ_2 and compute SNR and P_D with (7.1) and (6.5)
  end for
  Compute the overall probability detection: $P_D = max(P_D(n_i))$
  Compute the next time revisit: $T = t + k.dt$
end while
Output: T

Algorithm 4 Algorithm to perform update rate adaptation for a generic model

while $T(h) \leq N$ **do**
  $h := h+1$
  Apply algorithm 3, which gives $T(h)$, with inputs $\hat{X}^+_{T(h-1)}, P^+_{T(h-1)}$
  Propagation phase : Compute $\hat{X}_{T(h)}$ and $P_{T(h)}$
  Request a measurement at time $T(h)$
  Compute the update $\hat{X}^+_{T(h)}$ and $P^+_{T(h)}$
end while

performed as long as the target is not found, and the maximum number of illuminations N_{max} is not reached either.

$$\text{SNR} \approx \frac{SN_0 - \ln(P_F)}{1 + 2(\sigma_2/B)^2} \tag{7.1}$$

Let N be the duration of the whole trajectory, T be the function relating the number of the update with the time of the update. The adaptation algorithm is summed up in Algorithm 4.

The main difference between this adaptive method and the regular Blackman and Van Keuk criterion is that in our method, the optimisation is performed at each time step, and the update rate is thus perfectly suited to the instantaneous performances of the underlying estimation algorithm. It is possible to link experimentally the detection probability threshold required for the adaptive criterion and the threshold on the covariance matrix in the fixed criterion derivation. This will be explained in greater detail in section 7.4.

As an application of the proposed method, we use the non-linear target model, based on the use of the 3D Frenet-Serret frame presented in chapter 2. The method presented in chapter 6 will be referred to as the *fixed criterion*, and the one presented in this chapter as the *adaptive criterion*.

7.4 Application: Non-linear target model

To apply the algorithm of the previous section to a non-linear target model, we use the 3D Frenet-Serret model described in section 2.6.2, and the estimation algorithm is the IEKF of section 3.6.

We can compute approximately the detection probability threshold induced by the use of the fixed criterion, thanks to (7.2). On a trajectory simulated with a Singer model, the results obtained should be very similar. Indeed, the Singer model and the linear Kalman filter match the hypothesis of section 6.3.

$$P_{D_0} = P_F^{\frac{1}{1+SN_0}} \tag{7.2}$$

When using the Frenet-Serret based model instead, we anticipate an improvement of the optimisation rate L_c when we move from the fixed criterion to the adaptive one. An advantage of the adaptive criterion is also to adapt the detection probability threshold to the requirements of the client. This detection probability could also serve as an indication of the performance of the estimation filter.

Finally, for the fixed as for the adaptive criteria, the update rate adaptation algorithm is very dependent on the quality of the filtering algorithm, since it is based on its results. Indeed, if the algorithm provides erroneous covariance predictions, then the update rate computations will also be erroneous. And once again, what we expect of this update rate adaptation is an improvement of the radar load L_c and not of the precision of the estimation. So the filtering algorithm has to be reliable to perform these update rate algorithms. This is why we begin by assessing the performances of the linear Kalman filter on a Singer model trajectory, and the IEKF on a Frenet-Serret model trajectory, with a fixed update rate.

We have performed several experiments to show the differences between the filtering algorithms and the update rate adaptation methods, they are described in the next section.

7.5 Experiments

In this section, the adaptation algorithm is tested for the two different study cases mentioned in the previous section. The first study case is the linear Singer model tracked with the linear Kalman filter, and the second study case is the non-linear Frenet-Serret model tracked with the IEKF. We show the results for the fixed criterion and adaptive algorithms for both cases, and compare the results obtained. The trajectories are built differently for the two models, as will be explained in the following sections.

To begin with, we show the tracking performances of the linear Kalman filter with the Singer model and of the IEKF with the Frenet-Serret model without any update rate adaptation. Then, the update rate will be adaptive and the behaviours of the fixed and adaptive methods along with the radar load will be compared.

7.5.1 Tracking results with a Linear Kalman filter and an IEKF

In this part, we only illustrate the relevance of the models and filters involved, without any concern about the update rate adaptation problem yet.

Tracking with the Singer model with jumps

To test the filters, we first build a trajectory using the Singer model: we include one manoeuvre that occurs at time $t = 200s$, in which the two first coordinates of the acceleration undergo a sudden jump from $a_1 = -0.8m/s^2$ and $a_2 = -0.9m/s^2$ to $a_1 = 2.2m/s^2$ and $a_2 = 2m/s^2$. The position is initially $\begin{pmatrix} x_1 & x_2 & x_3 \end{pmatrix} = \begin{pmatrix} 6000 & 6000 & 6000 \end{pmatrix} m$, and the velocity $\begin{pmatrix} v_1 & v_2 & v_3 \end{pmatrix} = \begin{pmatrix} 0.5 & 0.5 & 0.5 \end{pmatrix} m.s^{-1}$. The parameters for the Singer model are $\alpha = 1/60s^{-1}$ and $\Sigma = 1.0$. The acceleration on the third coordinate does not jump and is initially equal to $a_3 = -0.9m.s^{-2}$. The measurement noise is of variance $1000m$. This trajectory is presented in figure 7.2.

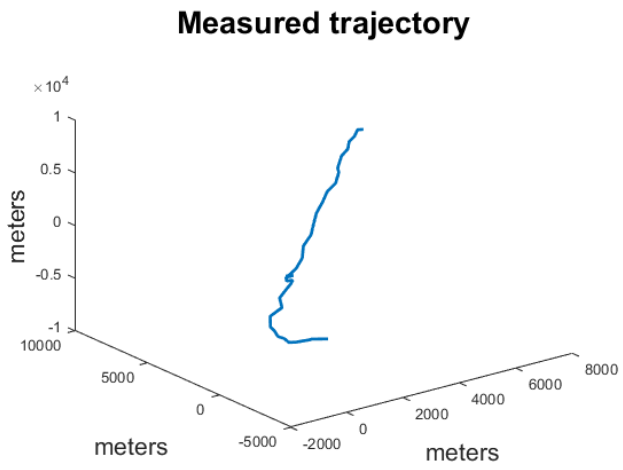


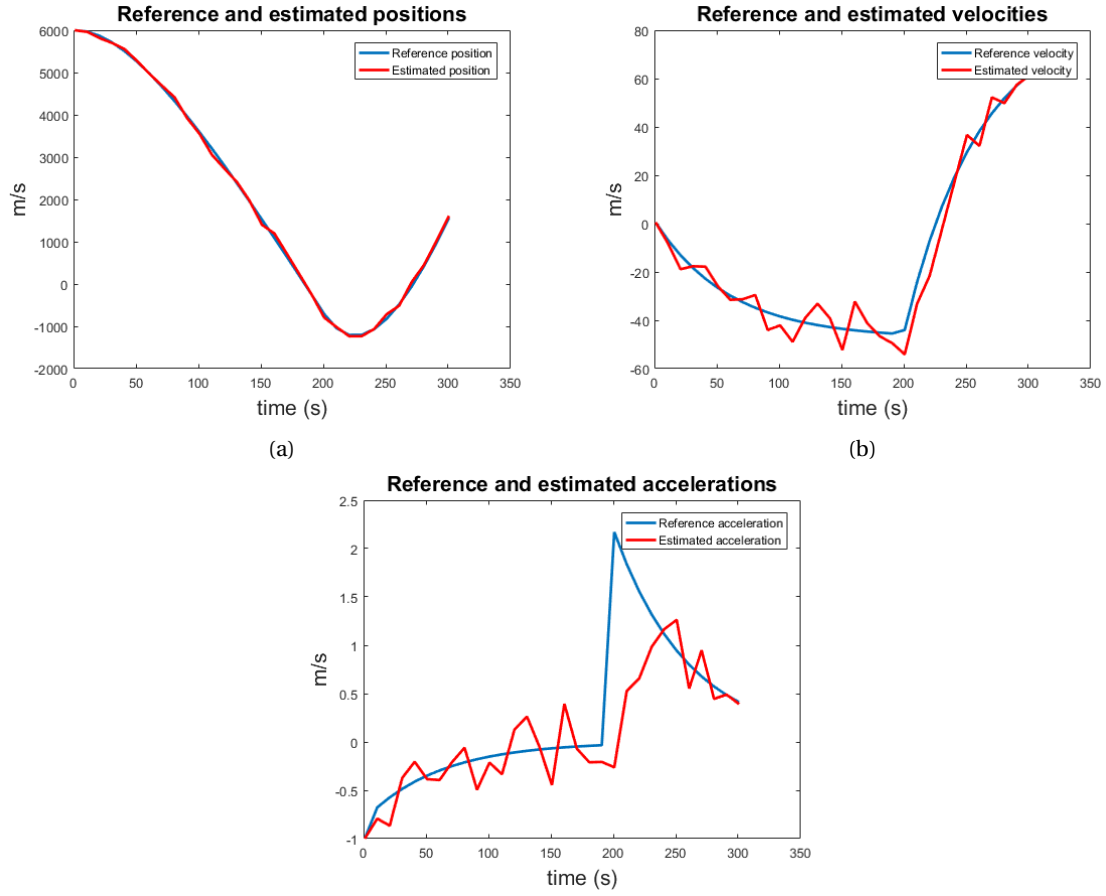
Figure 7.2 – Measured trajectory with the Singer model and one maneuver

The tracking results of the position, the velocity and the acceleration on the first coordinate of the Kalman filter for this trajectory are presented in figure 7.3. The update rate is set to be one measurement every ten seconds. The filter estimates quite well the position and the velocity. The estimation of the acceleration is more difficult, especially when the jump occurs. As for all Kalman-based filters, the tuning of the model noise covariance matrix plays a high role in finding a balance between the precision of the estimation and the ability to react efficiently when manoeuvres occur. For further information about noise tuning for Kalman filters, see [85], [72].

The lack of precision in the acceleration coordinate is not of crucial importance, as long as it does not result in a reduced precision for the velocity coordinate, which is a parameter that is relevant for some applications (among which intercepting missiles for instance).

Tracking with the Frenet-Serret model with jumps

We test our IEKF algorithm with the trajectory presented in figure 7.4. The trajectory is made of three Frenet-parameters constant parts. In the first part, the target has a constant velocity straight line motion. In the second part, the target has an helix trajectory, with constant velocity, curvature and torsion, and finally, the target does a constant planar turn. The maneuvers occur at times $t = 200s$ and $t = 450s$ respectively. The initial state is $R_0 = I_3$, $x_0 = \begin{pmatrix} 10^4 & 10^4 & 10^4 \end{pmatrix} m$, $u_0 = 100m.s^{-1}$, $\gamma_0 = 0.02s^{-1}$ and $\tau_0 = 0$. The first jump at $t = 200s$ is characterized by $u_{200} = 500m.s^{-1}$, $\gamma_{200} = 0.07s^{-1}$ and $\tau_{200} = 0.005$, and the second jump by $u_{450} = 200m.s^{-1}$, $\gamma_{450} = 0.002s^{-1}$ and $\tau_{450} = 0$, elsewhere, the velocity, curvature and torsion are constant. We assume Cartesian position measurements, and we add Gaussian measurement noise of variance $1000m$ independently on the three axis.



(a) (b) (c)

Figure 7.3 – Results of the linear Kalman filter estimations for the position in x fig. 7.3a, velocity in x fig. 7.3b and acceleration in x fig. 7.3c

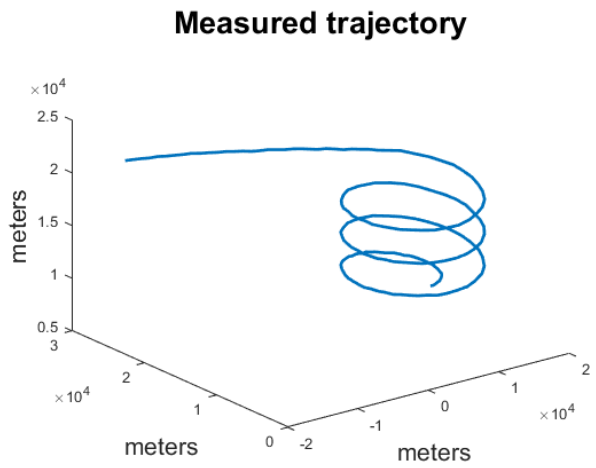


Figure 7.4 – Measured trajectory with two maneuvers

We show the performances of the IEKF, used with a fixed update rate of T = 5s. The tracking results of the IEKF alone are presented on figure 7.5. We see that the torsion is very difficult to track accurately, indeed, it is a third derivative of the position, and thus it is barely observable. The same problem as the linear Kalman filter with the noise tuning occurs because of the presence of jumps. However once again, the tracking results are satisfying to perform update rate adaptation with the same noise tunings. Indeed, the position and the norm of the velocity, and to a certain extent also the curvature are well estimated with this fixed update rate.

