Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

Magne Sirnes

Joint localization and tracking

a per-track error state approach

Master's thesis in Cybernetics and Robotics
Supervisor: Edmund Førland Brekke
Co-supervisor: Erik Wilthil
June 2021

NTNU
Norwegian University of
Science and Technology

Magne Sirnes

Joint localization and tracking

a per-track error state approach

Master's thesis in Cybernetics and Robotics
Supervisor: Edmund Førland Brekke
Co-supervisor: Erik Wilthil
June 2021

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

NTNU

Kunnskap for en bedre verden

Abstract

A novel solution (ES-JLAT) for joint localization and tracking (JLAT) using error states (ES) is developed and tested. The filtering performance of this solution is tested in a number of Monte Carlo simulations, and compared to the performance of an already established solution: an extended Kalman filter (EKF) comprised of both ownship and target states.

ES-JLAT is defined in terms of several tracking filters, each interacting with a localization filter. The contribution for this thesis stands as the development of tracking filters which can infer corrections on the localization estimate based on positional measurements of targets, as well as a method for condensing an arbitrary number of these corrections into a single correction, and finally two methods for injecting such a correction into the localization filter.

In the simulations, the localization objective is twofold, first is estimation of an ownship's position and velocity, generated using a constant velocity (CV) model. Second is estimation of its heading, generated as a random walk process. Note that velocity and heading are decoupled, to emulate the motion range of a fully actuated vessel. Furthermore, the tracking objective is estimation of the position and velocity of two independent targets, also generated using CV models. Either solution is provided with positional measurements of all vessels (fully associated), in a global coordinate system, as well as range-bearing measurements of the relevant targets, in a body (ownship) coordinate system. Crucially, no direct observations of the ownship heading are provided, so either solution has to infer heading from the range-bearing measurements in order to fulfill their localization objective.

The performance of both solutions is quantified as ensemble averaged timeseries' of estimation error (EE) and normalized estimation error squared (NEES) over multiple simulations, as well as root-mean-square (RMSE) and average NEES (ANEES) values for individual simulations. Resulting metrics are presented for a number of different configurations of ES-JLAT, as well as a 'baseline' configuration of ES-JLAT without error state injection, which effectively is a CV-filter.

After some design revision, the final variation of ES-JLAT presented was found to have localization performance comparable to that of EKF, with marginally worse estimation accuracy for position and velocity, and significantly more accurate heading estimation. For estimation of target states, the performance was again comparable, being marginally less accurate for position and velocity. As for consistency, the localization estimates were found to be consistent, while the target estimates were mostly overconfident, finally the EKF was under-confident on all counts.

Sammendrag

En ny løsning (error state joint localization and tracking: ES-JLAT) for felles navigasjon og sporing ved hjelp av feiltilstander er utviklet og testet. Filtreringsytelsen til denne løsningen er testet i en rekke Monte Carlo simuleringer, og sammenlignet med ytelsen til en tilsvarende, etablert løsning: et utvidet Kalman-filter (EKF) med både eget skip og sporingsmål i sin tilstandsvektor.

ES-JLAT er definert som mange sporingsfilter, der alle interagerer med et felles navigasjonsfilter. Bidragene i denne oppgaven er utviklingen av sporingsfilter som kan produsere korreksjoner på navigasjonsestimatet, en metode for å sammenfatte et vilkårlig antall av disse korreksjonene ned til en enkelt *konsensus-korreksjon*, og til slutt to metoder for å injisere denne korreksjonen inn i navigasjonsfilteret.

I simuleringene består navigasjon av to oppgaver, først er estimering av posisjon og hastighet for eget skip, som er generert vha. en konstant hastighet (CV) modell. Den andre oppgaven er å estimere skipets kurs, denne er generert med en tilfeldig gange (random walk) modell. Merk at skipets hastighet og kurs dermed er uavhengige, for å emulere et fullt aktuert skip. For sporing er målet å estimere posisjon og hastighet til to uavhengige mål, igjen generert med en CV modell. Begge løsningene blir tilført målinger i form av posisjonene til alle skip (både eget skip og begge sporingsmål) som er fullstendig assosiert, og gitt i et globalt koordinatsystem. I tillegg blir begge løsninger tilført avstand-peiling (range-bearing) målinger til begge sporingsmål, gitt i et lokalt koordinatsystem (altså relativt til eget skip). Merk at ingen målinger av kursen til eget skip er tilgjengelige, dermed kan kursen bare estimeres vha. de relative avstand-peiling målingene.

Ytelsen til begge løsninger er kvantisert med ensemble-gjennomsnitt tidserier av estimeringsfeil (EE) og normalisert, kvadrert estimeringsfeil (NEES), snittet over mange simuleringer. I tillegg er RMS-feil og gjennomsnittlig NEES (ANEES) regnet ut for hver enkelt simulering. Alle disse metrikkene er presentert for en rekke ulike konfigurasjoner av ES-JLAT, i tillegg til en 'standard-konfigurasjon' uten noen feilestimat-injeksjon, som tilsvarer et konvensjonelt CV-filter.

Etter noe tilpasning av ES-JLAT ble resulater for den siste varianten presentert og sammenlignet med EKF. Disse navigasjonsresultatene viste seg å være marginalt mindre presise for posisjon og hastighet, men markant mer presise for kurs. For sporingsmålene var resultatene også marginalt mindre presise, men fortsatt konkurransedyktige. Når det gjelder troverdighet (filter consistency) til estimatene, viste det seg at navigasjonsestimatene var helt troverdige, mens sporingsestimatene var stort sett for selvsikre. For EKFs estimater, var samtlige for lite selvsikre.

Preface

This project is in certain aspects a successor to a similar project undertaken and documented during the fall and winter of 2020, resulting in a unpublished project thesis. Certain sections of this report (introduction, literature review, theory - notation) will re-use some sections of this text, with varying degrees of paraphrasing. The relevant sections of text, and their degree of paraphrasing will be made abundantly clear in these chapters.

With that being said, although the two theses share an overarching theme of joint localization and tracking using onboard and onshore sensors, they differ in focus areas: previously gathering and scrutinizing experimental data; now suggesting and testing a novel filter structure. For this reason I hesitate to call this project a true continuation of the previous project. By extension, it is only in the highly abstracted chapters which I deem the contents of the previous report to be relevant for this one.

Huge thanks go out to both my supervisor Edmund Brekke, and co-supervisor Erik Wilthil, whom have been exceedingly helpful with brainstorming, feedback on results and the thesis, and overall guidance on this project. I am especially grateful for the opportunity to pursue this new and weird solution for joint localization and tracking, despite it being completely different from the approach we had planned at the start of this project, and for the patience in helping figure the kinks during the infant stages of this project.

Finally, my thanks to Gustav Omberg, for some much needed feedback on the writing, as well as my gratitude and appreciation for the remainder of my friends in the cybernetics & robotics class of 21. It's been a weird conclusion to our studies, but we've made the best of it.

Table of Contents

List of Tables

List of Figures

Abbreviations

EKF	=	Extended Kalman filter
ESCV	=	Error state constant velocity (motion model, see chapter 5)
ESCV-IC	=	Inherited covariance ESCV (motion model, see chapter 5)
ESKF	=	Error state Kalman filter
FPSKF	=	Fixed-weight partial Schmidt-Kalman filter
GNSS	=	Global navigation satellite system
IPDA	=	Integrated probabilistic data association
ISKF	=	Intermittent Schmidt-Kalman filter
JIPDA	=	Joint integrated probabilistic data association
JLAT	=	Joint localization and tracking
JPDAF	=	Joint probabilistic data association filter
KF	=	Kalman filter
MTT	=	Multi target tracking
PDAF	=	Probabilistic data association filter
PSKF	=	Partial Schmidt-Kalman filter
RESCV	=	Reduced error state constant velocity (motion model, see chapter 5)
RESCV-IC	=	Inherited covariance RESCV (motion model, see chapter 5)
RTK	=	Real-time kinematic (navigation)
SLAM	=	Simultaneous localization and mapping
SKF	=	Schmidt-Kalman filter (i.e. consider Kalman filter)

Chapter 1

Introduction

A disclaimer:
This chapter shares certain paragraphs with its namesake in an unpublished project thesis [1], written during the fall and winter of 2020, by myself, Magne Sirnes. Specifically, the immediately following paragraph, as well as the first two paragraphs under 'Motivation' are slightly paraphrased from this project thesis, and thus should not be considered part of the 'new effort' for the master thesis.

In the pursuit of increased autonomy, precise, fault tolerant absolute navigation and situational awareness are prerequisites to avoiding accidents. In chaotic, urban environments, situational awareness is particularly challenging, for example because potential obstacles can be obscured for the autonomous vehicle until near-collision situations, and thus present themselves too late for traditional tracking methods. By introducing additional onshore sensors which have a different perspective, one can detect otherwise obscured obstacles, and let the autonomous vehicle know about them earlier than it otherwise would. Such onshore sensors would also at times be able to detect the autonomous vehicle, which can aid with the navigation of the vehicle, especially if global navigation satellite system (GNSS) is unavailable or unreliable, as it is liable to be in urban environments. Finally, for obstacles and features which are detected by both onboard and onshore sensors, it should be possible to cross reference these features. This can provide more precise information about the features, but more importantly also infer information on the pose of the autonomous vehicle. If onboard inertial sensors fail, it is also interesting to see whether passable localization and tracking is achievable when only using vessel-detecting onboard and onshore sensors.

1.1 Motivation

In much of the literature, localization and tracking have been abstracted into separate problems, which allows for cleaner modularization, at the cost of disregarding knowledge about the inherently linked nature of these problems. While some solutions to this problem have been attempted, such solutions have typically been limited to relative motion navigation and scan-matching techniques, using only body mounted sensors which have strictly coupled movement with the navigational object. This limits possible sensors, requires substantial feature density in the surroundings, and is finally computationally expensive. This leaves a substantial hole in the literature, in terms of such methods that are more adaptible to various sensor types.

Further expanding such methods to also include external sensors, for example for detecting the ownship, detecting other objectives of tracking and the surroundings (e.g. features for simultaneous localization and mapping, SLAM) is similarly untouched in the literature.

In unpublished project theses (Kjønås; Sirnes, 2020), experimental data using both onshore and onboard sensors was gathered. These experiments consisted of multiple scenarios in which the ownship, milliAmpere, as well as two target vessels maneuvered in a cluttered harbor environment. This data was gathered for the sake of testing detection pipelines, challenging multi target tracking, but more importantly for eventually testing *joint localization and tracking* (JLAT) schemes.

The motivation for this thesis is to develop a JLAT scheme which should be able to achieve reasonable performance in such scenarios, thus demonstrating the potential of combining onboard and onshore sensors for joint localization and tracking in constrained (no GNSS, compass or inertial measurement unit) conditions.

1.2 Scope

In this thesis, a JLAT scheme is developed as a combination of a localization filter and multiple tracking filters, which interact with each other. This constitutes JLAT in the sense that the tracking filters will use the localization estimates in order to process onboard sensor measurements, after which they provide corrections to the localization estimate. These corrections leverage the inherent connection between the ownship pose and the target positions which is present in onboard sensor data.

For the tracking filters (which now also need to produce corrections for the localization estimate), a 'compound' filter structure is developed. This filter structure leverages a typical extended Kalman filter (EKF) for target state estimation, which is combined with a variation of an error state Kalman filter (ESKF) for the localization correction estimation. This effectively makes a *compound filter* in which EKF and ESKF are combined. A significant part of the effort undertaken in this thesis consists of formulating this compound

filter, and especially deciding how the error state component of this filter should interact with the localization filter.

With the new compound filter in hand, which is capable of harnessing *all* the information present in onboard (relative) scans, we need to develop motion and measurement models which fit this filter structure. A second significant part of the effort in this project is to develop a few such preliminary models, and after some testing adapt these models into their final forms. From a JLAT prespective, these models are crucial in the sense that they need to fulfill all the usual criterion of target modelling, while at the same time internalizing the connection between target and ownship states, which is needed for leveraging onboard scans to the fullest.

The third and final part of development of this new JLAT scheme, is to bridge the gap between a single compound filter interacting with a localization filter and a lots of independent compound filters (one for each target) all interacting with the same localization filter. In this thesis, a solution for this is suggested as a 'consensus model'. This consensus model is capable of consuming error state estimates from every compound filter, and using these calculate a single consensus estimate. This consensus estimate can then be used with the same tools that were developed for the single compound filter case, thus generalizing the JLAT scheme for an arbitrary number of targets.

In order to investigate the performance of this scheme, a secondary goal for this thesis becomes to implement a simulator which can run Monte Carlo simulations using this solution and calculate important metrics for both accuracy and consistency. This simulation is built to mimic the conditions of the experiments done during the fall of 2020, and thus features a (fully actuated) ownship, which is capable of linear motion decoupled from rotation, as well as two targets with independent motion.

Finally, to have some established solution with which to compare the results of our newly developed scheme, a tertiary goal of this thesis becomes to select and implement such a solution, in this case a full state EKF (filtering both ownship and targets states in the same filter). Which will use similar motion and measurement models of those of our new solution, and thus provide decent grounds for comparison. This 'full state EKF' solution is of course also implemented in the aforementioned simulator, with the same performance metrics.

The primary goal of this thesis thus becomes to develop, implement and test the filtering performance of the new JLAT solution, by using the developed simulator and comparing its results to those of the full state EKF solution. A secondary goal becomes to analyse the results of our new solution, and pinpoint any problems or shortcomings which need fixing. Fixing these problems will be done during the project as part of the development, but also as an effort for pinpointing and documenting all the problems which were not (usually for lack of time or skill) fixed in development.

1.3 Structure of thesis

This thesis is structured in the following manner:

First, a brief summary of some relevant literature is provided.

Next, a number of theory chapters are included, starting with an introduction to filtering, which leads into the formulation of the 'compound' (nominal and error state) filter. Next, a number of motion and measurement models are defined, for ordinary (KF/EKF) filters, as well as the new 'compound' filter. Following this, brief introductions to localization and IPDA tracking are provided. These are both written in context with the previously defined motion and measurement models, as well as extended to fit the suggested 'compound' filter. Finally, a chapter on joint localization and tracking is inlcuded. This chapter describes the high level system in terms of interacting modules, an algorithmic example as well as the functionality needed to combine the results of multiple tracking filters into a single consensus. Furthermore, this chapter also defines the alternative full state EKF JLAT solution, as a baseline for comparison.

Following the theory chapters, a simulation chapter is included. This chapter describes the design and implementation of the simulator, as well as the various performance metrics which the simulator calculates.

Finally, the results produced by the simulator, for both solutions, are presented and discussed in an analysis chapter. This is divided into bulks of results and discussion for every configuration of the proposed solution, with a focus on comparing its results with those of the full state solution with similar information available. These results are presented in a 'observe problem and propose solution' narrative, in this sense their order should reflect the workflow during the project, and become progressively more impressive.

Concluding the thesis are chapters describing relevant future work and overall conclusions. This serves as a summary of the current capabilities of the proposed solution, and which aspects of it need improving before it is useful for real-world systems.

Chapter 2

Literature Review

A disclaimer:
This chapter is similar to its namesake in an unpublished project thesis [1], written during the fall and winter of 2020, by myself, Magne Sirnes. Some changes have been made, especially wrt. theory surrounding error state Kalman filtering (ESKF) and Schmidt-Kalman filtering (SKF), which did not feature in said thesis.

In this thesis, the problems of localization and tracking are overarching themes. For our purposes, localization is defined as an estimation problem, in which the estimation objective is the pose of an autonomous vehicle (i.e. *the ownship*) in a global coordinate system. For our purposes, as is typical for surface vessels, this pose is restricted to three degrees of freedom. i.e. position, velocity and heading in the plane. Similarly, we treat tracking as an estimation problem for other vessels (i.e. *targets*) in which the estimation objective is the position and velocity in the plane. Typically, tracking is also subject to limited target visibility, cluttered measurements and uncertain data association (i.e. we need to figure out whether a given detection actually stemmed from the target in question, before we can use said detection for estimation), for this reason we usually talk about tracking as the compound problem of both data association and filtering.

In terms of combining localization and tracking, there has been a significant gap in the literature. This gap manifests itself in the fact that most tracking algorithms assume knowledge about the ownship pose (position and orientation) with negligible uncertainty. In cases where the ownship pose is not known, the problems of localization and tracking are usually solved separately to fit this abstraction, i.e. *localization then tracking*. Only in recent years has there been made made steps to address these as joint problems.

In this chapter, three categories of solutions for joint localization and tracking are reviewed. First are the tracking centered solutions, which assume the localization to be

known, which can then retroactively make corrections on this localization estimate based on tracking results. The second category of solutions are localization centered solutions, such as variations of simultaneous localization and mapping (SLAM) adapted for dynamic surroundings. Third are the true *joint* solutions, which are solutions designed from the ground up to solve both problems jointly.[1]

2.1 Adapting trackers to aid localization

Another disclaimer:
At this point it bears mentioning that *adapting trackers to aid localization* really has not been done much in the literature, and has historically been a no-no.

This is (among other reasons) because a 'localization then tracking' scheme will necessarily use the localization estimate in order to transform sensor data produced in the body (ownship) frame (relative scans) into a global frame. Any error present in the localization estimate will thus propagate into this transformed sensor data. If this data contains detections used for tracking, we would expect all tracks to be 'injected' with this localization error. Taking this one step further, if we now use these tracking results to influence the localization, we could potentially create a feedback loop in which an errant localization estimate spirals out of control, because it is allowed to validate itself using similarly errant tracking results.

What mitigates this problem, and why I endeavour to develop such a solution in spite of this risk, is the presence of onshore sensors (with known poses) also producing detections. These *absolute* detections can keep this divergence problem in check, and should allow us to make the most of the relative scans, while also minimizing the risk of drift or divergence.

What this section actually presents is literature related to dealing with localization uncertainty in tracking, as well as literature presenting filter types which make *adapting trackers to aid localization* plausible.

Assuming that the localization problem is solved first, separately, the consequence is that one can use the resulting pose estimates to shift either detections or tracking estimates from a (ownship) body-frame to a global frame. The nature of this approach, i.e. solving one problem before the other has two immediate downsides. Firstly, both problems are solved with less information, and secondly, whichever problem is solved last is affected by whatever errors were introduced by solving the former. This makes tracking (solved last) particularly tricky. This is because the two primary tasks of a tracker: data associ-

[1] Granted, there is a fine line between SLAM with dynamic surroundings and a truly joint solution, but I maintain the distinction nonetheless. Especially considering conventional SLAM is only a localization scheme, and that SLAM with dynamic surroundings does not include the elaborate target motion models we typically associate with tracking.

ation (figuring out which detections stemmed from what targets) and filtering (updating track estimates based on associated detections), become significantly more difficult if the ownship pose is unknown or uncertain. Efforts for consciously accounting for navigation uncertainty in target tracking include [2; 3; 4].

From a tracking perspective, one can treat the localization error as a collection of nuisance parameters, which may or may not be estimated as part of their mitigation. In the case of not estimating such nuisance parameters (i.e. only considering their effect), the Schmidt-Kalman filter is an established solution [5; 6]. It has some limitations though, namely the assumption that the nuisance parameters are zero mean, and that we effectively disregard any information that might be used to infer these parameters. In the literature, there are some indications that although the SKF can greatly improve filter consistency, it does not necessarily improve estimation accuracy to a significant degree (see [7] and particularly fig. 2 of [4]).

As for estimating nuisance parameters explicitly, one could simply expand the tracking filters to also include the localization error as part of the target state. Structurally this behaves like any other filter (KF or EKF) and simply has some additional (not inherent to the target) states. In [7] Brink notes that if the nuisance parameters are only intermittently observable, then any filter trying to estimate them continuously (thus also in periods of inobservability) is liable to diverge. If this is the case for the localization error, one would expect such a filter to fail spectacularly. A more obvious downside is that we now have multiple filters (both localization and tracking) that are all trying to maintain localization estimates. In terms of computational expense, this is of course negative.

Having to choose either estimation (EKF) or consideration (SKF) is not always acceptable though, especially in cases where these nuisance parameters are both intermittently observable and we wish to maintain estimates of them. A few filter structures which act as a combination (or compromise) of these two philosophies include the intermittent Schmidt-Kalman filter (ISKF) and fixed-weight partial Schmidt-Kalman filter (FPSKF), as suggested by Brink in [7]. A recurring theme with these schemes is that they first calculate estimates for the nuisance parameters, as an EKF would. After which, using *some* heuristic, decide how much of these posterior estimates to keep. In the case of ISKF, this decision is binary, i.e. either fully estimate or only consider, depending on a binary observability heuristic. For the PSKF (and subsequent FPSKF), the degree to which new estimates are kept is decided by a weight parameter instead, which provides some more granularity, at the cost of requiring a more elaborate heuristic. Although not used in this thesis, these schemes are mentioned for posterity, due to the obvious applicability of estimating and mitigating localization error as a set of (intermittently observable) nuisance parameters.

The choice between these two philosophies (SKF vs. EKF) will inevitably boil down to the question: ”Do I need to know what the nuisance parameters actually are?” In our case, having explicit estimates of the nuisance parameters constitutes an opportunity to achieve more accurate localization, for this reason, it is the latter which interests us in

this thesis. The converse choice, namely using an SKF with navigational uncertainty to improve tracking performance (without affecting localization) is explored in [4; 6].

Let's preface this notion of nuisance parameters with the assumption that *some* decently accurate localization estimate is already available. From there we can treat the residual error of this estimate as an *error state*, i.e. we assume that it is accurate (zero mean, as SKF requires), but we also acknowledge that there may be an error present. This error is presumably observable using external measurements. In the literature, this is an estimation problem for which the 'indirect' error state Kalman filter (ESKF) is an established solution [8; 9].

This filter achieves its localization objective by estimating the error between a nominal estimate and its true state in every update step, after which it *injects* this error into the nominal estimate, thus 'driving' it towards the true state. Assuming now 'post-injection' that there is no error in the nominal estimate, the ESKF internalizes this by performing a reset-step, which manually sets the error estimate back to zero. There is one catch though, namely that the traditional implementations of ESKF (see [8; 9]) typically leave an ESKF as *the lone guarantor of localization*, in other words, the nominal state estimate is *only* a state estimate, and does not have its own uncertainty estimate attached. To this end, it falls to the ESKF to maintain such a localization uncertainty estimate, which by definition should be just as uncertain as its error state.

Now we get into the contribution of this thesis. If we include one or more *external* filters (i.e. trackers), which are maintaining *error states* of the localization variables (on top of the nominal estimates for their respective targets), then it should be possible to use these error estimates to aid a separate localization filter. In some sense, the trackers are taking on the job of an ESKF, which can occasionally provide corrections (injections) to improve the overall localization estimate. This is somewhat unorthodox, as we are now introducing several redundant filters which all maintain some Gaussian estimate of the localization state (or its error). We reach a point of contention upon trying to inject the error state estimates into the localization filter, as the error state sections of the tracking filters (with their redundant estimates) now have to interact with another filter. I have opted to label such a scheme, which is the chief contribution of this thesis, as *error state joint localization and tracking* i.e. ES-JLAT.

At this point, we have touched on both ESKF both SKF as relevant tools for localization and tracking. It bears mentioning that both these schemes were originally designed for mitigation of biases associated with measurements from inertial measurement units (IMU). Such applications of these filters are presented in [5; 9; 8]. Noting that we are interested in correcting the localization estimate (pose) directly, and not the typical IMU biases, we are forced to dig a little deeper in the literature. On this topic, some relevant works I could find are [6; 4]. While these are similar in terms of filter structure, they are still on the side of 'considering' (SKF) and not 'mitigating' (ESKF). In this sense, the contribution of this thesis becomes to adapt the filters used in [6; 4] from an SKF framework to an ESKF framework, while also attempting to solve the problems introduced by injecting error states into a 'competing' nominal filter.

2.2 Adapting localization schemes to do tracking

We now move along to solving the localization problem (separately). For this section, we make no distinction between relative localization (how has my pose changed from my initial condition) and absolute localization (what is my pose in relation to some global frame). In state of the art localization without GNSS dependence, SLAM reigns supreme. This is due to certain ubiquitous 'one-size-fits-all' solutions which are able to combine vehicle odometry (either from actuators or inertial sensors) with cheap extrospective sensors such as cameras, and using these produces profoundly precise and drift free navigation (provided loop closure is possible). This is also awesomely modular, as SLAM algorithms will work with almost any sensor as long as it produces repeatable detections of landmarks (which can be anything vaguely recognizable within the sensor data). Most of the SLAM literature makes assumptions of the surroundings (i.e. landmarks) being fully stationary, and uses the ability to recognize features within the environment (and their locations) in order to solve the localization problem. Some state of the art SLAM approaches include [10; 11; 12]. To enforce this assumption of stationary surroundings, efforts are being made in the SLAM field to find clever ways of disregarding moving features [13], as they are likely to corrupt pose estimates. In terms of tracking moving objects, this causes an inherent conflict of interests.

With this conflict of interest in mind, some works have regardless relaxed this assumption and also included dynamic features (in various senses) as parts of the map. This brings us neatly into another joint-approach for localization and tracking, namely to let some SLAM algorithm also keep track of dynamic features, which could correspond to moving targets. Noting that some SLAM algorithms keep track of features in a unique manner, i.e. being able to say "I know exactly what each feature looks like, and can detect and differentiate them fully." as opposed to "I have a number of features in my map, and I am only able to recognize that something *is* a feature, not which one it is." For the latter, this translates to having to solve an association problem after detection, while I would characterize the former as the SLAM algorithm doing *hard association* during its detection step. For target tracking, part of which involves association, the prospect of having a SLAM algorithm being able to perfectly do this association bears some merit. Some recent efforts which include target tracking in SLAM in various senses include [14; 15].

2.3 True joint localization and tracking schemes

At this point we have presented a few schemes which are capable of localization and tracking in some sense. However, they were all (at least historically) designed to perform **one** of the two tasks, then retrofitted (or combined with other schemes) in order to perform both. This leads us neatly to the final category, namely schemes which are designed from the ground up to jointly perform both tasks, i.e. *joint localization and tracking* (JLAT) schemes.

One way of classifying such a joint-approach is as a single-cluster point process filtering problem. In the single-cluster approach, the localization of the ownship is regarded as the parent process, and the tracking of consequent targets as child-processes which are dependent on the former. In this sense, in a similar manner to a SLAM like algorithm (maintaining estimates of stationary features, and using the relative positions of these features to estimate ownship pose)[2], it is the target positions which bear information, and the ownship position is inferred from this. While this is still very much a work in progress, huge strides towards making this style of problems tractable have been made in [16; 17]

A recurring theme for the works referred to in this section is that any of them which address jointly solving the problems of localization and tracking, all *only* use body-mounted sensors. Further, to the best of my knowledge, no renowned works exist in the literature which address these problems, that also include the use of both stationary and body-mounted sensors. This summarizes the fundamental gap addressed in this chapter. The goal of this thesis, then becomes to suggest such a scheme.

[2]Note the duality between the SLAM and tracking problems here: while tracking maintains estimates of the dynamic parts of the surroundings (targets), SLAM maintains estimates of the stationary parts of the surroundings (features).

Chapter 3

Theory - Notation

For the sake of avoiding confusion, some over-arching choices have been made regarding the notation used in this thesis. These choices are influenced by publications by Musicki et. al. [18; 19], Bar-Shalom et. al. [20; 21; 22], and Brekke[1] [16; 2], as well as adjusted according to the author's personal taste. This chapter is mostly identical to its namesake in my specialization project thesis [1].

3.1 Vectors and matrices

Vectors have no special notation, so it is given by context whether italic characters, e.g. x, z, p are scalar or vector sizes. For the record, these indicate target states, measurements and probabilities or position, respectively.

Matrices are always indicated with bold font, EG $\mathbf{R}$, $\mathbf{Q}$, $\mathbf{P}$. By extension, any adjacent variable to a matrix is to be interpreted as linear algebra multiplication, EG $\mathbf{F}x$.

3.2 Subscripting and indexing

No indexing will be used for this thesis, this is to ensure that parentheses only ever imply precedence of calculation, or indicating parameters provided to a function. EG $a(b + c)$ and $f_x(x_k)$.

[1] This also includes a so far unpublished sensor fusion textbook by Brekke.

In some cases, there will be need for differentiating several variables with the same symbols using subscripts. In these cases, the following standard will be followed:

- j: Used for enumerating measurements, either of some sub-set given by context, or more commonly, all measurements from a scan of a given time-step.
- k: Used for enumerating time-steps, IE successive steps of a discrete process.
- t: This subscript is used for enumerating tracks, which, assuming a 'true' track, also corresponds to a target of interest.
- s: Used for enumerating sensors, for example to differentiate between measurements produced by different sensors (or their corresponding pipelines). Also used to denote a general subset of a vector, i.e. x_s is a vector containing some of the variables of x.
- n, m: Used for general purpose enumeration EG $\{x_1, ..., x_n\}$.

In cases where multiple subscripts are needed, they are delimited by commas, EG $z_{k,j,s}$, which reads as measurement number j from a scan at time-step k, provided by sensor s.

In general, capital letters of the aforementioned subscripts indicate the total number of each enumeration, EG, summing over all sensors for the aforementioned example will be:

$$\sum_{s=1}^{S} z_{k,j,s}$$

3.3 Hats, tildes and other modifiers

To differentiate between true, estimated and predicted values, a number of modifiers are used:

For all predictions, IE propagating an uncertain estimate to the next time-step, using the previous estimate and the process model, a hat will be used. EG $\hat{x}_{k|k-1}$, note that context regarding the current time-step k and the 'predicted from' time-step $k-1$ is indicated as $k|k-1$.

When such a prediction is updated using relevant measurements, i.e. converted from a prior to a posterior, a hat is still used, but the subscripts change. EG, $\hat{x}_{k|k}$. this reads as "some estimate of x for timestep k, updated with measurements from timestep k.

Similarly, variables calculated from estimated values also inherit this hat. EG $\hat{z}_t = \mathbf{H}\hat{x}_t$.

In the case of no modifier being present, we assume that it is a *true* value.

Finally, there will occasionally be a discrepancy between an estimate and its true value. An overarching objective for this thesis is estimating and minimizing this *error state*, and for this reason it is designated its own modifier tilde, i.e. $\tilde{x}$.

The relationship between *true*, x, *estimated*, $\hat{x}$ and error, $\tilde{x}$ states is expressed as follows:

$$\tilde{x} = x - \hat{x} \tag{3.1}$$

3.4 Block matrices, dimension and index subscripts

For the sake of not defining huge matrices on a per-element basis, I will endeavour to express these using block matrices wherever possible. In these cases, there will usually be need for filler matrices which are either identity matrices, $\mathbf{I}$ or zero matrices, $\mathbf{0}$. In order to denote the dimensions of these matrices, the subscript notation nxm is employed. This corresponds to a matrix of height n with width m, e.g.

$$\mathbf{0}_{2x3} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{3.2}$$

To that end, building a block matrix of matrices $\mathbf{A}$ and $\mathbf{B}$ which are $2x2$ and $3x3$, respectively, can be done as:

$$\begin{bmatrix} \mathbf{A} & \mathbf{0}_{2x3} \\ \mathbf{0}_{3x2} & \mathbf{B} \end{bmatrix} \tag{3.3}$$

As block diagonal matrices with the off diagonal elements being zero are needed particularly often, I include another shorthand notation for this, namely the *blkdiag*$(.)$ function. Building the aforementioned block diagonal matrix (eq. 3.3) can be expressed as *blkdiag*$(\mathbf{A}, \mathbf{B})$. This is equivalent behaviour to its scalar variety *diag*$(.)$, which also sees some use in this thesis, e.g.

$$\mathit{diag}(a, b) = \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix} \tag{3.4}$$

Finally, there will occasionally be need for pulling certain sub-matrices from a larger matrix, I will denote this as $a : b, c : d$. This reads as "select elements from the matrix, starting at row a to and including row b, as well as column c to and including column d, of course counting from top to bottom and left to right, respectively. For example:

$$\mathbf{M} = \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix}, M_{1,1:2} = \begin{bmatrix} a & b \end{bmatrix} \tag{3.5}$$

For covariance matrices, $\mathbf{P}$, of a multivariate Gaussian, the diagonal elements will correspond the to marginal covariance of each state element. In order to refer to such a marginal with a little more context, I will use the shorthand $\mathbf{P}_a$. This refers to the marginal covariance of state variable a, e.g.

$$\mathcal{N}\left(x = \begin{bmatrix} a \\ b \end{bmatrix}, \mathbf{P} = \begin{bmatrix} \sigma_a^2 & \sigma_{a,b}^2 \\ \sigma_{b,a}^2 & \sigma_b^2 \end{bmatrix}\right), \mathbf{P}_a = \sigma_a^2 \tag{3.6}$$

3.5 Substate selectors as linear transformations

A recurring theme in the derivations of this thesis is the need for selecting some subset of a state vector. For example, given a 2 DOF CV state vector $x_{\text{CV}} = [p_x, v_x, p_y, v_y]^\top$, we might want to *select* only the positional elements, $x_{\text{pos}} = [p_x, p_y]^\top$ of this vector. For the purposes of this thesis, I define such a *selector* as a linear transformation $\mathbf{H}_{\text{CV}\to\text{pos}}$, s.t. $x_{\text{pos}} = \mathbf{H}_{\text{CV}\to\text{pos}} x_{\text{CV}}$, i.e.:

$$\mathbf{H}_{\text{CV}\to\text{pos}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{3.7}$$

Note the context provided by the subscript $A \to B$ here, this reads as: from A, select B. This will prove particularly useful for certain filters which have some states in common.

This is essentially an extension of an *observation matrix* $\mathbf{H}$ typically associated with KF update steps. For the record, such observation matrices will also be defined with selectors wherever possible.

Chapter 4

Theory - Filtering and the error state Kalman filter

An overarching theme for this thesis is filtering, and especially the subtle art of handling several filters, all of which have their own estimation objective, while at the same time interact in some sense. In this chapter, an introduction to filtering is provided, as well as an outline of the error state Kalman filter. The latter is particularly crucial, as the localization corrections which make joint localization and tracking possible in the suggested scheme, are structured mostly as error state Kalman filters (ESKFs).