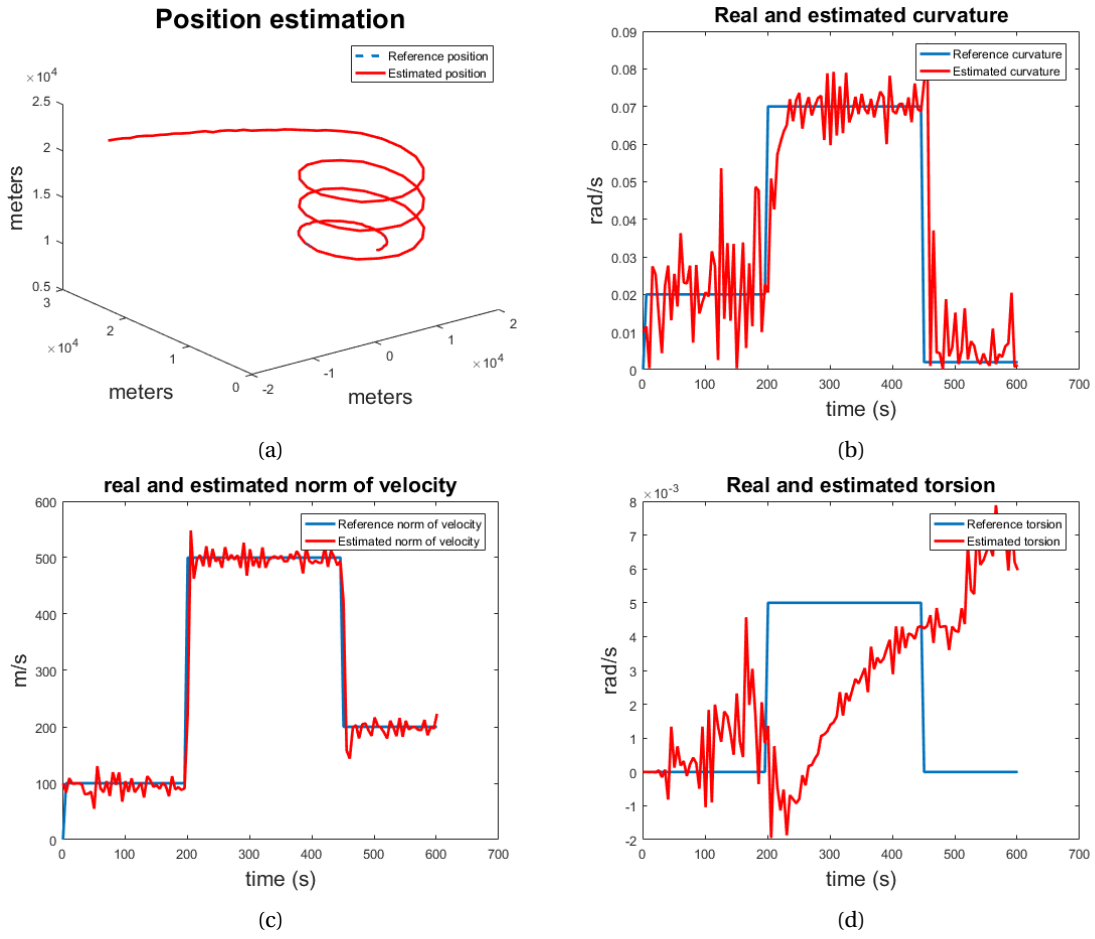


Figure 7.5 – Results for the tracking with the IEKF, with the x position fig. 7.5a, the curvature fig. 7.5b, the norm of the velocity fig. 7.5c and the torsion fig. 7.5d

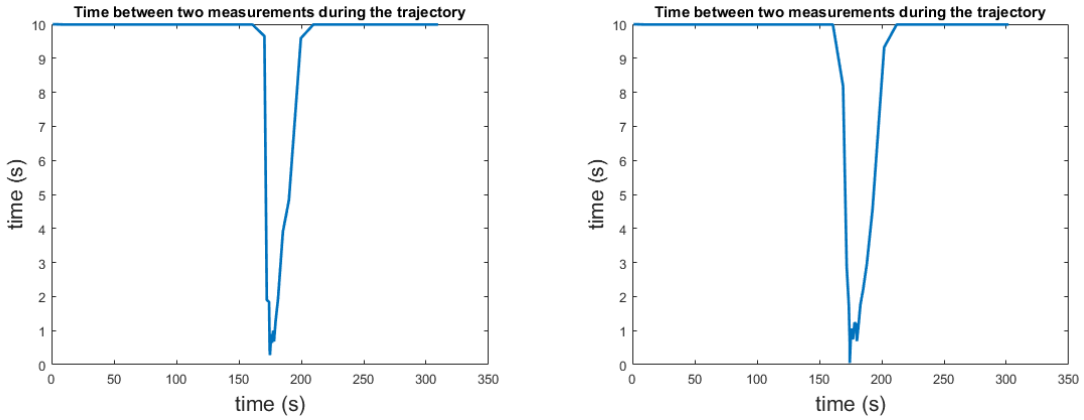
7.5.2 Update rate adaptation

We now implement the two update rate adaptation methods presented in chapters 6 and 7. The first one uses the fixed criterion of Blackman and Van Keuk described in section 6.3, and the second one is the adaptive criterion of algorithm 4. We test the algorithms on the trajectories presented in the previous section, and keep the same process noise tunings for the filtering algorithms.

Update rate adaptation with the Singer trajectory

We first present the results obtained with the Singer model based trajectory. We compare the fixed criterion with the adaptive one. The Singer model being the model used to compute the fixed criterion, the results should be very similar for both methods. The parameters used for the update rate adaptation for both algorithms are summarised in table 7.1.

Fixed update criterion The fixed criterion has been first implemented. We plot the duration computed by the algorithm between two measurements as a function of the time elapsed since the beginning of the trajectory. The higher bound for the duration is $T = 10s$ and the lower bound is $T = 0.01s$ and $dt = 0.01s$. The graph for the time interval between measurements is presented in figure 7.6a.



(a) Fixed criterion

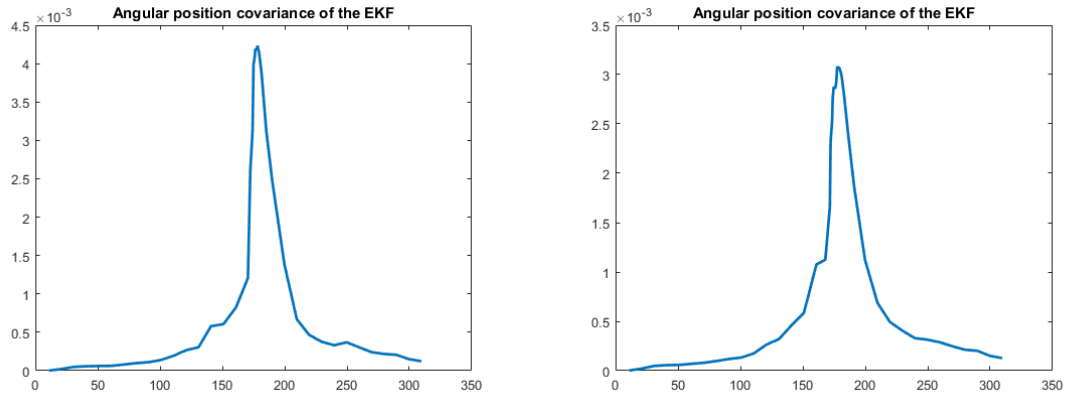
(b) Adaptive criterion

Figure 7.6 – Comparison of the duration between two measurements for the Singer model, with the fixed criterion on fig. 7.6a and for the adaptive criterion on fig. 7.6b. We see the behaviour of the adaptation is very similar for the two criteria. This was expected since the fixed criterion is derived for a Singer model. The peak corresponds to the moment the target becomes very close to the radar, so the angular covariance becomes large.

We can note that the adaptation is not necessarily linked to the presence of a manoeuvre (the update rate decreases before the manoeuvre occurs). In fact, it is due to an increase of the covariance in the u-coordinate at $t = 175s$, because the target is very close to the radar at this point, and we consider the angular dispersion. Let us now compare this behaviour to the one obtained with the adaptive algorithm described in this paper.

Adaptive criterion We make the same experiment, but we replace the fixed criterion with the adaptive one of algorithm 4, and plot the result in figure 7.6b. In order to achieve this, we compute the detection probability P_D that corresponds to $V_0 = 0.3$ thanks to the formula $P_{D_0} = P_F^{1/(1+SN_0)}$, which gives the threshold in algorithm 3 to be $s = 0.72$.

We also plot the lateral position covariances for both algorithms in figure 7.7. The duration general form curve echoes the covariance curve. When the covariance is low, then the maximum duration between two measurements is reached (in the first 150 seconds for instance).



(a) Fixed criterion

(b) Adaptive criterion

Figure 7.7 – Comparison of the covariances for the angular coordinate for the Singer model, with the fixed criterion on fig. 7.7a and for the adaptive criterion fig. 7.7b.

As expected, the update rate is very similar in both cases. Indeed, the fixed criterion is adapted to this Singer model, and the adaptive criterion is designed to match any model formulation. This shows our method encompasses and generalises the model of Blackman and Van Keuk indeed.

Optimisation ratio We have performed 100 Monte-Carlo experiments to compute the value of the rate $L_c = E(n)/E(T)$ for both methods in order to compare them. The loads L_c computed are given in table 7.2. The small difference is due to the approximations that are made in the fixed criterion derivation, and also to the fact that we are not necessarily in a stationary regime during all the trajectory, but the results are nevertheless very similar. We also collect the position estimation Root Mean Square Error, to ensure there is no major degradation of performance when using the adaptive criterion. Indeed, the radar load is optimised, but the tracking performances need to stay at least in the same magnitude order. The RMSE can be found in table 7.3.

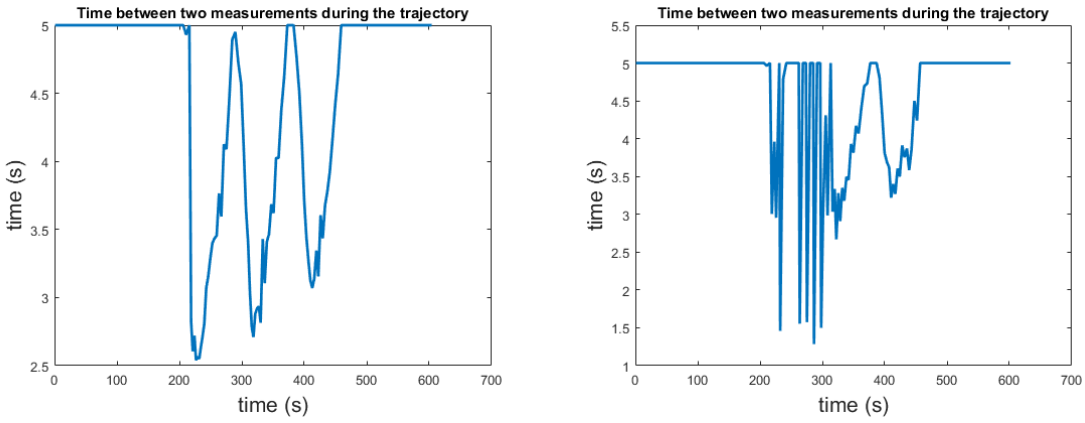
Update rate adaptation with the Frenet-Serret trajectory

The present experiment serves as an example for other models than the Singer model, that are non-linear, have to be estimated using a non-linear filtering algorithm and cannot have the same properties of convergence than the linear Kalman filter. The duration between two consecutive measurements can vary from $T_{max} = 5s$ to $T_{min} = 0.005s$ and $dt = 0.005s$. We perform again the simulations with the Blackman Van Keuk fixed criterion and with the adaptive algorithm. The results are presented in the following paragraphs. The values of the other parameters are gathered in table 7.1.

Parameter	Value
P_F	10^{-6}
SN_0	40
B	0.0175
s	0.72

Table 7.1 – Parameters for the update rate adaptation

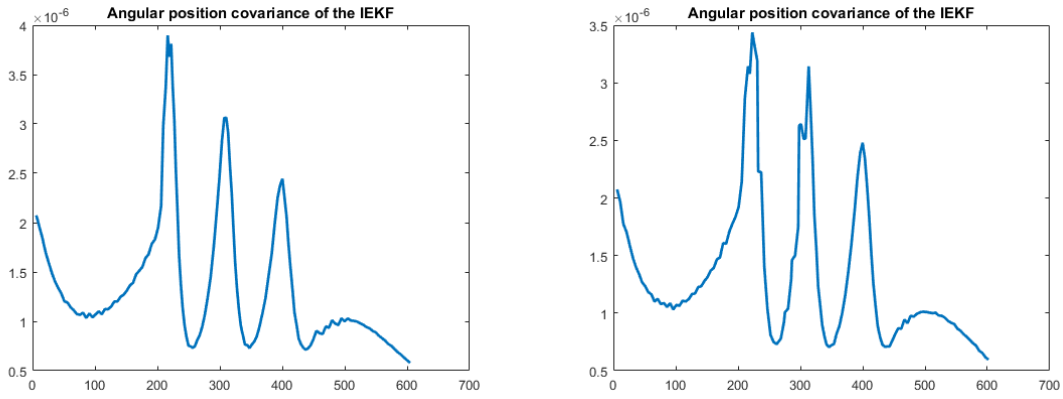
Fixed criterion The results are given on figures 7.8a and 7.9a. We can allow the duration between the measurements up to $T = 5s$ on some portions of the trajectory.



(a) Fixed criterion

(b) Adaptive criterion

Figure 7.8 – Comparison of the duration between two measurements for the Frenet-Serret model, with the fixed criterion on fig. 7.8a and the adaptive one on fig. 7.8b. The experiments give two different behaviours of the duration between two observations. Indeed, the fixed criterion is not suited to this model, and the adaptive criterion is required, as it directly minimizes L_c, contrary to the fixed criterion.



(a) Fixed criterion

(b) Adaptive criterion

Figure 7.9 – Comparison of the covariances for the angular coordinate for the Frenet-Serret model, with the fixed criterion on fig. 7.9a and for the adaptive criterion on fig. 7.9b.

Adaptive criterion Thanks to (7.2), we can compute the threshold $s = 0.72$ that we apply for the adaptive algorithm. The results are presented in figure 7.8b. The angular covariances are also displayed in figure 7.9. In this case, the covariances computed are still very similar, because they are the result of the filter, and the difference is only due to the period of refreshment.

However, contrary to the Singer model, the update rate results are quite different. The duration between maneuvers is often lower for the adaptive criterion, however, the number of necessary illuminations to find a target again is also lower. We see that the two algorithms lead to two different strategies, even though the general form of the adaptation is the same.

Optimization ratio The optimisation rate L_c is again computed with 100 Monte-Carlo simulations for the fixed and adaptive algorithms. The radar loads L_c obtained for the Frenet-Serret model are again collected in table 7.2. This shows that the optimisation is much better with an adaptive criterion, and that the fixed criterion is not sufficient to achieve the best radar load. The difference between the rates multiplied by the number of possible targets in a challenging scenario is therefore not negligible at all. The position RMSE are also collected in table 7.3, again no loss in the performances is observed.

	Fixed criterion	Adaptive criterion
Linear model	0.20	0.16
Nonlinear model	0.44	0.25

Table 7.2 – Radar load L_c computed for the experiments presented in this paper

	Fixed criterion	Adaptive criterion
Linear model	132	63
Nonlinear model	105	104

Table 7.3 – Position RMSE (in m) computed for the experiments presented in this paper

7.6 Discussion

The results obtained in the previous section show that the fixed criterion of Blackman and Van Keuk and the adaptive criterion proposed in this chapter are equivalent for the Singer model. However, with non-linear target models, the adaptive algorithm performs better than the fixed one, as anticipated. This saves the radar time budget: the update rate can be more decreased than when it is fixed, and the adaptive criterion algorithm is better suited to non-linear models and algorithms than the fixed one.

However, such adaptation algorithms can only adapt the update rate once the performances are beginning to decrease, they are thus not suited to detect or track high and abrupt manoeuvres. They are designed to detect a degradation in the filter's confidence of its own estimations, and to increase the update rate, so that the filter is updated more often, and the target has a higher probability to be in the radar beam for the next measurement, and so less energy is spent to find it again. Moreover, the covariance of the filter needs to be quite accurate, since it feeds the update rate adaptation algorithm. If the filter diverges, this means the covariance is not accurate anymore, so the update rate adaptation fails along with the filter.

This is corroborated by the results obtained in the previous section. Indeed, the update rate is not necessarily modified only during manoeuvres, but more generally whenever the angular covariance of the filtering algorithm is increasing.

7.7 Conclusion

In this chapter, we have first adapted the algorithm of chapter 6 to suit other types of models and estimation algorithms. This is required for industrial applications, since non-linear target models are used.

We have then applied the proposed algorithm to a model based on intrinsic coordinates, relying on the Frenet-Serret frame, along with an Invariant Extended Kalman Filter. The results both with the Singer model and the Frenet-Serret model corroborate the accuracy of the adaptive algorithm. The fixed and adaptive algorithms give the same results on the Singer model, and the adaptive algorithm gives better results on the Frenet-Serret model.

To develop the adaptive update rate algorithm, we have used the same optimisation criterion as in chapter 6. This criterion assumes the radar has a pencil beam, and the target search method is to look in a neighbourhood of the first guess given by the estimation algorithm. This search method is the optimal one, as stated in [108], so the same search method is used in this chapter. These two assumptions have not been questioned in this chapter, but some work can be done to have another optimisation function that takes into account the new possibilities of the newer generations of radars, that are becoming even more cognitive, including the fact that future radars will not necessarily be equipped with pencil beams, and so the radar load will have to be expressed differently.

Chapter 8

Conclusion

8.1 Résumé en français

Les travaux menés pendant cette thèse ont permis de proposer des solutions pour le pistage des cibles hyper-manœuvrantes. Le modèle de cible qui a été conçu est en effet dédié à ces cibles, et notamment à la description des mouvements en 3D, et aux virages avec de fortes vitesses angulaires. Un premier algorithme d'estimation a été proposé, utilisant le principe du filtrage, et construit spécifiquement à partir du modèle de cible développé. Cet algorithme donne des performances satisfaisantes et supérieures à des algorithmes couramment utilisés. Une autre méthode d'estimation, reposant sur un modèle de cible légèrement modifié a permis de prendre en compte des manœuvres extrêmes de cibles, comme des sauts dans le cap des cibles. Cette seconde méthode d'estimation d'état, du lissage avec sauts, est conçue pour des trajectoires avec des sauts extrêmes. On a observé que les algorithmes développés sont très performants pour l'estimation du cap de la cible, qui est indispensable pour avoir une bonne prédiction de position, dans le but d'orienter le faisceau de la poursuite active, pour avoir un bon affichage sur l'écran de l'opérateur radar, ou encore pour les radars de conduite de tir. Enfin, le travail sur la cadence adaptative permet d'envisager de réduire le budget du radar consacré à la tâche de poursuite active.

Plusieurs perspectives pour des travaux futurs peuvent être envisagées. Tout d'abord, il est possible de travailler à l'incorporation de l'IEKF à un IMM avec un EKF à modèle accélération constante, afin de permettre un réglage de bruit de modèle relativement faible pour l'IEKF, et avoir un fort bruit de modèle sur l'autre modèle pour pouvoir faire converger l'algorithme dans le cas de manœuvres extrêmes. Il est également possible de travailler sur le réglage du bruit de modèle de l'IEKF seul, afin de rendre le filtre encore plus robuste aux manœuvres. L'algorithme de lissage peut être généralisé en 3D, il faut alors être capable d'exprimer le flot de forme close dans l'espace mathématique du modèle de Frenet-Serret 3D. Enfin, on peut imaginer un algorithme de cadence adaptative encore plus général, et ne se limitant plus aux radars à faisceaux focalisés, mais étendus à des radars qui ont la capacité d'élargir leur faisceau si nécessaire.

8.2 Summary of the main contributions of the thesis

The challenge for modern multifunction radars is to face an increasing number of parallel tasks as well as a greater complexity of each task, including tracking highly manoeuvring targets. The arrival of phased-array fixed antenna radars brings new opportunities for numerical processing. The objective of this thesis was to exploit these state-of-the-art possibilities to provide more performing algorithms to track new targets, with high and unpredictable manoeuvres.

The new possibilities of modern radars include more computing power, a non-rotating antenna, dynamical beam-forming, among others. The radars are thus much more flexible in all these domains. This allows the design of more sophisticated algorithms to perform the tasks of tracking, and dynamic management of resources. The use of slightly more greedy algorithms is allowed, as long as it stays reasonable, and this opens the way to more suitable solutions to the

new stakes.

One of the main contributions of this work has been to propose a novel target model, based on intrinsic coordinates, and flexible to adapt different configurations. This model is adapted to highly manoeuvring targets, especially when they have very high normal accelerations. This target model can represent virtually any target motion, since the Frenet-Serret frame can be used to describe any continuous curve in the space. We have nonetheless restricted the model and made some light and frequently used assumptions to derive the equations to make the estimation problem tractable.

A robust filtering algorithm taking advantage of the mathematical space in which the target model is expressed has been developed to perform state estimation. This algorithm has been successfully tested on real scenarios. The IEKF used with the Frenet-Serret frame gives encouraging results, especially for the heading precision. This is one of the required performances of the tracking, and it is important for several reasons. First, the operator has a better readability on his screen when the heading is precise, and does not oscillate, then it is crucial for the radar to point in the right direction during active tracking, and if the covariance is thinner for the heading, then the beam formed will be thinner, thus saving the radar time budget, and facilitating the association problem. The filtering algorithm proposed can be easily implemented, since its structure is the same as the one of the widely used EKF. It has thus been successfully tested in a real tracking simulator for Air Traffic Control in Thales, which includes all the tracking process.

In a more academic field, target models modelling possible jumps in the trajectory have also been discussed, leading to algorithms that can handle the jumps, such as the Rao-Blackwell particle filter, and the smoothing algorithm. This latter algorithm performs very accurately on trajectories with really huge manoeuvres, that are the future targets the radars will have to face. The smoothing algorithm is more greedy in terms of computations than Kalman filters. However, the computation time can be reduced thanks to the use of sliding windows, or to the optimisation of matrix computation, especially to compute the inverse of sparse matrices. The computation time of the smoothing algorithm remains much smaller than the one of particle filters, even when considering the Rao-Blackwell particle filter, for which the number of required particles is reduced compared to standard particle filters. This smoothing algorithm is however more difficult to embed in an existing simulator, since its structure is completely different from that of a Kalman filter.

The last contribution is the development of a general update rate adaptation algorithm that can be applied to any filtering algorithm. This update rate adaptation algorithm computes the best update time to optimise the radar time budget spent on active tracking for one particular target. This is also of crucial importance for multifunction radars that are asked to perform surveillance as well as tracking, and are confronted to scenarios with a high number of targets.

8.3 Future work

Various questions and remarks have arisen from the contributions presented in this document. Some concern academic aspects, such as future development for the algorithms, and other concern more industrial issues that have been discussed with different teams inside Thales.

One of the first aspect that can be further investigated is the smoothing algorithm. Future research can be performed to extend it to the Lie group setting used for the 3D target model developed in this work. The solution is not straightforward, as the model is expressed partly in a Lie group and partly in a vector space.

The definition and tuning of the noises of the Frenet-Serret target model when used with an IEKF can be further examined. Indeed, as was shown in the simulation chapter, noise tuning is an issue with every filtering algorithm. The noises for the Frenet-Serret target model have been, in this document, defined partly in the Lie algebra, which might be an issue because when projected on the Lie group, the property of being white and Gaussian does not fully hold. It is indeed a theoretical difficulty of this model formulation that can lead to other research on the definition of the noises on a Lie group.

Another aspect already discussed in the document is the use of the IEKF inside an IMM. Indeed, we have seen that the IEKF has better performances than three models of the IMM, the constant velocity model, the horizontal turn and the vertical turn. However, to secure the algorithm, one can use a constant acceleration model in parallel. We have seen that the constant velocity Frenet-Serret model is not really adapted to targets which have a high tangential acceleration. Two solutions are possible: use the constant acceleration Frenet-Serret model, but this degrades the performances on constant parts, or couple the IEKF inside an IMM with a constant acceleration model. Work has to be done to provide a real mixing stage in the IMM, to merge the state of the IEKF and the one of a constant acceleration linear model.

Concerning the update rate adaptation, the method proposed in this document is very general, but it can be made even more adaptive if the beam forming (including the waveform for instance) in itself is included in the adaptation algorithm. One future field of research in this domain would be to link the time of refreshment of the observations with the formation of the radar beam.

Finally, all the methods presented in this document are designed for surface radars, which means they are not moving. One possible generalisation would be to extend the methods and test them for mobile radars, for instance embarked in an aircraft.

Appendix A

Mathematical definitions: Lie groups

A.1 General definitions

A **matrix Lie group** G is a set of matrices, stable by multiplication and inversion, and having a tangent space at the neutral element.

G is a matrix group means that its group operation is the matrix multiplication and that:

$$\mathrm{I}d \in \mathrm{G},\ \forall g \in \mathrm{G}, g^{-1} \in \mathrm{G},\ \forall g_1, g_2 \in \mathrm{G}, g_1 g_2 \in \mathrm{G}$$

The **Lie algebra** $\mathfrak{g}$ associated to the group is the tangent space at the neutral element. The Lie algebra is an $\mathbb{R}$-algebra, which means that it is a vectorial space with an intern multiplication which is bilinear. For a matrix Lie group, it is possible to represent the vectors of $\mathbb{R}^{\dim \mathfrak{g}}$ under the form of a matrix belonging to the Lie algebra, that is strictly equivalent, but may be more convenient to write the operations. If $\omega \in \mathbb{R}^{\dim \mathfrak{g}}$, then we note $\mathscr{L}_{\mathfrak{g}}$ the associated matrix in the Lie algebra $\mathfrak{g}$. In other words, we have a linear mapping $\mathscr{L}_{\mathfrak{g}} : \mathbb{R}^{\dim \mathfrak{g}} \rightarrow \mathfrak{g}$. The linear mapping will be detailed for each Lie group used in this thesis.

Finally, we also need the definition of the **exponential** on the Lie algebra. The Lie group exponential is defined in the Lie algebra and has values in the Lie group. For matrix Lie groups, this exponential is in fact $\exp_m$, the matrix exponential defined by the Taylor series:

$$\exp_m(\mathrm{A}) = \sum_{n=0}^{\infty} \frac{\mathrm{A}^n}{n!}$$

We can also define an exponential $\exp : \mathbb{R}^{\dim \mathfrak{g}} \rightarrow \mathrm{G}$ as follows:

$$\exp(\omega) = \exp_m(\mathscr{L}_{\mathfrak{g}}(\omega)),\ \forall \omega \in \mathfrak{g}$$

The description of the Lie groups encountered in this document, and the useful operations are provided in the following paragraphs.

A.2 Specific Lie groups used in the thesis

A.2.1 Group of 2D rotations SO(2)

The rotations in 2D form a matrix Lie group, SO(2), that is defined by:

$$\mathrm{SO}(2) = \left\{\mathrm{R} \in \mathscr{M}_2 | \mathrm{R}\mathrm{R}^{\mathrm{T}} = \mathrm{R}^{\mathrm{T}}\mathrm{R} = \mathrm{I}_2, det(\mathrm{R}) = 1\right\}$$

Matrices in this group can all be written in the following way, as a rotation of angle θ, assuming $\theta \in \mathbb{R}$:

$$\mathrm{R} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$$

The Lie algebra is of dimension 1, it is composed of the skew-symmetric matrices of size 2×2.

The notation $(.)_\times$ corresponding to the linear mapping from $\mathbb{R}$ to $\mathfrak{so}(2)$ is defined as:

$$(\omega)_\times = \begin{pmatrix} 0 & -\omega \\ \omega & 0 \end{pmatrix}$$

It in fact corresponds to the linear mapping $\mathscr{L}_{\mathfrak{so}(2)}$ for this Lie group, but the notation is more convenient.

The group exponential writes:

$$\exp_m \begin{pmatrix} 0 & -\omega \\ \omega & 0 \end{pmatrix} = \begin{pmatrix} \cos\omega & -\sin\omega \\ \sin\omega & \cos\omega \end{pmatrix}$$

As we have said, the exponential belongs indeed to the Lie group.