The one distinction, which also stands as the elephant in the room, in many ways, is the covariance estimate of the various filters. Provided one has a number of filters which all calculate Gaussian estimates of the same state, how on earth should their covariances interact? In this thesis, I have seemingly found a few reasonable ways to handle error state injection, but I still think some work remains on this topic.

This chapter is divided into three parts, the first of which introduces filtering as a whole, while the second introduces ESKF filtering, in terms of some (external) ESKF driving a localization estimate. Finally, the third part introduces some generalized tools which will be utilized later in this thesis. These tools constitute the adaptations made to a typical ESKF structure, which will result in a compound filter structure which is capable of having different filters with redundant state and covariance estimates interact in a meaningful way.

4.1 Introduction to filtering

For the purposes of this thesis, the words *filter* and *filtering* both refer to filters in the sense of state estimation filtering. These constitute algorithms (filters) which solve the state estimation problem: Given access to measurements z which are mathematically connected to the system state x, maintain an estimate of said system state. The notion of a filter in this sense stems from the fact that these measurements are usually subject to some form of uncertainty or noise. The filtering aspect of these solutions is defined by the fact that they somehow disregard or mitigate this noise, i.e. *filter it out*.

To further complicate matters, these measurements are often connected to the system state in some incomplete or convoluted sense, this relationship is typically defined by a measurement function $z = h(x)$. A secondary 'job' of a filter thus becomes to "put the pieces together" in order to infer whatever part of the system state which is not readily available in the measurement. For this reason, the question of *observability* is crucial within filtering, namely, is it mathematically possible to estimate the entire system state using only the measurements available? In other words, is there enough information present?

A simple example of a filter having to infer some subset of the state is as follows: A filter is tasked with estimating the position and velocity of a vehicle, but is only provided with periodic measurements of the vehicle position. Intuitively, we would say this is observable. If we for example receive two consecutive measurements, one second after each other, in which the position of the vehicle has changed by 10 meters, due north, it would be reasonable to assume that the velocity of the vehicle is approximately 10 m/s to the north (at the time of receiving the second measurement).

This highlights the final aspect of filtering, namely that we usually assume some knowledge about the system behaves, i.e. its dynamics (which in the nonlinear continuous case is defined as $\dot{x} = f(x)$). The inclusion of this knowledge has a profound effect on both state estimation and observability. E.g. if we in the previous example assume that the vehicle can *only* move southward, then our conclusion about the vehicle velocity would be dramatically different. We might now be forced to conclude that it was in fact measurement noise which produced the erroneous impression of the vehicle moving north, and that the vehicle is actually standing still (or potentially even moving slowly southward).

Finally, noting that the world is an uncertain place, in which both system dynamics and measurements are subject to errors and noise, it would be prudent to have filters also say something about how certain their estimates are. This prompts the use of probabilistic filters, which maintain state estimates in terms of probability distributions. The epitomal probabilistic state estimation filter is the Kalman filter (KF), which maintains a state estimate as a multivariate Gaussian distribution. For the purposes of this thesis, in which the KF is a crucial building block, the details of its inner workings are presumed known to the reader. To brush up on KF theory, I recommend [23].

For the purposes of this thesis, i.e. localization and tracking, we are usually interested in the position and velocity of some surface vessel. The system dynamics of such a vessel

(used for filtering) is referred to as a motion model henceforth. Similarly, the nature of the measurements (usually position of said vessels), and the noise they are subject to, are defined as measurement models. In chapter 5, I present all motion and measurement models which are used.

4.2 The error state Kalman filter (ESKF)

4.2.1 ESKF - Background

The ordinary KF has some limitations, and for our purposes we require more elaborate filters in order to achieve adequate estimation. The most obvious limitation is that the vanilla KF is only applicable to Linear-Gaussian systems, i.e. systems with linear equations for both dynamics and measurements, both of which are subject to (driven by) Gaussian white noise.

A well-established expansion to the KF is the Extended Kalman filter (EKF), this is a filter which now supports nonlinear dynamics and measurement functions, but still mostly behaves as a regular KF. This is achieved in the EKF by linearizing these functions about the current state estimate any time a prediction or update is being made, resulting in a set of matrices similar to the ones used in the regular KF equations.

A more problematic limitation of KF and EKF is an aversion to rotations. For an inertial system in which its attitude needs to be estimated, with the potential for angular rates being measured by an inertial measurement unit (IMU), keeping track of rotation can subject to over-parametrization and singularities. Similarly, the current attitude will have a profound and non-linear effect on the system dynamics.

Furthermore, assuming that the pose of some vehicle is the only estimation objective, a set of IMU measurements would now make it possible to maintain such an estimate *without a vehicle dynamics model* i.e. not knowing how the vehicle interacts with its surroundings (see [8]). In other words, our system dynamics model can be reduced to ”acceleration adds up to speed, angular rates add up to rotation”, and from these we can estimate pose and velocity in all 6 DOFs.

To this end, while at the same time solving the problems associated with rotation, the ESKF was developed. This is a filter which operates not on a state estimate $\hat{x}$ of a true state x, but rather the error between these two sizes, $\tilde{x} = x - \hat{x}$. Commonly referred to as an ’indirect’ Kalman filter, namely because the IMU measurements are included not as measurements, but as part of the system state. To this end, the ESKF is not a filter which ’filters’ IMU measurements, but rather a filter which ’after the fact’ (indirectly) tries to mitigate the known systematic errors introduced by IMU integration.

This proved to be profoundly useful for inertial navigation, and is the de facto standard for navigation of drones, small fixed wing aircraft, and other applications in which GNSS updates are not fast or precise enough by themselves. Similarly, a well initialized ESKF can provide outstanding localization performance for periods in which external navigation is unavailable. For a some historical context on the ESKF, I defer to [8]. For a *very* rigorous introduction to ESKF in the context of full 6 DOF intertial localization using quaternions, I recommend Sola's [9].

4.2.2 ESKF - Typical formulation

In this thesis, we are slightly abusing (adapting) the notion of a ESKF for our own purposes. This section introduces the high-level equations which a typical ESKF uses, and points out some aspects of this which we will need to adapt.

ESKF - Estimation objective

In general, the (indirect) error-state Kalman filter (ESKF), has the following estimation objectives:

- **Predict and update**: Maintain an estimate of δx_k (i.e. $\tilde{x}_o$), which corresponds to the current estimation error $x_k - \hat{x}_k$.
- **Predict and update**: Maintain an estimate of $\hat{\mathbf{P}}_k$, which is the covariance of δx_k, and by extension the current localization uncertainty.
- **Update and inject**: Whenever external sensor readings come in (i.e. updates), the posterior $\delta x_{k|k}$ is driven away from zero. Inject this nonzero estimate into the nominal state estimate as $\hat{x}_{k|k} = \hat{x}_{k|k-1} \oplus \delta x_{k|k}$.[1]
- **Reset**: After injection, set the posterior error estimate back to zero (since we have now accounted for it), $\delta x_{k|k} = \vec{0}$.

There are some subtle details omitted here, for example that the covariance estimate $\hat{\mathbf{P}}_{k|k}$ may need to be adjusted as part of the 'inject & reset' steps, in order to handle a sudden change in attitude. Since we are making significant changes regardless, such details are omitted here, again I defer to Sola's [9] for a more cohesive guide to ESKF.

[1] The operator $\oplus$ is used here to indicate that certain elements, such as quaternions, may indeed require other operators than $+$ in order to receive an injection. This constitutes an ad-hoc way to indicate that each element of the state vector should be injected according to whatever operator makes sense for that element.

ESKF - Prediction step

Being an error state filter (which by default assumes zero error), the prediction step (i.e. IMU updates) will essentially behave like dead-reckoning, and does the following:

- Integrate IMU samples in order to update nominal state, $\hat{x}_{k+1|k} = f(\hat{x}_k, u_k)$, i.e. dead-reckoning.
- Integrate IMU samples in order to update error state. Since no external information is available at this time, this prediction *has to* conclude that the error is still zero.
- Inflate covariance estimate $\hat{\mathbf{P}}_{k+1|k}$ according to process noise $\mathbf{Q}$, and crucially propagate correlations according to system dynamics $\mathbf{F}$, along the lines of $\hat{\mathbf{P}}_{k+1|k} = \mathbf{F}\hat{\mathbf{P}}_k\mathbf{F}^\top + \mathbf{Q}$.

ESKF - Update step

After receiving an update z_k from *some* external sensor at timestep k, the ESKF is able to calculate an update step, (similar to any other other KF). This results in two crucial changes:

- Update error estimate $\delta x_{k|k}$ according to typical KF equations. At this point, $\delta x_{k|k}$ can have nonzero values!
- Reduce the posterior covariance estimate $\hat{\mathbf{P}}_{k|k}$, as is typical for update steps. E.g. $\hat{\mathbf{P}}_{k|k} = \hat{\mathbf{P}}_{k|k-1} - \mathbf{W}\mathbf{H}\hat{\mathbf{P}}_{k|k-1}$.

At this point, we have the ESKF in a weird state, namely with a nonzero error estimate. Since the 'natural' state for our ESKF is to assume zero-mean error, we need to get back to this somehow. This is achieved with the injection and reset steps. This also reinforces the idea that the ESKF wants to operate with small errors and corrections, since it typically calculates its updates using linearizations about the prior state estimate $\hat{x}_{k|k-1}$. As with EKF, such an approximation is liable to introduce linearization errors if the update steps make too large strides from this.

ESKF - Injection and reset

As mentioned previously, the injection and reset steps simply shifts the current posterior error estimate $\delta x_{k|k}$ from the error state to the nominal state, again leaving the error state estimate $\delta x_{k|k}$ as zero (and zero mean). I.e.:

- $\hat{x}_{k|k} = \hat{x}_{k|k-1} \oplus \delta x_{k|k}$.
- $\delta x_{k|k} := \vec{0}$
- If necessary, make adjustments to $\hat{\mathbf{P}}_{k|k}$ to reflect injected state (typically needed for attitude changes).

4.3 Changes and tools developed for this thesis

For the purposes of this thesis, a number of changes to the aforementioned ESKF are needed, these changes, and the reasoning behind them, are presented in this chapter.

4.3.1 On state and covariance redundancy

First of all, note that the typical ESKF implementation only has one filter, despite managing estimates for both the nominal state *and* the error state. In this sense the covariance estimate $\hat{\mathbf{P}}$ produced by the ESKF as part of its Gaussian $\hat{\delta x} \sim \mathcal{N}(\vec{0}, \hat{\mathbf{P}})$, is defined as the covariance for *both* $\hat{\delta x}$ and $\hat{x}$.

This is not so weird, as we can treat the true state x as a constant for any given time. From the error state definition $\hat{\delta x} = x - \hat{x}$, it is apparent that if $\hat{\delta x} \sim \mathcal{N}(\vec{0}, \hat{\mathbf{P}})$ holds, a consequence has to be that $x \sim \mathcal{N}(\hat{x}, \hat{\mathbf{P}})$.

This is the first aspect in which things get spicy. Let's now instead assume that we have a localization filter which maintains its own estimate for both the nominal state, as $\hat{x}_o$, and a corresponding covariance matrix $\hat{\mathbf{P}}_o$. This could be a KF, an EKF, or even the nominal estimate of an ESKF (it really does not matter as long at it produces a multivariate Gaussian estimate). For our purposes, we will treat this filter as a unit of abstraction, namely a *localization filter*, which produces a Gaussian estimate according to $\hat{x}_o \sim \mathcal{N}(x_o, \hat{\mathbf{P}}_o)$.

In order to aid the estimates of this filter, we opt to include an external filter which produces an estimate of the error state (ES) $\tilde{x}_o = x_o - \hat{x}_o$, again with a (separate) covariance estimate $\tilde{\mathbf{P}}_o$, s.t. $\tilde{x}_o \sim \mathcal{N}(\vec{0}, \tilde{\mathbf{P}}_o)$.

This constitutes a break with the typical ESKF structure, wherein we would treat the ES covariance $\tilde{\mathbf{P}}_o$ as *the only* covariance available for the localization estimate $\hat{x}_o$. Instead, we are left with redundant covariance matrices, namely $\hat{\mathbf{P}}_o$ and $\tilde{\mathbf{P}}_o$. This prompts some important questions:

1. Which of these covariances should we trust?
2. How are they correlated?

3. If the external filter produces a nonzero estimate $\tilde{x}_o$ which we inject into the nominal estimate $\hat{x}_o$, how should that influence the localization filter's mean and covariance?

For the purposes of this thesis, I have opted to answer these questions thusly:

1. We trust the localization filter to maintain a 'best' estimate of covariance, i.e. we trust $\hat{\mathbf{P}}_o$. This is somewhat arbitrary, but I made this choice to enforce the abstraction that it is the localization filter's **job** to estimate both the ownship state and its uncertainty. This allows the external filters (which only intermittently have information about the ES, i.e. nonzero estimates) to occasionally have inconsistent or weird states.

   I.e. from an outsiders perspective, it is *always* the localization filter's estimates which are assumed to be accurate and consistent. One could even go as far as to say that the error state estimates should remain hidden in terms of a larger system.

2. Depending on how the external filter is designed, the posterior covariances may or may not be correlated.

   In this thesis, two different solutions were designed and tested: one which lets these covariances be completely de-coupled, and another which imposes that the external prior covariance $\tilde{\mathbf{P}}_{o,k|k-1}$ *has to be* equal to the (internal) prior covariance $\hat{\mathbf{P}}_{o,k|k-1}$. After an update step in the external filter, this of course implies a *very strong* correlation for the latter.

3. For the mean, we treat it as any other ESKF, i.e. the error state mean has to remain at zero. As for the covariance, we are left with a bit of a conundrum.

   On one hand, such an injection *should* make the localization estimate more precise, and by extension reduce its covariance. On the other hand, since the normal ESKF does not maintain redundant covariance estimates, we have no obvious way to perform such a reduction.

   In this thesis, answering this question of **post-injection localization covariance** (or at least finding a half decent solution) stands as one of the key contributions.

4.3.2 On compound (nominal and error state) filters

The second bastardization of ESKF made in this thesis is as follows: in the normal ESKF, the 'indirect' filter which estimates the error state will **only** estimate the error state. For the purposes for this thesis, in which the localization state x_o is profoundly connected to other target states x_t, we leverage this by creating external (target state) filters which maintain

both nominal estimates of a target state x_t, **and** an error state estimate of (some of)[2] the localization state $\tilde{x}_o$.

What this amounts to is that the *target state filters*, are now effectively *compound filters* i.e. some of the state (x_t) is estimated along the lines of a KF or EKF, while the remainder of the state ($\tilde{x}_o$) is estimated along the lines of an ESKF, with all the injection and reset steps which that entails. Later in this section, some tools to handle the logistics of these operations are included.

This is in many ways similar to a 'consider' Schmidt-Kalman filter (SKF) structure, in which a nominal state is estimated, but one also 'considers' the effect of additional zero-mean *nuisance parameters* with some known covariance. For such a filter, it is imposed that these parameters have to stay at zero, and that their marginal covariance does not change over predict and update steps. Applying such a filter to our "tracking subject to localization error" problem, we could for example set the target state x_t as the nominal state, and the localization error $\tilde{x}_o$ as the nuisance parameters. Such a structure is used in [6; 4], and stands as vital points of inspiration for the solutions suggested in this thesis.

For our purposes, it is crucial that the ownship localization error estimate ($\tilde{x}_o$) is allowed to differ from zero during update steps (as we wish to inject it into the localization filter). For this reason, we shift from an SKF style filter into a so called compound filter, in which only some of the states (previously nuisance parameters) are used in injection and reset steps. For lack of better terms, this filter structure becomes a combination of SKF and ESKF, which again is very similar to ISKF and FPSKF as suggested by Brink [7].

Structure of compound filters

In this thesis, these compound filters are structured as:

$$\begin{bmatrix} \hat{x}_t \\ \tilde{x}_o \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} x_t \\ \vec{0} \end{bmatrix}, \begin{bmatrix} \hat{\mathbf{P}}_t & \hat{\mathbf{P}}_{t,o} \\ \hat{\mathbf{P}}_{o,t} & \tilde{\mathbf{P}}_o \end{bmatrix} \right) \tag{4.1}$$

Depending on the design of these external filters, the ownship-target correlation covariances $\hat{\mathbf{P}}_{t,o}$ and $\hat{\mathbf{P}}_{o,t}$ may or may not be estimated, or indeed reset to zero along with the error state estimates ($\tilde{x}_o$) after an injection step. Similarly, a few different solutions for the handling of the marginal error state covariance $\tilde{\mathbf{P}}_o$ are suggested. As for the marginal target state covariance, $\hat{\mathbf{P}}_t$, it is left entirely in the hands of the prediction and update steps. In other words, the marginal target estimate $\hat{x}_t \sim \mathcal{N}(x_t, \hat{\mathbf{P}}_t)$ is left *entirely unaffected* by reset and injection steps.[3]

[2] i.e. subset ESKF, which is described in section 4.3.3

[3] This becomes problematic when we start looking at multiple, interacting compound filters, but left as a reasonable choice for the single filter case.

Useful functions for compound filters

For the sake of later convenience, as well as increased abstraction, I include some high-level functions for interacting with these compound filters.

The first of these is simply pulling the marginal error state estimate from the current (full) estimate $\{\hat{x}, \hat{\mathbf{P}}\}$. I define this as the function **get_error_state**(.), which returns the marginalized Gaussian $\{\tilde{x}_o, \tilde{\mathbf{P}}_o\}$, i.e.:

$$\{\tilde{x}_o, \tilde{\mathbf{P}}_o\} = \textbf{get\_error\_state}(\left\{ \begin{bmatrix} x_t \\ \tilde{x}_o \end{bmatrix}, \begin{bmatrix} \hat{\mathbf{P}}_t & \hat{\mathbf{P}}_{t,o} \\ \hat{\mathbf{P}}_{o,t} & \tilde{\mathbf{P}}_o \end{bmatrix} \right\}) \tag{4.2}$$

Considering we will eventually have one such compound filter for every target t being tracked, a similarly useful function would be to pull the error state from *every* filter. I define this as **get_error_states**$(\{\hat{x}_1, \hat{\mathbf{P}}_1\}, ..., \{\hat{x}_T, \hat{\mathbf{P}}_T\})$, which returns a list of marginalized error states $\{\tilde{x}_{o,1}, \tilde{\mathbf{P}}_{o,1}\}, ..., \{\tilde{x}_{o,T}, \tilde{\mathbf{P}}_{o,T}\}$.

Regardless of how the redundant covariances are handled, we *always* want to reset the error state estimate $\tilde{x}_o$ to zero after an injection. I define this (for a single target) as **reset_error_state**(.), i.e.:

$$\left\{ \begin{bmatrix} x_t \\ \vec{0} \end{bmatrix}, \begin{bmatrix} \hat{\mathbf{P}}_t & \hat{\mathbf{P}}_{t,o} \\ \hat{\mathbf{P}}_{o,t} & \tilde{\mathbf{P}}_o \end{bmatrix} \right\} = \textbf{reset\_error\_state}(\left\{ \begin{bmatrix} x_t \\ \tilde{x}_o \end{bmatrix}, \begin{bmatrix} \hat{\mathbf{P}}_t & \hat{\mathbf{P}}_{t,o} \\ \hat{\mathbf{P}}_{o,t} & \tilde{\mathbf{P}}_o \end{bmatrix} \right\}) \tag{4.3}$$

Note how this function enforces the zero-mean error assumption we use in ESKF, and essentially serves as the ESKF reset step.

For one of the two 'redundant-covariances' solutions suggested in this thesis, we impose that the prior ES covariance $\tilde{\mathbf{P}}_o$ must be equal to its localization filter counterpart $\hat{\mathbf{P}}_o$ (with no correlation to the ownship state). In order to force the compound filter into such a state, we use the the **reset_error_covariance**(.) function. This function accepts a current filter state, as well as a current localization covariance $\hat{\mathbf{P}}_o$, and returns a 'reset' compound filter state as follows:

$$\left\{ \begin{bmatrix} x_t \\ \tilde{x}_o \end{bmatrix}, \begin{bmatrix} \hat{\mathbf{P}}_t & \mathbf{0} \\ \mathbf{0} & \hat{\mathbf{P}}_o \end{bmatrix} \right\} = \textbf{reset\_error\_covariance}(\left\{ \begin{bmatrix} x_t \\ \tilde{x}_o \end{bmatrix}, \begin{bmatrix} \hat{\mathbf{P}}_t & \hat{\mathbf{P}}_{t,o} \\ \hat{\mathbf{P}}_{o,t} & \tilde{\mathbf{P}}_o \end{bmatrix} \right\}, \hat{\mathbf{P}}_o) \tag{4.4}$$

4.3.3 On estimating subsets of the localization state

The third and final way in which I have adapted the ESKF for external filters, is to set only a subset of the localization state x_o as error state parameters $\tilde{x}_o$. I deemed this prudent,

as the inherent connection between the localization state and target states, is present only for the ownship pose. In other words, from a target's perspective, there is *no* information present in the measurements with which to estimate the ownship velocity.

In a more generalized sense, I deem it prudent to only include localization error states if they have some influence on the measurement function used for updating the target state. This is very connected to the question of whether these error states *are observable* from the provided measurements, however, we may also choose to omit additional elements from the localization state, for example to reduce the ambiguity present in measurements.

Imagine if you will all the different combinations of target and ownship poses which can lead to the same range-bearing detection. Reducing the dimensionality here (by for example only considering the ownship position, or v.v. only the ownship heading as an error state), will reduce this ambiguity quite a bit.

There is also a pragmatic aspect of this, namely the choice of whether a part of the localization estimate actually needs correcting. If we already have amazing estimates of ownship position and velocity, while the ownship heading estimation is lackluster, it might be prudent to only include the ownship heading as an error state parameter. Thus resulting in a 'bespoke' ESKF structure, which only makes injections on the states which need it.

The practical aspects of handling subsets

For defining which subset of the localization estimate $\hat{x}_o$ we use as error states $\tilde{x}_o$, I opt to use a kind of observation matrix $\mathbf{H}$. This is similar to the observation matrix used in a standard KF, which is used to map the filter estimate to a measurement.

For some subset s, I define this observation matrix as $\mathbf{H}_{o\to s}$, s.t.:

$$\tilde{x}_o = \mathbf{H}_{o\to s}(x_o - \hat{x}_o) \tag{4.5}$$

E.g. if our full localization estimate consists of positions p and velocities v in x and y directions, as well as a heading φ, i.e. $x_o = [p_x, v_x, p_y, v_y, \varphi]^\top$, and we wish to use only the position and heading as our error state, i.e. $\tilde{x}_o = [\tilde{p}_x, \tilde{p}_y, \tilde{\varphi}]$, we would define $\mathbf{H}_{o\to s}$ as:

$$\mathbf{H}_{o\to s} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.6}$$

Conversely, if we wish to expand our error state estimate $\tilde{x}_o$ into the dimensionality of our localization estimate (assuming the non-estimated states are zero), we can use the transpose of this matrix, i.e.:

$$\tilde{x}_{o,\text{expanded}} = \mathbf{H}_{o\to s}^{\top}\tilde{x}_o \tag{4.7}$$

Finally, if we wish to select only the relevant parts $\tilde{\mathbf{P}}_o$ of the localization covariance estimate $\hat{\mathbf{P}}_o$ for some subset s, the rules of linear transformations of multivariate Gaussians apply, i.e.:

$$\tilde{\mathbf{P}}_o = \mathbf{H}_{o\to s}\hat{\mathbf{P}}_o\mathbf{H}_{o\to s}^{\top} \tag{4.8}$$

Challenges with subset ESKF

There is one major aspect which makes subset ESKF tricky, in my eyes. Namely injection, should we inject only the variables which we have estimated directly, or should we consider that these parameters may be correlated with the other variables which we *have not* estimated, and infer these as well? By extension, how (if at all) should this injection affect the covariance estimates of the unestimated stated?

4.3.4 Injection of the external estimate

So far, we have glossed over the details of error state injection into the localization filter. I will define this problem as injection of a error state estimate $\{\tilde{x}_s, \tilde{\mathbf{P}}_s\}$, which has elements according to some subset s of the localization estimate, according to the selector $\mathbf{H}_{o\to s}$. The objective then becomes to update both the mean $\hat{x}_o$ and the covariance $\hat{\mathbf{P}}_o$ estimates of the localization filter.

Such an injection would necessarily take place after any update on the external filters which correlates the target state to the ownship (error) state, thus, I leave it as implied that all of these vectors are *posterior* estimates.

Injection: naïve state update

Let's begin with the most naïve choice, which is to expand the error state estimate using the selector $\mathbf{H}_{o\to s}$, and injecting this directly into the localization filter state, i.e.:

$$\hat{x}_{o,\text{post-injection}} := \hat{x}_o + \mathbf{H}_{o\to s}^{\top}\tilde{x}_s \tag{4.9}$$

Injection: finding an analytical covariance update for the naïve state update

As for the covariance, we are left with a bit of a conundrum. On one hand, we need to make sure that the navigational filter remains consistent. On the other hand, we would like to update it to reflect the fact that new, presumably more precise estimates are present. Intuitively, this would mean reducing the covariance. As eluded to previously, the normal ESKF does not handle this kind of 'covariance redundancy' and thus does not provide an obvious answer on how to do such a reduction.

Taking a look at the analytical expressions, we can try to find out how this injection affects the final localization covariance:

$$\text{Var}(\hat{x}_{o,\text{post-injection}}) = \text{Var}(\hat{x}_o) + \text{Var}(\mathbf{H}_{o\to s}^\top \tilde{x}_s) + \text{Cov}(\hat{x}_o, \mathbf{H}_{o\to s}^\top \tilde{x}_s) \tag{4.10}$$

Using the rules of linear transformation of multivariate Gaussians, and that the (pre-injection) localization covariance is $\hat{\mathbf{P}}_o$, we can insert these into the equation, resulting in:

$$\text{Var}(\hat{x}_{o,\text{post-injection}}) = \hat{\mathbf{P}}_o + \mathbf{H}_{o\to s}^\top \tilde{\mathbf{P}}_s \mathbf{H}_{o\to s} + \text{Cov}(\hat{x}_o, \mathbf{H}_{o\to s}^\top \tilde{x}_s) \tag{4.11}$$

Expectedly enough, we rely on the covariance between the localization state $\hat{x}_o$ and the injected correction $\mathbf{H}_{o\to s}^\top \tilde{x}_s$ in order to reduce the overall variance (since this can and should have negative values, given our error state definition). This of course begs the question, how on earth are these correlated? Intuitively, they have to be, considering that the injection should be an estimate of $x - \hat{x}_o$, otherwise, our injections would be nonsensical.

If I were a better statistician with more time, I might fiddle around with $\text{Cov}(\hat{x}_o, \mathbf{H}_{o\to s}^\top \tilde{x}_s)$ to see if I found some analytical expression which would make an optimal covariance update possible. That being said, I am not, for this reason I will instead endeavour to find some sub-optimal covariance update equation which appears to maintain consistency in simulation.

The second reason for avoiding this (granted alluring) rabbit-hole of analytical statistics, is that the relationship between the error state estimate $\tilde{x}_s, \tilde{\mathbf{P}}_s$ and the state estimate $\hat{x}_o$ becomes **very complex** as we start calculating the error state estimate as a consensus of **many** independent compound filters, each with posteriors depending on their incoming measurements and current target estimates, not to mention the localization estimate.

Injection: finding a sub-optimal covariance update instead

Looking back to the ESKF, we note that it reduces its covariance using an update step, which reduces the covariance estimate as a function of $\mathbf{W}$ and $\mathbf{H}$. Imitating this, I opt to

'spoof' a KF update step for the sake of calculating the post-injection covariance. Our job then becomes to define functions for $\mathbf{W}$ and $\mathbf{H}$ s.t. this update maintains consistency in the localization filter.

Starting with the most obvious option, I elect to pretend that the injection is in fact an innovation vector, with an associated innovation covariance $\mathbf{S}$. This matrix is a function of the current localization covariance $\hat{\mathbf{P}}_o$, as well as a measurement noise matrix $\mathbf{R}$. For our purposes, we pretend that the measurement noise is in fact the injection covariance $\tilde{\mathbf{P}}_s$, this yields:

$$\mathbf{S}_s = \mathbf{H}_{o\to s}\hat{\mathbf{P}}_o\mathbf{H}_{o\to s}^\top + \tilde{\mathbf{P}}_s \tag{4.12}$$

With $\mathbf{S}_s$, $\mathbf{H}$ and the prior localization covariance $\hat{\mathbf{P}}_o$, we can calculate the Kalman gain matrix $\mathbf{W}_s$ in the usual fashion:

$$\mathbf{W}_s = \hat{\mathbf{P}}_o\mathbf{H}_{o\to s}^\top\mathbf{S}_s^{-1} \tag{4.13}$$

Finally, we can calculate the posterior (post-injection) ownship covariance matrix $\hat{\mathbf{P}}_{o,\text{post-injection}}$ as follows:[4]

$$\hat{\mathbf{P}}_{o,\text{post-injection}} = (\mathbf{I} - \mathbf{W}_s\mathbf{H}_{o\to s})\hat{\mathbf{P}}_o \tag{4.14}$$

Injection: why not just treat state injection as an update?

At this point, we have thrown optimality out the window, and instead opted for a simple update which guarantees a symmetric positive definite (SPD) posterior covariance. To maintain optimality here, we would have to treat $\tilde{x}_s$ as an innovation vector instead, and calculate the posterior state estimate as $\hat{x}_{o,\text{post-injection}} = \hat{x}_o + \mathbf{W}_s\tilde{x}_s$.

There is something deeply disturbing about this supposedly 'optimal' update equation though, namely that it is effectively running an already optimal injection through another update step. What I mean by this is that we already assume the error state estimate to be the (optimal) result of an update step in an external filter (or a consensus of multiple external filters, although I will get back to that). By extension, this estimate should already be weighted against the measurement noise which afflicted the update, as well as the pre-injection localization covariance $\hat{\mathbf{P}}_o$.

At this point, it makes sense for me to trust this estimate to be true (or at least an improvement), and just inject its value directly. What the above equation would have me do, is to instead treat this optimal estimate as a raw measurement, with corresponding noise $\mathbf{R}$.

[4]This form of posterior covariance is of course evil, and a more sound (numerically stable) option is the Joseph form update. In implementation, this is used instead.

By doing this, we are essentially trusting the consensus less, where the relative trust is defined by how much smaller $\tilde{\mathbf{P}}$ is than its corresponding components in $\hat{\mathbf{P}}_o$. This relationship becomes apparent by inspecting the expanded equation for the Kalman update gain:

$$\mathbf{W}_{s,\text{real}} = \hat{\mathbf{P}}_o \mathbf{H}_{o\to s}^\top (\mathbf{H}_{o\to s} \hat{\mathbf{P}}_o \mathbf{H}_{o\to s}^\top + \tilde{\mathbf{P}}_s)^{-1} \tag{4.15}$$

Let's contextualize this with a hypothetical: assuming we have a perfect consensus estimate (with zero covariance), we arrive at the following equation:

$$\mathbf{W}_{s,\text{ideal}} = \hat{\mathbf{P}}_o \mathbf{H}_{o\to s}^\top (\mathbf{H}_{o\to s} \hat{\mathbf{P}}_o \mathbf{H}_{o\to s}^\top)^{-1} \tag{4.16}$$

Premultiplying both sides of this equation with $\mathbf{H}_{o\to s}$, we see that the inverses cancel out to an identity matrix:

$$\mathbf{H}_{o\to s} \mathbf{W}_{s,\text{ideal}} = \mathbf{I} \tag{4.17}$$

Assuming that the observation matrix $\mathbf{H}_{o\to s}$ is simply an observation of individual states, i.e. a number of zero-filled rows, where a single (unique) element of each row is a 1, the conclusion has to be that $\mathbf{W}_{s,\text{ideal}}$ is structured as a transpose of this observation matrix, with corresponding elements also being 1.

E.g. if our localization estimate has five elements, in which only the final element (the heading) is part of the subset s, we arrive at the following solution for $\mathbf{W}_{s,\text{ideal}}$:

$$\mathbf{W}_{s,\text{ideal}} = \begin{bmatrix} a & b & c & d & 1 \end{bmatrix}^\top \tag{4.18}$$

Note that elements a, b, c, d can be anything here, as they are multiplied by zeroes from $\mathbf{H}_{o\to s}$, thus their values are indeterminable from this equation. (In practice, these elements would be defined by whatever correlation the localization filter has learned between heading and the other states.) What is crucial here is the final element being 1, in other words: **we would fully trust the error state estimate, and inject it directly into the fifth element of the localization estimate**. This is the behavior we seek in terms of the posterior state estimate, so why not just set $\tilde{\mathbf{P}}$ to zero?

This is because running an update step with a supposed measurement noise $\mathbf{R}$ of zero, will by design drive the covariance of all measured states to zero. This is of course completely unrealistic, as it would be impossible to use noisy measurements to arrive at a perfect estimate. This puts us in a spot of bother in terms of the injection update. On one hand,

for the sake of getting an accurate injection, we would like to trust it fully, i.e. set $\tilde{\mathbf{P}}_s$ to zero.[5]

On the other hand, we acknowledge that a measurement noise of zero is out of the question for covariance updates. At the same time, we need *some way* to reduce the covariance upon making an injection, since it presumably makes the state estimate more accurate. One way of achieving this would then be to run the state injection using an assumed zero measurement noise, while running its corresponding covariance update with an adequately sized measurement noise instead.

What I eventually landed on was two subtly different approaches here, which I have named direct injection (DI) and weighted injection (WI). Their only difference is how the state injection is calculated.

Direct injection

For direct injection, it was opted to simply inject all error estimates directly into their relevant localization estimate. This effectively disregards any correlations between the error states and the other states in the localization filter. This means that from a KF update perspective, we are manually setting the gain matrix to $\mathbf{H}_{o\to s}^{\top}$, resulting in the following equation:

$$\hat{x}_{o,\text{post-injection}} = \hat{x}_o + \mathbf{H}_{o\to s}^{\top}\tilde{x}_s \tag{4.19}$$

For the covariance update, the standard KF update equations (eqs: 4.12, 4.13, 4.14) are used. This corresponds to 'pretending' that the consensus is actually a measurement, with observation matrix $\mathbf{H}_{o\to s}$ and measurement noise $\tilde{\mathbf{P}}_s$.