A.2.2 Group of 2D rotations and translations $\mathrm{SE}(2)$

We can add the 2D translations to SO(2) to form the group of 2D rotations and translations, SE(2). This group is defined by

$$\mathrm{SE}(2) = \left\{ \begin{pmatrix} \mathrm{R} & x \\ 0_{1,2} & 1 \end{pmatrix}, \mathrm{R} \in \mathrm{SO}(2), x \in \mathbb{R}^2 \right\}$$

The associated Lie algebra, called $\mathfrak{se}(2)$ is

$$\mathfrak{se}(2) = \left\{ \begin{pmatrix} 0 & -\omega & \alpha \\ \omega & 0 & \beta \\ 0 & 0 & 0 \end{pmatrix}, \omega \in \mathbb{R}, \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \in \mathbb{R}^2 \right\}$$

It is thus of dimension 3. In this case, $\mathscr{L}_{\mathfrak{se}(2)}$ is defined by:

$$\mathscr{L}_{\mathfrak{se}(2)} \begin{pmatrix} \omega \\ \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} 0 & -\omega & \alpha \\ \omega & 0 & \beta \\ 0 & 0 & 0 \end{pmatrix}$$

The exponential of $(\omega, u) \in \mathbb{R} \times \mathbb{R}^2$ is defined by

$$\exp_m \begin{pmatrix} (\omega)_\times & u \\ 0_{1,2} & 0 \end{pmatrix} = \begin{pmatrix} \mathrm{R}(\omega) & \mathrm{V} \\ 0_{1,2} & 1 \end{pmatrix}$$

with

$$\mathrm{R}(\omega) = \begin{pmatrix} \cos\omega & -\sin\omega \\ \sin\omega & \cos\omega \end{pmatrix}$$

and

$$\mathrm{V} = \begin{pmatrix} u_1 \frac{\sin\omega}{\omega} - u_2 \frac{1-\cos\omega}{\omega} \\ u_1 \frac{1-\cos\omega}{\omega} + u_2 \frac{\sin\omega}{\omega} \end{pmatrix}$$

if $\omega \neq 0$, otherwise we have

$$\exp_m \begin{pmatrix} 0_{2,2} & u \\ 0_{1,2} & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & u_1 \\ 0 & 1 & u_2 \\ 0 & 0 & 1 \end{pmatrix}$$

A.2.3 Group of 3D rotations $\mathrm{SO}(3)$

The group of rotations SO(3) is composed of the rotation matrices of dimension 3×3:

$$\mathrm{SO}(3) = \{\mathrm{R} \in \mathscr{M}_3 | \mathrm{R}\mathrm{R}^\mathrm{T} = \mathrm{R}^\mathrm{T}\mathrm{R} = \mathrm{I}_3, \det(\mathrm{R}) = 1\}$$

The associated Lie algebra is of dimension 3 and is defined by:

$$\mathfrak{so}(3) = \left\{ \begin{pmatrix} 0 & -c & b \\ c & 0 & -a \\ -b & a & 0 \end{pmatrix}, \begin{pmatrix} a \\ b \\ c \end{pmatrix} \in \mathbb{R}^3 \right\}$$

It is again possible to define the notation $(.)_\times$. If $\omega \in \mathbb{R}^3$, then

$$(\omega)_\times = \begin{pmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{pmatrix}$$

One also needs the definition of the matrix exponential $\exp : \mathfrak{so}(3) \rightarrow \mathrm{SO}(3)$. Let $\Omega \in \mathfrak{so}(3)$, and $\theta = \|\Omega\|$. The exponential map is:

$$\exp_m(\Omega) = \mathrm{I}_3 + \frac{\sin\theta}{\theta}\Omega + \frac{1-\cos\theta}{\theta^2}\Omega^2$$

$$\exp_m(0_{3,3}) = \mathrm{I}_3$$

A.2.4 Group of 3D rotations and translations SE(3)

SE(3) is the group that describes the possible motions of a point mass in the 3D space. It represents the rotations and the translations:

$$\mathrm{SE}(3) = \left\{ \begin{pmatrix} \mathrm{R} & x \\ 0_{1,3} & 1 \end{pmatrix}, \mathrm{R} \in \mathrm{SO}(3), x \in \mathbb{R}^3 \right\}$$

The associated Lie algebra is of dimension 6 and is defined by:

$$\mathfrak{se}(3) = \left\{ \begin{pmatrix} 0 & -c & b & \alpha \\ c & 0 & -a & \beta \\ -b & a & 0 & \gamma \\ 0 & 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} a \\ b \\ c \end{pmatrix} \in \mathbb{R}^3, \begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix} \in \mathbb{R}^3 \right\}$$

The mapping $\mathscr{L}_{\mathfrak{se}(3)}$ is defined by

$$\mathscr{L}_{\mathfrak{se}(3)} \begin{pmatrix} a \\ b \\ c \\ \alpha \\ \beta \\ \gamma \end{pmatrix} = \begin{pmatrix} 0 & -c & b & \alpha \\ c & 0 & -a & \beta \\ -b & a & 0 & \gamma \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

The exponential map of $(\omega, u) \in \mathbb{R}^3 \times \mathbb{R}^3$ is defined as follows

$$\exp \begin{pmatrix} \omega \\ u \end{pmatrix} = \exp_m \begin{pmatrix} (\omega)_\times & u \\ 0_{1,3} & 0 \end{pmatrix} = \begin{pmatrix} \exp_m((\omega)_\times) & \mathrm{V}u \\ 0_{1,3} & 1 \end{pmatrix}$$

with $\theta = \sqrt{\omega^{\mathrm{T}}\omega}$ if $\omega \neq 0$ and

$$\mathrm{V} = \mathrm{I}_3 + \frac{1-\cos\theta}{\theta^2}(\omega)_\times + \frac{\theta - \sin\theta}{\theta^3}(\omega)_\times^2$$

$$\exp \begin{pmatrix} 0_{3,1} \\ u \end{pmatrix} = \begin{pmatrix} \mathrm{I}_3 & u \\ 0_{1,3} & 0 \end{pmatrix}$$

Appendix B

More details about the Kalman Filter

In this appendix, we show how the filter's equations are derived. A tutorial paper fort this is [77]. The problem we want to solve is to estimate a vector x_k from partial and noisy observations y_k.

$$y_k = \mathrm{H}x_k + v_k \tag{B.1}$$

x_k evolves according to equation

$$x_{k+1} = \mathrm{F}x_k + w_k \tag{B.2}$$

v_k and w_k are assumed to be white Gaussian noises, with covariances $\mathrm{N} = \mathrm{E}[v_k v_k^{\mathrm{T}}]$ and $\mathrm{Q} = \mathrm{E}[w_k w_k^{\mathrm{T}}]$ respectively. We do not consider the input vector u_k in this appendix.

B.1 Maximum Likelihood Estimator

We seek to compute the probability distribution $\mathrm{P}(x_k|y_1,\ldots,y_k)$. Due to the fact that the noises are Gaussian and independent, and that Gaussians are stable by linear transformations, we can show that this distribution is a Gaussian $\mathcal{N}(\hat{x}_{k|k}, \mathrm{P}_{k|k})$.

Otherwise, because the distribution is Gaussian, the conditional mean $\hat{x}_{k|k}$ corresponds to the most probable value of x_k given the past observations $y_1,\ldots,y_k$. This is also the best least squares estimate.

B.2 Algorithm derivation

Let us define the error as $e_k = x_k - \hat{x}_{k|k}$. The covariance matrix associated to the error is $\mathrm{P}_{k|k} = \mathrm{E}[e_k e_k^{\mathrm{T}}]$. This gives:

$$\mathrm{P}_{k|k} = \mathrm{E}[(x_k - \hat{x}_{k|k})(x_k - \hat{x}_{k|k})^{\mathrm{T}}]$$

We first need to compute the step going from $\hat{x}_{k|k}$ to $\hat{x}_{k+1|k}$, and the prediction of the covariance matrix P. The state projection is

$$\hat{x}_{k+1|k} = \mathrm{F}\hat{x}_{k|k}$$

For the error, we have

$$\begin{aligned} e_{k+1|k} &= x_{k+1} - \hat{x}_{k+1|k} \\ &= (\mathrm{F}x_k + w_k) - \mathrm{F}\hat{x}_{k|k} \\ &= \mathrm{F}e_{k|k} + w_k \end{aligned}$$

So, using the fact that $\mathrm{E}[e_{k|k} w_k] = 0$, we obtain:

$$\begin{aligned}
\mathrm{P}_{k+1|k} &= \mathrm{E}[e_{k+1|k} e_{k+1|k}^{\mathrm{T}}] \\
&= \mathrm{E}[(\mathrm{F}e_{k|k} + w_k)(\mathrm{F}e_{k|k} + w_k)^{\mathrm{T}}] \\
&= \mathrm{E}[\mathrm{F}e_{k|k}(\mathrm{F}e_{k|k})^{\mathrm{T}}] + \mathrm{E}[w_k w_k^{\mathrm{T}}] \\
&= \mathrm{F}\mathrm{E}[e_{k|k} e_{k|k}^{\mathrm{T}}]\mathrm{F}^{\mathrm{T}} + \mathrm{Q} \\
&= \mathrm{F}\mathrm{P}_{k|k}\mathrm{F}^{\mathrm{T}} + \mathrm{Q}
\end{aligned}$$

The observation must now be taken into account to obtain the update equations, for the state $\hat{x}$ and the covariance P. Kalman proved that the updated state has the form:

$$\hat{x}_{k+1|k+1} = \hat{x}_{k+1|k} + \mathrm{K}_{k+1}(y_{k+1} - \mathrm{H}\hat{x}_{k+1|k})$$

where K_k is the Kalman gain. Once this form is assumed, the optimal Kalman gain is easily derived as follows, and using equation (B.1):

$$\hat{x}_{k+1|k+1} = \hat{x}_{k+1|k} + \mathrm{K}_{k+1}(\mathrm{H}x_{k+1} + v_{k+1} - \mathrm{H}\hat{x}_{k+1|k})$$

We now use this equation, and the fact that $x_{k+1} - \hat{x}_{k+1|k}$ and v_{k+1} are uncorrelated to compute $\mathrm{P}_{k+1|k+1}$:

$$\begin{aligned}
\mathrm{P}_{k+1|k+1} &= \mathrm{E}\left[[(\mathrm{I} - \mathrm{K}_{k+1}\mathrm{H})(x_{k+1} - \hat{x}_{k+1|k}) - \mathrm{K}_{k+1} v_{k+1}][(\mathrm{I} - \mathrm{K}_{k+1}\mathrm{H})(x_{k+1} - \hat{x}_{k+1|k}) - \mathrm{K}_{k+1} v_{k+1}]^{\mathrm{T}}\right] \\
&= (\mathrm{I} - \mathrm{K}_{k+1}\mathrm{H})\mathrm{E}[x_{k+1} - \hat{x}_{k+1|k}](\mathrm{I} - \mathrm{K}_{k+1}\mathrm{H})^{\mathrm{T}} + \mathrm{K}_{k+1}\mathrm{E}[v_{k+1} v_{k+1}^{\mathrm{T}}]\mathrm{K}_{k+1}^{\mathrm{T}} \\
&= (\mathrm{I} - \mathrm{K}_{k+1}\mathrm{H})\mathrm{P}_{k+1|k}(\mathrm{I} - \mathrm{K}_{k+1}\mathrm{H})^{\mathrm{T}} + \mathrm{K}_{k+1}\mathrm{N}\mathrm{K}_{k+1}^{\mathrm{T}} \\
&= \mathrm{P}_{k+1|k} - \mathrm{K}_{k+1}\mathrm{H}\mathrm{P}_{k+1|k} - \mathrm{P}_{k+1|k}\mathrm{H}^{\mathrm{T}}\mathrm{K}_{k+1}^{\mathrm{T}} + \mathrm{K}_{k+1}(\mathrm{H}\mathrm{P}_{k+1|k}\mathrm{H}^{\mathrm{T}} + \mathrm{N})\mathrm{K}_{k+1}^{\mathrm{T}}
\end{aligned}$$

As we want the mean squared error minimiser, we want to minimise $\mathrm{E}[||x_{k+1} - \hat{x}_{k+1|k+1}||^2]$, which is the same as minimising the trace of $\mathrm{P}_{k+1|k+1}$. So we have to solve the equation

$$\frac{\partial tr(\mathrm{P}_{k+1|k+1})}{\mathrm{K}_{k+1}} = 0$$

The matrix derivative rules give

$$0 = \frac{\partial tr(\mathrm{P}_{k+1|k+1})}{\mathrm{K}_k} = -2(\mathrm{H}\mathrm{P}_{k+1|k})^{\mathrm{T}} + 2\mathrm{K}_{k+1}(\mathrm{H}\mathrm{P}_{k+1|k}\mathrm{H}^{\mathrm{T}} + \mathrm{N})$$

Solving this equation, we can obtain the value of the gain K_{k+1}:

$$\mathrm{K}_{k+1} = \mathrm{P}_{k+1|k}\mathrm{H}^{\mathrm{T}}(\mathrm{H}\mathrm{P}_{k+1|k}\mathrm{H}^{\mathrm{T}} + \mathrm{N})^{-1}$$

The quantity $\mathrm{H}\mathrm{P}_{k+1|k}\mathrm{H}^{\mathrm{T}} + \mathrm{N}$ is often denoted S_{k+1} and corresponds to the measurement prediction covariance. Finally, we can compute the updated covariance matrix:

$$\begin{aligned}
\mathrm{P}_{k+1|k+1} &= \mathrm{P}_{k+1|k} - \mathrm{K}_{k+1}\mathrm{H}\mathrm{P}_{k+1|k} - \mathrm{P}_{k+1|k}\mathrm{H}^{\mathrm{T}}\mathrm{K}_{k+1}^{\mathrm{T}} + \mathrm{K}_{k+1}(\mathrm{H}\mathrm{P}_{k+1|k}\mathrm{H}^{\mathrm{T}} + \mathrm{N})\mathrm{K}_{k+1}^{\mathrm{T}} \\
&= \mathrm{P}_{k+1|k} - \mathrm{P}_{k+1|k}\mathrm{H}^{\mathrm{T}}(\mathrm{H}\mathrm{P}_{k+1|k}\mathrm{H}^{\mathrm{T}} + \mathrm{N})^{-1}\mathrm{H}\mathrm{P}_{k+1|k} \\
&= \mathrm{P}_{k+1|k} - \mathrm{K}_{k+1}\mathrm{H}\mathrm{P}_{k+1|k} \\
&= (\mathrm{I} - \mathrm{K}_{k+1}\mathrm{H})\mathrm{P}_{k+1|k}
\end{aligned}$$

When deriving the optimal filtering algorithm, we find the Kalman equations, for the prediction and the update steps.

Appendix C

Linearisation of the 2D Frenet-Serret model for smoothing

We present in this appendix the linearisation of the 2D Frenet-Serret model to apply the smoothing algorithm. We take the notations of section 5.7.

Let us linearise the full system defined by (5.26), assuming that $\omega_0 \neq 0$, which is recalled here:

$$x_n = x_0 + u_0 \Delta t \sum_{k=0}^{n-1} \begin{pmatrix} \cos(\theta_0 + k\omega_0\Delta t)\frac{\sin(\omega_0\Delta t)}{\omega_0} - \sin(\theta_0 + k\omega_0\Delta t)\frac{1-\cos(\omega_0\Delta t)}{\omega_0} \\ \sin(\theta_0 + k\omega_0\Delta t)\frac{\sin(\omega_0\Delta t)}{\omega_0} + \cos(\theta_0 + k\omega_0\Delta t)\frac{1-\cos(\omega_0\Delta t)}{\omega_0} \end{pmatrix} \tag{C.1}$$

Let us call

$$A_n = \sum_{k=0}^{n-1} \cos(\theta_0 + k\omega_0\Delta t)\frac{\sin(\omega_0\Delta t)}{\omega_0} - \sin(\theta_0 + k\omega_0\Delta t)\frac{1-\cos(\omega_0\Delta t)}{\omega_0}$$

$$B_k = \cos(\theta_0 + k\omega_0\Delta t)\frac{\sin(\omega_0\Delta t)}{\omega_0}$$

$$C_k = \sin(\theta_0 + k\omega_0\Delta t)\frac{1-\cos(\omega_0\Delta t)}{\omega_0}$$

and

$$\theta_0 = \tilde{\theta}_0 + \delta\theta_0$$

$$\omega_0 = \tilde{\omega}_0 + \delta\omega_0$$

We want to linearise A_n around $\tilde{\theta}_0$ and $\tilde{\omega}_0$. We have:

$$B_k = \cos\left(\tilde{\theta}_0 + \Delta\theta_0 + k(\tilde{\omega}_0 + \delta\omega_0)\delta t\right)\frac{\sin\left((\tilde{\omega}_0 + \delta\omega_0)\Delta t\right)}{\tilde{\omega}_0 + \Delta\omega_0}$$

$$B_k \approx \left[\cos(\tilde{\theta}_0 + k\tilde{\omega}_0\Delta t) - (\delta\theta_0 + k\delta\omega_0\Delta t)\sin(\tilde{\theta}_0 + k\tilde{\omega}_0\Delta t)\right]\frac{\sin(\tilde{\omega}_0\Delta t) + (\delta\omega_0\Delta t)\cos(\tilde{\omega}_0\Delta t)}{\tilde{\omega}_0 + \delta\omega_0}$$

$$B_k \approx \left[\cos(\tilde{\theta}_0 + k\tilde{\omega}_0\Delta t) - (\delta\theta_0 + k\delta\omega_0\Delta t)\sin(\tilde{\theta}_0 + k\tilde{\omega}_0\Delta t)\right]\frac{\sin(\tilde{\omega}_0\Delta t) + (\delta\omega_0\Delta t)\cos(\tilde{\omega}_0\Delta t)}{\tilde{\omega}_0}\frac{1}{1+\frac{\delta\omega_0}{\tilde{\omega}_0}}$$

$$B_k \approx \left[\cos(\tilde{\theta}_0 + k\tilde{\omega}_0\Delta t) - (\delta\theta_0 + k\delta\omega_0\Delta t)\sin(\tilde{\theta}_0 + k\tilde{\omega}_0\Delta t)\right] \times \left[\sin(\tilde{\omega}_0\Delta t) + (\delta\omega_0\Delta t)\cos(\tilde{\omega}_0\Delta t)\right]\left[\frac{1}{\tilde{\omega}_0} - \frac{\delta\omega_0}{\tilde{\omega}_0}\right]$$

$$B_k \approx \frac{\cos(\tilde{\theta}_0 + k\tilde{\omega}_0\Delta t)\sin(\tilde{\omega}_0\Delta t)}{\tilde{\omega}_0} - \cos(\tilde{\theta}_0 + k\tilde{\omega}_0\Delta t)\sin(\tilde{\omega}_0\Delta t)\frac{\delta\omega_0}{\tilde{\omega}_0^2}$$

$$+\cos(\tilde{\theta}_0 + k\tilde{\omega}_0\Delta t)\frac{\delta\omega_0}{\tilde{\omega}_0}\Delta t\cos(\tilde{\omega}_0\Delta t) - (\delta\theta_0 + k\delta\omega_0\Delta t)\sin(\tilde{\theta}_0 + k\tilde{\omega}_0\Delta t)\frac{\tilde{\omega}_0\Delta t}{\tilde{\omega}_0}$$

and:

$$C_k \approx \left[\sin(\tilde{\theta}_0 + k\tilde{\omega}_0\Delta t) + (\delta\theta_0 + k\delta\omega_0\Delta t)\cos(\tilde{\theta}_0 + k\tilde{\omega}_0\Delta t)\right] \times \frac{1-\cos(\delta\theta_0\Delta t) + \delta\omega_0\Delta t\sin(\tilde{\omega}_0\Delta t)}{\tilde{\omega}_0}\frac{1}{1+\frac{\delta\omega_0}{\tilde{\omega}_0}}$$

$$C_k \approx \sin(\tilde{\theta}_0 + k\tilde{\omega}_0\Delta t)\left(\frac{1}{\tilde{\omega}_0} - \frac{\delta\omega_0}{\tilde{\omega}_0^2}\right) - \sin(\tilde{\theta}_0 + k\tilde{\omega}_0\Delta t)\frac{1}{\tilde{\omega}_0}\cos(\tilde{\omega}_0\Delta t)$$
$$+\sin(\tilde{\theta}_0 + k\tilde{\omega}_0\Delta t)\frac{\delta\omega_0}{\tilde{\omega}_0^2}\cos(\tilde{\omega}_0\Delta t) + \sin(\tilde{\theta}_0 + k\tilde{\omega}_0\Delta t)\frac{\delta\omega_0\Delta t\sin(\tilde{\omega}_0\Delta t)}{\tilde{\omega}_0}$$
$$+(\delta\theta_0 + k\delta\omega_0\Delta t)\cos(\tilde{\theta}_0 + k\tilde{\omega}_0\Delta t)\left(\frac{1}{\tilde{\omega}_0}(1-\cos(\tilde{\omega}_0\Delta t))\right)$$

Finally, we obtain the approximation for A_n with (C.2), and the following notations:

$$C_1 = \sum_{k=0}^{n-1}\cos(\tilde{\theta}_0 + k\tilde{\omega}_0\Delta t)$$

$$C_2 = \sum_{k=0}^{n-1} k\cos(\tilde{\theta}_0 + k\tilde{\omega}_0\Delta t)$$

$$S_1 = \sum_{k=0}^{n-1}\sin(\tilde{\theta}_0 + k\tilde{\omega}_0\Delta t)$$

$$S_2 = \sum_{k=0}^{n-1} k\sin(\tilde{\theta}_0 + k\tilde{\omega}_0\Delta t)$$

$$\begin{aligned} A_n = (&\frac{\sin(\tilde{\omega}_0\Delta t)}{\tilde{\omega}_0}C_1 - \frac{1-\cos(\tilde{\omega}_0\Delta t)}{\tilde{\omega}_0}S1) \\ &+ (-\frac{\sin(\tilde{\omega}_0\Delta t)}{\tilde{\omega}_0}S_1 - \frac{1-\cos(\tilde{\omega}_0\Delta t)}{\tilde{\omega}_0}C_1)\delta\theta_0 \\ &+ (-\frac{\sin(\tilde{\omega}_0\Delta t)}{\tilde{\omega}_0^2}C_1 + \frac{\Delta t\cos(\tilde{\omega}_0\Delta t)}{\tilde{\omega}_0}C_1 + \frac{1-\cos(\tilde{\omega}_0\Delta t)}{\tilde{\omega}_0^2}S_1 \\ &- \frac{\Delta t\sin(\tilde{\omega}_0\Delta t)}{\tilde{\omega}_0}S_1 - \frac{\Delta t\sin(\tilde{\omega}_0\Delta t)}{\tilde{\omega}_0}S_2 - \frac{\Delta t(1-\cos(\tilde{\omega}_0\Delta t))}{\tilde{\omega}_0}C_2)\delta\omega_0 \end{aligned} \tag{C.2}$$

The same linearisation apply to the second coordinate of x_n, and the end of the linearisation, and the concatenations to obtain the final matrices for the least squares problem is the same as in section 5.7.

Appendix D

Particle filters with jumps

D.1 The Rao-Blackwell particle filter

D.1.1 Description

The Rao-Blackwell particle filter is a special particle filter, introduced in [49]. The principle of this filter is to isolate one or several parameters that, being known, allow to derive the entire state distribution.

For our problem, the idea is to use the Kalman filter in an optimal way during piecewise constant motions and use the Rao-Blackwell particle filter to detect the jumps. Indeed, we know that in the linear Gaussian case, the Kalman filter is optimal, we thus seek to use this property as widely as possible, or to get close to it. The isolated parameter is sampled, and approximated by particles. The Rao-Blackwell particle filter associates a weight to each particle. A resampling step is necessary to keep only the most likely particles. The final state is a weighted sum of the states associated to each particle.

Let us present the algorithm on the simple generic model (D.1), this is the jump Markov linear system of section 2.3.4. u_k is an input, x_t is the state of the system, y_t is the observation, w_t and v_t are white Gaussian noises. r_k is a Markov chain with discrete states. $\mathrm{F}, \mathrm{B}, \mathrm{G}, \mathrm{C}, \mathrm{A}, \mathrm{D}$ are matrices that represent the evolution of the state and of the observations.

$$\begin{cases} x_{k+1} = \mathrm{F}(r_{k+1})x_t + \mathrm{B}(r_{k+1})u_{k+1} + \mathrm{G}(r_{k+1})w_{k+1} \\ y_k = \mathrm{C}(r_k)x_k + \mathrm{A}(r_k)u_k + \mathrm{D}(r_k)v_k \end{cases} \tag{D.1}$$

In our case, the sampled parameter is r_k. Indeed, when the instant of the jumps are fixed, then the trajectory is deterministic, and piecewise linear. The particles thus sample the presence of a jump.

Let $a_{1:n} = \{a_1, \ldots, a_n\}$. We want to evaluate $p(x_k, r_k | y_{1:k})$. The Bayes formula gives (D.2). We can compute $p(x_k | r_{1:k}, y_{1:k})$ with a standard Kalman filter. The remaining part $p(r_{1:k} | y_{1:k})$ has to be evaluated, which will be done by sampling. A standard particle filter would directly sample $p(x_k, r_{1:k} | y_{1:k})$, which is more demanding computationally. The Rao-Blackwell particle filter samples only along one dimension, r_k.

$$p(x_k, r_{1:k} | y_{1:k}) = p(x_k | r_{1:k}, y_{1:k}) p(r_{1:k} | y_{1:k}) \tag{D.2}$$

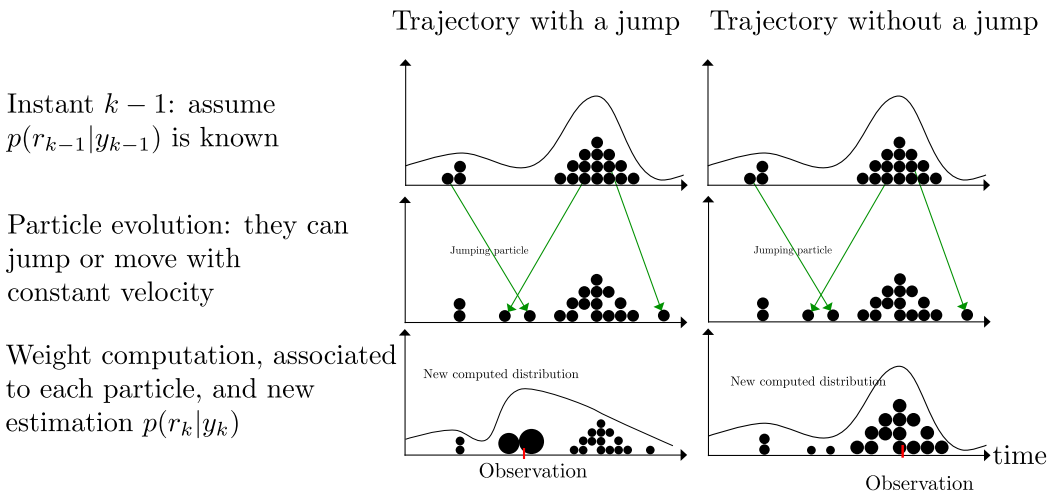


Figure D.1 – Rao-Blackwell Particle Filter

Let us describe more precisely how this partial sampling works. Suppose we have an approximation of $p(r_{1:k-1}|y_{1:k-1})$ thanks to the particles $r^i_{1:k-1}$. The sampling steps are detailed thereafter, and summarised on figure D.1. We detail the step from time instant k to $k+1$. N is the number of particles.