Weighted injection

After some observed problems in testing (pure positional injections degrading the velocity estimates of the localization filter, as correlations between position and velocity were not accounted for) it was deemed prudent to find an injection option which more accurately accounted for learned correlations in the localization filter. However, as previously discussed, a regular KF update will account for the injection covariance as well, and by extension trust the injection less. To remedy this, I opted to use the 'ideal' Kalman gain, as previously discussed (eq. 4.16). This gain assumes zero measurement noise, and will

[5]For posterity: it is worth noting that all of this discussion is in the context of two targets. It is perfectly possible that the injection covariance $\tilde{\mathbf{P}}_s$ will be more than small enough if there are lots of targets present. There is some merit to finding a solution here which deals with an arbitrary number of targets, which I leave as a point of future work.

thus inject the estimated error states to their *full* extent, while also inferring changes for the non-estimated states.

In other words, this injection step calculates two different Kalman gain matrices. One of which $\mathbf{W}_{s,\text{real}}$ uses the real injection covariance $\tilde{\mathbf{P}}_s$ (as in eqs. 4.12, 4.13), and the other $\mathbf{W}_{s,\text{ideal}}$ assumes this covariance to be zero (as in eq. 4.16).

Using the ideal gain matrix, we arrive at the following equation for state injection:

$$\hat{x}_{o,\text{post-injection}} = \hat{x}_o + \mathbf{W}_{s,\text{ideal}}\tilde{x}_s \tag{4.20}$$

Just as with the direct injection, we use the Kalman gain $\mathbf{W}_{s,\text{real}}$ for covariance updates, eq. 4.14.

Chapter 5

Theory - Motion and measurement models

For both tracking and localization, modelling the behaviour of the system is a prerequisite for estimation. Between the different solutions discussed in this document, a few common models for both vessel motion and detection will be utilized. These models are presented in this chapter, as well as modified for more specialized needs.

5.1 Motion models

For describing vessel motion, we largely defer to the constant velocity (CV) model in two dimensions. In this section, the CV model, as well as some useful variations are presented.

5.1.1 CV

For our purposes, this motion model has a 4 variable state vector x_{CV}, with position and velocity in x and y directions. It is assumed that the velocity remains constant, although subject to a zero mean, Gaussian (white) noise, v, in the form of acceleration, with variance σ^2_{CV}.

Discretizing this model, assuming a constant ΔT seconds between each time-step, we arrive at the following equations to describe the linear-Gaussian state and its prediction.

$$x_k = \begin{bmatrix} p_x \\ v_x \\ p_y \\ v_y \end{bmatrix} \tag{5.1}$$

$$x_{k+1} = \mathbf{F}_{CV} x_k + v_k \tag{5.2}$$

$$\mathbf{F}_{CV} = \begin{bmatrix} 1 & \Delta T & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta T \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.3}$$

$$v_k \sim \mathcal{N}(0, \mathbf{Q}_{CV}) \tag{5.4}$$

$$\mathbf{Q}_{CV} = \sigma_{CV}^2 \begin{bmatrix} \Delta T^3/3 & \Delta T^2/2 & 0 & 0 \\ \Delta T^2/2 & \Delta T & 0 & 0 \\ 0 & 0 & \Delta T^3/3 & \Delta T^2/2 \\ 0 & 0 & \Delta T^2/2 & \Delta T \end{bmatrix} \tag{5.5}$$

Noting that the prior estimate of the state for some time-step k is distributed as $\mathcal{N}(x_k, \mathbf{P}_k)$, we can find the distribution for the consecutive time-step, x_{k+1}, by combining equations 5.3 and 5.4, which is simply a KF prediction step:

$$\mathcal{N}(x_{k+1}, \mathbf{P}_{k+1}) = \mathcal{N}(\mathbf{F}x_k, \mathbf{F}_{CV}\mathbf{P}_k\mathbf{F}_{CV}^\top + \mathbf{Q}_{CV}) \tag{5.6}$$

5.1.2 CV with heading

In the case of simple tracking, the CV model is often enough, as we rarely care about the heading or rotation of targets[1]. In localization on the other hand, the orientation of the ownship is of utmost importance. This is for example because we need the heading to predict the influence of actuator inputs, or in order to transform onboard sensor readings to a global frame. For the sake of being applicable for fully actuated vessels, which are capable of motion which is decoupled from the heading, it is prudent to model the heading as an independent process.

Working from the CV model, we append the heading estimate φ to the CV state defined in 5.1. We assume that this heading remains constant, although also being subject to a zero mean, white noise rotational speed, with variance σ_φ^2.

[1] Or indeed assume that the direction of travel, IE velocity vector, corresponds perfectly with heading.

Again, we discretize the model, assuming a fixed time ΔT between each timestep:

$$x_k = \begin{bmatrix} p_x & v_x & p_y & v_y & \varphi \end{bmatrix}^\top \tag{5.7}$$

$$x_{k+1} = \mathbf{F}_{\text{CV + heading}} x_k + v_k \tag{5.8}$$

$$\mathbf{F}_{\text{CV + heading}} = \begin{bmatrix} \mathbf{F}_{\text{CV}} & 0_{4x1} \\ 0_{1x4} & 1 \end{bmatrix} \tag{5.9}$$

$$v_k \sim \mathcal{N}(0, \mathbf{Q}_{\text{CV + heading}}) \tag{5.10}$$

$$\mathbf{Q}_{\text{CV + heading}} = \begin{bmatrix} \mathbf{Q}_{\text{CV}} & 0_{4x1} \\ 0_{1x4} & \sigma_\varphi^2 \Delta T \end{bmatrix} \tag{5.11}$$

Similarly to its CV counterpart, we can define a prediction step from some prior $\mathcal{N}(x_k, \mathbf{P}_k)$. This is possible by virtue of the prediction equation 5.9 being a linear transformation subject to Gaussian white noise.

$$\mathcal{N}(x_{k+1}, \mathbf{P}_{k+1}) = \mathcal{N}(\mathbf{F}_{\text{CV + heading}} x_k, \mathbf{F}_{\text{CV + heading}} \mathbf{P}_k \mathbf{F}_{\text{CV + heading}}^\top + \mathbf{Q}_{\text{CV + heading}}) \tag{5.12}$$

This concludes the motion modelling chosen for the ownship localization. When including frequent outside positional fixes, such as external detections or GNSS/RTK, such a navigational scheme becomes viable, at least for position and velocity. I.e. the scheme is able to maintain estimates of these sizes.

The heading remains unobserved, although certain solutions, such as RTK, or multiple GNSS receivers will be able to provide an orientation fix as well, solving the problem. The more obvious solution here is a compass, however I have opted to consider such sensors unavailable for this thesis. This choice is made to emulate an emergency case in which some sensors (IMU, GNSS, compass) have failed. In terms of testing our solution, we have the added bonus of some states in the motion model being directly observable, and others (in our case the heading) which are not. By extension, we get to test the behaviour and performance of our developed system both as an aid for existing estimators (for position/velocity), and as a lone guarantor of estimation (for heading).

Moving forward with the assumption that no direct measurements of the heading are available, we are forced to look into indirect options. One such option is to leverage range-bearing sensors mounted on the ownship, and **some**[2] expectation of what readings these

[2]This would be features of the (static) surroundings for SLAM, or the locations of targets in our case.

sensors are supposed to produce. Cross referencing this expectation with the actual data, we are usually able to solve for the heading, or at least make corrections for small errors. This brings us neatly to the final two motion models I have included, these have additional error states appended, which correspond to inferred corrections in the ownship's localization.

5.1.3 ESCV - Error state constant velocity

The first such motion model, which I have named error state constant velocity (ESCV), has two distinct parts to its state vector. The first of which is a CV model describing target motion, as introduced in 5.1.1, furthermore it *also* contains a full 3 DOF error state of the ownship, i.e. position in x and y, as well as heading.

This is the first motion model presented which fits into the 'compound filter' framework, as presented in sec 4.3.2. In this case the nominal state is the target CV state (eq. 5.1), and the error state becomes the aforementioned position and heading elements. Thus the selector for this error state subset (henceforth referred to as ES), assuming the localization filter uses the *CV + heading* motion model, becomes:

$$\mathbf{H}_{o\to\text{ES}} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.13}$$

These error states are assumed to be constant, although subject to noise for the purposes of prediction. Just like in ESKF, these parameters remain at zero through prediction, until they are inferred during update steps. This may seem excessive, so what follows is some motivation for such a model, in the context of joint localization and tracking.

ESCV: motivation

Say that we detect some target to the left (i.e. wrong bearing) of where we expected, in an ownship (relative) range-bearing scan. This example is illustrated in figure 5.1. For the CV motion model, in which the only state present is the target's, the conclusion has to be that the target has moved, or that the original estimate was wrong (Case B). By extension, one would infer changes in position and velocity in light of this new information. While this is perfect if we assume that the localization estimates are perfect, the flip side is that it can propagate any present error in localization into the tracking estimates. I.e. a **localization, then tracking** scheme will effectively inject any localization error into the track's error.

If we also include ownship error states in the tracker, an equivalent conclusion is that the target has remained stationary, and that it is in fact the ownship which has moved to the right (Case C). Similarly, we could conclude that the ownship has rotated (Case D). Finally,

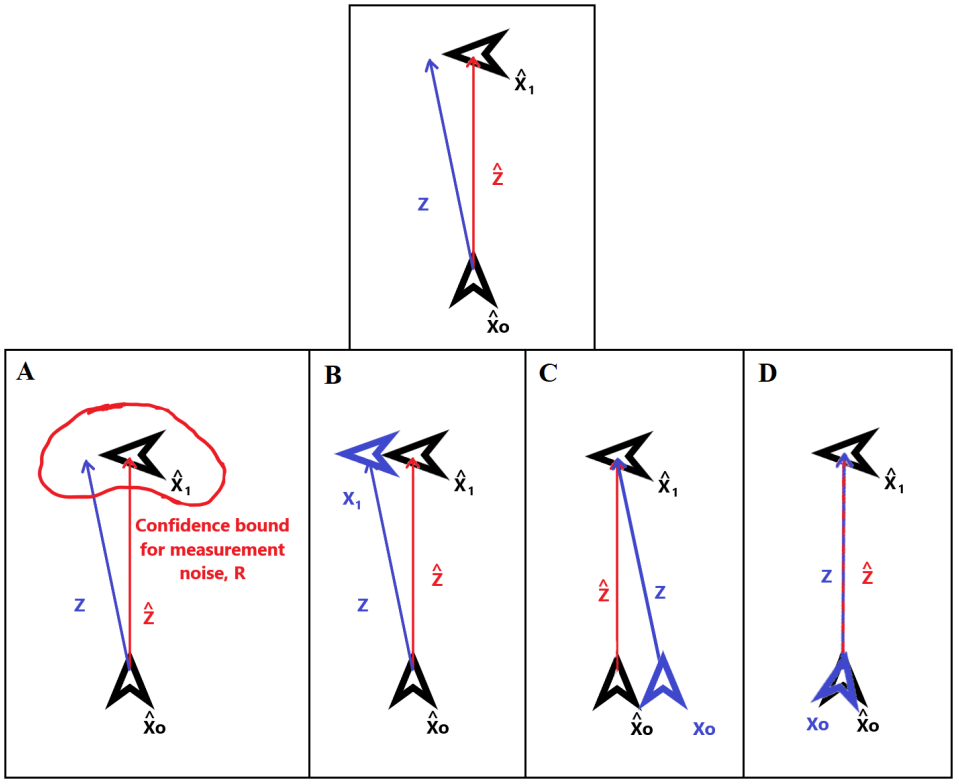

Figure 5.1: Demonstration of ambiguity present in relative range bearing scans. Expected detection is indicated with $\hat{z}$ while the received detection is indicated with z.
Case A: measurement noise has caused the errant detection.
Case B: the target x_1 is in fact left of its estimate $\hat{x}_1$.
Case C: the ownship x_o is in fact right of its estimate $\hat{x}_o$.
Case D: the ownship x_o has a different heading than its estimate $\hat{x}_o$.

we could always conclude that no estimates are wrong, and that it is measurement noise which has caused the errant detection (Case A). This illustrates the intrinsically connected nature of localization and tracking, namely that when using relative (ownship mounted) sensors, it is difficult to figure out whether it is the ownship or the target which has moved.

Although including these error states introduces this ambiguity, which can be tricky to deal with, I regardless deem it a useful modification, in the sense that it allows for acknowledging the possibility that it is in fact the navigation which is errant, and not the target state estimates. Including this at a modelling level could potentially improve tracking performance (by reducing the corruption from injecting errant localization), and using estimates to make corrections in localization. This is **very** similar to including the ownship pose as a set of nuisance parameters, in the style of an SKF. However, there are two key distinctions which separate this scheme from a conventional SKF:

1. The parameters are allowed to be estimated at every timestep, which makes it an expanded state KF or EKF instead.

2. The nuisance parameters are modeled as being driven by Gaussian white noise, i.e. nonzero $\mathbf{Q}_{ES}$. The elements of this matrix act as tuning parameters for the final filter. As opposed to the standard SKF formulation, in which the marginal covariance matrix for the nuisance parameters is constant.

   By extension, each track estimate will maintain a localization covariance matrix (and covariances between target and nuisance parameters) which is decoupled from the localization filter.

After some testing, I came to the conclusion that the process of tuning the matrix $\mathbf{Q}_{ES}$ to get reasonably robust performance was nigh on impossible, and actively shortening my lifespan. I suspect that there is something fundamentally flawed about letting the error state covariance be completely decoupled from the corresponding covariance estimate $\hat{\mathbf{P}}_o$ in the localization filter (and growing according to $\mathbf{Q}_{ES}$). Given that the motion of the ownship and targets is presumed to be independent, it stands to reason that they should be uncorrelated, by extension, there is little sense in letting a filter try to learn such a correlation where there is none.
For this reason, I opted to relax point 2 in the above list, and create additional models which instead inherit the relevant covariance estimates $\hat{\mathbf{P}}_o$, from the localization filter (which is more along the lines of traditional SKF). These models, which are introduced later in this chapter, are appended with "-IC" to imply *inherited covariance*.

This flies in the face of the abstraction in the well-established 'localization, then tracking' schemes[3]. I.e. trackers should **only** do tracking, and that allowing for information to flow from tracking to localization is dangerous. There is certainly some merit to this, as it does open up for the possibility of erroneous tracking propagating back to the localization, effectively corrupting it by trying to *confirm* its own results. In its own right, this is a good reason to avoid the schemes which are suggested in this thesis, however, it is interesting to inspect the performance of such schemes nonetheless.

This challenge can be remedied both by introducing absolute scans (sensors mounted elsewhere, which provide detections which are independent of localization estimates), or by introducing **enough** (independent) targets. Assuming that the association is known, the question changes to "Has **every** target unexpectedly moved in the same direction, or has the ownship moved?". Assuming that the targets are independent of each other, it becomes increasingly unlikely (as the number of targets increase) for every target to suddenly move in a coordinated fashion which can be misunderstood for ownship movement.

Assuming on the other hand that association is NOT known, a large amount of closely grouped targets can be difficult to discern and track, and by extension, inferring ownship localization from ambiguous tracks can be dangerous.

[3] In some sense, this is a step in the direction of SLAM, and once again hints at the inherent connections between JLAT and SLAM.

ESCV: model definition

This model, when discretized, becomes very similar to the CV model defined earlier, with extra parameters $\tilde{p}_{o,x}, \tilde{p}_{o,y}, \tilde{\varphi}_o$ driven by corresponding white noise variances $\sigma^2_{\text{o,pos}}, \sigma^2_{\text{o,rot}}$.

$$x_k = \begin{bmatrix} x_{\text{CV}} \\ x_{\text{ES}} \end{bmatrix} = \begin{bmatrix} p_x & v_x & p_y & v_y & \tilde{p}_{o,x} & \tilde{p}_{o,y} & \tilde{\varphi}_o \end{bmatrix}^\top \tag{5.14}$$

$$x_{k+1} = \mathbf{F}_{\text{ESCV}} x_k + v_k \tag{5.15}$$

$$\mathbf{F}_{\text{ESCV}} = \begin{bmatrix} \mathbf{F}_{\text{CV}} & \mathbf{0}_{4x3} \\ \mathbf{0}_{3x4} & \mathbf{I}_{3x3} \end{bmatrix} \tag{5.16}$$

$$v_k \sim \mathcal{N}(0, \mathbf{Q}_{\text{ESCV}}) \tag{5.17}$$

$$\mathbf{Q}_{\text{ES}} = \Delta T \cdot diag(\sigma^2_{\text{o,pos}}, \sigma^2_{\text{o,pos}}, \sigma^2_{\text{o,rot}}) \tag{5.18}$$

$$\mathbf{Q}_{\text{ESCV}} = \begin{bmatrix} \mathbf{Q}_{\text{CV}} & \mathbf{0}_{4x3} \\ \mathbf{0}_{3x4} & \mathbf{Q}_{\text{ES}} \end{bmatrix} \tag{5.19}$$

Noting the linear-Gaussian form of the prediction equation 5.15, this is equivalent to the previously defined prediction equations (such as eq. 5.6) but now using the new $\mathbf{F}_{\text{ESCV}}$ and $\mathbf{Q}_{\text{ESCV}}$ matrices.

5.1.4 RESCV - Reduced error state constant velocity

The second model introduced, the reduced ESCV model (RESCV), is also a CV model with an appended error state. This is a reduction of the aforementioned ESCV model, which now only contains an error state for ownship heading. This makes the total state vector a stack of the typical 2 DOF, 4 variable CV model, as well as the ownship heading error $\tilde{\varphi}_o$. Again, a brief motivation is included, as well as the model equations.

As with ESCV, the RESCV fits into the 'compound filter' framework, in which the nominal target state is the usual CV model, and the error state is a subset RES of the localization state (assumed to be using the CV + heading model). This results in the following selector:

$$\mathbf{H}_{o \to \text{RES}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.20}$$

RESCV: motivation

While the ESCV model is useful in the sense that it can infer 3 DOF corrections on the localization results, it introduces a substantial amount of ambiguity in the measurement models. I.e. any error in the position of a detected target can be attributed to four potentially interchangeable effects (see figure 5.1, as discussed for ESCV):

1. The target has moved, i.e. make correction to target position, infer target velocity.
2. The ownship has moved, i.e. make correction to ownship position, infer ownship velocity.
3. The ownship has rotated, i.e. make correction to ownship heading.
4. The target is in fact exactly where we expect, and it is measurement noise which has produced the errant detection.

Using the measurement and motion models in some filter update, i.e. letting the tracker calculate some (supposedly optimal) combination of these effects is no small feat. When we furthermore consider the number of tuning parameters, there is some merit to reducing the complexity here. For this reason, I have included a **reduced** error state constant velocity model (RESCV, henceforth) which only includes the targets CV state, as well as an error state for the ownship heading. This corresponds to eliminating case C from figure 5.1, which should help limit these effects being mistaken for each other.

This is a useful simplification, especially if we assume that some absolute positional fixes are available for the ownship. (For example from detections of the ownship produced by shore mounted sensors or GNSS.) Such positional fixes would be able to stabilize the position and velocity estimates of the ownship, leaving only the heading as an unknown. In such a case the RESCV motion model becomes a purpose built model, which ”fills the gap” for heading estimation.

RESCV: model definition

As with ESCV, we define the discretized model as follows:

$$x_k = \begin{bmatrix} x_{\text{CV}} \\ x_{\text{RES}} \end{bmatrix} = \begin{bmatrix} p_x & v_x & p_y & v_y & \tilde{\varphi}_o \end{bmatrix}^\top \tag{5.21}$$

$$x_{k+1} = \mathbf{F}_{\text{RESCV}} x_k + v_k \tag{5.22}$$

$$\mathbf{F}_{\text{RESCV}} = \begin{bmatrix} \mathbf{F}_{\text{CV}} & \mathbf{0}_{4x1} \\ \mathbf{0}_{1x4} & 1 \end{bmatrix} \tag{5.23}$$

$$v_k \sim \mathcal{N}(0, \mathbf{Q}_{\text{RESCV}}) \tag{5.24}$$

$$\mathbf{Q}_{\text{RESCV}} = \begin{bmatrix} \mathbf{Q}_{\text{CV}} & \mathbf{0}_{4x1} \\ \mathbf{0}_{1x4} & \Delta T \sigma^2_{\text{o,rot}} \end{bmatrix} \tag{5.25}$$

Again, noting the linear-Gaussian prediction equation 5.22, we can define a KF prediction with $\mathbf{F}_{\text{RESCV}}$ and $\mathbf{Q}_{\text{RESCV}}$, which is omitted for brevity.

5.1.5 ESCV-IC - ESCV with inherited nuisance covariance

Both the RESCV and ESCV models were found to have lackluster performance. To solve some of these problems, a new variant of both ESCV and RESCV was introduced, namely the *inherited covariance* (-IC) varieties. This new model was not permitted to learn correlations between target and ownship error states, and furthermore inherits the nuisance covariance estimate $\hat{\mathbf{P}}_{\tilde{x}}$ from relevant elements of the localization filter covariance estimate: $\hat{\mathbf{P}}_o$. This means that at any point in time, the estimated prior covariance matrix $\hat{\mathbf{P}}_{\text{some-IC}}$ of such an *inherited covariance* model will look like:

$$\hat{\mathbf{P}}_{\text{some-IC}} = \begin{bmatrix} \hat{\mathbf{P}}_{\text{CV}} & \mathbf{0}_{\text{CV},\tilde{x}} \\ \mathbf{0}_{\tilde{x},\text{CV}} & \hat{\mathbf{P}}_{\tilde{x}} \end{bmatrix} \tag{5.26}$$

Since the ESCV-IC shares the same subset of error states as the ESCV, it uses the same selector, as defined in eq. 5.13. As for the inherited covariance $\hat{\mathbf{P}}_{\tilde{x}}$, this can be fetched from the localization covariance $\hat{\mathbf{P}}_o$ using this selector, i.e.

$$\hat{\mathbf{P}}_{\tilde{x}} = \mathbf{H}_{o\to\text{ES}} \hat{\mathbf{P}}_o \mathbf{H}^\top_{o\to\text{ES}} \tag{5.27}$$

ESCV-IC: Motivation

As eluded to previously, tuning the process noise parameters of the nuisance parameters, i.e. $\sigma^2_{\text{o,pos}}, \sigma^2_{\text{o,rot}}$ turned out to be tricky. The observed behaviour (which I will elaborate on in the analysis section) was that the accumulated variance estimates for the nuisance parameters reached wholly unreasonable magnitudes, which caused similarly large injection steps.

This was in part mitigated by similarly large covariances between nuisance and target states (i.e. $\hat{\mathbf{P}}_{1:4,5:7}$ and $\hat{\mathbf{P}}_{5:7,1:4}$ for ESCV). When the system behaviour matched these

learned correlations, filtering performance was fine, however, the flip side is that any aberrant vessel behaviour caused a significant estimation error, and occasionally even track loss.

The conclusion at the time was that these learned correlations were nonsensical, particularly when considering that the motion of any vessel is assumed to be independent of the other vessels. By extension, letting the filter *learn* such (false) correlations of inter-vessel behaviour was subjecting it to worse performance. To sidestep this, I opted to bar the filter from learning such correlations, by manually setting the target-nuisance covariance matrices to zero. These elements are denoted by $\mathbf{0}_{\tilde{x},\mathrm{CV}}$ and $\mathbf{0}_{\mathrm{CV},\tilde{x}}$ in eq. 5.26.

Note that letting a KF learn correlations that are not necessarily *present* is not always dangerous. In fact, the full-filter solution presented in this thesis seems to handle it just fine (barring a tendency for over-confidence...), why is it then a problem for the ESCV and RESCV models? I posit that this is because the full filter also gets ownship-specific position updates, which helps to keep the localization covariance bounded, and furthermore hinders the learning of errant correlations. As the ESCV and RESCV models do not get this sanity check, their ownship (co)variance estimates are much more likely to spiral out of control. This stands as another argument for the IC variety, namely that it instead uses the localization covariance, which should remain bounded as it is periodically updated with absolute detections.

The EKF also has a few known consistency problems associated with the linearization points chosen for updates, which are inherent to the nonlinear nature of the (relative) measurement functions. [24] explores this in the context of SLAM (and suggests some solutions). In the more generalized terms of EKFs learning spurious correlations where there are none (and its effect on filter consistency), it stands as an interesting read.

A second and more profound question is how one should model the uncertainty in localization error. In ESCV and RESCV, it was arbitrarily decided to let these parameters also be subject to process noise, i.e. nonzero elements in $\mathbf{Q}$ for these parameters. This implies that the (nuisance) localization error is ever growing, and has to be kept bounded by the update steps done in the tracking filter.

Another more problematic aspect of this, is that each ESCV or RESCV model maintains a (redundant) covariance estimate which is completely independent of the corresponding estimate in the localization filter. In our case, it would be prudent to trust the localization filter to do its job, thus it would make sense to trust its covariance estimate to be consistent. We can leverage this information by using $\hat{\mathbf{P}}_o$ as a measure of *How wrong can the localization estimate be*. By forwarding this information to the tracking filters, they are provided with what should be an updated, optimal estimate of $\hat{\mathbf{P}}_{\tilde{x}}$.

These changes correspond to a huge step in the direction of an SKF-like filter. At this point, the only real distinction is the fact we still calculate online estimates of the nuisance

parameters.[4] Still, we have achieved a less complex system in the sense that we now have fewer tuning parameters, and can skip calculation of large parts of the covariance estimate posteriors.

ESCV-IC: model definition

For the model definition I largely defer to the equations listed in the ESCV model (eqs. 5.14, 5.15, 5.16). The exceptions being the distribution of the noise vector v_k, as well as custom equations for prior $\hat{\mathbf{P}}_{k|k-1}$ and posterior $\hat{\mathbf{P}}_{k|k}$ covariances:

$$v_k \sim \mathcal{N}(0, \mathbf{Q}_{\text{ESCV-IC}}) \tag{5.28}$$

$$\mathbf{Q}_{\text{ESCV-IC}} = \begin{bmatrix} \mathbf{Q}_{\text{CV}} & \mathbf{0}_{4x3} \\ \mathbf{0}_{3x4} & \mathbf{0}_{3x3} \end{bmatrix} \tag{5.29}$$

$$\hat{\mathbf{P}}_{k|k-1} = \begin{bmatrix} \mathbf{F}_{\text{CV}}\mathbf{H}_{\text{ESCV}\to\text{CV}}\hat{\mathbf{P}}_{k-1|k-1}\mathbf{H}_{\text{ESCV}\to\text{CV}}^{\top}\mathbf{F}_{\text{CV}}^{\top} + \mathbf{Q}_{\text{CV}} & \mathbf{0}_{4x3} \\ \mathbf{0}_{3x4} & \mathbf{H}_{\text{o}\to\text{ES}}\hat{\mathbf{P}}_{o,k|k-1}\mathbf{H}_{\text{o}\to\text{ES}}^{\top} \end{bmatrix} \tag{5.30}$$

$$\hat{\mathbf{P}}_{k|k,\text{after update}} = \hat{\mathbf{P}}_{k|k-1} - \mathbf{W}\mathbf{H}\hat{\mathbf{P}}_{k|k-1} \tag{5.31}$$

$$\hat{\mathbf{P}}_{k|k,\text{after reset}} = \begin{bmatrix} \mathbf{H}_{\text{ESCV}\to\text{CV}}\hat{\mathbf{P}}_{k|k,\text{after update}}\mathbf{H}_{\text{ESCV}\to\text{CV}}^{\top} & \mathbf{0}_{4x3} \\ \mathbf{0}_{3x4} & \mathbf{H}_{\text{o}\to\text{ES}}\hat{\mathbf{P}}_{o,k|k}\mathbf{H}_{\text{o}\to\text{ES}}^{\top} \end{bmatrix} \tag{5.32}$$

Note the $\mathbf{0}_{3x3}$ element in the bottom right of the process noise matrix $\mathbf{Q}_{\text{ESCV-IC}}$ (eq. 5.29). This is included to indicate that the covariance of these elements **is not supposed to be inflated with prediction steps, but rather overwritten with fresh covariance estimates from the localization filter**. In this sense, we could conceivably have skipped building this matrix for all states, and rather just re-used the $\mathbf{Q}_{\text{CV}}$ matrix from earlier. I opted against this for the sake of compatibility with existing software, which means that the prior covariance is actually calculated for all 4 block matrices. After this their elements are overwritten to reflect the prior covariance equation (eq. 5.30).

[4] This is **very** similar to an ISKF or PSKF as introduced by Brink, [7]. Formulating the (R)ESCV-IC to match these filter structures (and doing a corresponding comparison) would be an interesting point of future work.

Note that I have included explicit equations for the posterior covariance after an update (i.e. before reset) and after a reset step. This serves as a reminder that in the interim, *the entirity* of the covariance matrix can be populated. This is also the only time in which the error state estimates $\hat{x}_{\text{ES}}$ can have nonzero values.

Furthermore, note that for both the prior covariance $\hat{\mathbf{P}}_{k|k-1}$ and the 'after-reset' posterior $\hat{\mathbf{P}}_{k|k,\text{after reset}}$, the bottom right block matrix is set as the ES subset of the localization covariance $\hat{\mathbf{P}}_o$, i.e. $\mathbf{H}_{\text{o}\to\text{ES}}\hat{\mathbf{P}}_o\mathbf{H}_{\text{o}\to\text{ES}}^\top$. This is how the 'inherited covariance' property of this filter is imposed, and constitutes the model being provided with *outside information* in the form of this covariance estimate.

5.1.6 RESCV-IC - RESCV with inherited nuisance covariance

Just as with the changes made from ESCV to ESCV-IC, this model is just a variant of RESCV which assumes zero process noise for the heading error state, and instead inherits the nuisance covariance from the localization filter. Finally it is not permitted to learn correlations between the CV state and the error state.

RESCV-IC: model definition

Again, I defer to the model definition in the RESCV section, i.e. equations 5.21, 5.22, 5.23. Barring this, the process noise and covariance estimates bear re-definition:

$$v_k \sim \mathcal{N}(0, \mathbf{Q}_{\text{RESCV-IC}}) \tag{5.33}$$

$$\mathbf{Q}_{\text{RESCV-IC}} = \begin{bmatrix} \mathbf{Q}_{\text{CV}} & \mathbf{0}_{4x1} \\ \mathbf{0}_{1x4} & 0 \end{bmatrix} \tag{5.34}$$

$$\hat{\mathbf{P}}_{k|k-1} = \begin{bmatrix} \mathbf{F}_{\text{CV}}\hat{\mathbf{P}}_{\text{CV},k-1|k-1}\mathbf{F}_{\text{CV}}^\top + \mathbf{Q}_{\text{CV}} & \mathbf{0}_{4x1} \\ \mathbf{0}_{1x4} & \mathbf{H}_{\text{o}\to\text{ES}}\hat{\mathbf{P}}_{o,k|k-1}\mathbf{H}_{\text{o}\to\text{ES}}^\top \end{bmatrix} \tag{5.35}$$

$$\hat{\mathbf{P}}_{k|k,\text{after update}} = \hat{\mathbf{P}}_{k|k-1} - \mathbf{W}\mathbf{H}\hat{\mathbf{P}}_{k|k-1} \tag{5.36}$$

$$\hat{\mathbf{P}}_{k|k,,\text{after reset}} = \begin{bmatrix} \mathbf{H}_{\text{RESCV}\to\text{CV}}\hat{\mathbf{P}}_{k|k,\text{after update}}\mathbf{H}_{\text{RESCV}\to\text{CV}}^\top & \mathbf{0}_{4x1} \\ \mathbf{0}_{1x4} & \mathbf{H}_{\text{o}\to\text{RES}}\hat{\mathbf{P}}_{o,k|k}\mathbf{H}_{\text{o}\to\text{RES}}^\top \end{bmatrix} \tag{5.37}$$

5.2 Measurement models

Moving along to measurement models, these provide the probabilistic connection between the target state, and detections produced by sensors. From a KF perspective, one might state that a motion model handles the prediction step, while a measurement model handles the update step. This section is made somewhat complicated due to the presence of many different state vectors, depending on the choice of motion model. Each type of state vector will have its own measurement function, hence the need for multiple measurement functions associated with each measurement model.

5.2.1 Cartesian, absolute

Starting with the simplest measurement model: the Cartesian absolute model. It is intended to be equivalent to some shore mounted sensor (with known and constant pose) which detects the position of both the ownship and targets. For all the aforementioned motion models, the measurement function $h_{\text{cart}}(x)$ remains the same:

$$h_{\text{cart}}(x) = \begin{bmatrix} p_x \\ p_y \end{bmatrix} \tag{5.38}$$

Noting that any sensor is subject to noise, we model this in the Cartesian absolute model as a zero mean, Gaussian white noise, defined with the covariance matrix $\mathbf{R}_{\text{cart}}$.

This makes the distribution of Cartesian absolute detection the following, where x is the true target state:

$$z_{\text{cart}} \sim \mathcal{N}(h_{\text{cart}}(x), \mathbf{R}_{\text{cart}}) \tag{5.39}$$

For the sake of simplicity, we define the measurement uncertainty as being constant, equal in x and y directions, and finally both directions being independent of each other. This makes the covariance matrix diagonal, and equal to:

$$\mathbf{R}_{\text{cart}} = \begin{bmatrix} \sigma^2_{\text{cart}} & 0 \\ 0 & \sigma^2_{\text{cart}} \end{bmatrix} \tag{5.40}$$

Where the parameter σ^2_{cart} corresponds to the variance of the detection error, in m^2.

For the sake of later convenience, I also include some Jacobians of the measurement function h_{cart} in this section. For each motion model X, the jacobian of h_{cart} wrt. its state vector is denoted as $\mathbf{H}_{\text{X}\to\text{cart}}$, these matrices are almost identical, with varying lengths of appended zeros, but all included for completeness.

$$\mathbf{H}_{\text{CV}\to\text{cart}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{5.41}$$

$$\mathbf{H}_{\text{CV + heading}\to\text{cart}} = \begin{bmatrix} \mathbf{H}_{\text{CV}\to\text{cart}} & \mathbf{0}_{2x1} \end{bmatrix} \tag{5.42}$$

$$\mathbf{H}_{\text{ESCV}\to\text{cart}} = \begin{bmatrix} \mathbf{H}_{\text{CV}\to\text{cart}} & \mathbf{0}_{2x3} \end{bmatrix} \tag{5.43}$$

$$\mathbf{H}_{\text{RESCV}\to\text{cart}} = \begin{bmatrix} \mathbf{H}_{\text{CV}\to\text{cart}} & \mathbf{0}_{2x1} \end{bmatrix} \tag{5.44}$$

5.2.2 Range bearing, relative

The second and more complex model included for our purposes is the range-bearing relative model. This is intended to be equivalent to a ownship-mounted range bearing sensor (such as radar or lidar), which is capable of detecting other targets. By extension, its detections are dependent both on the ownship pose x_o, as well as the target state x_t.

We begin with the true measurement function $h_{\text{rel}}(x_o, x_t)$, i.e. assuming that both the true target state x_t and sensor pose x_o are available:

$$h_{\text{rel}}(x_o, x_t) = \begin{bmatrix} ||p_t - p_o||_2 \\ \angle(p_t - p_o) - \varphi_o \end{bmatrix} \tag{5.45}$$

Note the shorthand p_t and p_o here, these are the elements of x_t and x_o which correspond to the position, s.t. $p_t = [p_{t,x}, p_{t,y}]^\top$, etc. Also note the angle function, $\angle(.)$, which for all intents and purposes is the atan2$(.)$ function, i.e. the four quadrant arctan function.

As for its Jacobians $\mathbf{H}_{\text{MM}\to\text{rel}}$, I leave these as implicitly defined as the measurement function, differentiated w.r.t. the motion model (MM) state. For motion models which include certain error states of x_o, these are of course also differentiated w.r.t. their corresponding localization states (i.e. for $\tilde{\varphi}_o$ differentiate $h_{\text{rel}}(.)$ w.r.t. φ_o). Such that the resulting Jacobian also defines the connection between the relevant error state and the provided measurement.

As with the Cartesian absolute measurement model, we subject this model to zero mean Gaussian noise, defined by the covariance matrix $\mathbf{R}_{\text{rel}}$, which is made up of constant radial and tangential variances, $\sigma^2_{\text{rel,r}}, \sigma^2_{\text{rel},\theta}$, $[m^2, rad^2]$:

$$\mathbf{R}_{\text{rel}} = \begin{bmatrix} \sigma^2_{\text{rel,r}} & 0 \\ 0 & \sigma^2_{\text{rel},\theta} \end{bmatrix} \tag{5.46}$$

Combining equations 5.45 and 5.46, we arrive at the distribution for a detection z_{rel} produced by the relative range bearing measurement model:

$$z_{\text{rel}} \sim \mathcal{N}(h_{\text{rel}}(x_o, x_t), \mathbf{R}_{\text{rel}}) \tag{5.47}$$

Chapter 6

Theory - Localization

For this chapter, we consider localization in the plane for our ownship, a surface vessel. This corresponds to a 3 DOF model with two Cartesian variables for position as well as one rotational variable for heading. In general, I denote these variables with the vector $x_o = [p_x, p_y, \varphi]^\top$, of which its elements p_x, p_y are positional variables, in meters, and φ is the heading, in radians.

For the purposes of this thesis, I define *localization* as the task of maintaining estimates of this 3 DOF ownship pose. As it is prudent to calculate reasonably precise predictions of this pose while the ownship is moving, we also include estimating velocities v_x, v_y as part of this estimation task. This makes the state vector equal to $[p_x, v_x, p_y, v_y, \varphi]^\top$. Assuming that the ownship velocity and heading are more or less constant, this also provides us with the Markov property, i.e. that a current state estimate is the only thing necessary to calculate a prediction. This is particularly useful from a modeling perspective, which will be elaborated on later.

Furthermore, it is particularly useful to know something about how accurate our localization estimate $\hat{x}_o$ is (i.e. navigational uncertainty), for this reason, we again expand the localization task to also include estimation of a covariance matrix $\hat{\mathbf{P}}_o$. Combining these two changes, the localization task becomes to estimate the following Gaussian:

$$\hat{x}_o \sim \mathcal{N}(x_o, \hat{\mathbf{P}}_o) \tag{6.1}$$

At this point, we have defined localization as estimation of a multivariate Gaussian with the Markov property, which is a prime task for a Kalman filter. In the remainder of this chapter, I will elaborate on how the previously defined motion and measurement models defined in chapter 5 can be leveraged to this end.

Note again that inertial measurement such as an IMU is assumed to be unavailable. I must then reiterate the assumption that both velocites and the heading are constant (driven by zero-mean noise). A typical extension to our localization state would be to include accelerations and angular velocities (potentially with bias estimates) as provided by an IMU, at which point an ESKF-style filter would be applicable, and we could relax the assumption to 'the IMU measures acceleration and angular rates accurately (subject to noise and bias)'.

With that being said, this was opted against, and we are forced to include other ways of observing the position, velocity and heading in order to make this filtering problem solvable. It was opted to only include periodic absolute positional measurements, which would make position and velocity observable. By extension, the heading is *not* directly observable. This puts our localization filter in a 'built to fail' situation, since it has no inherent way of updating the heading estimate.

An obvious solution would be to include a compass in order to make heading observable, however, as previously mentioned, this was also opted against (see sec. 5.1.2 for details). The saving grace being that the heading is observable via range-bearing measurements, since the heading can be inferred from the expected range-bearing measurements of known targets, in this sense, we have constructed a localization problem which actually requires tracking results in order to be solved.

6.1 Dead reckoning with positional updates

Assuming that periodic, absolute positional measurements are available, we can define a Kalman filter structure which solves the localization problem (barring heading). Such a positional measurement can stem from any number of systems, for example ownship detections produced by shore mounted sensors or onboard GNSS/RTK. For modelling this detection, we employ the *Cartesian absolute* measurement model defined in section 5.2.1. Combining the *CV with heading* motion model with this measurement model, we arrive at the following KF equations for predictions and updates, in which the prior state estimate $\{\hat{x}_{o,k-1}, \hat{\mathbf{P}}_{o,k-1}\}$ and an ownship detection $z_{cart,k}$ are presumed available.

6.1.1 prediction and update steps: KF level

$$\begin{aligned}\mathcal{N}(\hat{x}_{k|k-1}, \hat{\mathbf{P}}_{k|k-1}) = \mathcal{N}(&\mathbf{F}_{\text{CV + heading}}\hat{x}_{k-1|k-1}, \\ &\mathbf{F}_{\text{CV + heading}}\hat{\mathbf{P}}_{k-1|k-1}\mathbf{F}_{\text{CV + heading}}^{\top} + \mathbf{Q}_{\text{CV + heading}})\end{aligned} \tag{6.2}$$

$$\nu_{\text{cart,k}} = z_{cart,k} - h_{\text{cart}}(\hat{x}_k) \tag{6.3}$$

Conveniently enough, $h_{\text{cart}}(x_k)$ is a linear function on x_k, which spares us the need to estimate a jacobian $\mathbf{H}_{\text{cart}}$ for computing the update step. For the *CV with heading* state vector, this linear function is:

$$\mathbf{H}_{\text{cart}} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{6.4}$$

Noting that $\nu_{\text{cart,k}}$ is distributed as $\mathcal{N}(0, \mathbf{S}_{\text{cart,k}})$, in which $\mathbf{S}_{\text{cart,k}}$, the innovation covariance, is defined as:

$$\mathbf{S}_{\text{cart,k}} = \mathbf{H}_{\text{cart}} \hat{\mathbf{P}}_k \mathbf{H}_{\text{cart}}^{\top} + \mathbf{R}_{\text{cart}} \tag{6.5}$$

Filling in the remaining, bog-standard KF update equations:

$$\mathbf{W}_{\text{cart,k}} = \hat{\mathbf{P}}_{k|k-1} \mathbf{H}_{\text{cart}}^{\top} \mathbf{S}_{\text{cart,k}}^{-1} \tag{6.6}$$

$$\mathcal{N}(\hat{x}_{k|k}, \hat{\mathbf{P}}_{k|k}) = \mathcal{N}(\hat{x}_{k|k-1} + \mathbf{W}_{\text{cart,k}} \nu_{\text{cart,k}}, (\mathbf{I} - \mathbf{W}_{\text{cart,k}} \mathbf{H}_{\text{cart}}) \hat{\mathbf{P}}_{k|k-1}) \tag{6.7}$$

6.1.2 prediction and update steps: model level

At last, for the sake of abstraction, observe that equation 6.2 effectively accepts the following parameters: $\{\hat{x}_{k-1|k-1}, \hat{\mathbf{P}}_{k-1|k-1}\}, \{\mathbf{F}_{\text{CV + heading}}, \mathbf{Q}_{\text{CV + heading}}\}$. Of which, the first set is the posterior state estimate of timestep $k-1$, and the second set is matrices associated with the motion model in question. These matrices are a function of internal model parameters such as covariances, as well as the time difference ΔT between steps k and $k-1$. Letting the prediction be a function of the motion model $M_{\text{motion,odo}}$, and internalising any parameters which are motion model specific, we arrive at the following definition:

$$\{\hat{x}_{k|k-1}, \hat{\mathbf{P}}_{k|k-1}\} = M_{\text{motion,CV+heading\_predict}}(\{\hat{x}_{k-1|k-1}, \hat{\mathbf{P}}_{k-1|k-1}\}, \Delta T) \tag{6.8}$$

In a similar fashion, we can define an update function which belongs to the measurement model $M_{\text{meas,cart}}$, comprised of the calculations made in equations 6.3, 6.5, 6.6, 6.7, which consumes the prior (predicted) state estimate $\{\hat{x}_{k|k-1}, \hat{\mathbf{P}}_{k|k-1}\}$ and a detection $z_{cart,k}$, and returns a posterior state estimate $\{\hat{x}_{k|k}, \hat{\mathbf{P}}_{k|k}\}$.

$$\{\hat{x}_{k|k}, \hat{\mathbf{P}}_{k|k}\} = M_{\text{meas,cart\_update}}(\{\hat{x}_{k|k-1}, \hat{\mathbf{P}}_{k|k-1}\}, z_{cart,k}) \tag{6.9}$$

This syntax is preferred because it internalises (abstracts away) the model specific calculations, which for measurements are: $h_{\text{some\_model}}(.), \mathbf{H}_{\text{some\_model}}, \mathbf{S}_{\text{some\_model},k}, \mathbf{W}_{\text{some\_model},k}$, and leaves us with the higher level action of "Hey, I have a new measurement, please update the state.". In the remainder of this thesis, I will endeavour to use syntax similar the one defined in equations 6.8, 6.9 wherever possible.

At this point, we are left with a localization scheme able to stabilize position and velocity estimates, while the heading estimate still blindly assumes no change over time, and will inevitably diverge. In the joint solutions (i.e. localization and tracking) suggested in this thesis, two different schemes for stabilizing heading (simultaneously with multi target tracking, MTT) will be explored.

Chapter 7

Theory - Tracking

For the work done on target tracking in this thesis, several parallelized Markov Chain 1 IPDA trackers are employed. For derivation of the IPDA tracker, I defer to [18].

For the sake of brevity, only a brief introduction of the IPDA tracker's features and core functions is included in this chapter. For the sake of abstraction, we define these functions i.e. *prediction* and *update* as functions of *some* target state combined with a motion model and a measurement model. This is useful for a number of reasons, including but not limited to, a stateless *one size fits all* implementation, as well as support for arbitrary motion and measurement models.

7.1 IPDA features

7.1.1 State filtering

One defining feature of any tracker is some kind of state estimation. In the case of our IPDA implementation, this is achieved with a KF esque solution, which provides a framework with which to estimate both target state and covariance. For the sake of abstraction, the interface for this filtering is the Kalman prediction and update functions, as should be familiar. We note the recurring theme in which motion models are used for prediction steps, while measurement models are used for update steps.

7.1.2 Data association

The second defining feature of tracking is the need for data association, i.e. solving the problem of: "I have lots of measurements, and it is uncertain which, if any, originate from my target.". In the IPDA case, this is solved with *probablistic data association*, i.e. we calculate the probability of each measurement stemming from the target (association probability), and use these probabilities to calculate some weighted average with which to update our target state.

7.1.3 Validation gating

In some cases, looking at *all* measurements from a given scan is both unnecessary and wasteful. For example, any measurements with sufficiently large distance to the current target state estimate cannot possibly have originated from the target. Intuitively, such measurements will have negligible association probability, and by extension have a negligible effect on the updated target state estimate.

Validation gating allows us to disregard such measurements, by imposing a minimal required gate probability, p_{gate} and simply dropping any measurement below this threshold. This allows the tracker to only regard the *relevant* measurements for some target, making for more efficient computation.

We note from the measurement model section 5.2 that the innovation ν, i.e. the difference between some measurement and its predicted value, is distributed according to a zero mean multivariate Gaussian with covariance $\mathbf{S}$. Leveraging this distribution, one can evaluate the probability of some measurement stemming from the target. Doing this for every ν_j in the set of innovations $\{\nu_1, ..., \nu_J\}$, we can drop the ones with a probability below our designated threshold.

A computationally sound way of calculating this gate threshold is to calculate the NIS (normalized innovation squared) associated with each measurement, and checking this against the gate size (which is a function of p_{gate} and measurement dimensionality). This is an equivalent test, and corresponds to evaluating one sided χ^2 confidence interval instead of a multivariate Gaussian PDF.

7.1.4 Existence probability estimation

What sets the IPDA apart from its simpler PDAF variety, is the inclusion of target existence probability estimation. In essence, this is a scalar variable $\hat{p}$, predicted using a Markov chain, and updated by cross referencing the association probabilities against the 'detected and gated probability', which says something about whether the gated measurements match up with the estimated target state as well as one would expect.

An intuitive way to think about this is using the 'detected and gated probability' ($P_{\text{gate}} \cdot P_d$) associated with some sensor (i.e. measurement model). For example, if some sensor has a P_d of 0.9 and a P_{gate} of 0.9, we would expect the association probabilities of the gated measurements to (on average) add up to around 0.81. If the sum of association probabilities is lower than this, it amounts to saying "We would expect more plausible detections to be gated for a *real target*, thus this scan does **not** adequately prove the existence of our target." and as a result we would reduce the target existence probability estimate to reflect this. Conversely, if the sum exceeds this expected value, we treat it as an indication that the target does in fact exist, and would increase our estimate.[1]

The sum of association probabilities is profoundly important in this sense, since it becomes a measure of whether a track estimate has plausible (enough) detections in a scan. Conversely the 'detected and gated probability' becomes the expected probability of such plausible detections being present for a *real target* (and by extension also a supposedly accurate track estimate).

7.2 IPDA functions

For the sake of abstraction, I will outline some syntax for high-level IPDA functions in this section, i.e. predict and update. These functions operate on the target state estimate, $\{\hat{x}_{k-1|k-1}, \hat{P}_{k-1|k-1}, \hat{p}_{k-1|k-1}\}$, here denoted as the posterior estimate of timestep $k-1$. As well as the time in seconds between steps k and $k-1$, i.e. ΔT, and finally arbitrary motion and measurement models, M_{motion} and $M_{\text{measurement}}$, respectively.

Note that the IPDA is this thesis can be used with motion models that utilize 'compound filters' as introduced in section 4.3.2 (such as ESCV, RESCV, ESCV-IC, RESCV-IC), thus we also need to define some functions in order to interact with the error-state part of these filters, i.e. get_error_state and reset_error_state functions.

7.2.1 Predict

Starting with prediction, which propagates some state estimate $\{\hat{x}_{k-1|k-1}, \hat{P}_{k-1|k-1}, \hat{p}_{k-1|k-1}\}$ some time ΔT forward, according to a motion model M_{motion}. I define this as:

$$\begin{aligned} \{\hat{x}_{k|k-1}, \hat{P}_{k|k-1}, \hat{p}_{k|k-1}\} = \\ \text{IPDA\_predict}(\{\hat{x}_{k-1|k-1}, \hat{P}_{k-1|k-1}, \hat{p}_{k-1|k-1}\}, \Delta T, M_{\text{motion}}) \end{aligned} \tag{7.1}$$

Or more specifically, noting that our IPDA implementation simply uses the predict functions of its submodules:

[1] This is a gross simplification for the sake of illustration, and I defer to [18] for a more detailed derivation.

$$\begin{aligned} \mathcal{N}(\hat{x}_{k|k-1}, \hat{P}_{k|k-1}) &\sim M_{\text{motion\_predict}}(\mathcal{N}(\hat{x}_{k-1|k-1}, \hat{P}_{k-1|k-1}), \Delta T) \\ \hat{p}_{k|k-1} &= \text{Markov\_predict}(\hat{p}_{k-1|k-1}, \Delta T) \end{aligned} \tag{7.2}$$

7.2.2 Update

Completely analogously to the predict step, we also define an update step as follows, with the following parameters:

- $Z_{k,\text{meas\_model}}$: A scan (set of measurements) produced by some measurement model.
- $\{\hat{x}_{k|k-1}, ...\}$: The current prior, predicted to timestep k in order to match the timestamp of $Z_{k,\text{meas\_model}}$.
- $M_{\text{meas\_model}}$: the measurement model which produced the scan $Z_{k,\text{meas\_model}}$.

$$\begin{aligned} &\{\hat{x}_{k|k}, \hat{P}_{k|k}, \hat{p}_{k|k}\} = \\ &\text{IPDA\_update}(\{\hat{x}_{k|k-1}, \hat{P}_{k|k-1}, \hat{p}_{k|k-1}\}, Z_{k,\text{meas\_model}}, M_{\text{meas\_model}}) \end{aligned} \tag{7.3}$$

A final thing worth noting is that the IPDA_update function will accept any state estimate (not just a prior). In other words, we can (and will) call this function (with both priors and posteriors) multiple times, depending on how many different scans we have at a given timestep. Similarly, we could just as well call predict multiple times in a row, if it turns out that no scans were produced for some timesteps.

Since I will not subject the implemented IPDA to any ambiguous (cluttered) scans in this thesis, it will serve essentially as a KF. To this end, their elaborate update schemes are not in use, and I opted to omit them for brevity. For a more rigorous definition of the IPDA update, I defer to [18].

7.2.3 Get error state

For getting the error state, we defer to its namesake as defined for a general compound filter (eq. 4.2). This of course leaves the probability of existence estimate $\hat{p}_{k|k}$ untouched.

Although omitted here, it is of course the motion model (M_{motion}) which defines what elements of the target state $\hat{x}_{k|k}$ are nominal target states $\hat{x}_{t,k|k}$, and what elements are error states $\tilde{x}_{o,k|k}$. This again determines which elements are marginalized for in the **get_error_state**$(.)$ function.

7.2.4 Reset error state

For resetting after an injection, things get a little more complicated. This is because the 'inherited covariance' motion models (ESCV-IC and RESCV-IC) handles this a little differently than their 'decoupled covariance' counterparts (ESCV and RESCV).

In both cases, we run the **reset_error_state**$(.)$ function as defined for the compound filter in eq. 4.3.

For only the inherited covariance varieties (i.e. ESCV-IC and RESCV-IC), we also run the **reset_error_covariance**$(.)$ function as defined the compound filter in eq. 4.4.

This of course leaves the probability of existence estimate $\hat{p}_{k|k}$ untouched.

As with getting the error state, it is the motion model ($M_{\text{motion\_model}}$) which defines what elements of the target state $\hat{x}_{k|k}$ are nominal target states $\hat{x}_{t,k|k}$, and what elements are error states $\tilde{x}_{o,k|k}$. This again determines which elements are reset in the reset functions.

Chapter 8

Theory - Joint localization and tracking

Having defined localization and tracking separately, we can introduce the concept of joint localization and tracking. What I mean by this is some system which performs both tasks *and* in which the results from either task is used as part of the calculation of the other. In laymans terms, *the information flows both ways*. The idea of course being that such a joint solution will somehow provide better performance, more robustness, or otherwise some improvement over similar, separated schemes. In this chapter, two such schemes are presented in detail.

The first of which is a new addition suggested in this thesis, namely **navigational correction based on error state estimation as part of each track**. Much of the appeal for such a solution stems from the substantial support for modularization, in terms of separate and arbitrary localization and tracking schemes, which again support arbitrary motion and measurement models. This is also a true tracking solution in the sense that the implementation can deal with data association, misdetections and clutter. A second but significant appeal is that such a structure is similar to existing typical navigation and tracking systems, and as such would require less dramatic changes to implement.

The second approach is a full state EKF, which includes both the ownship localization state as well as the CV state of all targets. This requires fully associated, clutter free scans, and is included more so as a best-case performance example, with which to compare the sub-optimal error state approach. One would also expect such a filter to be decently

consistent in simulation, as it contains accurate models of all (target) behaviour, and is able to maintain all correlations.[1]

8.1 Error state joint localization and tracking - ES-JLAT

In this section, an error state oriented solution to joint localization and tracking is presented. For brevity, I will refer to this solution (in the broadest sense) as ES-JLAT, i.e. error state joint localization and tracking. This is divided into the following parts: firstly, a visual presentation of the various modules included, as well as which functions each module fulfills. Secondly, a suggested sequence of actions (function calls) which demonstrates the solution in an algorithmic perspective. The third part is a presentation of some suggested consensus models. Such models are the only truly intrinsic aspect this solution and thus a crucial design choice. The fourth and final part is a discussion on adjacent theory within state estimation and joint localization and tracking as a whole, including Schmidt-Kalman filters, error state Kalman-filters, and track to track fusion.

8.1.1 ES-JLAT: a visual guide

Before delving into the diagram, a brief outline of the information assumed available is prudent. Note also the arbitrary discrete timestep k here, for any such timestep, it is assumed that the various pieces of information are synchronized in time. This information is divided into three main categories:

1. Ownship odometry, i.e. u_k. This is the starting point for ownship localization, and provides a means for noisy, relative localization. This information is assumed to be produced in a 3 DOF discrete format, via IMU preintegration. As such, it is also assumed to be available in whatever timing is required, as the IMU is presumed to have a sampling frequency which is orders of magnitude larger than the other sensors.
   Note that odometry is **not used** for the simulations in this thesis. This illustrates some of the modularity of ES-JLAT, namely that we don't really care how the localization filter does its job, as long it can provide us with *current* Gaussian state estimates.
2. Absolute scans, i.e. $Z_{\text{abs},k}$. These are scans which contain one or more detections for some timestamp. Each detection is presumably the position of some vessel. This also includes the ownship, which is crucial for mitigating drift caused by IMU integration. As the *absolute* label implies, these detections are not subject to navigational uncertainty, i.e. the sensor pose is presumed known and *has* been used to transform the scan to a global frame.

[1] As opposed to a separate solution, in which the ownship pose is considered external information, which is "taken for granted" in some sense.

3. Relative scans, i.e. $Z_{\text{rel},k}$. These are scans which contain one or more detections for some timestamp. Each detection presumably contains the relative position between the ownship and another target. By extension, the sensor pose is assumed to be equal to the ownship pose. Using the online localization estimate to transform these scans to a global frame will necessarily subject this scan not only to the measurement uncertainty inherent to the sensor (measurement model) but *also* the navigational uncertainty of the ownship pose.

In terms of joint localization and tracking, it is perhaps the relative scans which are the most interesting. These contain a very explicit connection between the ownship and the targets, which, by extension, if you know something about the state of one of the two, it should be possible to use relative scans to infer something about the state of the other.

There is of course some reciprocity to this, in the sense that we also need the absolute scans to get some information about the individual positions to begin with. In this solution, it is very much the absolute scans which are ”the voice of reason” which stabilize the whole affair. One would expect inaccurate or infrequent absolute scans to make this approach subject to drift or divergence.

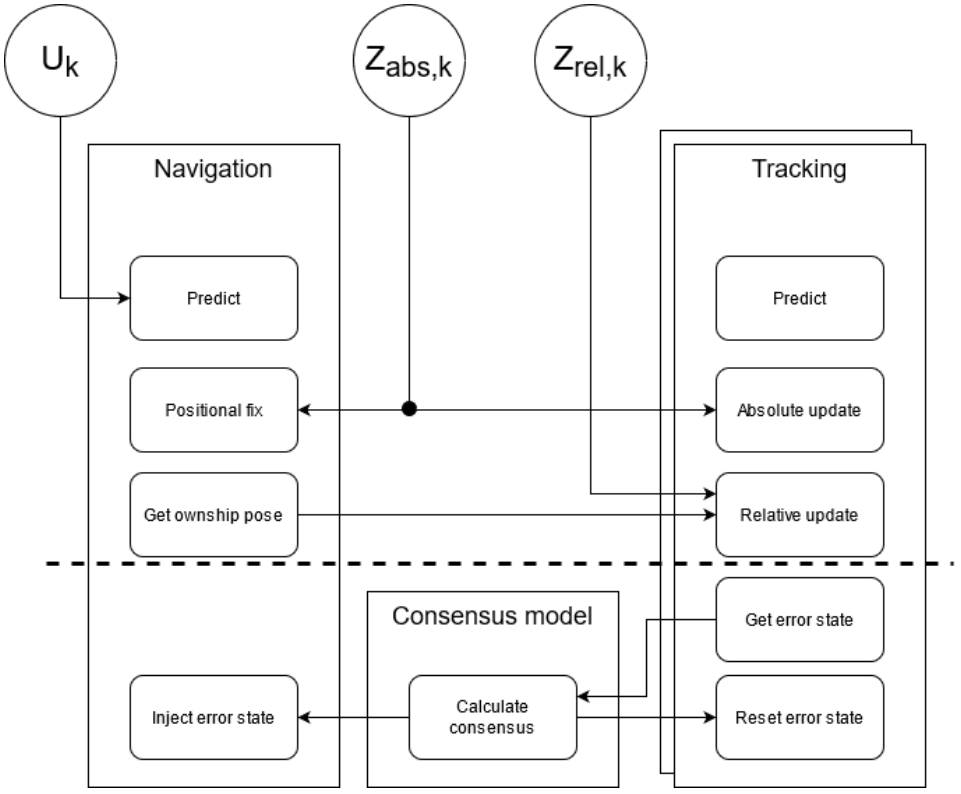

Figure 8.1: ES-JLAT overview diagram

A diagram of the different suggested modules for ES-JLAT is shown in figure 8.1. Note the layered tracking modules, which are intended to indicate the possibility for using multiple single target trackers (as done in this thesis) or a multi target tracker, furthermore, note that the update step has been divided into absolute and relative varieties, as it is only the relative updates which require knowledge about the ownship pose.

Finally, note that in typical system abstraction, one would simply omit the lower part of this diagram, i.e. the area below the dashed line. By only examining the top part, we see the traditional one sided flow (left to right) of information from navigation to tracking. By introducing the error state aspect, we enable a flow of information from tracking to navigation (right to left). It is this reciprocal flow of information which makes ES-JLAT a *joint* tracking and localization system.

8.1.2 ES-JLAT: an algorithmic example

Making use of the actions defined in figure 8.1, we can define an algorithm for a typical timestep of this scheme, doing all actions necessary to arrive at sound posteriors for both localization and tracking. For the record, some action A within a module M will be referred to as M_A.

1. Run **Navigation_predict**.
2. Run **Tracking_predict** to match the timestamp of the ownship.
3. At this point, we have calculated predicted estimates $\{\hat{x}_{k|k-1}, \hat{P}_{k|k-1}\}$ for both the ownship and all targets.
4. If an ownship detection is present within $Z_{\text{abs},k}$ at this timestep, run **Navigation_positional_fix** with this detection.
5. Run **Tracking_absolute_update** with the remainder of $Z_{\text{abs},k}$.
6. Run ES-JLAT injection and reset steps*:
   (a) Run **Consensus_model_calculate_consensus** with all error states (**Tracking_get_error_state**) from trackers.
   (b) Inject consensus into navigation model: **Navigation_inject_error_state**.
   (c) Reset error state in trackers: **Tracking_reset_error_state**
7. Run **Tracking_relative_update** with $Z_{\text{rel},k}$ and the current ownship pose.
8. Run ES-JLAT injection and reset steps:
   (a) Run **Consensus_model_calculate_consensus** with all error states (**Tracking_get_error_state**) from trackers.
   (b) Inject consensus into navigation model: **Navigation_inject_error_state**.

(c) Reset error state in trackers: **Tracking_reset_error_state**

*: After further consideration, it was found to be problematic to run injection steps after absolute updates. This is likely because the absolute scans do not contain any correlation between the target and ownship states, and thus only provide spurious estimates. Adjusting for this change, one would skip steps 6a and 6b, and only run the reset step, 6c. This disregards any error states estimated, and leaves all filters in an expected state, ready for new updates or predictions.

8.1.3 ES-JLAT: consensus models

So far in the elaboration of ES-JLAT, the consensus model has been left undetermined, as an arbitrary module which provides some **calculate consensus** function. In this section, various definitions for this function, i.e. various *consensus models* are discussed, finally concluding on a full definition for the choice I deemed worked best.

This function must accept multiple, independent error state estimates from each target state $\{\hat{x}_1, \hat{P}_1, \hat{p}_1\}, ..., \{\hat{x}_T, \hat{P}_T, \hat{p}_T\}$, and using these, make some correction on the localization estimate $\{\hat{x}_o, \hat{P}_o\}$. Furthermore, some preliminary discussion on the different models' pros and cons is included.

Consensus model: first attempts

Before arriving at the 'covariance weighted error' I experimented with a couple of other consensus models. The first of these were a simple average of every error estimate $\tilde{x}_t$, after some testing, this was found to be problematic. This is mostly because such a scheme does not, in any way, account for the fact that a track may be uncertain or inaccurate at certain times. It was deemed prudent to find some consensus model which is able to disregard error estimates from bad (or even diverged) tracks.

The first attempt to this end was a consensus model which still calculated an average of the error states, but now weighted against that track's probability of existence estimate $\hat{p}_t$. Given that a number of the simulations I ran was without clutter, and with fully associated scans (i.e. just filtering), the $\hat{p}_t$ of *every* track was driven exponentially fast to 1. Since the existence probability of every track was equal in these cases, the resulting estimate became equal to that of the aforementioned 'average consensus model', with the same problems as before.

Consensus model: covariance weighted error

In order to leverage a more dynamic metric (hopefully reflecting the track's accuracy), it was opted to instead use the covariance estimates of (the error states in) a track to weigh its contribution to the consensus. Furthermore, now considering the track's own estimate of the error state covariance, it becomes possible to not only calculate a consensus, but also the covariance of that consensus estimate. In this sense, it starts to look more like any other KF update step, in the sense that a measurement has both a value z and an associated noise matrix $\mathbf{R}$. In our case, this expands the consensus model's job from providing just an injection, to also estimating the uncertainty of that injection.

In order to estimate these sizes, we first consider the simple case in which the error state estimate $\tilde{x}_t$ associated with each track, t, is a scalar. By extension, its marginal covariance estimate $\hat{P}_{t,\tilde{x}}$ is also a scalar. Let's assume that each error state $\tilde{x}_t$ is independently distributed according to $\mathcal{N}(\tilde{x}, \hat{P}_{t,\tilde{x}})$, where the mean $\tilde{x}$ is the supposed *true* error state.

This assumption ($\tilde{x}_t \sim \mathcal{N}(\tilde{x}, \hat{P}_{t,\tilde{x}})$), is a consequence of assuming that the error state $\tilde{x}$ is observable from the measurements provided to the tracking filter. If there is an error present, we trust the tracking filter's post-update error state to reflect this (and in fact be an optimal estimate of).

This is a dangerous assumption, since we have not actually accounted for the observability of the various localization states in the tracking filter update. In other words, for unobservable error states, we are liable to produce zero (or close to zero, depending on correlations in $\hat{\mathbf{P}}_o$) estimates of these states in $\tilde{x}_t$. As it stands, these are naïvely injected, and their zero estimates are interpreted as "ok, there must be no error for that state", when in reality, we have no idea whether there is an error present. This is something I will get back to, but stands as a pressing point of future work, since the current implementation will inject these spurious estimates (and reduce their covariance) regardless.

Leveraging this distribution, we can calculate a maximal likelihood estimator (MLE) as:

$$\tilde{x} = \frac{1}{\sum_{t=1}^{T} \frac{1}{\hat{P}_{t,\tilde{x}}}} \sum_{t=1}^{T} \frac{\tilde{x}_t}{\hat{P}_{t,\tilde{x}}} \tag{8.1}$$

For $T = 2$, i.e. two targets being tracked, this simplifies to:

$$\tilde{x} = \tilde{x}_1 \frac{\hat{P}_{2,\tilde{x}}}{\hat{P}_{1,\tilde{x}} + \hat{P}_{2,\tilde{x}}} + \tilde{x}_2 \frac{\hat{P}_{1,\tilde{x}}}{\hat{P}_{1,\tilde{x}} + \hat{P}_{2,\tilde{x}}} \tag{8.2}$$

Which intuitively enough becomes a normalized weight on each error state, which is determined by how much of the total covariance is taken up by *the other* targets. For example,

if the covariance of target 1 is much larger than that of target 2, it follows that the weight of target 1 will be close to zero, while the weight of target 2 will be close to 1. This matches our expectations, as we would want to trust the estimate which has the lower covariance of the two. In a similar fashion, if the two covariances are of similar size, the two weights both become roughly 0.5, as an evenly weighted average.

These expressions become significantly more horrid as T grows, but for the purposes of the simulations used in this thesis, $T = 2$ is enough. So simplified expressions of this equation are omitted.