1. Importance sampling step:

   - For $i = 1,\dots,N$, sample $\rho^i \sim \pi(r_k|y_{1:k}, r^i_{1:k-1})$ and let $\rho^i_{1:k} \triangleq \left(r^i_{1:k-1}, \rho^i_k\right)$.
   - For $i = 1,\dots,N$, evaluate the importance weights (up to a normalising constant):

$$\tilde{\omega}^i_t \propto \frac{p(y_k|y_{1:k-1}, \rho^i_k)\, p(\rho^i_k|\rho^i_{k-1})}{\pi(\rho^i_k|y_{1:k}, \rho^i_{1:k-1})} \tag{D.3}$$

   - For $i = 1,\dots,N$, normalise the importance weights:

$$\omega^i_k = \left[\sum_{j=1}^{N} \tilde{\omega}^j_k\right]^{-1} \tilde{\omega}^i_k \tag{D.4}$$

2. Selection step: duplicate/suppress the particles $(\rho^i_{1:k}, i = 1,\dots,N)$ given their normalised weights ω^i_k to obtain N particles $(r^i_{1:k}, i = 1,\dots,N)$. This step can be done in different ways.

The probability π is called the importance distribution. It can be chosen freely, as long as it can be sampled from. Some distributions reduce the variance of the samples, as the two ones that we will present later.

Where is the Kalman filter used ?

- It is used to compute the weights ω^i_k associated to each particle. Indeed, the probability $p(y_k|y_{1:k-1}, \rho^i_{1:k})$ is exactly the probability given by the Kalman filter (it is a normal law, with mean $\hat{x}_k - y_k$ and variance S, with S the prediction variance).

- It can also be used to compute the sampling distribution π, before the sampling step.

In the two following paragraphs, the choice of the importance distribution π is detailed, one is based on the prior distribution, the other one on the optimal distribution.

Prior importance distribution

The prior distribution is

$$\pi(\rho_k^i | y_{1:k}, \rho_{1:k-1}^i) = p(\rho_k^i | \rho_{k-1}^i)$$

It is the simplest choice for the importance distribution. From a practical point of view, this means that particle i has a given probability q to jump, with q the *a priori* jumping probability. The weight of the particle i is

$$\omega_k^i = p(y_k | y_{1:k-1}, \rho_{1:k})$$

The use of this prior distribution requires the use of one Kalman filter per particle, to compute the sampling weight.

D.1.2 Results on piecewise linear trajectories

Let us use the following toy model, with piecewise constant velocity in 2D. The target model and observation model are given in (D.5).

$$\begin{cases} x_{k+1} = \mathrm{F}(r_k) x_k + \mathrm{B}(r_k) w_k \\ y_k = \mathrm{H} x_k + v_k \end{cases} \tag{D.5}$$

The state x_k contains the position and the velocity: $x_k = \begin{pmatrix} x_k^1 & \dot{x}_k^1 & x_k^2 & \dot{x}_k^2 \end{pmatrix}^{\mathrm{T}}$, and w_k is the noise on the velocity. $y_k = \begin{pmatrix} x_k^1 & x_k^2 \end{pmatrix}^{\mathrm{T}}$ is the Cartesian observation. v_k is a Gaussian white noise. This type of model, without model noise, is inspired from the variable rate model of [32], that will be detailed in section D.2.

To represent piecewise constant velocity, we use a Markov chain r_n with values 0 or 1:

$$r_n = \begin{cases} 1 & \text{with probability} \quad q \\ 0 & \text{with probability} \quad 1-q \end{cases} \tag{D.6}$$

The matrices F, B and H are defined as follows:

$$\mathrm{F}(0) = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \ \mathrm{F}(1) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\mathrm{B}(0) = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}, \ \mathrm{B}(1) = \begin{pmatrix} 0 & 0 \\ \sigma & 0 \\ 0 & 0 \\ 0 & \sigma \end{pmatrix}$$

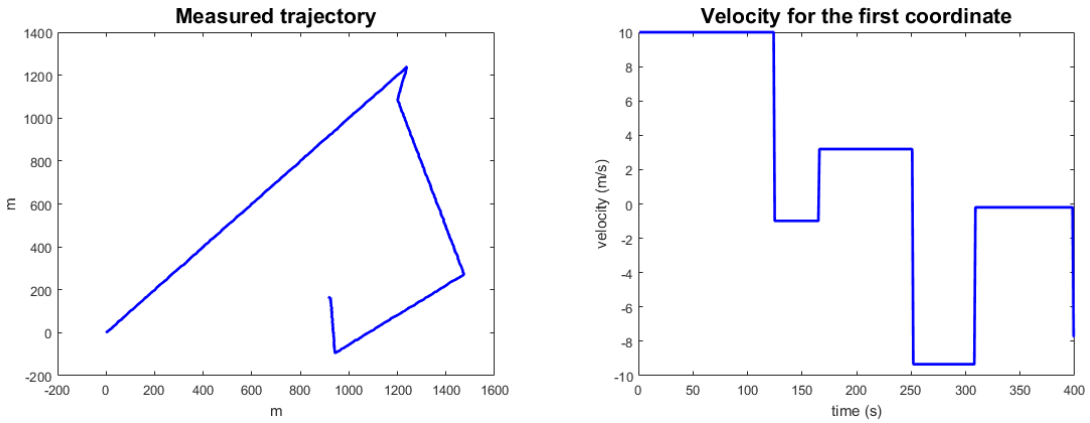
$$\mathrm{H} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

where σ is the variance of the jump allowed for the velocity. This parameter depends typically on the type of target considered.

Since the trajectory is straight with piecewise constant velocity, the stake of the filtering is to detect the jump as early as possible, and re-initialise the velocity in the best possible way once the jump is detected, so that the Kalman filter can operate in the optimal way.

An example of a trajectory is given on figure D.2, along with the velocity along the x-axis.

We perform estimation on this trajectory with different number of particles. The estimation of the position is very accurate whatever the number of particles. This position estimation is shown on figure D.3. The velocity estimation, on figure D.4, especially at the moment of the jumps depend on the number of particles used. We used 10, 100 and 1000 particles. For the case where only 10 particles are used, the jumps are detected with a delay, and the convergence takes time after a



(a) (b)

Figure D.2 – Measured trajectory fig. D.2a and real x-velocity fig. D.2b

jump. For 100 or 1000 particles, the results are quite equivalent, and the jump detection is faster, and the velocity is better estimated right after the jumps. We also observe that when the number of particles is low (10 particles for instance), there are more jumps. This is due to the observation noise.

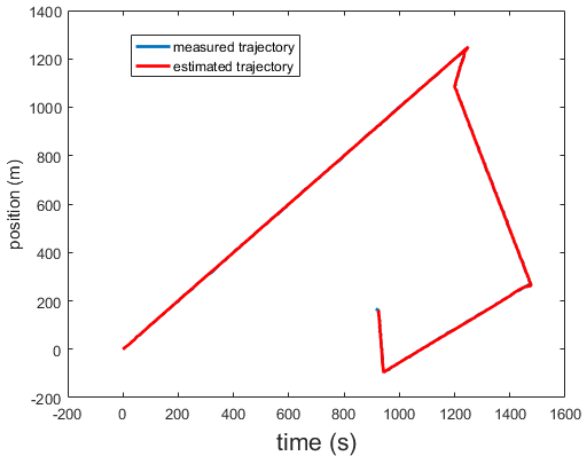
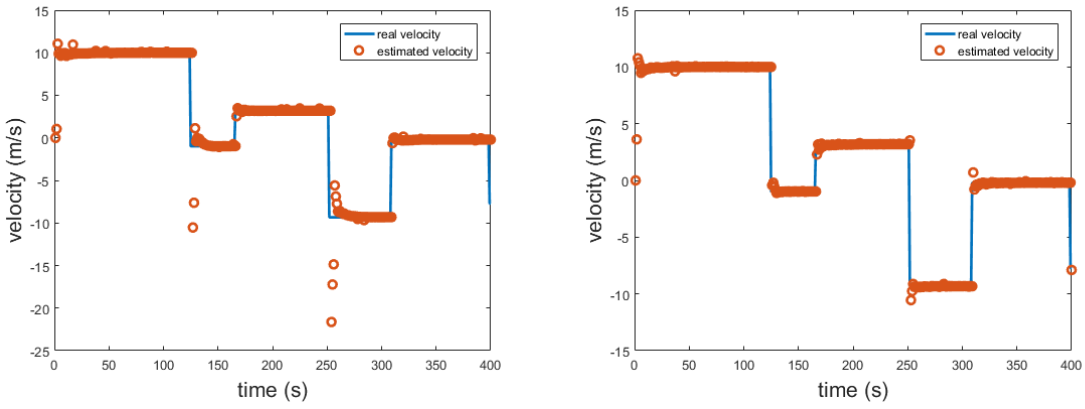


Figure D.3 – Estimated position by a Rao-Blackwell Particle Filter

The same results can be observed with piecewise constant accelerations. Piecewise constant accelerations give smoother trajectories than piecewise constant velocities, but account for less brutal jumps.

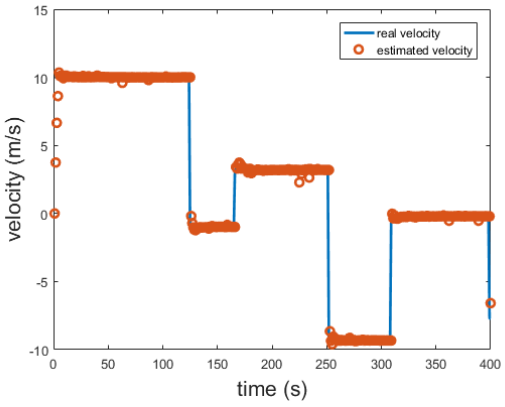
Does this filter converges to the optimal filter ? It is first necessary to specify what "convergence" means. If $p^{\mathrm{N}}(x_n|y_{1:n})$ is the distribution given by the Rao-Blackwell Particle Filter for N particles, then the convergence to the optimal filter means that $p^{\mathrm{N}}(x_n|y_{1:n}) \to p(x_n|y_{1:n})$ when N goes to the infinity. We need to specify in which sense this convergence has to be considered. A common approach is to use the Total Variation norm, see [34], or the Hilbert metric, see [79]. Under fairly large hypotheses, this convergence is easily obtained for fixed n, see [65]. However, what would be really interesting would be to have uniform convergence in time, *i.e.* uniform on n, so that the filter can be used online. This last type of convergence is much more difficult to prove, even if it is often observed in practice, if we look at the beautiful results obtained with particle filters in recent years, see [96].

The convergence that we are mentioning here is thus not the convergence of the industrials, who want to know the convergence time of a filter when initialised far from the true state.



(a)

(b)



(c)

Figure D.4 – Estimated velocities in the x-coordinate for a particle filter with 10 particles fig. D.4a, 100 particles fig. D.4b and 1000 particles fig. D.4c

We did not have time to investigate deeply these aspects, but some results can already be found in the literature, see for example [43], [44] and [34]. This type of convergence is intimately related to the fact that the exponential is memoryless with respect to the initial state of the system.

However, when applied to a non-linear Frenet-Serret model, the convergence properties no longer hold. For the Frenet-Serret model, when confronted to real trajectories, the model noise is imperative. So the Rao-Blackwell particle filter has troubles to detect the jumps, because they can be hidden in process noise.

D.2 Variable Rate Particle Filter

The Variable Rate Particle Filter (VRPF) is extensively described in [59], and we will here recall the basics of this filter. True target trajectories look like deterministic trajectories, with jumps that occur at some times. These jumps may take place at any moment. The VRPF accommodates these jumps in a continuous time setting. Indeed, it computes jumping times, with the hypothesis that these jumps can occur at any moment, unlike the Jump Markov linear models, which consider a discrete-time model where jumps and output measurements are synchronized, which leads to algorithms such as the Rao-Blackwell Particle Filter described in the previous section.

The Variable Rate Particle Filter (VRPF), see [58], [59] and [32], is a particle filter used to track dynamical systems subject to jumps at unknown random times. It is in particular well suited to the case where the target's dynamics are governed by a deterministic differential equation $\frac{d}{dx}x = f(x,\theta)$ whose model parameters θ are fixed between two successive jumps, *i.e.* the dynamics are deterministic between the jumps. The VRPF samples the jumping times and the parameters θ, and evaluates likelihoods based on measurements that occur at times that differ from the jumping times.

To describe the Variable Rate Particle Filter, we apply it directly to our Frenet-Serret target model (2.33).

The state for the VRPF is defined as $\mathbf{x}_k = (\delta_k, \theta_k) \in \mathbb{R}^+ \times \mathrm{E}$. $k \in \mathbb{N}$ is a discrete state time index, $\delta_k > \delta_{k-1} > ... > \delta_0$ denote the state jump times, and θ_k is the vector of parameters necessary to recover the complete target state. We make a Markov assumption and denote $\mathbf{x}_k \sim f(\mathbf{x}_k|\mathbf{x}_{k-1})$. At time t_n, we let $\mathscr{N}_{t_n} = \{k, k-1; \delta_k > t_n \geq \delta_{k-1}\}$ and denote $\mathscr{N}_{t_n}^+ := k$ the maximum of $\mathscr{N}_{t_n}$.

Like standard particle filters, this one works in two steps. In the first step, the particles' states are propagated with the model given to the filter. In the second step, weights are computed for each particle, according to their position with respect to the measurement. The particles can then be resampled according to their weights (the resampling can be done with many different techniques, see for instance [48]). The simplest way to implement the VRPF for our specific model would be to use the pseudo code given in algorithm 5, where N is the number of particles.