For this estimator, we can also evaluate its variance, again treating all covariances as constant, and treating each $\tilde{x}_t$ as independent of the others:

$$\text{Var}(\tilde{x}) = \frac{1}{\sum_{t=1}^{T} \frac{1}{\hat{P}_{t,\tilde{x}}}} \tag{8.3}$$

Once again simplifying this for T = 2, we arrive at:

$$\text{Var}(\tilde{x}) = \frac{\hat{P}_{1,\tilde{x}}\hat{P}_{2,\tilde{x}}}{\hat{P}_{1,\tilde{x}} + \hat{P}_{2,\tilde{x}}} \tag{8.4}$$

Now looking at the non-scalar case, leveraging linear algebra instead of calculus, we arrive at the following MLE estimate for the error state vector $\tilde{x}$:

$$\tilde{x}^\top = \left(\sum_{t=1}^{T} \tilde{x}_t^\top \hat{\mathbf{P}}_{t,\tilde{x}}^{-1}\right)\left(\sum_{t=1}^{T} \hat{\mathbf{P}}_{t,\tilde{x}}^{-1}\right)^{-1} \tag{8.5}$$

$$\text{Var}(\tilde{x}^\top) = \tilde{\mathbf{P}} = \left(\sum_{t=1}^{T} \hat{\mathbf{P}}_{t,\tilde{x}}^{-1}\right)^{-1} \tag{8.6}$$

At this point, it is worth noting that computing these expressions will result in a copious amount of matrix inversions. This is typically an expensive (slow) computation, so it would be prudent to look for ways of speeding this calculation up either by algebraic tricks, or 'close enough' estimates which are more computationally sound. For the purposes of this thesis, I have not made any efforts towards this end, so this is left as a point of future work. Mitigating this problem has seen some work in the literature, especially in the context of SLAM [25], this leverages an information filter, which maintains an estimate of the inverse covariance matrix, i.e. *an information matrix*. Depending on the overall needs for the system, it might be prudent to swap the error state estimation to information filters, thus inherently producing the necessary inverse matrices.

Overconfidence for positional updates

After some tests with the ESCV and ESCV-IC models, it was found that the positional corrections they produced were very small, and not really enough to make much of a difference. Furthermore, the fact that the ownship already receives positional detections frequently, will drive these covariance values quite low. I.e. the filter already has a fairly accurate estimate, and does not need much help. When using the injection covariance $\tilde{\mathbf{P}}$ in order to calculate posterior covariance for the localization filter, it was observed that positional estimates suddenly grew grossly overconfident.

I expect this to be a result of the positional injections not really making the estimates more accurate, while at the same time, the covariance update causes the positional variances to be greatly reduced. At this time it is tricky for me to diagnose whether this will be a recurring theme for injections on variables which already have decent estimates, or similarly whether there is simply not enough information in two range bearing detections to accurately estimate all three error states.

Nonetheless, I ended up doing an ad-hoc adjustment here in order to give the ESCV(-IC) a fighting chance. This adjustment consisted of creating an inflation matrix $\mathbf{K}$, which is used to increase the size of $\tilde{\mathbf{P}}$, before using it to calculate the posterior covariance (after injection). In KF terms, the injection covariance $\tilde{\mathbf{P}}$ is used as the measurement noise $\mathbf{R}$, thus its size will influence how much the covariance estimate is reduced during an update. Intuitively, setting this $\mathbf{R}$ matrix to be very small would result in a similarly small posterior covariance, whereas setting to something huge, would result a relatively unchanged posterior covariance.

We can leverage this by saying ”ok, we still trust the heading corrections to be accurate, whereas the positional corrections are too small to make a difference.”. Modifying the injection covariance $\tilde{\mathbf{P}}$ to reflect this insight, we would want to make the positional elements huge, while leaving the heading elements unchanged. I chose to achieve this by scaling the $\tilde{\mathbf{P}}$ matrix on either side by the diagonal matrix $\mathbf{K}$:

$$\tilde{\mathbf{P}}_{\text{inflated}} = \mathbf{K}\tilde{\mathbf{P}}\mathbf{K} \tag{8.7}$$

For the sake of generality, I defined this inflation matrix using the tuning parameters K_{p} and K_{φ}, which are scalars for inflation of heading and positional elements, respectively. This makes the inflation matrix for an ESCV injection defined as such: $\mathbf{K}_{\text{ESCV}} = \text{diag}(K_{\text{p}}, K_{\text{p}}, K_{\varphi})$. In the general case, we would want to define a inflation parameter K_i for each element $\tilde{x}_i$ of the error state. If multiple relative measurement models are present, each with different observability on the error states, it might even be prudent to define such a vector of inflation parameters for every measurement model, in order to ensure that they adequately affect posterior covariance according to their properties.

This exponential increase of tuning parameters (which may even need to be adjusted according to number of targets) highlights an obvious ’rabbit-hole’ with this solution, and I

deem the posterior covariance of ES-JLAT injection to be the decidedly weakest part of this algorithm. This stands a crucial point of future work, namely to find some more elegant (robust and simple) way of calculating this posterior covariance which would guarantee consistent results.

For the inherited covariance motion models (ESCV-IC and RESCV-IC), these inflation parameters are actually the only tuning parameters inherent to ES-JLAT, and thus become important tools for achieving consistency.

It is worth noting here that the number of targets being tracked will have a huge influence on the injection covariance $\tilde{\mathbf{P}}$, as it is an MLE estimate of all individual error states. All of the insights discussed here are in the context of tracking two targets. It is perfectly possible that these inflation parameters will need to vary depending on targets, in order to ensure that the $\tilde{\mathbf{P}}$ estimate does not grow too confident.

8.1.4 ES-JLAT: adjacent theory

With ES-JLAT now being fully defined, its similarities to existing solutions bears mentioning.

Schmidt-Kalman filters

In terms of a single tracking filter (referred to as a 'compound filter') the most adjacent thing I can think of are the navigational uncertainty SKF solutions for tracking, as presented in [6; 4]. With very similar filter structures, the main distinction is the fact that we allow the nuisance parameters to be estimated, as opposed to an SKF, which imposes zero mean nuisance parameters, also on updates.

Error state Kalman filters

Since we also perform injection and reset steps, a single compound filter becomes an ESKF (see sec. 4.2 as well as [8; 9]) in some sense. Noting of course that it is only the nuisance parameters (localization error) which are affected by the injection/reset steps, it would be more prudent to describe it as a partial ESKF, where the remainder of the filter simply becomes a KF or EKF.

Track to track fusion

Moving along to the combination of multiple tracking filters, and the consensus injection, it becomes reminiscent of a track to track fusion structure (see [26] for an introduction).

This being the case by virtue of every tracker calculating an update step independently, and that a consensus is calculated 'after the fact' using these updated track estimates. To that end, the 'track fuser' becomes the consensus model, which handles both the fusion of data from each separate track, while also handling the reset step which constitutes a 'central tracks update'. An extension of the 'track fuser' thus becomes its influence on the localization update, which goes beyond typical track to track fusion.

In terms of a posterior 'central tracks update' this becomes a a bit of a stretch, as the current formulation of ES-JLAT does not perform any retroactive corrections on track estimates (beyond the error state reset). However, I posit a relevance nonetheless, as such a modification is an immediate point of future work. I.e. using information from the calculated consensus in order to retroactively correct track posteriors. E.g. correcting tracks which had an error state estimate far from the consensus, while leaving the seemingly accurate (close to consensus) tracks unchanged. This is something I will discuss later in the thesis under the label 'injection innovation'.

If we swap perspectives, we can treat the ownship localization as the objective of tracking. In such a view, we clearly see how every tracking filter provides its own 'tracking estimate', which is fused in the consensus model into a 'central track update' as a consensus injection. In such a perspective, it is clearly a track to track fusion system.

8.2 Full state EKF

To include some reasonably sane filter with which to compare the results of ES-JLAT, a full state filter was also implemented (henceforth referred to as full-filter, FF). Its state includes the ownship state (with heading) and an arbitrary number of CV targets. In this section, the details surrounding this filter are presented.

8.2.1 Full state EKF: motion model

In our case, we assume that the ownship is modeled according to the *CV + heading* (as defined in section 5.1.2) and that all targets (1 through T) are modeled according to the CV model (as defined in 5.1.1). By stacking this into a state vector x_{FF}, we arrive at:

$$x_{\text{FF}} = \begin{bmatrix} x_{\text{CV+heading},o} \\ x_{\text{CV},1} \\ \vdots \\ x_{\text{CV},T} \end{bmatrix} \tag{8.8}$$

Noting that $x_{\text{CV+heading},o}$ has 5 elements, and each vector $x_{\text{CV},t}$ has 4 elements, it follows that x_{FF} will have a grand total of $(5+4\cdot T)$ elements. Stacking together a similar transition matrix $\mathbf{F}_{\text{FF}}$ for discretized states, we arrive at:

$$\mathbf{F}_{\text{FF}} = \textit{blkdiag}(\mathbf{F}_{\text{CV + heading}}, \mathbf{F}_{CV}, ..., \mathbf{F}_{CV}) \tag{8.9}$$

Note that the number of $\mathbf{F}_{CV}$ matrices in the tail end of this parameter list corresponds to the number of targets, T, s.t. the dimensions of the square matrix $\mathbf{F}_{\text{FF}}$ matches up to the number of elements in x_{FF}.

In a similar sense, we can stack together a process noise matrix $\mathbf{Q}_{\text{FF}}$:

$$\mathbf{Q}_{\text{FF}} = \textit{blkdiag}(\mathbf{Q}_{\text{CV + heading}}, \mathbf{Q}_{CV}, ..., \mathbf{Q}_{CV}) \tag{8.10}$$

Combining these into a classic KF prediction step assuming a Gaussian posterior from the previous step $\{x_{\text{FF},k|k}, P_{\text{FF},k|k}\}$ (no need for linearization, as matrices $\mathbf{F}$ and $\mathbf{Q}$ are already discretized), we arrive at:

$$\mathcal{N}(x_{\text{FF},k+1|k}, \mathbf{P}_{\text{FF},k+1|k}) = \mathcal{N}(\mathbf{F}_{\text{FF}} x_{\text{FF},k|k}, \mathbf{F}_{\text{FF}} \mathbf{P}_{\text{FF},k|k} \mathbf{F}_{\text{FF}}^{\top} + \mathbf{Q}_{\text{FF}}) \tag{8.11}$$

8.2.2 Full state EKF: measurement models

For the update steps of the EKF, we once again defer to the models defined in chapter 5. In this case, we assume that an absolute (associated) detection is available for the ownship, as well as every target. Similarly, we assume that a relative detection is available between the ownship and every target. Furthermore, we assume that these detections are all associated, i.e. sorted to match their corresponding targets. Thus, the job of modeling both these update steps amounts to stacking together the appropriate amount of measurement models.

For both models, the task of actually calculating the posterior state is reduced to defining $h(x), \mathbf{H}, \mathbf{R}$, with which a KF update can be calculated.

Absolute detections

Noting the detection of the ownship o and all targets 1 through T, and operating with the previously defined measurement function $h_{\text{cart}}(.)$ (eq. 5.38) and its corresponding noise $\mathbf{R}_{\text{cart}}$ (eq. 5.40), we arrive the the model:

$$z_{\text{FF,cart},k} \sim \mathcal{N}(h_{\text{FF,cart}}(x_{\text{FF},k}), \mathbf{R}_{\text{FF,cart}}) \tag{8.12}$$

$$h_{\text{FF,cart}}(x_{\text{FF}}) = \begin{bmatrix} h_{\text{cart}}(p_o) \\ h_{\text{cart}}(p_1) \\ \vdots \\ h_{\text{cart}}(p_T) \end{bmatrix} \tag{8.13}$$

$$\mathbf{R}_{\text{FF,cart}} = \textit{blkdiag}(R_{\text{cart}}, ..., R_{\text{cart}}) \tag{8.14}$$

Noting that the EKF estimates a linear measurement matrix $\mathbf{H}$ to compute the update step, we find this via linearization of the measurement function $h_{\text{FF},abs}(s)$. Conveniently, this is already a linear function, and corresponds to selecting the positional elements from the ownship and every target. Using $\mathbf{H}_{\text{CV + heading}\to\text{cart}}$ and $\mathbf{H}_{\text{CV}\to\text{cart}}$, as defined in equations 5.42 and 5.41, we build the full measurement matrix $\mathbf{H}_{\text{FF,cart}}$ as:

$$\mathbf{H}_{\text{FF,cart}} = \textit{blkdiag}(\mathbf{H}_{\text{CV + heading}\to\text{cart}}, \mathbf{H}_{\text{CV}\to\text{cart}}, ..., \mathbf{H}_{\text{CV}\to\text{cart}}) \tag{8.15}$$

At this point, the remainder of this update step is a KF update, as a function of the state prior $\{x_{\text{FF},k|k-1}, \mathbf{P}_{\text{FF},k|k-1}\}$, the measurement vector $z_{\text{FF},cart,k}$, as well as its measurement and noise covariance matrices, $\mathbf{H}_{\text{FF},cart}$ and $\mathbf{R}_{\text{FF},cart}$.

For the sake of brevity[2] the remaining nitty gritty of this update is omitted, and simply defined as:

$$\begin{aligned} &\{x_{\text{FF},k|k}, \mathbf{P}_{\text{FF},k|k}\} = \\ &\quad \textbf{KF\_update}(\{x_{\text{FF},k|k-1}, \mathbf{P}_{\text{FF},k|k-1}\}, z_{\text{FF},cart,k}, \mathbf{H}_{\text{FF},cart}, \mathbf{R}_{\text{FF},cart}) \end{aligned} \tag{8.16}$$

Relative detections

Just as the absolute detections model amounted to a stack of $T+1$ *Cartesian absolute* measurement models, the relative detections model is simply a stack of T *Range bearing relative* measurement models (see sec. 5.2.2):

$$z_{\text{FF},rel,k} \sim \mathcal{N}(h_{\text{FF},rel}(x_{\text{FF},k}), \mathbf{R}_{\text{FF},rel}) \tag{8.17}$$

$$h_{\text{FF},rel}(x_{\text{FF},k}) = \begin{bmatrix} h_{\text{rel}}(x_o, x_1) \\ \vdots \\ h_{\text{rel}}(x_o, x_T) \end{bmatrix} \tag{8.18}$$

[2]Furthermore, a standard KF update step has already been defined in section 6.1.1.

Assuming independence of the measurement noise for each target, we define the noise covariance matrix $\mathbf{R}_{\text{FF,rel}}$ as:

$$\mathbf{R}_{\text{FF,rel}} = \mathit{blkdiag}(\mathbf{R}_{\text{rel}}, ..., \mathbf{R}_{\text{rel}}) \tag{8.19}$$

As for $\mathbf{H}_{\text{FF,rel}}$, i.e. the Jacobian of $h_{\text{FF},rel}(.)$, it is significantly more hairy than its Cartesian counterpart. This is because its elements are derivatives of Euclidian norms and atan2 functions, which will vary with both ownship and target positions. For this reason, I have opted to not compute this matrix analytically, but rather numerically.[3] As the EKF is designed to compute such jacobians from nonlinear measurement functions, we simply defer to an EKF update step, and leave it as implied that $\mathbf{H}_{\text{FF,rel}}$ will be estimated from $h_{\text{FF},rel}(.)$ in some way. This leaves us with the following definition for this update step:

$$\begin{aligned}&\{x_{\text{FF},k|k}, \mathbf{P}_{\text{FF},k|k}\} = \\ &\quad \textbf{EKF\_update}(\{x_{\text{FF},k|k-1}, \mathbf{P}_{\text{FF},k|k-1}\}, z_{\text{FF},rel,k}, h_{\text{FF},rel}, \mathbf{R}_{\text{FF},rel})\end{aligned} \tag{8.20}$$

[3] For example using finite differences to differentiate $h_{\text{FF},rel}(.)$ wrt. x_{FF} about the newest state estimate.

Chapter 9

Simulation

In order to validate and compare the solutions presented in chapter 8, they were implemented in MATLAB and tested in various Monte Carlo simulations. This chapter outlines how these simulations were implemented, as well as the different variants which were tested. This chapter is divided into six parts. First the simulator is described in broad terms. Secondly, the vessel behaviour is defined, with adjustments being made for the extra states necessary for the ownship (in addition to the planar position and velocity required for CV). Third, the generation of sensor data is described, i.e. generating detections in the form of absolute and relative scans. Fourth, the chosen performance metrics for the simulations are introduced, and their merits are briefly discussed. Fifth, an algorithmic outline of how the simulations are executed is included. Sixth and finally, some sample output from the simulator is shown. I.e. vessel trajectories in XY plots and an ownship heading timeseries.

For the sake of brevity and abstraction, this chapter relies heavily on the previously defined functions and models. Thus it is prudent to revisit chapter 5 for any definitions, distributions, measurement functions which are not readily defined here.

9.1 Overview

The simulation features 2 targets of interest, as well as an ownship. All three of these vessels are defined as points moving in a planar coordinate system (2 DOF), with the ownship also including a heading (3 DOF).

Otherwise, there are no additional vessels, no obstructions present, and no environmental/external influence on these vessels. I.e. just an open, calm ocean with all three vessels moving about freely.

Each simulation is set to start at time $t_0 = 0s$, and runs in discrete steps until $t_{\text{max}} = 50s$.

9.2 Initial conditions

Initial conditions, both in terms of true states from which to generate trajectories, as well as filter initialization, will have a huge impact on performance. Especially the ES-JLAT scheme, which relies on tiny incremental corrections, has been found to struggle if the filters are initialized far from the corresponding true values. For this reason, relatively easy initial conditions were decided on.

9.2.1 Initial conditions: true values

The ownship trajectory always starts in $[0, 0]$ with zero initial velocity. The initial heading is sampled from a continuous uniform distribution as $\varphi_{o,0} = \text{uniform}(-\pi, \pi)$.

Both targets were initialized to start a nominal distance (50 meters) from the ownship. This was achieved by sampling polar coordinates r_t, θ_t, for either target t. The radius was sampled similarly for both targets, with a variance of 10 m^2, i.e. $r_t \sim \mathcal{N}(50, 10)$. It was deemed prudent to ensure a decent starting distance between the targets, in order to give the trackers a decent chance of establishing tracks before being challenged on association from proximal vessels. For this reason, it was imposed that either target should start on opposite sides of the ownship. This was achieved by sampling the angles θ_t as follows:

$$\begin{aligned} \theta_1 &= \text{uniform}(\pi/4, 3\pi/4) \\ \theta_2 &= \text{uniform}(-3\pi/4, -\pi/4) \end{aligned} \tag{9.1}$$

This makes the initial target position $p_{t,0}$ defined as:

$$p_{t,0} = \begin{bmatrix} r_t \cos(\theta_t) \\ r_t \sin(\theta_t) \end{bmatrix} \tag{9.2}$$

Finally, the initial target velocity $v_{t,0}$ was sampled from a zero mean Guassian distribution, with a variance of 3 $(m/s)^2$, i.e.:

$$v_{t,0} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 3 & 0 \\ 0 & 3 \end{bmatrix} \right) \tag{9.3}$$

9.2.2 Initial conditions: filter initialization

For all filters used in this simulation, their inherent assumption is that their produced estimate is distributed as a Gaussian about their true value. To enforce this in their initialization, it was opted to sample the initial values from said Gaussian.

This was achieved by sampling the relevant true states x_0 from their sampled values, and defining some initial covariance $\hat{\mathbf{P}}_0$. This initial covariance was set as constant across all vessels and simulations, with the following values:

- The marginal covariance of all positional elements p were set as $3\ m^2$.
- The marginal covariance of all velocity elements v were set as $3\ (m/s)^2$.
- The marginal covariance of the ownship heading φ_o was set as $0.3\ (rad)^2$. This corresponds to a standard deviation of roughly 31 degrees.
- All cross correlations (off diagonal elements) were set as 0.

This produces the following initial covariance estimates $\hat{\mathbf{P}}_{o,0}, \hat{\mathbf{P}}_{t,0}$ for the ownship and targets, respectively:

$$\begin{aligned} \hat{\mathbf{P}}_{o,0} &= \text{diag}(3, 3, 3, 3, 0.3) \\ \hat{\mathbf{P}}_{t,0} &= \text{diag}(3, 3, 3, 3) \end{aligned} \tag{9.4}$$

Using these initial covariances as well as the corresponding true values, the initial state estimates were sampled from:

$$\begin{aligned} \hat{x}_{o,0} &\sim \mathcal{N}(x_{o,0}, \hat{\mathbf{P}}_{o,0}) \\ \hat{x}_{t,0} &\sim \mathcal{N}(x_{t,0}, \hat{\mathbf{P}}_{t,0}) \end{aligned} \tag{9.5}$$

9.3 Vessel behaviour

For all 3 vessels, their motion is generated using the discrete CV model defined in section 5.1.1. I.e. from some initial state $x_{CV,0} = [p_x, v_x, p_y, v_y]^\top$, we sample a random acceleration according to $\mathbf{Q}_{\text{CV}}$, which is then used to propagate it to the consecutive timestep, $x_{CV,1}$. This matrix is a function of both the acceleration noise σ^2_{CV} as well as the time between each step ΔT. For **all** simulations, these parameters are constant, and defined as $0.5(m/s^2)^2$ and $0.01s$, respectively.

This is perfectly analogous to the CV model, thus we would expect the CV filters used in these solutions to be consistent, given that they are tuned to the correct noise parameter σ^2_{CV}.

9.3.1 Ownship

For the ownship, part of the localization objective is to estimate the heading. Thus we also need a ground truth heading to be generated by the simulator. I have opted to implement the heading as a random walk, i.e. a constant value, which is driven by a zero mean Gaussian noise at every timestep.

This makes it analogous to the *CV + heading* model, as defined in section 5.1.2. For propagating an initial heading value φ_0 to its next timestep, we again require a tuning parameter σ^2_φ as well as a time difference ΔT. These parameters are defined as $0.16(rad/s)^2$ and $0.01s$, respectively, for **all** simulations.

At this point it worth noting that the ownship heading, with this random walk model, is completely decoupled from the ownship direction of travel. This is somewhat uncharacteristic for surface vessels, as such vessels are typically directionally stable, i.e. the hull, thrusters and rudder are designed in such a way that the heading always stabilizes to the direction of travel (much like fixed wing aircraft). However, fully actuated vessels, outfitted with multiple azimuth thrusters (such as milliAmpere) can easily control heading and linear motion separately, for this reason, it is prudent to be able to estimate heading without inferring it from motion. In order to enforce this independence, while also producing readily available angular rate values, this random walk model was chosen.

Note the choice of sample time as $\Delta T = 0.01s$ for the vessel motion. At this point, we have effectively chosen the simulation frequency as $100Hz$, by extension, it is now prudent to choose all other sample rates as some whole number fraction of $100Hz$. This ensures that accurate ground truth data is available for other modules (without having to interpolate between timesteps), whenever they need to sample it.

9.4 Sensor data

9.4.1 Absolute scans

Next, we need to generate absolute, positional detections of all targets. These are modeled according to the *Cartesian absolute* model, as defined in section 5.2.1, and running at a constant sample rate of $10Hz$.

Noting that the ground truth trajectories are available at $100Hz$, we generate the absolute scans by selecting the position of all 3 vessels at every 10th sample of the ground truth

timeseries. These positions are then sampled according to the Gaussian distribution of the Cartesian absolute model, i.e. equation 5.39.

These samples are subject to noise defined by the covariance matrix $\mathbf{R}_{\text{cart}}$, which again is defined by the Cartesian measurement noise parameter, σ^2_{cart}. For **all** simulations and all vessels, this parameter is set as 3 m^2.

These scans are *not* subject to any misdetections or clutter, i.e. all targets are detected in every scan, and no additional false detections are included. This means that for a 50 second simulation, the generated scans $Z_{\text{cart},k}$ (and their corresponding timestamps t_k) can be expressed as the collection:

$$\{Z_{\text{cart},0}, t_0\}, ..., \{Z_{\text{cart},K}, t_K\} = \left\{ \begin{bmatrix} z_{\text{cart},o} \\ z_{\text{cart},1} \\ z_{\text{cart},2} \end{bmatrix}, 0 \right\}, ..., \left\{ \begin{bmatrix} z_{\text{cart},o} \\ z_{\text{cart},1} \\ z_{\text{cart},2} \end{bmatrix}, 50 \right\} \tag{9.6}$$

9.4.2 Relative scans

For the relative detections, we defer to the *Range bearing, relative* measurement model, as defined in 5.2.2. In this case, the model runs at 20 *Hz*, i.e. we select every fifth sample from the ground truth trajectories in order to generate the scans. As for tuning parameters $\sigma^2_{\text{rel,r}}$, $\sigma^2_{\text{rel},\theta}$, these define the measurement noise matrix $\mathbf{R}_{\text{rel}}$, and are radial and tangential variance, respectively.

Again, we define these as constants for every simulation and vessel, and set their values to $\sigma^2_{\text{rel,r}} = 2\ m^2$ and $\sigma^2_{\text{rel},\theta} = 0.1\ rad^2$. Now that the ground truth data and covariance matrix are defined, we can sample every scan according to its distribution: eq. 5.47. As with the absolute detections, I have not included any misdetections or clutter, and an example collection of all scans for a 50 second simulation thus becomes:

$$\{Z_{\text{rel},0}, t_0\}, ..., \{Z_{\text{rel},K}, t_K\} = \left\{ \begin{bmatrix} z_{\text{rel},1} \\ z_{\text{rel},2} \end{bmatrix}, 0 \right\}, ..., \left\{ \begin{bmatrix} z_{\text{rel},1} \\ z_{\text{rel},2} \end{bmatrix}, 50 \right\} \tag{9.7}$$

9.4.3 Disclaimer

A number of arbitrary choices have been made with regards to generating sensor data for the simulations. Specifically, sample rates and noise parameters. Crucially, the choice to not include misdetections or clutter is an important one. For the purposes of demonstrating ES-JLAT as a full fledged tracking system, it needs to be able to handle both misdetections and clutter. To this end, this simulator is by no means a final answer for such a demonstration.

A very obvious point of future work becomes to implement these error sources in order to test ES-JLATs resilience to them. Alternatively, and even more interesting, is to test ES-JLAT on a real, experimental dataset instead, as simulations after all cannot capture all the complexities of a real life system.

On the topic of coping with misdetections and clutter, it is worth noting that the choice of tracker(s) will have a huge impact on this. A similar point of future work will be to test the ES-JLAT with various trackers. Given the large degree of modularity inherent to ES-JLAT, i.e. support for arbitrary trackers, motion, measurement and consensus models, testing all kinds of combinations becomes a Herculean task. This is of course far too much for this thesis, and I leave this too as a point of future work.

9.5 Execution of a simulation

The execution of a single simulation is divided into four parts:

1. Setup, i.e. generation of trajectories, scans, initial conditions.
2. Iteration of ES-JLAT, i.e. running prediction, update and injection steps in sequence.
3. Iteration of the full filter solution, i.e. running prediction and update steps in sequence.
4. Calculation of performance metrics (EE, NEES, RMSE, ANEES) of chosen state subsets, for both solutions.

This section describes how these four parts are executed in the simulation, resulting in a set of performance metrics for either solution.

9.5.1 Simulation setup

1. Generate vessel trajectories
2. Generate absolute and relative scans
3. Generate initial conditions for all filters.

9.5.2 ES-JLAT iteration

1. Start a for loop at $t_k = 0s$, incrementing $0.05s$ per iteration, i.e. 20 Hz:
   (a) Run prediction step in localization filter.

(b) Run prediction step in tracker(s).

(c) If an absolute scan $Z_{\text{cart},k}$ is available for time t_k:

i. Run positional update in localization filter with first element of $Z_{\text{cart},k}$.

ii. Run update step in tracker(s) with $Z_{\text{cart},k}$.

iii. Calculate and inject consensus according to consensus model.*

iv. Reset error states in tracker(s).*

(d) If a relative scan $Z_{\text{rel},k}$ is available for time t_k:

i. Run update step in tracker(s) with $Z_{\text{rel},k}$.

ii. Calculate and inject consensus according to consensus model.

iii. Reset error states in tracker(s).

*: Running an injection after updates from absolute scans was found to be problematic. For this reason, steps 1 c iii and 1 c iv were skipped in some simulations. This will be specified in the results section of these simulations.

9.5.3 Full filter iteration

1. Start a for loop at $t_k = 0s$, incrementing $0.05s$ per iteration, i.e. 20 *Hz*:

(a) Run prediction step.

(b) If an absolute scan $Z_{\text{cart},k}$ is available for time t_k:

i. Run update step with $Z_{\text{cart},k}$.

(c) If a relative scan $Z_{\text{rel},k}$ is available for time t_k:

i. Run update step with $Z_{\text{rel},k}$.

9.5.4 Calculation of performance metrics

Performance metrics: description of data

After iteration of both filters, we are left with posterior estimates $\{\hat{x}_k, \hat{\mathbf{P}}_k\}$ for every 0.05 seconds. For the ES-JLAT solution, these estimates are separate for each vessel, and look like:

$$\{\hat{x}_{o,0}, \hat{\mathbf{P}}_{o,0}\}, \{\hat{x}_{1,0}, \hat{\mathbf{P}}_{1,0}\}, \{\hat{x}_{2,0}, \hat{\mathbf{P}}_{2,0}\}, ..., \{\hat{x}_{o,K}, \hat{\mathbf{P}}_{o,K}\}, \{\hat{x}_{1,K}, \hat{\mathbf{P}}_{1,K}\}, \{\hat{x}_{2,K}, \hat{\mathbf{P}}_{2,K}\} \tag{9.8}$$

As the full filter includes all three vessels as part of its state, its estimates look like:

$$\left\{ \begin{bmatrix} \hat{x}_{o,0} \\ \hat{x}_{1,0} \\ \hat{x}_{2,0} \end{bmatrix}, \hat{\mathbf{P}}_0 \right\}, ..., \left\{ \begin{bmatrix} \hat{x}_{o,K} \\ \hat{x}_{1,K} \\ \hat{x}_{2,K} \end{bmatrix}, \hat{\mathbf{P}}_K \right\} \tag{9.9}$$

Pulling the true data from the vessel trajectories, for the corresponding timestamps t_0 ... t_K, we get:

$$\{x_{o,0}, x_{1,0}, x_{2,0}\}, ..., \{x_{o,K}, x_{1,K}, x_{2,K}\} \tag{9.10}$$

At this point, we have estimated and true states, with corresponding covariance estimates, which are all synchronized in time. We are now ready to calculate the two performance metrics of choice, namely RMSE and NEES.

Performance metrics: useful state subsets

For any timestep k, we have a total of 13 different variables with which to calculate various metrics. For the sake of scrutinizing some of these in greater detail, and considering that the ES-JLAT does not maintain covariance estimates of inter-vessel values, it becomes prudent to define some subsets of these states.

- p_o, p_1, p_2 are the positional elements of their respective vessel states x_o, x_1, x_2.
- φ_o: the heading of the ownship.
- x, without any subscript, refers to the full 13 variable state vector, as used by the full filter.

For the record, I will also refer to covariance matrices for these subsets like so: $\hat{\mathbf{P}}_{p,k}$. This reads as matrix $\hat{\mathbf{P}}_k$, marginalized over the positional elements in p.

Note that for the ESCV and RESCV models, the track state vectors $\hat{x}_1, \hat{x}_2$ are appended with zero-valued nuisance parameters corresponding to *some* ownship localization error. For the purposes of calculating metrics, these elements are ignored, i.e. $\hat{x}_1, \hat{x}_2$ now refer to *only* the CV state of a vessel (the first four elements). Similarly, the corresponding covariance matrices $\hat{\mathbf{P}}_1, \hat{\mathbf{P}}_2$ are assumed to be marginalized over these same four states.

Performance metrics: estimation error

For some estimate $\hat{x}$ and a corresponding true value x, we define the estimation error (EE) as the Euclidian norm of $\hat{x} - x$, i.e.:

$$\text{EE}(x, \hat{x}) = ||\hat{x} - x||_2 = \sqrt{(\hat{x} - x)^\top (\hat{x} - x)} \tag{9.11}$$

As opposed to the root mean square error (RMSE), which is another common metric, the EE is **not** adjusted (scaled down) for degrees of freedom. Thus, we would in general expect vectors with more elements (i.e. higher dimensionality) to produce larger estimation errors. This is something to keep in mind as we scrutinize this metric.

Performance metrics: normalized estimation error squared

The second metric used is the normalized estimation error squared (NEES). This is a measure of consistency, as its magnitude is adjusted for the estimated covariance $\hat{\mathbf{P}}$ of the state. It is calculated as follows:

$$\text{NEES}(x, \hat{x}, \hat{\mathbf{P}}) = (\hat{x} - x)^\top \hat{\mathbf{P}}^{-1} (\hat{x} - x) \tag{9.12}$$

This metric is particularly interesting, as it answers the question of filter consistency. In other words, it answers the question: ”Are you as inaccurate as you think you are?”. In the realm of state estimation, this is supremely important, as we need to know whether the covariance estimates can be trusted.

Furthermore, assuming that a state estimate $\hat{x}$ is distributed according to a multivariate Gaussian, with mean x and covariance $\hat{\mathbf{P}}$, (i.e. the filter model corresponds with reality), it follows that the NEES of $\hat{x}$ will be chi-squared, χ^2, distributed with degrees of freedom equal to the number of elements in x. Leveraging this, we now have a notion of how *probable* a given NEES value is. To contextualize this, NEES values are usually provided with bounds of a confidence interval, calculated using the appropriate χ^2 distribution.

Performance metrics: RMSE and ANEES

Both of the above metrics (EE and NEES) are calculated on a per timestep, per state subset basis. In other words, for every chosen state subset, we calculate a timeseries of that metric. This is still a lot of information which is produced per simulation, which can be overwhelming. For the sake of introducing some scalar, easy-to-digest metrics as well, I have opted to include root mean square error (RMSE) and average NEES (ANEES). These are produced as follows:

$$\text{RMSE}(x_1, ..., x_K, \hat{x}_1, ..., \hat{x}_K) = \sqrt{\frac{1}{K} \sum_{k=1}^{K} ||\hat{x}_k - x_k||_2^2} \tag{9.13}$$

$$\text{ANEES}(x_1, \hat{x}_1, \hat{\mathbf{P}}_1, ..., x_K, \hat{x}_K, \hat{\mathbf{P}}_K) = \frac{1}{K}\sum_{k=1}^{K}(\hat{x}_k - x_k)^\top \hat{\mathbf{P}}_k^{-1}(\hat{x}_k - x_k) \tag{9.14}$$

Performance metrics: ensemble averages

For both EE and NEES, it was opted to calculate ensemble averages over a number of Monte Carlo simulations. This was achieved by gathering timeseries data of EE and NEES values from several simulations, then averaging these values separately for each timestamp. In other words, the produced ensemble average is still a timeseries, but now every element is now the average of some metric across multiple simulations.

This is particularly useful for filters which are liable to occasionally diverge, or whose performance is very dependent on initial condition and vessel behaviour. (Such as ES-JLAT, in some configurations) For example, if the filter diverges horribly in 1 out of a 100 simulations, this constitutes a substantial problem which should be addressed. By running *enough* simulations, it is intended that such errant runs should be visible in the produced ensemble average. This helps demonstrate that scenarios were not cherry-picked to produce exaggerated performance.

Ensemble average NEES vs. ANEES:
It may appear redundant to include two different kinds of NEES averages as metrics, it was opted to do so for two reasons:

Firstly, it was deemed necessary to have some kind of timeseries metric in order to scrutinize the transient behaviour (from initialization to 'steady state'), the ensemble averaged NEES provides such a metric, which is also condensed from multiple simulations. This timeseries metric is also necessary for the purposes of seeing whether the metric in question is noisy, which is lost in a scalar metric.

Secondly, the ensemble averaged metric is not enough by itself. Since it smooths out its value across all simulations, it can hide certain behaviour. I.e. over- and under-confident runs can 'cancel out' into something that appears consistent. While it certainly is consistent in an averaged sense, we still want some per-simulation metric (i.e. ANEES) in which this behaviour is visible.

9.6 Sample output

To illustrate how the simulated vessels behave, some sample output is included in this section. This includes an XY plot of all three vessel trajectories, for a single simulation, as well as a plot of the generated ownship heading, for that same simulation.

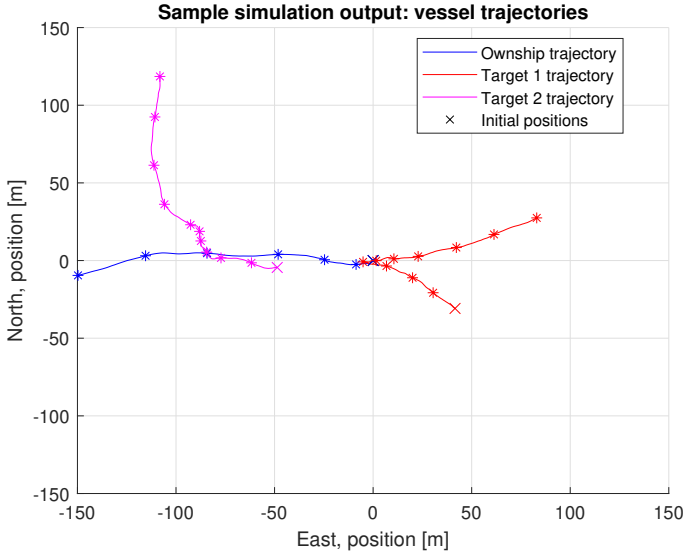

Figure 9.1: Vessel trajectories for a single simulation.

The motion of every vessel is shown in figure 9.1. This is an XY plot showing the position of every vessel, over the course of a 50 second simulation. Note that the starting positions are denoted with an $\times$ marker, and that $*$ markers are placed along the trajectories for every 5 seconds of the simulation. These are intended to give some notion of the vessel velocities, as well as their locations at specific times.

As specified in the initial conditions section, we observe that the ownship always starts out stationary in $[0, 0]$ and that both targets start on either side of the ownship, roughly 50 meters away from it.

Finally, observe that the vessels can start close enough to each other, (considering the total simulation time and the CV acceleration parameter), to cross, or even crash, at certain points during the simulation. This is of particular importance, as it proves the presence of (albeit mild) challenges of target association, as several targets can be close enough together to be mistaken for one another. Similarly, if the targets are able to get too close to the ownship, a minute error in their position estimates can result in a huge error in their supposed bearing (relative to the ownship). This makes inferring the heading from relative range-bearing scans very error prone, and at risk of filter divergence. Both these problems will be explored in further detail in the analysis section.

As for the ownship heading, a sample sequence is shown in figure 9.2. As specified, it starts out somewhere between $-\pi$ and π, and changes according to a random walk process from there. Although the heading (coincidentally) stays around -2.9 radians in this run, we observe that it changes quickly enough to potentially vary quite a bit over the course of 50 seconds.

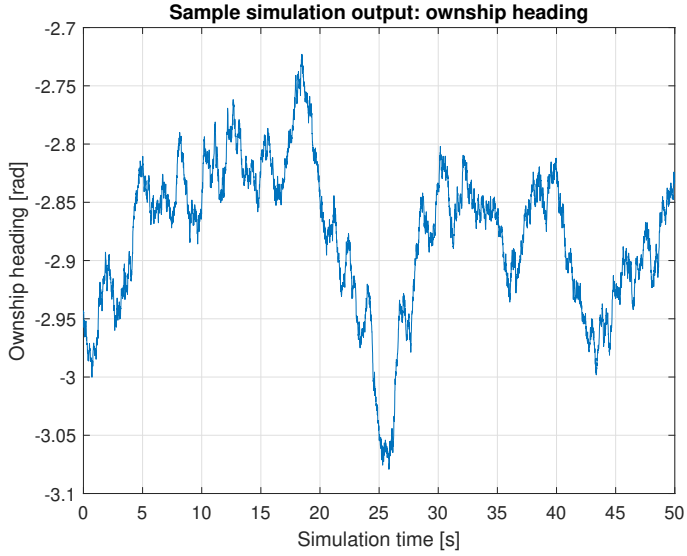

Figure 9.2: Ownship heading for a single simulation

In retrospect I realise that this random walk might still be a bit too ”nice”. A prudent piece of future work would be to increase the driving noise parameter (from $0.16\ (rad/s)^2$) in order to make heading estimation more difficult.

Chapter 10

Analysis - Simulation results

In this chapter, simulation results are presented for a number of configurations. These results are usually presented in the context of N Monte Carlo simulations, with some metrics (RMSE and ANEES) being produced *per-simulation*, while others, such as ensemble averages of EE and NEES are produced from the set of all N simulations.

As for simulation configuration in this chapter, the main parameter which is varied between runs is the choice of motion model for the ES-JLAT scheme. The following five sets are included:

1. CV (without relative scans)
2. RESCV-IC
3. RESCV-IC (with injections only on relative scans)
4. ESCV-IC (using direct injection)
5. ESCV-IC (using weighted injection)

A common theme for all these results is that no odometry is considered, and that every filter is provided with scans that *only* contain detections of the relevant target. In other words, these results **do not demonstrate any kind of target association performance**, nor does it challenge the filters in terms of consistency. The latter is a result of the following: the vessel models used to generate the simulation data **exactly match** the motion and measurement models used in both FF and ES-JLAT filters. By extension, full consistency should be achievable in these simulations. There is no 'real-life' black box which introduces modelling errors which might cause inconsistency.

Category	Setting	Symbol	Value	Unit
Simulation	Number of simulations	N	30	
	Simulation time	T_{sim}	50	s
	Odometry included		no	
ES-JLAT	Motion model		CV	
	Pre-associated scans		yes	
	Relative scans included		no	
	Clutter rate	λ	0	
	Detection probability	P_D	1.0	
FF	Pre-associated scans		yes	
	Relative scans included		no	

Table 10.1: Simulation configuration for 'CV (without relative scans)'

A recurring theme in this chapter is comparison between the full filter (FF) solution and the ES-JLAT solution. For the sake of consistency, as well as more compact presentation, the results of FF are always plotted in blue, while the results of ES-JLAT are always plotted in red.

For all NEES and ANEES values presented in this chapter, a corresponding 95% confidence interval is indicated in the plots. The bounds of this interval are plotted with dashed black lines, and are of course calculated from the CDF of the χ^2 distribution, with the appropriate degrees of freedom.

Furthermore, for the ensemble averages of NEES values, the percentages of timesteps which fell within the confidence bounds are indicated in the sub-title for each plot. These are denoted with the shorthand "FF: $x\%$, ES-JLAT: $y\%$", where x and y are the percentage-within value for that metric, for the full filter and the ES-JLAT solution, respectively.

10.1 CV (without relative scans)

This first run included does not feature any of the newly introduced ES-JLAT functionality. It is simply a comparison between the full filter (FF) solution and a more modular solution with separate filters for localization and tracking. Since it does not include relative scans, it becomes a testament to what can be achieved with only absolute detections. The configuration used for this run is shown in table 10.1.

Since no relative scans are present, there is **no** information present with which to infer the heading. For this reason, no plots for heading estimation are included in this section. There is still a subtle influence of heading error on the ownship state x_o, however this is mostly dominated by much larger errors in position and velocity, and can safely be neglected.

10.1.1 Results: CV (without relative scans)

The simulation results for this run are shown in figures 10.1, 10.2 , 10.3, 10.4. Of these, the first two are ensemble averages over all 30 simulations, with time along the x-axis, while the latter two are per-simulation results, with the simulation number indicated along the x-axis of each plot (1 through 30).

Analysis: Estimation accuracy

For the estimation accuracy, see the EE and RMSE metrics, figures 10.2 and 10.4, respectively. Observe in figure 10.2 that the EE values for FF and ES-JLAT are approximately identical for all states. In the RMSE plot, observe that the ES-JLAT solution is marginally worse for x_1, x_2, p_1, p_2. I posit that the FF is slightly better here because it can learn inter-vessel correlations during each simulation. However this difference is very subtle, and corresponds to a positional error of a few centimeters.

Analysis: Estimation consistency

To scrutinize how well the estimated covariances match with the actual errors, I refer to the NEES and ANEES plots, figures 10.1, 10.3. For both of these, observe that the FF grossly over-estimates the covariances. I will refer to this type of inconsistency as under-confidence, i.e. the actual errors are much smaller than their covariance estimates indicate. This manifests as NEES and ANEES values which are **much** smaller than their corresponding confidence bounds (indicated with dashed black lines).

As for the ES-JLAT, which in this case uses the exact same motion models as the FF, we observe much better consistency on all counts (except for φ_o). This manifests as NEES and ANEES values which are either inside or near their confidence bounds. In this case we would expect perfect consistency, as all filter models *perfectly* match the models used to produce the simulated data. Perfect consistency in this case should manifest as all NEES percentages being *exactly* 95%.[1]

Note that there is only one difference between these two solutions in this configuration. Namely that the ES-JLAT has separate filters for each vessel, while the FF has a single filter containing all three models. In other words, the ES-JLAT *only* has marginal covariance estimates ($\hat{\mathbf{P}}_{x_1}$, $\hat{\mathbf{P}}_{x_2}$, etc.), while the full filter also has cross correlations such as ($\hat{\mathbf{P}}_{x_o,x_2}$, $\hat{\mathbf{P}}_{x_1,x_2}$, etc.). This difference in information available is made obvious if we stack together the covariance estimates provided by either solution as $\hat{\mathbf{P}}_{\text{ES-JLAT}}$ and $\hat{\mathbf{P}}_{\text{FF}}$:

[1] If I were to set the number of simulations, N, much higher, I would expect these results to exactly match their confidence bounds.

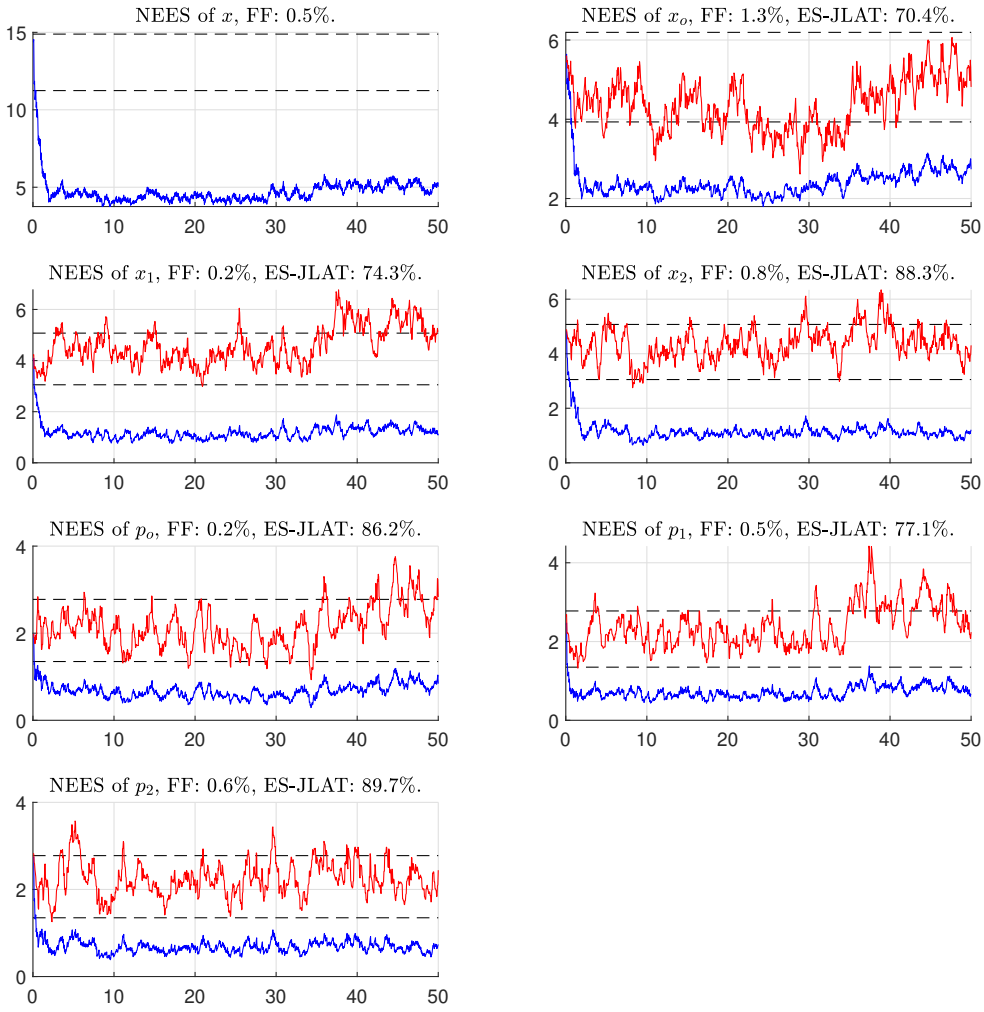

Figure 10.1: Ensemble averaged NEES values for 'CV (without relative scans)'

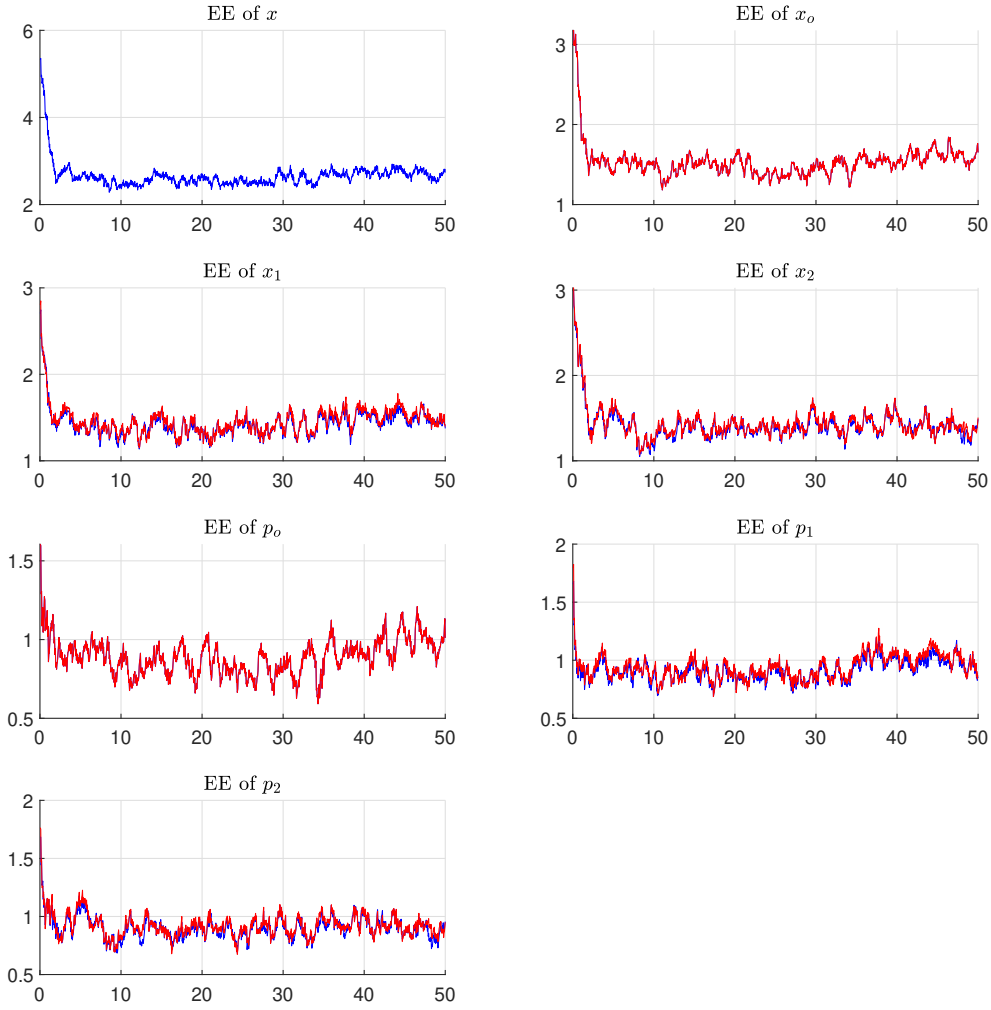

Figure 10.2: Ensemble averaged EE values for 'CV (without relative scans)'

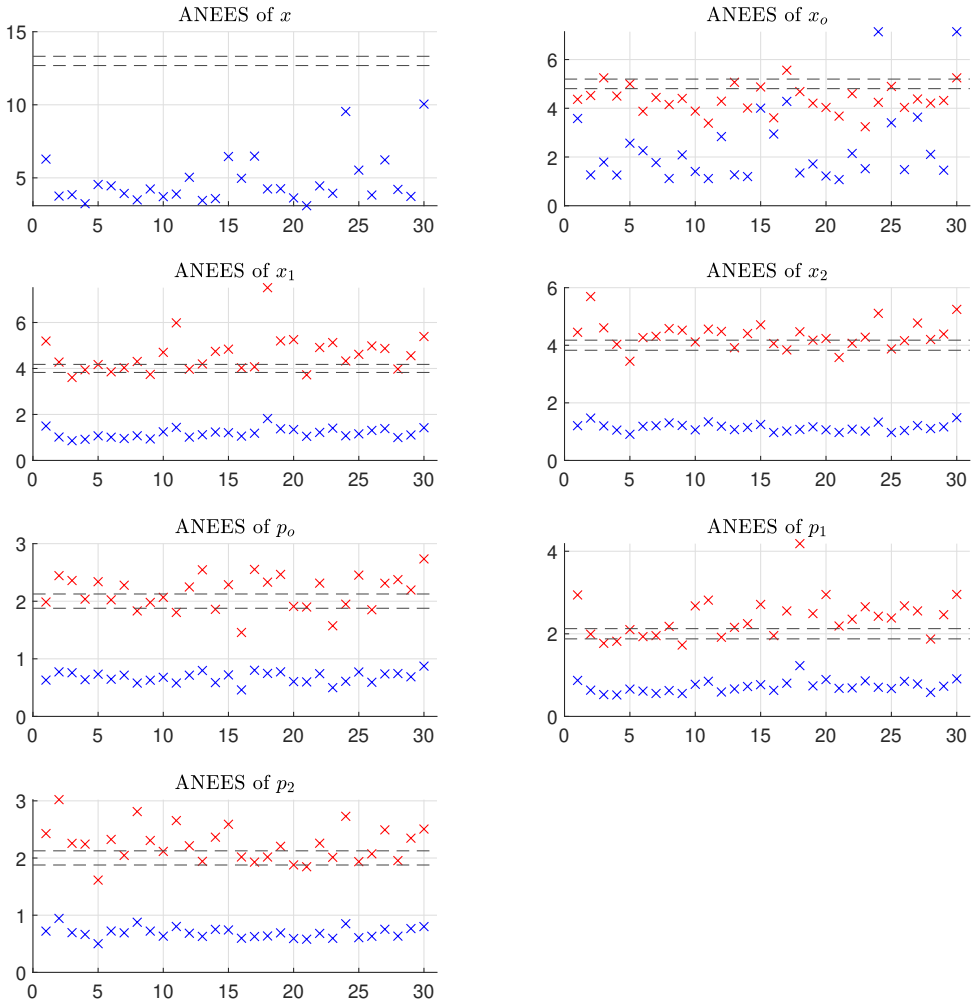

Figure 10.3: Per simulation ANEES values for 'CV (without relative scans)'

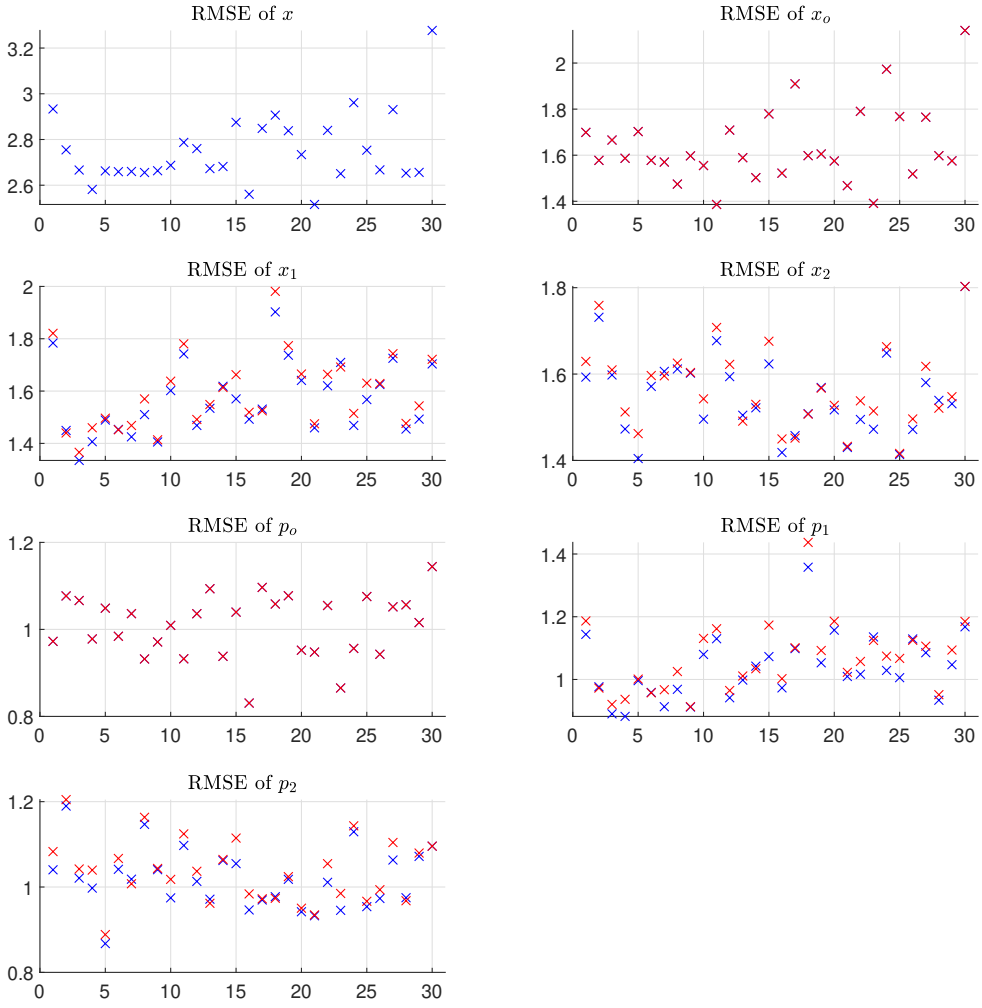

Figure 10.4: Per simulation RMSE values for 'CV (without relative scans)'

$$\hat{\mathbf{P}}_{\text{ES-JLAT}} = \begin{bmatrix} \hat{\mathbf{P}}_{x_o} & \mathbf{0}_{5x4} & \mathbf{0}_{5x4} \\ \mathbf{0}_{4x5} & \hat{\mathbf{P}}_{x_1} & \mathbf{0}_{4x4} \\ \mathbf{0}_{4x5} & \mathbf{0}_{4x4} & \hat{\mathbf{P}}_{x_2} \end{bmatrix} \qquad \hat{\mathbf{P}}_{\text{FF}} = \begin{bmatrix} \hat{\mathbf{P}}_{x_o} & \hat{\mathbf{P}}_{x_o,x_1} & \hat{\mathbf{P}}_{x_o,x_2} \\ \hat{\mathbf{P}}_{x_1,x_o} & \hat{\mathbf{P}}_{x_1} & \hat{\mathbf{P}}_{x_1,x_2} \\ \hat{\mathbf{P}}_{x_2,x_o} & \hat{\mathbf{P}}_{x_2,x_1} & \hat{\mathbf{P}}_{x_2} \end{bmatrix} \tag{10.1}$$

We observed that the ES-JLAT has decent consistency while the FF is grossly under-confident, I posit that this is a result of the aforementioned inter-vessel correlations. These correlations matrices will "allow" the marginal covariances ($\hat{\mathbf{P}}_{\text{FF},x_1}$, $\hat{\mathbf{P}}_{\text{FF},x_2}$, etc.) to reach larger values, since their effect (magnitude) is being mitigated by matching correlations. This becomes a problem when we marginalize $\hat{\mathbf{P}}_{\text{FF}}$ into ($\hat{\mathbf{P}}_{x_1}$, $\hat{\mathbf{P}}_{x_2}$, etc.) as part of the NEES calculation. By doing this we are effectively ignoring the context of why these elements were so large in the first place, and the resulting NEES values become disproportionately small.

On marginalization vs. conditioning:
This observed behaviour is partly caused by the subset NEES values being marginalized from a multivariate Gaussian. In fact, if we were to extract individual state subsets $\{\hat{x}_1, \hat{\mathbf{P}}_{x_1}\}$ by instead conditioning the elements of $\{\hat{x}_{\text{FF}}, \hat{\mathbf{P}}_{\text{FF}}\}$ on the remainder of the state vector, we would expect more consistent results. This is a recurring theme for all FF results presented in this thesis, and worth bearing in mind.

Noting that the full state x is also grossly under-confident (despite not being marginalized), there has to be something else at play as well. I expect this to be a result of the inter-vessel correlations learned by the FF being exceedingly accurate. When the vessel behaviour matches these correlations, the NEES values should become correspondingly lower (resulting in perceived under-confidence). Since the filter is readily updated with full (all vessels correlated) measurement updates, the FF gets excellent information with which to learn these correlations (and adapt them quickly to new behaviour).

10.2 RESCV-IC

This next run switches to the RESCV-IC motion model in the ES-JLAT. Furthermore, we now also provide relative scans to both filters, which provides a means of inferring the ownship heading. Otherwise, all tuning parameters are set equal to their true values, which should be a good premise for consistent results. The full simulation configuration is shown in table 10.2, with relevant changes from the previous run being indicated in **bold font**. The only error state included in the RESCV-IC model is the ownship heading. I.e. we would expect similar performance to the CV model on all accounts except for the heading, which should now hopefully have a sound estimate.

Category	Setting	Symbol	Value	Unit
Simulation	Number of simulations	N	30	
	Simulation time	T_{sim}	50	s
	Odometry included		no	
	Relative scans included		**yes**	
ES-JLAT	**Motion model**		**RESCV-IC**	
	Injection type		direct	
	Pre-associated scans		yes	
	Clutter rate	λ	0	
	Detection probability	P_D	1.0	
	Heading ES covariance gain	K_φ	$\mathbf{1.2}$	
FF	Pre-associated scans		yes	

Table 10.2: Simulation configuration for 'RESCV-IC'

10.2.1 Results: RESCV-IC

The simulation results for this run are shown in figures 10.5, 10.6 , 10.7, 10.8. Of these, the first two are ensemble averages over all 30 simulations, while the latter two are per-simulation results, with the simulation number indicated along the x-axis of each plot (1 through 30).

Analysis: Estimation accuracy

Starting with the estimation accuracy, this is demonstrated by the EE and RMSE values, shown in figures 10.6 and 10.8, respectively.

Noting that the change from CV to RESCV-IC provided us with a mechanism with which to estimate heading, we start with scrutinizing the heading estimation in detail. In general, both schemes appear to be able to estimate heading with reasonable success, staying within 0.1 radians (roughly 6 degrees) of error on average. Considering that the angular measurement uncertainty σ_θ^2 is set to $0.1\ rad^2$ (STD of roughly 18 degrees), and that the true heading changes constantly with time, we would expect these two effects to create a upper bound on achievable estimation accuracy. Finding the exact value of this bound would be subject of a Cramer-Rao lower bound analysis (or something similar), but I leave this as a point of future work. Instead, I simply make the qualitative claim of being pleased with heading estimation which is more precise than the angular measurement noise variance.