With the Frenet-Serret model, we let $\theta_k = \left(\gamma_k, \tau_k, u_k\right)^{\mathrm{T}}$. Indeed, when (γ_k, τ_k, u_t) is known, the trajectory follows a fully specified deterministic equation given by (2.33). Denoting by $\Gamma(\alpha,\beta)$ a Gamma law of parameters α and β, the sampling distributions defining f are assumed to be

$$\begin{aligned} \delta_k - \delta_{k-1} - \delta_{\min} &\sim \Gamma(\alpha,\beta), \delta_{\min} > 0 \\ \gamma_k - \gamma_{k-1} &\sim \mathrm{N}(\gamma_{k-1}, \sigma_\gamma^2) \\ \tau_k - \tau_{k-1} &\sim \mathrm{N}(\tau_{k-1}, \sigma_\tau^2) \\ u_k - u_{k-1} &\sim \mathrm{N}(u_{k-1}, \sigma_u^2). \end{aligned} \tag{D.7}$$

Let also $g(y_n|x_{t_n}) \sim \mathscr{N}(x_{t_n}, \mathrm{N}_n)$ be the measurement density.

This VRPF works fine on deterministic trajectories with random jumps. However, it is computationally demanding, especially when the initial parameters are not known.

Algorithm 5 VRPF to perform estimation at time t

for $i = 1$ to N **do**
  $k = \mathcal{N}_{t_{n-1}}^{+,(i)}$
  while $\delta_k^{(i)} < t_n$ **do**
    $k \leftarrow k+1$
    If $rand < q$ Draw $\mathbf{x}_k^{(i)} \sim f(\mathbf{x}_k|\mathbf{x}_{k-1}^{(i)})$ and integrate (2.33) between times $\delta_{k-1}^{(i)}$ and $\delta_k^{(i)}$.
  end while
  $\mathcal{N}_{t_n}^{+,(i)} = k$
  Add the new states to the particles: $\mathbf{x}_{0:\mathcal{N}_{t_n}^{+,(i)}}^{(i)} = \left(\mathbf{x}_{0:\mathcal{N}_{t_{n-1}}^{+,(i)}}^{(i)}, \mathbf{x}_{\mathcal{N}_{t_{n-1}}^{+,(i)}+1:\mathcal{N}_{t_n}^{+,(i)}}^{(i)}\right)$
  Compute the weights, and then normalize them: $w_{t_n}^{(i)} \sim w_{t_{n-1}}^{(i)} g(y_n|x_{t_n}^{(i)})$, $w_{t_n}^{(i)} := w_{t_n}^{(i)} / \sum_{j=1}^{\mathrm{N}} w_{t_n}^{(j)}$,
  Resample is necessary.
end for

Bibliography

[1] Pieter Abbeel, Adam Coates, Michael Montemerlo, Andrew Y. Ng, and Sebastian Thrun. Discriminative training of kalman filters. In *Robotics: Science and systems*, volume 2, page 1, 2005. 56

[2] Cesare Alippi and Manuel Roveri. An adaptive cusum-based test for signal change detection. In *2006 IEEE International Symposium on Circuits and Systems*, pages 4 pp.–, May 2006. 119

[3] Vladimir I Arnold. Sur la géométrie différentielle des groupes de lie de dimension infinie et ses applicationsa l'hydrodynamique des fluides parfaits. *Ann. Inst. Fourier*, 16(1):319–361, 1966. 28

[4] M. Sanjeev Arulampalam, Simon Maskell, Neil Gordon, and Tim Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188, February 2002. 44

[5] Yaakov Bar-Shalom, Kevin C. Chang, and Henk A. P. Blom. Tracking a maneuvering target using input estimation versus the interacting multiple model algorithm. *Aerospace and Electronic Systems, IEEE Transactions on*, 25(2):296–300, 1989. 39, 46

[6] Yaakov Bar-Shalom, X. Rong Li, and Thiagalingam Kirubarajan. *Estimation with Applications to Tracking and Navigation: Theory Algorithms and Software*. Wiley, 2004. 16, 18, 20, 37, 39, 46, 61

[7] Yaakov Bar-Shalom, X. Rong Li, and Thiagalingam Kirubarajan. *Estimation with Applications to Tracking and Navigation: Theory Algorithms and Software*. Wiley, 2004. 86

[8] Yaakov Bar-Shalom, Peter K Willett, and Xin Tian. *Tracking and data fusion*. YBS publishing, 2011. 36

[9] Axel Barrau. *Non-linear state error based extended Kalman filters with applications to navigation*. Theses, Mines Paristech, September 2015. 43

[10] Axel Barrau and Silvère Bonnabel. Navigating with highly precise odometry and noisy GPS: a case study. In *IFAC Proceedings Volumes*. IEEE, 2016. 22

[11] Axel Barrau and Silvère Bonnabel. The invariant extended kalman filter as a stable observer. *IEEE Transactions on Automatic Control*, 62(4):1797–1812, 2017. 37, 41, 43, 47, 48, 51, 52, 53, 55

[12] Michèle Basseville, Igor V Nikiforov, et al. *Detection of abrupt changes: theory and application*, volume 104. Prentice Hall Englewood Cliffs, 1993. 119

[13] H Benoudnine, M Keche, A Ouamri, and MS Woolfson. Fast adaptive update rate for tracking a maneuvering target with a phased array radar, using imm and mrimm algorithms. *Journal of Applied Science*, 9(2), 2009. 119

[14] Dieter Bestle and Michael Zeitz. Canonical form observer design for non-linear time-variable systems. *International Journal of control*, 38(2):419–431, 1983. 31

[15] Robert H. Bishop. *Geometric nonlinear filtering theory with application to the maneuvering aircraft tracking problem*. PhD thesis, Rice University, 1990. 41

[16] Robert H. Bishop and A.C. Antoulas. Nonlinear approach to aircraft tracking problem. *Journal of Guidance, Control, and Dynamics*, 17(5):1124–1130, 1994. 15, 31, 41

[17] Yves Blanchard. *Le radar, 1904-2004: Histoire d'un siècle d'innovations techniques et opérationnelles*. Ellipses, 2004. 3

[18] Henk AP Blom and Yaakov Bar-Shalom. The interacting multiple model algorithm for systems with markovian switching coefficients. *IEEE transactions on Automatic Control*, 33(8):780–783, 1988. 23

[19] Yvo Boers and Johannes N. Driessen. Interacting multiple model particle filter. *IEE Proceedings-Radar, Sonar and Navigation*, 150(5):344–349, 2003. 61

[20] Philip L. Bogler. Tracking a maneuvering target using input estimation. *IEEE transactions on Aerospace and Electronic Systems*, (3):298–310, 1987. 17

[21] Silvère Bonnabel, Philippe Martin, and Pierre Rouchon. Symmetry-preserving observers. *IEEE Transactions on Automatic Control*, 53(11):2514–2526, 2008. 52

[22] Silvère Bonnabel. Left-invariant extended kalman filter and attitude estimation. In *Decision and Control, 46th IEEE Conference on*, pages 1027–1032. IEEE, 2007. 37

[23] Silvère Bonnabel, P. Martin, and P. Rouchon. Non-linear observer on lie groups for left-invariant dynamics with right-left equivariant output. In *IFAC World Congress*, pages 8594–8598, 2008. 52

[24] Silvère Bonnabel, P. Martin, and E. Salaün. Invariant extended Kalman filter: theory and application to a velocity-aided attitude estimation problem. In *Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. CDC/CCC 2009. Proceedings of the 48th IEEE Conference on*, pages 1297–1304. IEEE, 2009. 37

[25] Jérémie Boulanger, Salem Said, Nicolas Le Bihan, and Jonathan H Manton. Filtering from observations on stiefel manifolds. *Signal Processing*, 122:52–64, 2016. 44

[26] Guillaume Bourmaud, Rémi Mégret, Marc Arnaudon, and Audrey Giremus. Continuous-discrete extended Kalman filter on matrix lie groups using concentrated gaussian distributions. *Journal of Mathematical Imaging and Vision*, 51(1):209–228, 2014. 37, 44

[27] Guillaume Bourmaud, Rémi Megret, Audrey Giremus, and Yannick Berthoumieu. Discrete extended Kalman filter on lie groups. In *Signal Processing Conference (EUSIPCO), 2013 Proceedings of the 21st European*, pages 1–5, September 2013. 37, 44

[28] Yann Briheche, Frederic Barbaresco, Fouad Bennis, and Damien Chablat. Update rates constraints in fixed-panel radar search pattern optimization with limited time budget. In *Radar Symposium (IRS), 2017 18th International*, pages 1–10. IEEE, 2017. 110

[29] Yann Briheche, Frederic Barbaresco, Fouad Bennis, Damien Chablat, and François Gosselin. Non-uniform constrained optimization of radar search patterns in direction cosines space using integer programming. In *Radar Symposium (IRS), 2016 17th International*, pages 1–6. IEEE, 2016. 110

[30] Martin Brossard, Silvère Bonnabel, and Jean-Philippe Condomines. Unscented kalman filtering on lie groups. In *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, pages 2485–2491. IEEE, 2017. 61, 62

[31] TE Bullock and S Sangsuk-Iam. Maneuver detection and tracking with a nonlinear target model. In *Decision and Control, 1984. The 23rd IEEE Conference on*, volume 23, pages 1122–1126. IEEE, 1984. 15

[32] Pete Bunch and Simon Godsill. Dynamical models for tracking with the variable rate particle filter. In *Information Fusion (FUSION), 2012 15th International Conference on*, pages 1769–1775. IEEE, 2012. 15, 21, 46, 86, 91, 145, 148

[33] Pete Bunch and Simon Godsill. Particle smoothing algorithms for variable rate models. *IEEE Transactions on Signal Processing*, 61(7):1663–1675, 2013. 86, 91

[34] Olivier Cappé, Eric Moulines, and Tobias Rydén. Inference in hidden markov models, 2009. 21, 146, 148

[35] Frank R. Castella. An adaptive two-dimensional Kalman tracking filter. *IEEE Transactions on Aerospace and Electronic Systems*, (6):822–829, 1980. 23, 61, 69

[36] Josip Cesic, Ivan Markovic, and Ivan Petrovic. Moving object tracking employing rigid body motion on matrix lie groups. *CoRR*, abs/1708.05548, 2017. 22

[37] Raja Chatila and Jean-Paul Laumond. Position referencing and consistent world modeling for mobile robots. In *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*, volume 2, pages 138–145. IEEE, 1985. 87

[38] Daniel Clark. Bayesian filtering for multi-object systems with independently generated observations. 02 2012. 37

[39] S. A. Cohen. Adaptive variable update rate algorithm for tracking targets with a phased array radar. *Communications, Radar and Signal Processing, IEE Proceedings F*, 133(3):277–280, June 1986. 119

[40] Richard G. Curry. *Radar System Performance Modeling*. Number vol. 1 in Artech House radar library. Artech House, 2005. 5, 26

[41] Jacques Darricau. *Physique et théorie du radar: principes et éléments de base*. Ed. Deniaud, 2002. 3

[42] Giorgio de Moura Magalhães, Eloi Dranka, Yusef Cáceres, João B. R. do Val, and Rafael S. Mendes. Ekf on lie groups for radar tracking using polar and doppler measurements. In *2018 IEEE Radar Conference (RadarConf18)*, pages 1573–1578, April 2018. 22, 27, 44

[43] Pierre Del Moral. Non-linear filtering: interacting particle resolution. *Markov processes and related fields*, 2(4):555–581, 1996. 148

[44] Pierre Del Moral and Alice Guionnet. On the stability of interacting processes with applications to filtering and genetic algorithms. In *Annales de l'IHP Probabilités et statistiques*, volume 37, pages 155–194, 2001. 148

[45] Daniel Delahaye, Stéphane Puechmorel, and Loïc Boussouf. Trajectory prediction: a functional regression approach. In *ICRAT 2008, International Conference on Research in Air Transportation*, pages pp–xxxx, 2008. 81

[46] Frank Dellaert and Michael Kaess. Square root sam: Simultaneous localization and mapping via square root information smoothing. *The International Journal of Robotics Research*, 25(12):1181–1203, 2006. 87

[47] Tue-Cuong Dong-Si and Anastasios I Mourikis. Motion tracking with fixed-lag smoothing: Algorithm and consistency analysis. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 5655–5662. IEEE, 2011. 89

[48] Randal Douc and Olivier Cappé. Comparison of resampling schemes for particle filtering. In *ISPA 2005. Proceedings of the 4th International Symposium on Image and Signal Processing and Analysis, 2005.*, pages 64–69. IEEE, 2005. 44, 148

[49] Arnaud Doucet, Nando De Freitas, Kevin Murphy, and Stuart Russell. Rao-blackwellised particle filtering for dynamic bayesian networks. In *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence*, pages 176–183. Morgan Kaufmann Publishers Inc., 2000. 45, 86, 143

[50] Arnaud Doucet, Neil J Gordon, and Vikram Krishnamurthy. Particle filters for state estimation of jump Markov linear systems. *Signal Processing, IEEE Transactions on*, 49(3):613–624, 2001. 21, 37, 44

[51] Arnaud Doucet and Adam M Johansen. A tutorial on particle filtering and smoothing: Fifteen years later. *Handbook of nonlinear filtering*, 12(656-704):3, 2009. 44

[52] Nour Elhouda Dougui, Daniel Delahaye, Marcel Mongeau, and Stéphane Puechmorel. Aircraft trajectory planning under uncertainty by light propagation. *Procedia-Social and Behavioral Sciences*, 54:201–210, 2012. 81

[53] Zhansheng Duan, X Rong Li, Chongzhao Han, and Hongyan Zhu. Sequential unscented kalman filter for radar target tracking with range rate measurements. In *Information Fusion, 2005 8th International Conference on*, volume 1, pages 8–pp. IEEE, 2005. 27