Looking at the ES-JLAT results, we observe its heading estimates are in fact better than the FF results! This is somewhat unexpected from the authors standpoint, but certainly stands as a proof-of-concept for ES-JLAT being able to improve localization estimates. Specifically, we observe that the EE of the heading stays between 0.065 to 0.1 radians for

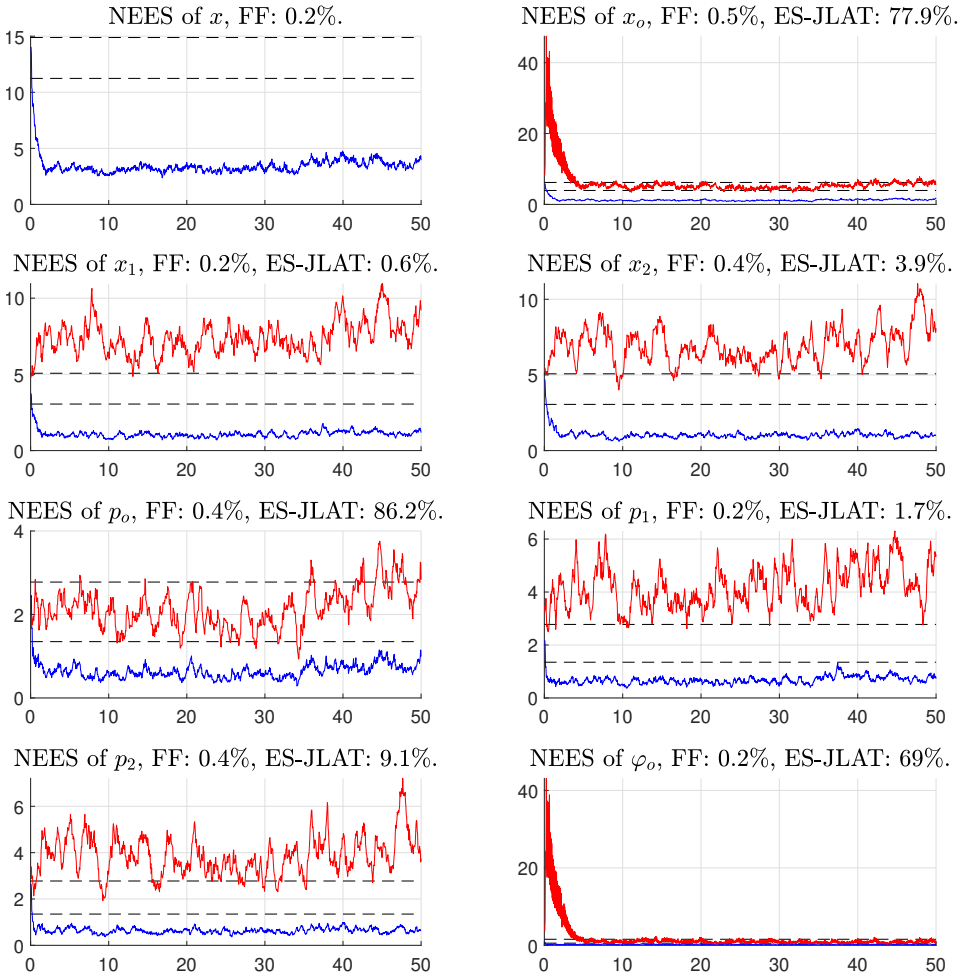

Figure 10.5: Ensemble averaged NEES values for 'RESCV-IC'

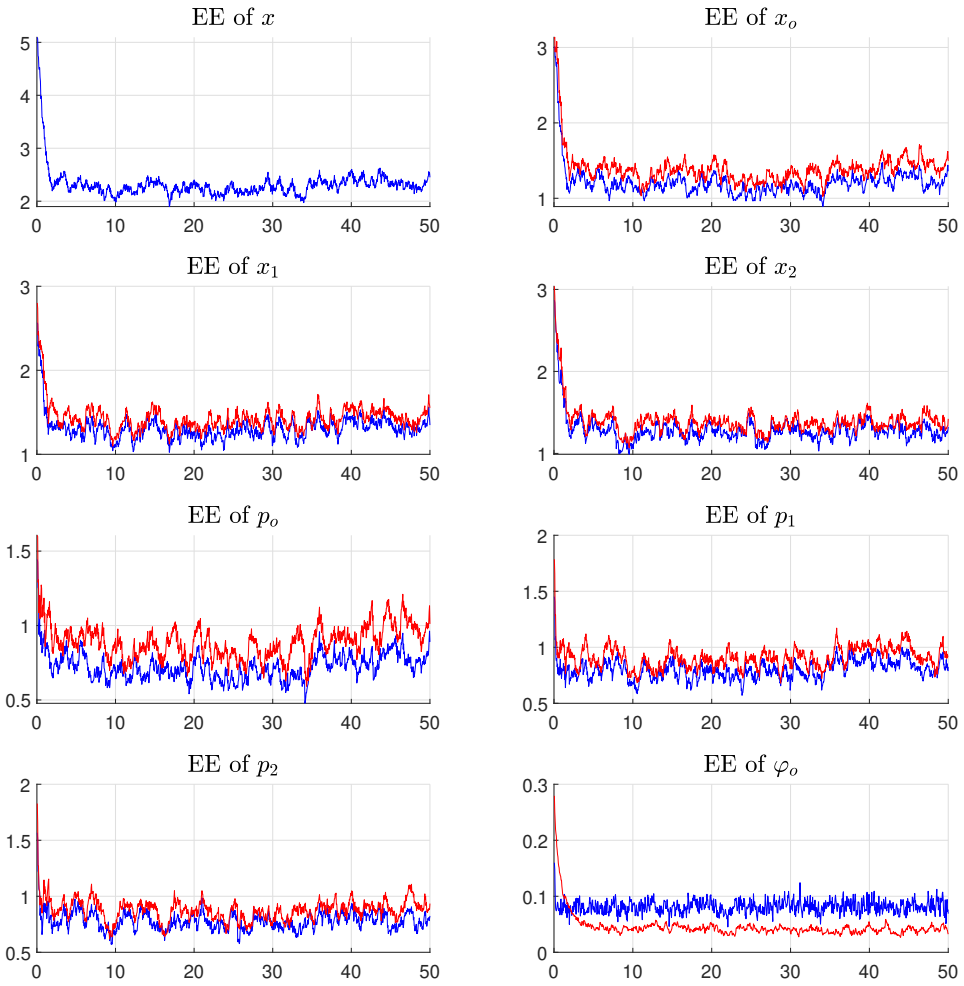

Figure 10.6: Ensemble averaged EE values for 'RESCV-IC'

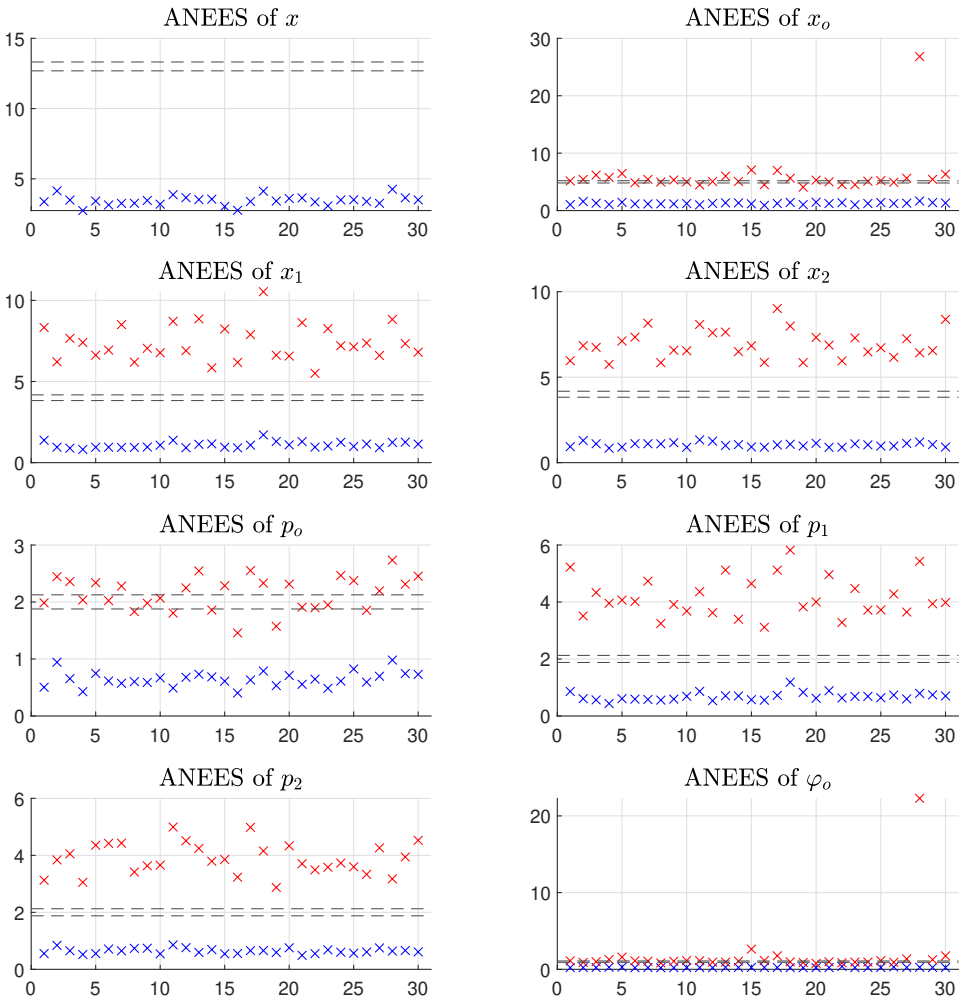

Figure 10.7: Per simulation ANEES values for 'RESCV-IC'

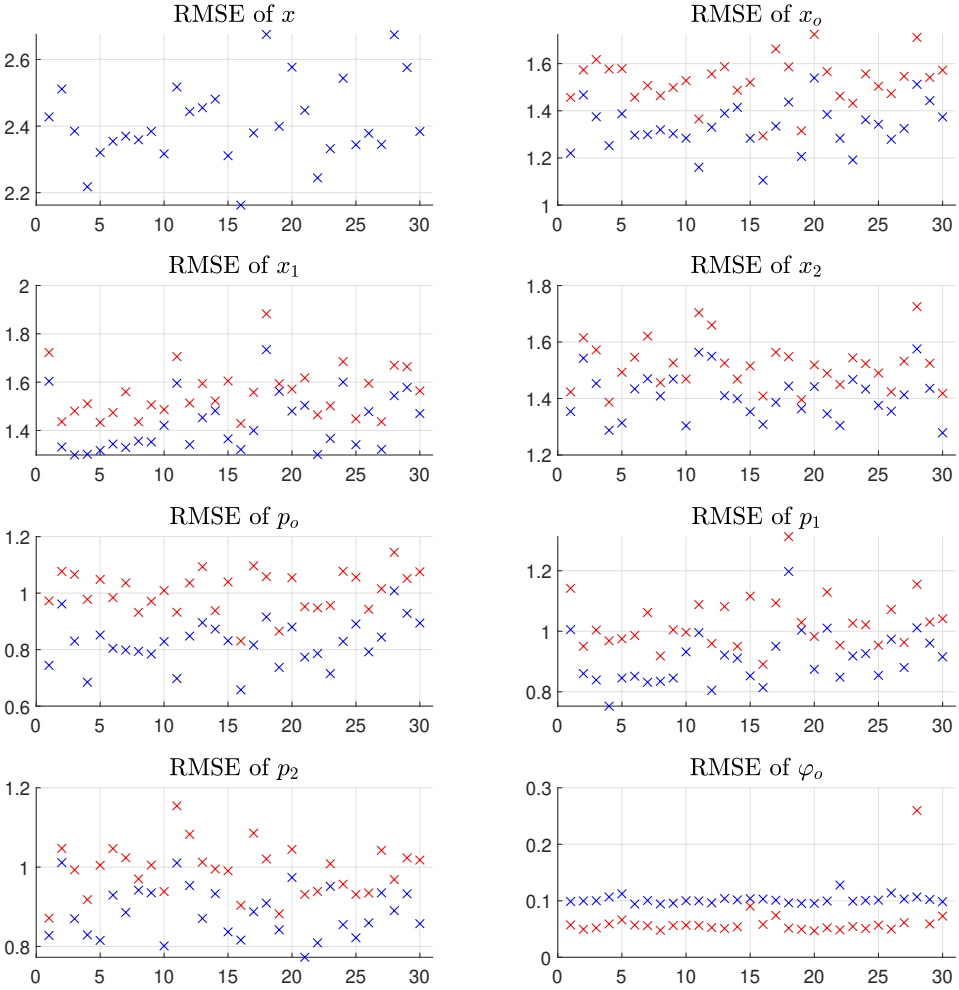

Figure 10.8: Per simulation RMSE values for 'RESCV-IC'

the FF, while the steady state values for the ES-JLAT stay between 0.04 and 0.05 radians. (Twice as accurate!)

Comparing the accuracy of the remaining states ($x_o, x_1, x_2, p_o, p_1, p_2$), we observe that this improvement in heading accuracy did not come without a price. In fact, *all* of these metrics are marginally worse with the ES-JLAT scheme. For the positional estimates, this manifests as an increased error of between 10 and 20 centimeters (on average).

To some extent, this is expected, as the RESCV-IC model provides the tracker(s) with the option to 'blame' a heading estimate error instead of using all information available to update the target estimate (position and velocity). By assuming that at some points in time, a target moves unexpectedly in a tangential direction (relative to the ownship) its motion is now liable to be 'blamed' on the heading error state instead.

Let's illustrate this with an example. Say that both targets move unexpectedly and equally fast in opposite tangential directions, they have similar distances to the ownship, similar prior covariances etc. Each of them will produce an equal and opposite heading error state estimate, which when sent to the consensus model should cancel out. Thus we are left with the localization estimate being unaffected (as it should be) and both targets experiencing less of an update to their tangential motion than was the case. This illustrates an obvious way in which the ES-JLAT scheme can be detrimental to tracking performance.[2]

On the effects of letting trackers 'blame' the localization error:
It was stated previously that letting a tracker attribute some unexpected measurement to a localization error (instead of a change in target state), might have a detrimental effect on tracking performance.

This is a null argument in some sense, as I could just as easily state that this effect should in fact be positive for the tracking performance, since the localization error associated with using relative scans is now *less* likely to be injected into the track estimates.

At the time of writing this I can only conclude that these are two interchangeable effects which are counteracting each other, and thus that speculating on their net effect is tricky. For this reason, I opt to not speculate much further on these effects, and instead try to observe which of them appear to dominate in the various simulations.

A more unexpected result is that the ownship position estimate p_o also appears to have taken a hit (i.e. 20-30 cm worse accuracy) in the ES-JLAT scheme, when compared to FF. Considering that the RESCV-IC model does not make any correction on the ownship position, this difference in accuracy has to stem from something else. By comparing the RSME performance of this RESCV-IC run to the previous CV run (fig. 10.4 vs. fig. 10.8), we get to the heart of the matter. When comparing the FF performance of these

[2]This presents a particularly interesting point of future work. Namely to look at the individual error state estimates associated with each track, and see how that matches up with the final consensus. If they do not match up, it might be prudent to inject these differences back into the target estimates! I think this would be a very interesting way to counteract the problems illustrated by the aforementioned 'cancel-out' hypothetical.

two runs, we find a reduction in error of roughly 0.2 to 0.3 across the board (i.e. for $x_o, x_1, x_2, p_o, p_1, p_2$). Recall that for the RESCV-IC run, we included relative scans as part of the simulation. This is the **only** difference in the FF configuration for these two runs, and thus where this improvement has to come from. Intuitively enough, when providing a measurement which intrinsically connects the positions of all three vessels, their estimates improve.

Now comparing the ES-JLAT performance instead (still fig. 10.4 vs. fig. 10.8), we observe that the positional elements p_o, p_1, p_2 actually have very similar values. From this, we draw a very crucial observation: **Introducing error state injection does not necessarily cause worse tracking accuracy, but it does lessen the improvement on tracking accuracy we would expect to see by introducing relative scans.**

In some sense, we have effectively traded off the potential for improved tracking performance, by instead choosing to use the relative scans in order to improve the heading estimate. Note of course that the RESCV(-IC) model is only capable of improving the heading estimate, as that is its only error state. It remains to be seen whether we are also able to improve the rest of the localization estimate by also including these as error states, which will be explored in the results of the ESCV(-IC) models.

A final observation for the heading estimation of the ES-JLAT, is the presence of a fairly slow transient. In the EE plot of φ_o (fig. 10.6), we observe a transient from 0.3 to 0.05, lasting about 7 seconds. This is dramatically slower than the FF, which appears to stabilize to 0.1 almost momentarily. This illustrates a significant weakness in the current ES-JLAT approach, namely that it seems to struggle with a large initial heading error. In fact, looking at the φ_o RMSE value for simulation 28 (fig. 10.8), which is at a huge 0.3 radians, it would appear that the ES-JLAT never recovered from a too large initial error.

It seems prudent then to impose that such a system (ES-JLAT using RESCV-IC) should never be the lone guarantor of heading estimation, and rather be responsible for small corrections on an already decent estimate. In other words, the fact that this system (as the lone guarantor of heading estimation) was able to maintain a (better) estimate of heading in 29 out of 30 cases is nigh on a miracle.

Analysis: Estimation consistency

Looking now at the consistency performance (figures 10.5 and 10.7), we see that the introduction of the error states has had a largely adverse effect for the ES-JLAT. In fact, for all target metrics x_1, x_2, p_1, p_2, both the NEES and ANEES values are decidedly overconfident. I.e. the actual error is much larger than the covariance estimates would suggest. Since the absolute values of the errors have not changed much (when compared to the CV run) I deem it likely that the new covariance estimates are disproportionally small.

This also seems plausible, considering the trackers are now provided with another set of scans, which results in another set of update steps. Each update step will necessarily reduce

the covariance estimate, thus it comes as no surprise that more frequent updates should result in an overall lower covariance. Since we do not observe a matching reduction in estimation error, only one of the two aspects of consistency have been reduced, resulting in over-confident results.

It seems prudent then to investigate whether a different filter structure (for such error state motion models) might be more suited to this kind of update, as the current choice appears to tank the target-specific covariance far too much. I leave this as a point of future work, with the idea that something like an (F)PSKF, as introduced by Brink in [7], might be suited to the task.

As for the ownship estimates x_o, p_o, φ_o, some consistency appears to have been retained. Especially for the NEES of p_o, which is at a decent 86% (where we would ideally want 95%). As in the estimation accuracy section, we see a slow transient present in the heading estimate. Furthermore, we see how noisy and 'jittery' the covariance estimates of φ_o are during this transient (this is also present in the x_o plot, as φ_o is in fact an element of x_o). This is a significant problem, and again proves how this RESCV-IC ES-JLAT configuration struggles with large (initial) localization errors.

After recovering from the transient, the NEES percentages of x_o, φ_o are decent (78% and 69%), but still a far cry from consistency. It was attempted to tune the filter to achieve better consistency (for φ_o and consequently x_o), by adjusting the covariance injection gain K_φ. However, this was the best I could achieve with the current configuration. As previously mentioned, we observe a lot of high-frequency noise in the NEES of φ_o and x_o. Which I suspect is caused by over-compensation in the covariance reduction of some update steps. I.e. an update step (or combination of multiple) drives the covariance too low, the predict step drives the covariance up again, and this results in an oscillation of sorts.

On the high frequency noise present in the φ_o NEES:
Interestingly, I found that this noise was greatly reduced if I switched the ES-JLAT scheme to only run injection steps after updates with relative scans. Whereas the configuration for this run will do injection steps after *all* updates. Seemingly, the injection steps (from absolute scans) did not adequately reduce error as much as they reduced covariance, resulting in inconsistent behaviour. For the next run of simulations, this is explored in detail.

In retrospect, I should have seen this coming, considering the injection step of a cartesian scan does not contain any correlation between the ownship heading and a target's position. The result of this is an injection step which does not inject any changes to the heading estimate, **but still blindly reduces the covariance** according to the KF covariance update equation. Trying to tune such a filter is of course a fool's errand.

This highlights an inherent problem with the way 'post injection localization covariance' is implemented. Namely that the observation matrix $\mathbf{H}$ used for this reduction: $\mathbf{P}_{o,\text{post-injection}} = (\mathbf{I} - \mathbf{WH})\mathbf{P}_{o,\text{pre-injection}}$ is actually just the error state observation matrix $\mathbf{H}_{\text{o}\to\text{RES}}$. This does **not** correspond with which states are observable in the measurement, but rather which localization states are estimated as error states. By extension, it naïvely reduces the covariances for all error states, regardless of whether they are observed. This is a **pressing** point of future work, namely to replace this observation matrix with one that also imposes that a localization state has to be observable in the measurement for its covariance to be reduced in an injection.

Another interesting point of future work thus becomes to design a consensus scheme which able to decide for itself whether a scan (and its update step) is relevant for injection. This sounds a lot like ISKF of (F)PSKF, which are obvious candidates. The required heuristic for such a scheme may be as simple as: "Does the sensor provide relative or absolute scans?" or equivalently: "Is the sensor mounted on the ownship or somewhere else?". Where it is of course the former which are relevant.

Regarding the FF, we still observe massive under-confidence on all states. As with the CV run, I expect this to be caused largely by disregarding inter-vessel correlations upon calculating NEES and ANEES values. Otherwise, there is not much interesting to remark on here.

10.3 RESCV-IC (with injections only on relative scans)

This next run is configured exactly like previous one, with one exception: the ES-JLAT scheme now only runs injection steps after updates using relative scans. This was explored as a means of reducing high-frequency noise observed in the heading covariance estimate, and turned out to have greatly superior consistency properties.

Since there are now fewer overall injections made on the localization estimate, I was also able to reduce the ES covariance gain, K_φ, from 1.2 to 1.0. This parameter essentially controls how much a injection step is allowed to reduce the localization covariance. A higher gain would result in a higher innovation covariance, and consequently a smaller covariance reduction. The relevant configuration is shown in table 10.3, and as usual, the relevant changes are indicated with **bold font**.

Category	Setting	Symbol	Value	Unit
Simulation	Number of simulations	N	30	
	Simulation time	T_{sim}	50	s
	Odometry included		no	
	Relative scans included		yes	
ES-JLAT	Motion model		RESCV-IC	
	Injection type		direct	
	Pre-associated scans		yes	
	Clutter rate	λ	0	
	Detection probability	P_D	1.0	
	Heading ES covariance gain	K_φ	**1**	
	Injection on abs update		**no**	
FF	Pre-associated scans		yes	

Table 10.3: Simulation configuration for 'RESCV-IC (with injections only on relative scans)'

10.3.1 Results: RESCV-IC (with injections only on relative scans)

The simulation results for this run are presented in figures 10.9, 10.10, 10.11, 10.12. Due to the large similarity with the previous run, the analysis included here will be brief, and largely focus on the changes in the ES-JLAT performance for ownship heading, φ_o. For the FF, there are no changes between this run and the previous one, so its results are only included here for convenient comparison.

Analysis: Estimation accuracy

Looking at the EE for φ_o in figure 10.10, we observe almost identical steady-state performance to that of the previous run (fig. 10.6). By zooming in painfully far, we can observe an increased EE of about 0.005 radians (0.3 degrees), although this is certainly small enough to write off as standard deviation.

There are two distinctions I wish to point out. First of all, we have no diverged runs for the ES-JLAT, which is a slight improvement. A reduction from 1/30 to 0/30 is hardly statisti-

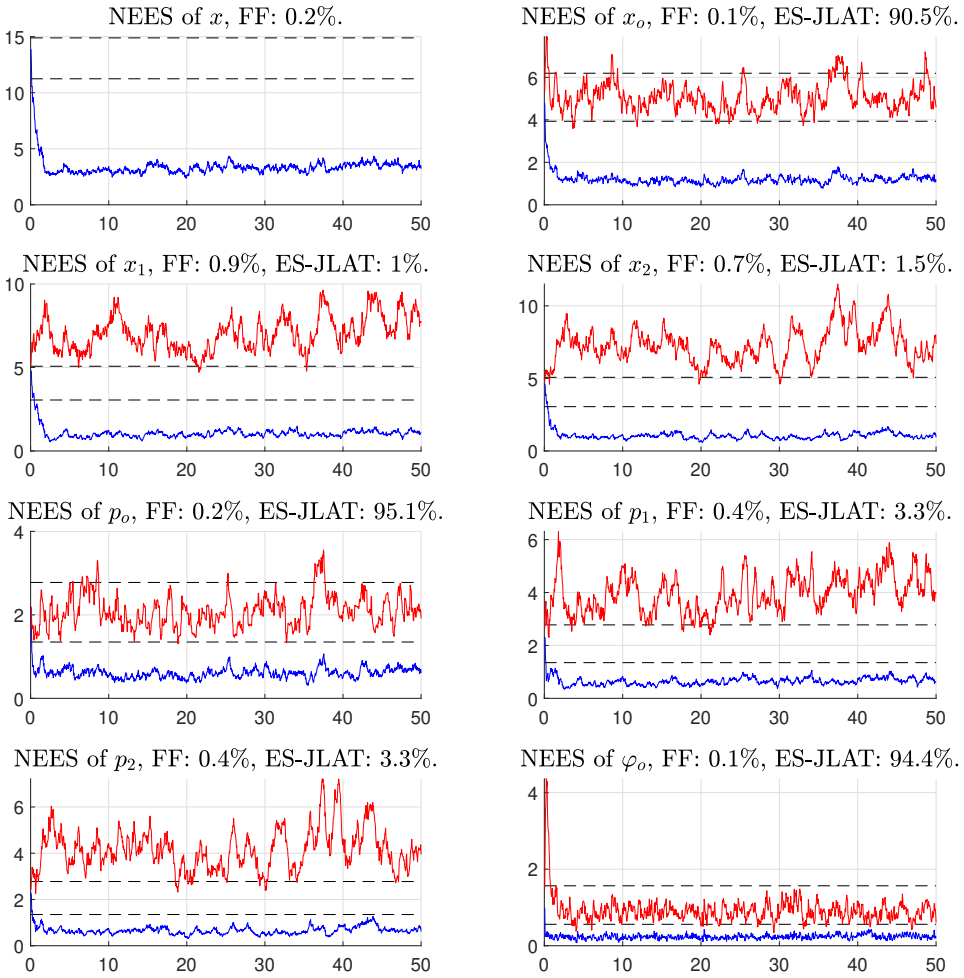

Figure 10.9: Ensemble averaged NEES values for 'RESCV-IC (with injections only on relative scans)'

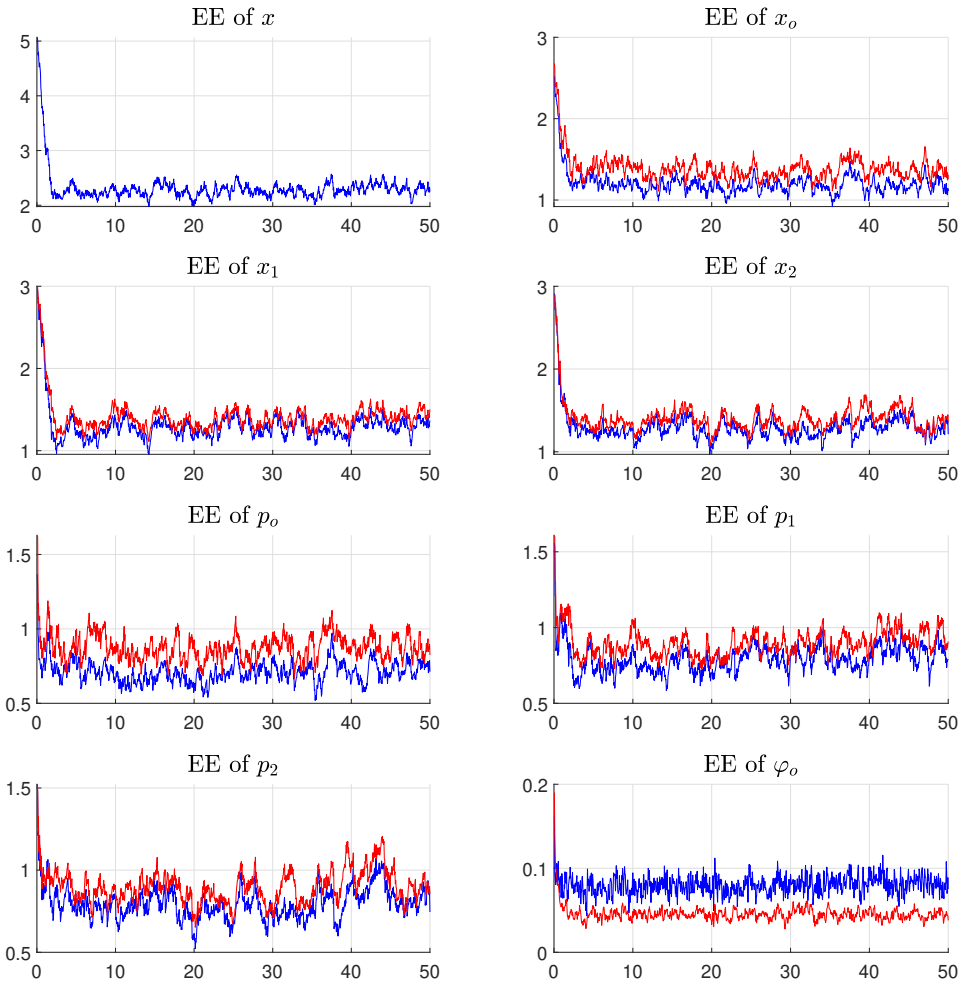

Figure 10.10: Ensemble averaged EE values for 'RESCV-IC (with injections only on relative scans)'

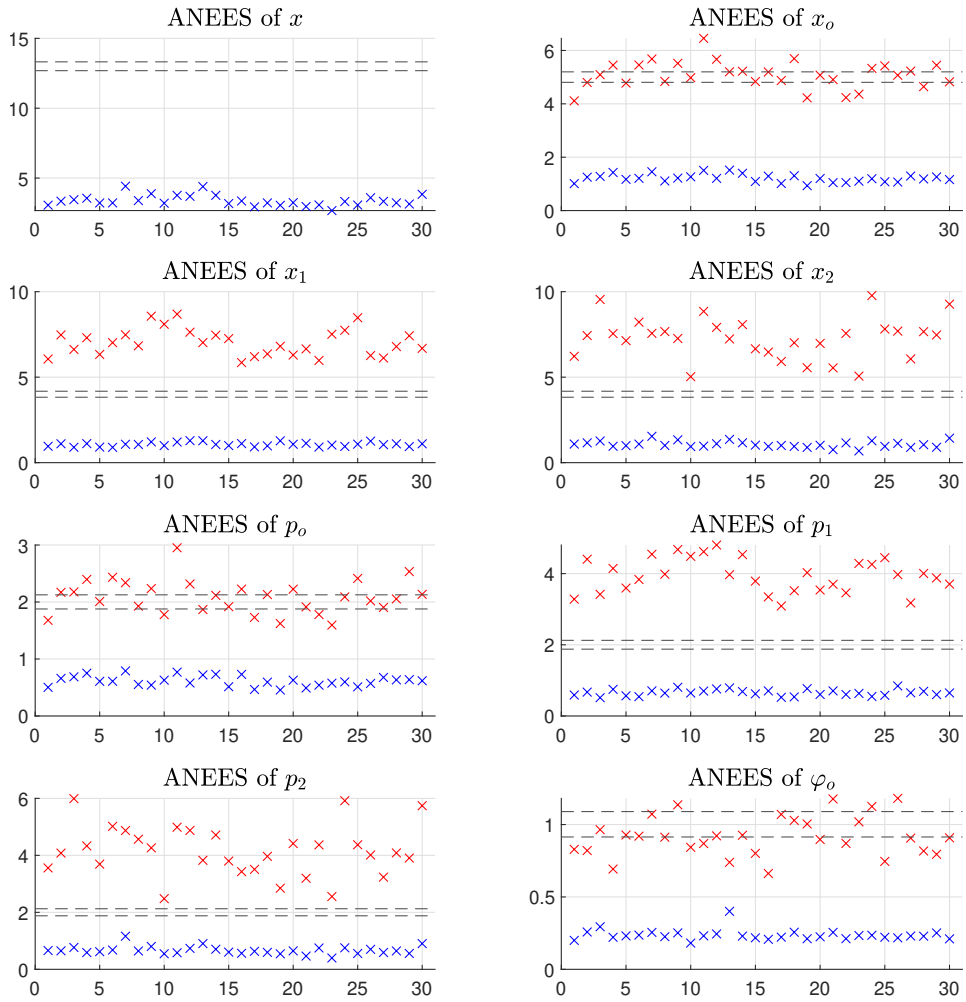

Figure 10.11: Per simulation ANEES values for 'RESCV-IC (with injections only on relative scans)'

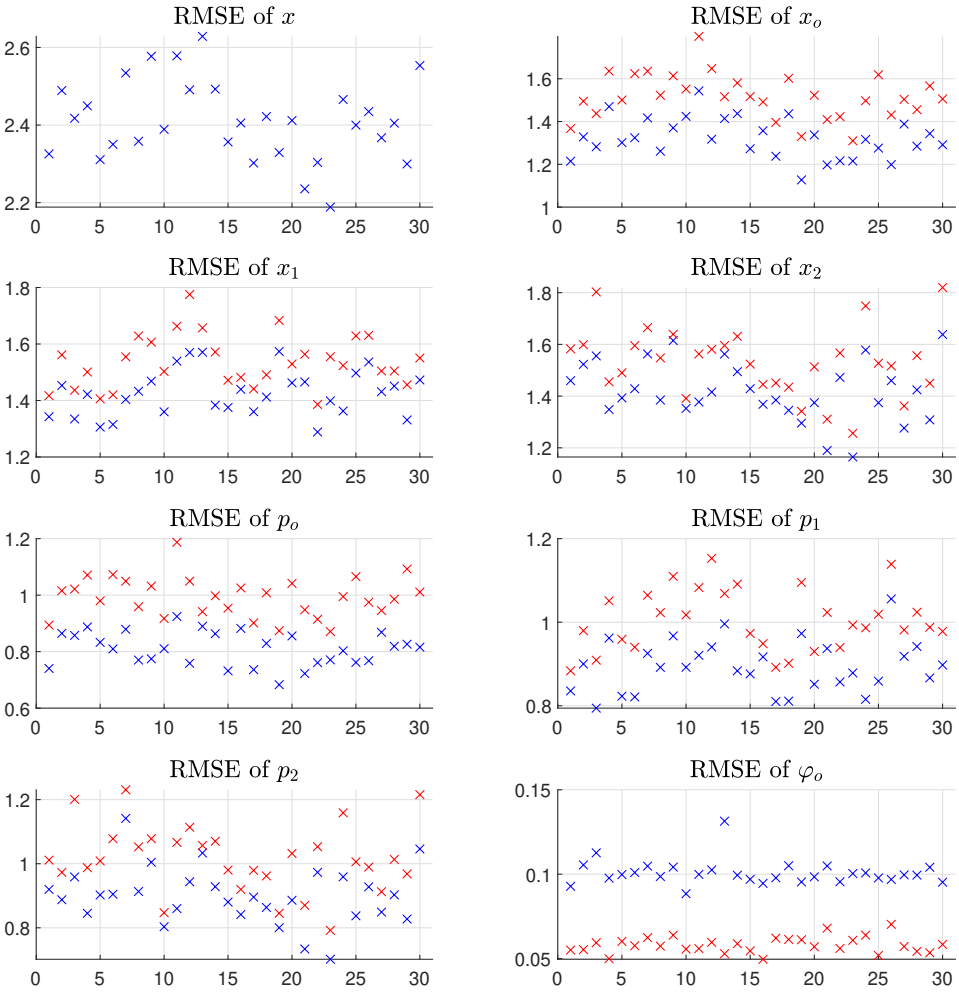

Figure 10.12: Per simulation RMSE values for 'RESCV-IC (with injections only on relative scans)'

cally significant though, so take this with a grain of salt. It could simply be that this run of simulations was more lucky with initial conditions or the random vessel trajectories.[3]

The second distinction, and much more important in my eyes, is the **much faster** transient for this setup. By transient I mean the time spent (from simulation start) until the heading estimate stabilizes at around 0.05 radians of error. Comparing it to the previous run (fig. 10.6), we see that the transient now lasts approximately 1 second, where it previously lasted around 7 seconds. This is a huge improvement, and in this regard the ES-JLAT now has performance comparable to that of the FF.

I suspect this to be mitigated now, as we have eliminated the spurious injections during cartesian updates. These injections, although not affecting the heading estimate, can still drive the marginal heading covariance to erroneously small values. In the following the update steps, the tracking filters would be less likely to attribute measurement errors to error in the heading estimate (since the prior covariance is lower). By extension, ES-JLAT is potentially making small corrections where large corrections are warranted. I expect this new configuration of RESCV-IC to more aptly reduce the covariance only when the heading estimate has become correspondingly accurate, thus resulting in a faster (smoother) transient.

Analysis: Estimation consistency

Excluding the heading transient, it is in consistency of the ownship estimates, x_o, p_o, φ_o, where we see huge improvements in this run. Looking at their NEES percentages for ES-JLAT (fig. 10.9), being $90.5, 95.1, 94.4$ percent, respectively. This is **very** close to the ideal 95%, and from this I conclude that the localization filter maintains consistency throughout its interaction with this variety of ES-JLAT. This constitutes a resounding success for ES-JLAT, and it would appear that we are nearing the goal of a JLAT scheme with both excellent accuracy and consistency.

In the previous run, we observed a very high frequency oscillation in the φ_o NEES, mostly during the transient, but still somewhat present during steady state. This has been mostly eliminated, and is of course also a big improvement over the previous run.

Glancing quickly at the target states, we observe no change from the previous run. That is, all target state estimates x_1, x_2, p_1, p_2 are still over-confident.

Finally, looking at all ANEES values for the ES-JLAT scheme (fig. 10.11) we observe quite a spread in their values (as opposed to the FF, which produced very similar ANEES values for every simulation). For even the supposedly consistent ownship states x_o, p_o, φ_o, we observe that a few them fall above and below their confidence bounds. This implies that the consistency of the RESCV-IC + ES-JLAT scheme is very dependent on filter initialization or vessel trajectory, and may occasionally be either under-confident or overconfident.

[3] A point of future work would be running a similar setup with a lot more simulations in both runs, which should give more conclusive results.

10.4 ESCV-IC (using direct injection)

Having noted that the RESCV-IC previously tested is seemingly capable of maintaining a consistent heading estimate, we move along to ESCV-IC. The goal with this scheme is to take the success of the RESCV-IC and expand it to also make corrections on the ownship position estimate. In this section, we adopt the simulation settings used in the previous RESCV-IC run, and investigate how the ESCV-IC fares in these conditions.

Expanding to also include positional error states, we are forced to include another ES covariance gain parameter K_p, which after some tuning was set as 13. This parameter, as with K_φ (used in RESCV-IC), artificially inflates the innovation covariance matrix $\mathbf{S}$, which is used to calculate the posterior localization covariance during an injection step.