[54] Johannes Jisse Duistermaat and Johan AC Kolk. *Lie groups*. Springer Science & Business Media, 2012. 28

[55] Ryan M Eustice, Hanumant Singh, and John J Leonard. Exactly sparse delayed-state filters. 2005. 87

[56] James Richard Forbes, Anton HJ de Ruiter, and David Evan Zlotnik. Continuous-time norm-constrained kalman filtering. *Automatica*, 50(10):2546–2554, 2014. 37

[57] Crispin Gardiner. *Stochastic methods*, volume 4. springer Berlin, 2009. 19

[58] Simon Godsill and Jaco Vermaak. Variable rate particle filters for tracking applications. In *IEEE/SP 13th Workshop on Statistical Signal Processing, 2005*, pages 1280–1285. IEEE, 2005. 21, 91, 148

[59] Simon J Godsill, Jaco Vermaak, William Ng, and Jack F Li. Models and algorithms for tracking of maneuvering objects using variable rate particle filters. *Proceedings of the IEEE*, 95(5):925–952, 2007. 20, 21, 74, 148

[60] Maria S Greco, Fulvio Gini, Pietro Stinco, and Kristine Bell. Cognitive radars: A reality? *arXiv preprint arXiv:1803.01000*, 2018. 110

[61] Frederik Gustafsson, Fredrik Gunnarsson, Niclas Bergman, Urban Forssell, Jonas Jansson, Rickard Karlsson, and P.-J. Nordlund. Particle filters for positioning, navigation, and tracking. *IEEE Transactions on signal processing*, 50(2):425–437, 2002. 44

[62] Fredrik Gustafsson. Particle filter theory and practice with positioning applications. *IEEE Aerospace and Electronic Systems Magazine*, 25(7):53–82, 2010. 44

[63] Simon Haykin. Cognitive radar: a way of the future. *IEEE Signal Processing Magazine*, 23(1):30–40, Jan 2006. 110

[64] Xiao-Li Hu, Thomas Schön, and Lennart Ljung. A basic convergence result for particle filtering. *IEEE Transactions on Signal Processing*, 56(4):1337–1348, 2008. 45

[65] Xiao-Li Hu, Thomas Schön, and Lennart Ljung. A general convergence result for particle filtering. *IEEE Transactions on Signal Processing*, 59(7):3424–3429, 2011. 146

[66] Simon J. Julier and Jeffrey K. Uhlmann. New extension of the kalman filter to nonlinear systems. In *AeroSense'97*, pages 182–193. International Society for Optics and Photonics, 1997. 41, 42

[67] Simon J. Julier and Jeffrey K. Uhlmann. Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92(3):401–422, 2004. 41, 42

[68] Michael Kaess, Hordur Johannsson, Richard Roberts, Viorela Ila, John J Leonard, and Frank Dellaert. isam2: Incremental smoothing and mapping using the bayes tree. *The International Journal of Robotics Research*, 31(2):216–235, 2012. 87

[69] Michael Kaess, Ananth Ranganathan, and Frank Dellaert. isam: Incremental smoothing and mapping. *IEEE Transactions on Robotics*, 24(6):1365–1378, Dec 2008. 87, 89

[70] Rudolf E. Kalman and Richard S. Bucy. New results in linear filtering and prediction theory. *J. Basic Eng.*, 1961. 111

[71] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960. 38

[72] Bernt M. Åkesson, John Bagterp Jørgensen, Niels Kjølstad Poulsen, and Sten Bay Jørgensen. A tool for kalman filter tuning. In Valentin Pleşu and Paul Şerban Agachi, editors, *17th European Symposium on Computer Aided Process Engineering*, volume 24 of *Computer Aided Chemical Engineering*, pages 859 – 864. Elsevier, 2007. 123

[73] Alireza Khosravian, Jochen Trumpf, Robert Mahony, and Christian Lageman. Observers for invariant systems on lie groups with biased input measurements and homogeneous outputs. *Automatica*, 55:19–26, 2015. 52

[74] Mikhail Koldychev and Christopher Nielsen. Local observers on linear lie groups with linear estimation error dynamics. *IEEE Transactions on Automatic Control*, 59(10):2772–2777, 2014. 52

[75] Kurt Konolige. Large-scale map-making. In *AAAI*, pages 457–463, 2004. 87

[76] Jonathan Korn, Sol W. Gully, and Alan S. Willsky. Application of the generalized likelihood ratio algorithm to maneuver detection and estimation. In *1982 American Control Conference*, pages 792–798, June 1982. 119

[77] Tony Lacey. Tutorial: The kalman filter. *Georgia Institute of Technology*. 139

[78] Nicolas Le Bihan. The geometry of proper quaternion random variables. *Signal Processing*, 138:106–116, 2017. 73

[79] François Le Gland, Nadia Oudjane, et al. Stability and uniform approximation of nonlinear filters using the Hilbert metric and application to particle filters. *The Annals of Applied Probability*, 14(1):144–187, 2004. 146

[80] X Rong Li and Vesselin P Jilkov. Survey of maneuvering target tracking. part i. dynamic models. *IEEE Transactions on Aerospace and Electronic Systems*, 39(4):1333–1364, October 2003. 16

[81] X Rong Li and Vesselin P Jilkov. Survey of maneuvering target tracking. part v. multiple-model methods. *IEEE Transactions on Aerospace and Electronic Systems*, 41(4):1255–1321, October 2005. 40

[82] David G Luenberger. Observing the state of a linear system. *IEEE transactions on military electronics*, 8(2):74–80, 1964. 31

[83] Ronald Mahler. Phd filters of higher order in target number. *IEEE Transactions on Aerospace and Electronic Systems*, 43(4), 2007. 37

[84] Philippe Martin and Erwan Salaün. Generalized multiplicative extended kalman filter for aided attitude and heading reference system. In *AIAA Guidance, Navigation, and Control Conference*, page 8300, 2010. 37

[85] Peter Matisko and Vladimír Havlena. Noise covariances estimation for kalman filter tuning. *IFAC Proceedings Volumes*, 43(10):31 – 36, 2010. 10th IFAC Workshop on the Adaptation and Learning in Control and Signal Processing. 123

[86] Michael Montemerlo, Sebastian Thrun, Daphne Koller, Ben Wegbreit, et al. Fastslam: A factored solution to the simultaneous localization and mapping problem. In *Aaai/iaai*, pages 593–598, 2002. 87

[87] Mark R. Morelande and Neil J. Gordon. Target tracking through a coordinated turn. In *Proceedings. (ICASSP '05). IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005.*, volume 4, pages iv/21–iv/24 Vol. 4, March 2005. 20, 86

[88] Nassib Nabaa and Robert H. Bishop. Validation and comparison of coordinated turn aircraft maneuver models. *IEEE Transactions on aerospace and electronic systems*, 36(1):250–259, 2000. 31

[89] Bernt Øksendal. Stochastic differential equations. In *Stochastic differential equations*, pages 65–84. Springer, 2003. 17, 37

[90] Marion Pilté, Silvere Bonnabel, and Frederic Barbaresco. Fully-adaptive update rate for nonlinear trackers. *IET Radar, Sonar & Navigation*, 2018. 9

[91] Marion Pilté, Silvère Bonnabel, and Frédéric Barbaresco. Tracking the frenet-serret frame associated to a highly maneuvering target in 3d. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 1969–1974, Dec 2017. 9, 23

[92] Marion Pilté and Frédéric Barbaresco. Tracking quality monitoring based on information geometry and geodesic shooting. In *Radar Symposium (IRS), 2016 17th International*, pages 1–6. IEEE, 2016. 119

[93] Marion Pilté, Silvere Bonnabel, and Frédéric Barbaresco. Drone tracking using an innovative ukf. In *International Conference on Geometric Science of Information*, pages 301–309. Springer, 2017. 9

[94] Marion Pilté, Silvère Bonnabel, and Frédéric Barbaresco. An innovative nonlinear filter for radar kinematic estimation of maneuvering targets in 2d. In *Radar Symposium (IRS), 2017 18th International*, pages 1–10. IEEE, 2017. 8

[95] Marion Pilté, Silvère Bonnabel, and Frédéric Barbaresco. Maneuver detector for active tracking update rate adaptation. In *2018 19th International Radar Symposium (IRS)*, pages 1–10. IEEE, 2018. 9

[96] Patrick Rebeschini, Ramon Van Handel, et al. Can local particle filters beat the curse of dimensionality? *The Annals of Applied Probability*, 25(5):2809–2866, 2015. 146

[97] Konrad Reif, Frank Sonnemann, and Rolf Unbehauen. An EKF-based nonlinear observer with a prescribed degree of stability. *Automatica*, 34(9):1119–1123, 1998. 68

[98] Branko Risti, Daniel Clark, and Ba-Ngu Vo. Improved smc implementation of the phd filter, 08 2010. 37

[99] Peter Sarunic. Adaptive update rate target tracking for a phased array radar. Master's thesis, University of South Australia, 1995. 119

[100] H. J. Shin, S. M. Hong, and D. H. Hong. Adaptive-update-rate target tracking for phased-array radar. *IEE Proceedings - Radar, Sonar and Navigation*, 142(3):137–143, Jun 1995. 119

[101] Gabe Sibley, Larry Matthies, and Gaurav Sukhatme. Sliding window filter with application to planetary landing. *Journal of Field Robotics*, 27(5):587–608, 2010. 89

[102] Iliyana Simeonova and Tzvetan Semerdjiev. Specific features of imm tracking filter design. *Information and Security*, 9:154–165, 2002. 24, 25

[103] Robert A Singer. Estimating optimal tracking filter performance for manned maneuvering targets. *IEEE Transactions on Aerospace and Electronic Systems*, (4):473–483, 1970. 19, 27, 31, 118

[104] George M Siouris. *Missile guidance and control systems*. Springer Science & Business Media, 2004. 14

[105] Merrill Ivan Skolnik. Radar handbook. 1970. 3

[106] Gabriel A Terejanu. Unscented kalman filter tutorial. *University at Buffalo, Buffalo*, 2011. 42

[107] Sebastian Thrun and Michael Montemerlo. The graph slam algorithm with applications to large-scale mapping of urban structures. *The International Journal of Robotics Research*, 25(5-6):403–429, 2006. 87

[108] Guenter van Keuk and Samuel S Blackman. On phased-array radar tracking and parameter control. *IEEE Transactions on Aerospace Electronic Systems*, 29:186–194, 1993. 10, 111, 112, 113, 114, 118, 120, 129

[109] Eric A. Wan and Rudolph 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, 2000. 42

[110] Gregory A. Watson and W. Dale Blair. Tracking performance of a phased array radar with revisit time controlled using the imm algorithm. In *Proceedings of 1994 IEEE National Radar Conference*, pages 160–165, Mar 1994. 119

Résumé

Les nouvelles générations de radars sont confrontées à des cibles de plus en plus menaçantes. Ces radars doivent effectuer plusieurs tâches en parallèle, dont la veille et la poursuite. Pour cela, ils peuvent être équipés de panneaux fixes, pour éviter les contraintes liées à la rotation de l'antenne. Le pistage du radar doit donc être renouvelé pour répondre à la double difficulté posée par le pistage des cibles très manoeuvrantes et la gestion des ressources.

Dans ce contexte, cette thèse étudie de nouvelles méthodes de pistage pour les cibles hyper-manoeuvrantes. Un nouveau modèle de cible, en coordonnées intrinsèques, est proposé. Ce modèle est exprimé directement dans le repère de la cible, afin de décrire au mieux des manoeuvres fortes avec des accélérations normales bien supérieures à la gravité terrestre. Un algorithme de filtrage utilisant la formulation intrinsèque du modèle est développé. Cet algorithme ayant la même structure qu'une filtre de Kalman étendu, il a été testé sur de vraies données. La comparaison avec d'autres algorithmes de filtrage a montré de réelles améliorations sur un ensemble important de trajectoires. Une nouvelle méthode d'estimation, reposant sur la formulation en termes de moindres carrés de l'approche de lissage, et permettant de tenir compte de sauts dans la trajectoire est également proposée, et les bénéfices sur des méthodes plus classiques de sauts entre modèles sont montrés. Indépendamment, le problème de cadence adaptative est également traité. Un algorithme très général permettant d'optimiser la cadence de mesure pour ménager le budget temps du radar pour la surveillance est présenté.

Mots clefs : Estimation, pistage de cibles, filtre de Kalman, groupes de Lie, cadence adaptative

Abstract

The new generation of radars is facing increasingly threatening targets. These radars are asked to perform several tasks in parallel, including surveillance and tracking. To this aim, they can be equipped with staring antennas, so they overcome the constraints induced by the rotation of the antenna. The tracking function of the radar has thus to be upgraded to respond to the double issue of tracking highly manoeuvring targets and managing the resources to balance time between tasks.

In this context, this thesis investigates new means of tracking highly manoeuvring targets. A new target model based on intrinsic coordinates to perform target tracking is proposed. This new target model is expressed in the frame of the target itself, and uses the Frenet-Serret frame, which is well suited to the description of highly dynamic manoeuvres involving normal accelerations that are much larger than earth gravity. A filtering algorithm using the special intrinsic formulation of the target model is developed. This filtering algorithm is very similar in terms of implementation to an Extended Kalman filter, and was implemented using real data. The comparison with standard target models and filtering algorithms show improvements over simple models and algorithms on a large set of trajectories. A new estimation method, relying on the least squares formulation of the smoothing approach, and taking into account kinematic jumps in the trajectory is also developed. This method also shows improvements over a set of common algorithms based on standard manoeuvre detection. Independently, we investigate the issue of update rate adaptation for radar measurements. A very general update rate adaptation algorithm is derived to optimise the time of revisit of each target, allowing to preserve the radar time budget for other tasks simultaneously performed, such as surveillance.

Keywords: State estimation, Target tracking, Kalman filter, Lie groups, Update rate adaptation