This is dramatically large, and corresponds to scaling the positional elements of injection covariance by a whopping 169. At this point, we are essentially saying that "Yes, there has been a positional update, but you should not reduce its covariance at all." The need for such a large scaling is certainly an indication that the injections are not making the position estimates any more accurate, which does not bode well. The simulation configuration is shown in table 10.4, again with changes in **bold font**.

Category	Setting	Symbol	Value	Unit
Simulation	Number of simulations	N	30	
	Simulation time	T_{sim}	50	s
	Odometry included		no	
	Relative scans included		yes	
ES-JLAT	**Motion model**		**ESCV-IC**	
	Injection type		direct	
	Pre-associated scans		yes	
	Clutter rate	λ	0	
	Detection probability	P_D	1.0	
	Positional ES covariance gain	K_p	**13**	
	Heading ES covariance gain	K_φ	1	
	Injection on absolute update		no	
FF	Pre-associated scans		yes	

Table 10.4: Simulation configuration for 'ESCV-IC (using direct injection)'

10.4.1 Results: ESCV-IC (using direct injection)

The simulation results for this run are presented in figures 10.13, 10.14, 10.15, 10.16. For the FF, there are no changes between this run and the previous one, so its results are only included here for convenient comparison.

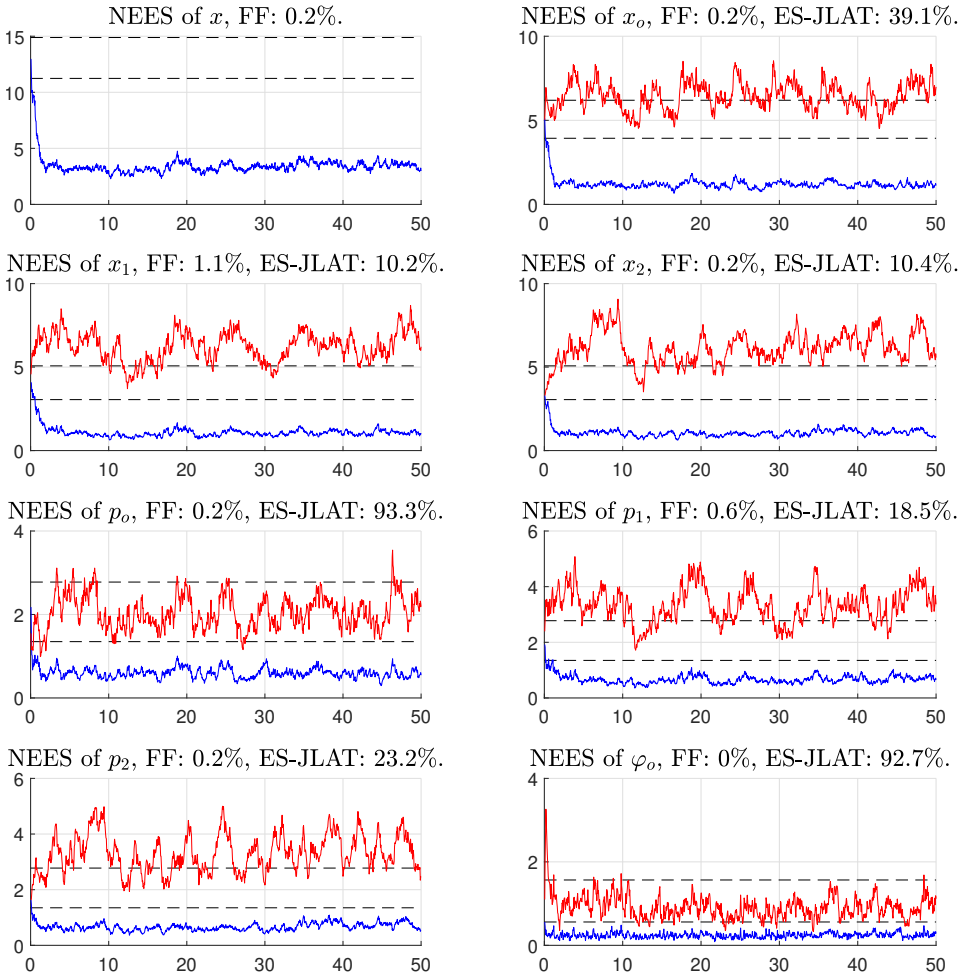

Figure 10.13: Ensemble averaged NEES values for 'ESCV-IC (using direct injection)'

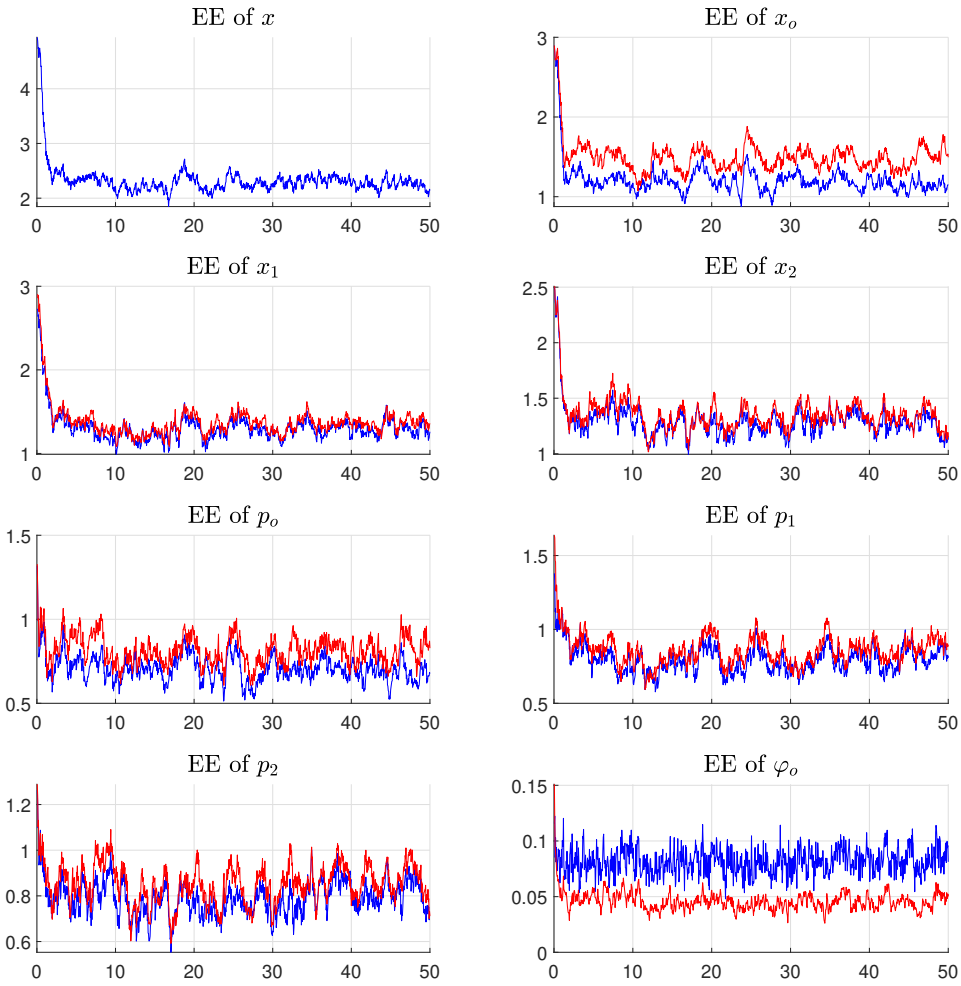

Figure 10.14: Ensemble averaged EE values for 'ESCV-IC (using direct injection)'

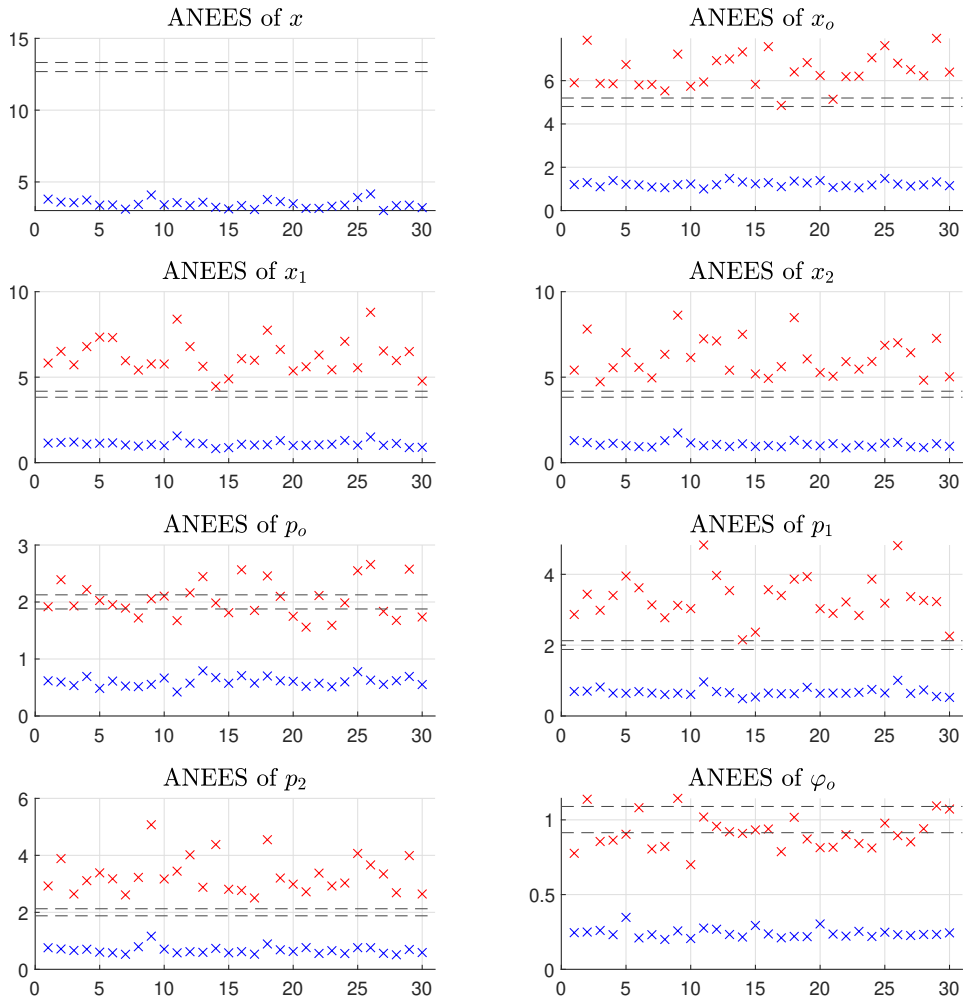

Figure 10.15: Per simulation ANEES values for 'ESCV-IC (using direct injection)'

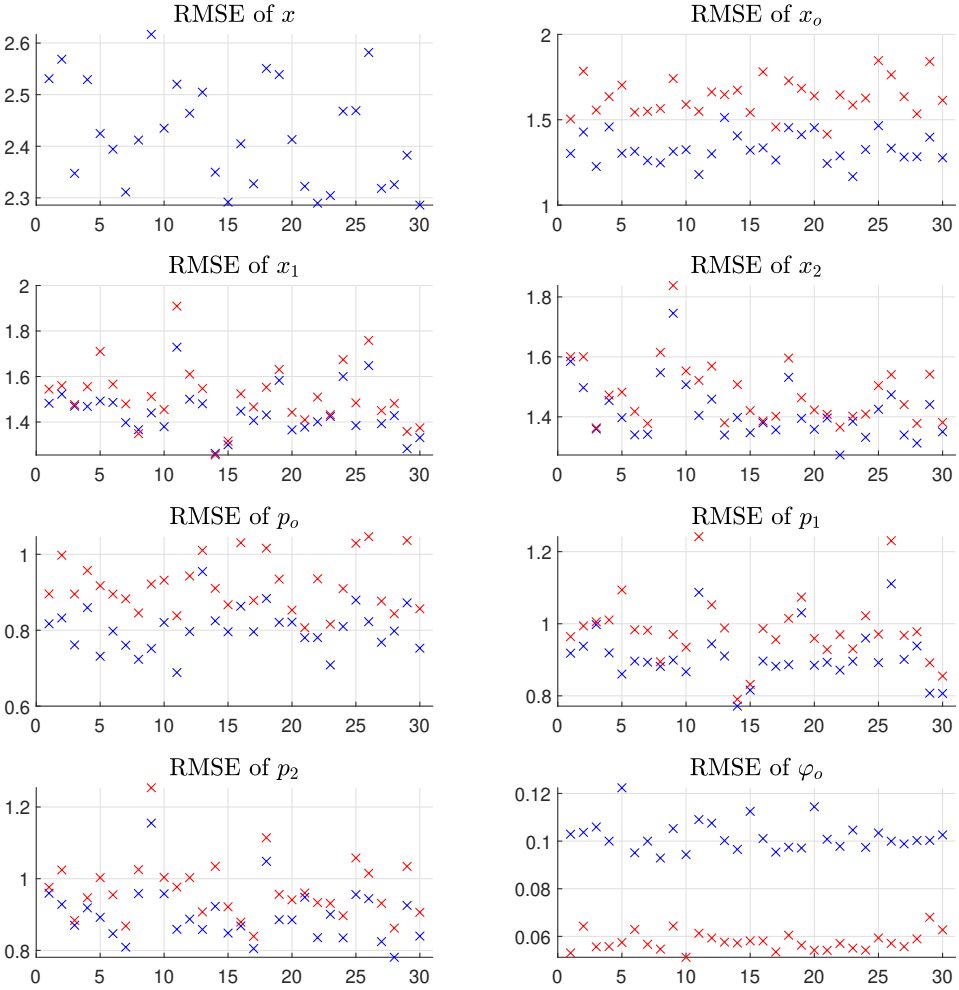

Figure 10.16: Per simulation RMSE values for 'ESCV-IC (using direct injection)'

Analysis: Estimation accuracy

For the estimation accuracy, we look at the EE and RMSE values shown in figures 10.14 and 10.16.

Starting with the ownship position p_o, we were hoping to see a significant improvement here, as this is one of the states which the ESCV makes injections on. This does not appear to be the case. While we do observe a subtle improvement of around 0.05 meters in the p_o RMSE values, there is still a significant gap between the the FF and ES-JLAT RMSE values. Similarly the other positional estimates p_1, p_2 also appear to have similar subtle improvements of between 0.05 and 0.1 meters. Most positional RMSE values are now around (or lower than) 1 meters, there are still some outlier RMSE values around 1.2 for target positions, but there appears to be fewer of them than in the RESCV runs.

Looking at the heading estimates, it is interesting to see how the inclusion of additional error states has affected its accuracy. We observe that the RMSE values of φ_o are now around 0.06 radians. This is slightly worse than the 0.055 radians achieved by the RESCV scheme (fig. 10.12), although still more than acceptable. Especially considering that it still has a comfortable margin to the FF performance, at an average RMSE of 0.1 radians.

Finally, for the full vessel states x_o, x_1, x_2, we do not observe much of a difference. The RMSE values of x_1, x_2 appear to be slightly closer to the FF values (than in the RESCV run, fig. 10.12), however this is probably caused by the subtle improvement in position estimates. In other words, there does not appear to be any improvement in target velocity estimation. Looking at the ownship state, x_o, its RMSE appears to actually have gotten slightly worse (as opposed to the RESCV run). Noting that the position estimate p_o became slightly more precise, and the heading estimate φ became slightly less precise, I would think that these should mostly cancel out. Since the degradation of the x_o RMSE is noticeable nonetheless, I suspect that the ownship velocity estimates have degraded slightly, resulting in these worse RMSE values.

This argument is a little thin, by virtue of trying to infer velocity error from the RMSE of x_o (which also contains position and heading errors). A very prudent point of future work would be to simply plot the velocity error separately, allowing for a definitive demonstration instead of the current speculative argument.

On the suspected worse ownship velocity estimates of ESCV:
I realise now that the direct injection of positional error in the localization filter (i.e. the ESCV injection step) might work against its purpose. The reason for this is that the ownship motion is described using a CV model, which defines a strong correlation between position and velocity. In fact, CV models actually need to be updated with (correct) positional errors in order to accurately infer velocity.

In a regular KF update, this correlation would be imposed via the gain matrix $\mathbf{W}$, which is calculated using the heavily correlated state covariance $\mathbf{P}_o$. Since we in the (direct) injection step instead just add on the estimated error directly, this does not get the chance to influence the velocity estimates. Intuitively, I would expect that a long-term result might be a localization estimate which very accurately tracks position, but ends up with only a fraction of the true velocity. Where the 'lagging' or 'missing' velocity is compensated for by perpetual positional injections.

This leads to another point of future work: namely to test different ways of by-passing this problem. One solution could be to replace the current direct injection with a traditional KF update step. I.e. act as if the error states are an innovation vector, with a corresponding innovation covariance, resulting in 'typical' equations for posterior state and covariance estimates. Exploring this option further led to the inclusion of the *weighted injection* scheme, as described in section 4.3.4, and tested in the next set of results.

Another possible solution is to instead include the velocities in the nuisance parameters of the motion model. Assuming that we use the 'inherited-covariance' scheme, i.e. steal the ownship covariance estimate to use as the prior nuisance parameter covariances, we would get these position-velocity correlations for free. This solution crossed my mind in the idea phase of this project, and is more similar to the SKF structure used in [4], however I opted against it, as I was concerned it would result in too much (computational) complexity. I also deemed there to be some prudence to **only** including nuisance parameters which are directly coupled to the relative measurements, again for the sake of reducing computational costs.

Analysis: Estimation consistency

Finally we arrive at consistency analysis for the ESCV-IC model. Looking at the NEES percentages for p_o and φ_o, (fig. 10.13), which are both around 93%, we have apparently managed to keep these estimates consistent. Sadly, the percentage of x_o is now much worse, at only 39%. I suspect this is caused by an estimation error in ownship velocity, which is not matched by similarly large covariance values. As discussed previously, I think this is a result of the ESCV-IC scheme actually worsening ownship velocity estimates, while also reducing their covariance.

There is one silver lining here though, and that is a slight improvement on target specific NEES percentages, i.e. x_1, x_2, p_1, p_2, which are around 10 to 20 percent. These esti-

mates are still overconfident, but less so than in similar RESCV runs (fig. 10.9). I think this matches the purpose of a SKF, namely that we can 'consider' the localization error slightly corrupting the incoming measurements, and thus not inject this error into tracking estimates.

10.5 ESCV-IC (using weighted injection)

In the first ESCV-IC run, we observed some problems with maintaining consistent ownship velocity estimates. Having diagnosed this as a consequence of purely positional injections (which undercuts velocity estimation), I opted to develop another injection scheme which more accurately accounts for learned correlations in the localization filter. This resulted in the *weighted injection* scheme, which is tested with the ESCV-IC in this run. The hope being that this injection type will achieve consistency for the ownship state x_o while at the same time maintaining the previously achieved accuracy and consistency for ownship position and heading.

The configuration for this run is shown in table 10.5, and as usual the relevant changes from the previous run are indicated with **bold font**.

Category	Setting	Symbol	Value	Unit
Simulation	Number of simulations	N	30	
	Simulation time	T_{sim}	50	s
	Odometry included		no	
	Relative scans included		yes	
ES-JLAT	Motion model		ESCV-IC	
	Injection type		**weighted**	
	Pre-associated scans		yes	
	Clutter rate	λ	0	
	Detection probability	P_D	1.0	
	Positional ES covariance gain	K_p	13	
	Heading ES covariance gain	K_φ	1	
	Injection on absolute update		no	
FF	Pre-associated scans		yes	

Table 10.5: Simulation configuration for 'ESCV-IC (using weighted injection)'

10.5.1 Results: ESCV-IC (using weighted injection)

The simulation results for this run are presented in figures 10.17, 10.18, 10.19, 10.20. For the FF, there are no changes between this run and the previous one, so its results are only included here for convenient comparison. The addition of this run was prompted by a

lacking ownship velocity estimate in the previous run, thus, I will limit analysis to mostly x_o performance in this section.

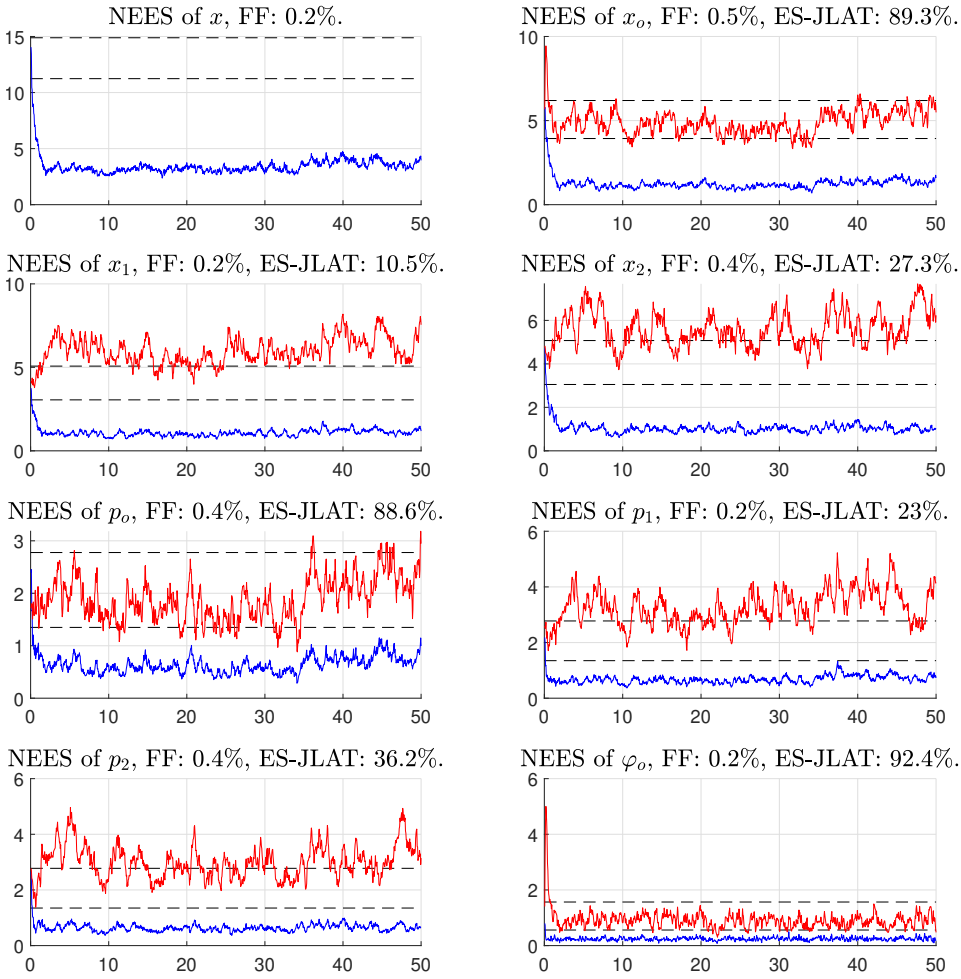

Figure 10.17: Ensemble averaged NEES values for 'ESCV-IC (using weighted injection)'

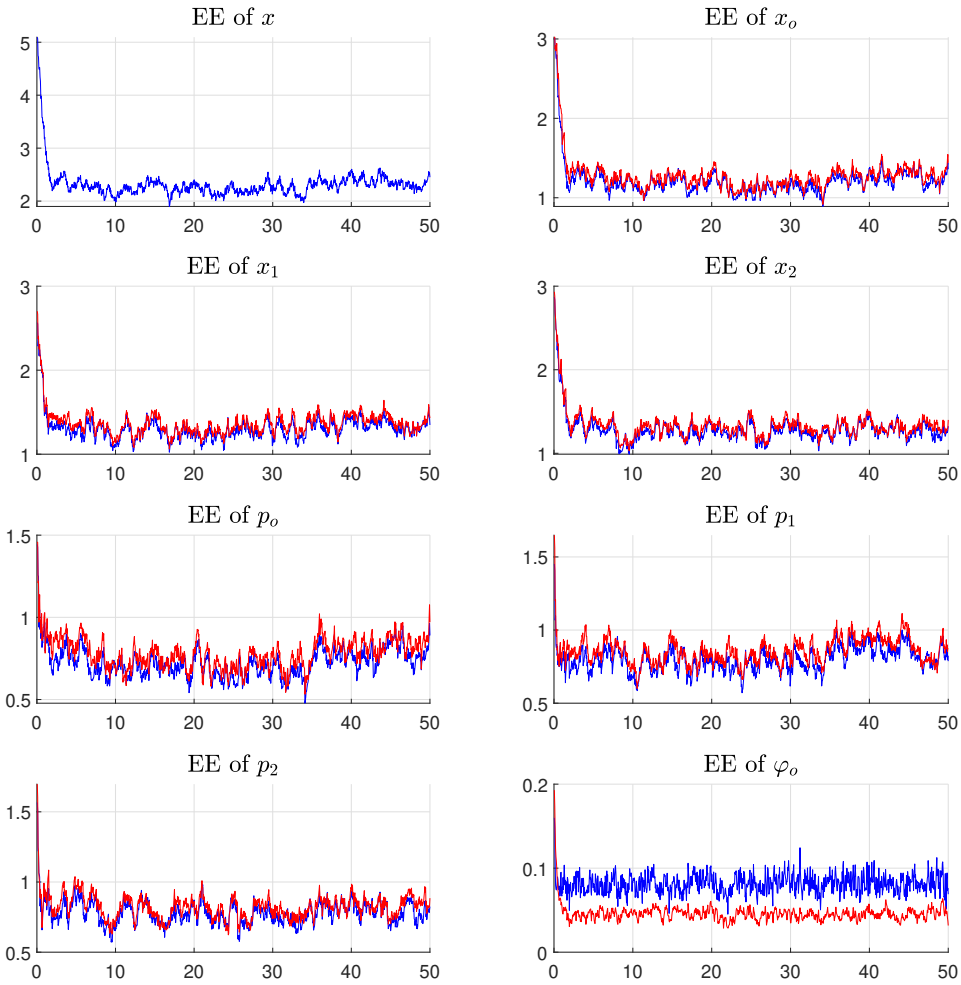

Figure 10.18: Ensemble averaged EE values for 'ESCV-IC (using weighted injection)'

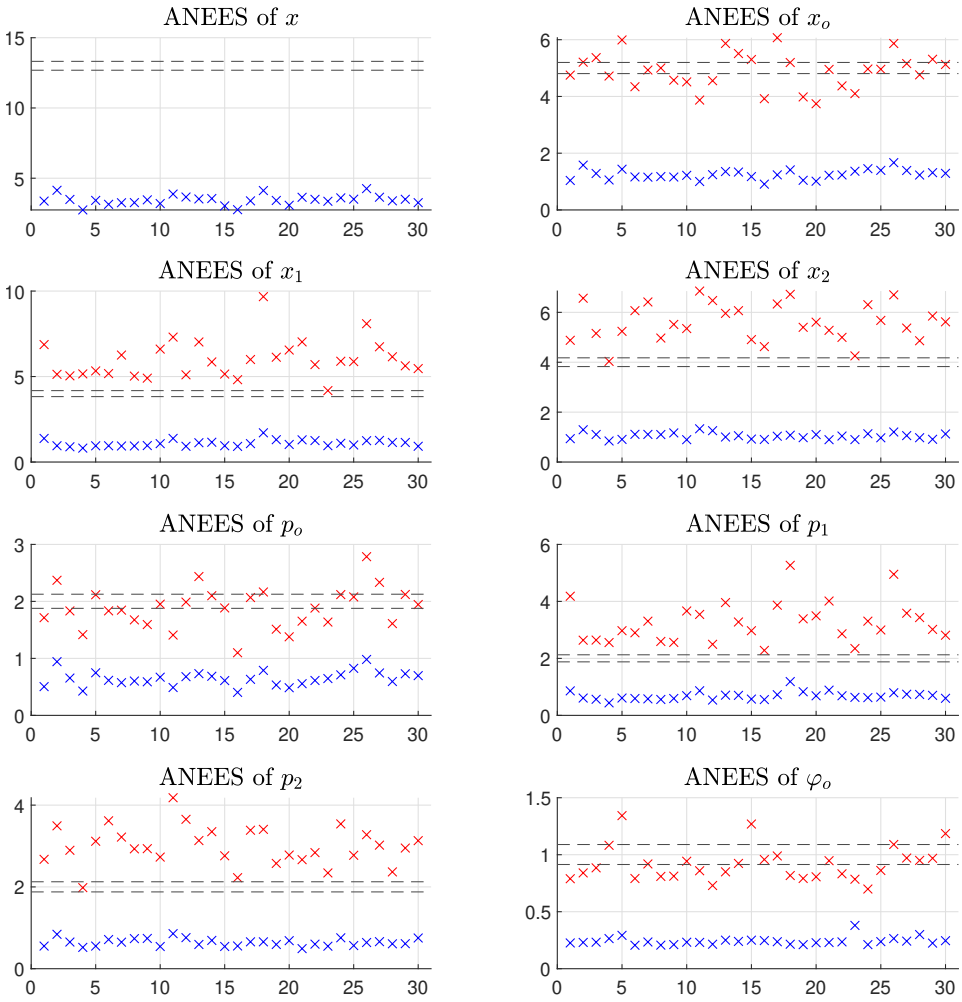

Figure 10.19: Per simulation ANEES values for 'ESCV-IC (using weighted injection)'

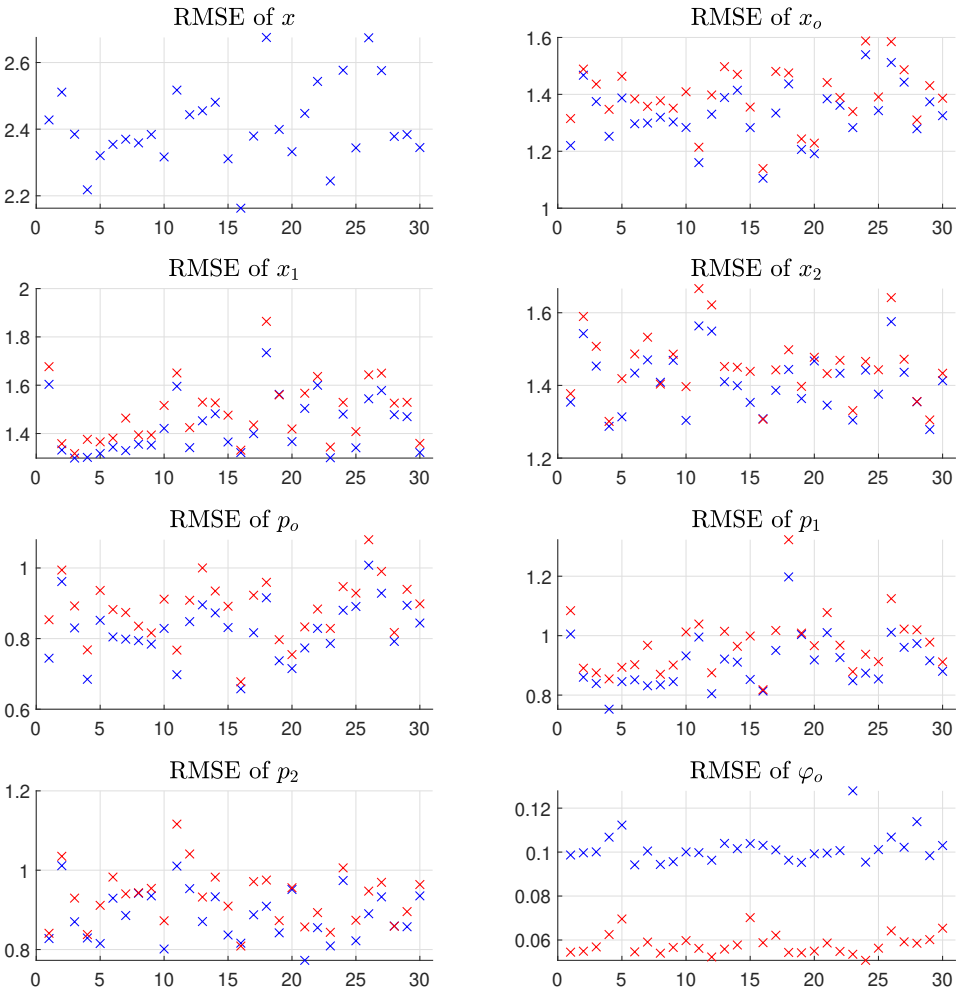

Figure 10.20: Per simulation RMSE values for 'ESCV-IC (using weighted injection)'

Analysis: Estimation accuracy

Comparing the difference of RMSE values between ES-JLAT and FF, for this run and the previous (fig. 10.20 vs. 10.16), we see a significant improvement for both p_o and x_o. Where we previously saw a difference between 0.2 and 0.5 for x_o, we now see a difference of between 0 and 0.1. Similarly, for the ownship position p_o, which previously had differences between 0.1 and 0.2, these differences are now all 0.1 or smaller. This means that the localization accuracy is now comparable to that of the target states x_1, x_2, p_1, p_2, *and also* comparable to that achieved with the FF.

By extension, it would appear that the switch to the *weighted injection* scheme gave the desired effect, namely that the system is now able to account for (and infer) the localization states which are not explicitly estimated as nuisance parameters (in this case the ownship velocity).

The fact that all substates $x_o, x_1, x_2, p_o, p_1, p_2, \varphi_o$ now have comparable accuracy in the ES-JLAT and FF varieties, constitutes an important achievement for ES-JLAT, namely that it is able to leverage the information present in the relative range-bearing scans to the same degree that the FF is. In other words, it achieves comparable accuracy to that of the FF, with the same information available, despite its more modular structure. This is one of the significant selling points for ES-JLAT, namely its scalability and support for arbitrary localization and tracking filters. (Assuming of course that they operate with Gaussian estimates, along the lines of a KF.)

It of course remains to be seen whether the ES-JLAT will have similar performance under more demanding circumstances (especially in terms of tracking), and how it will behave for different motion models, measurement models, localization filters, different numbers of targets, etc.. These simulations provide a very narrow insight into its achievable performance, but it remains possible that the algorithm (in its current form) will have lackluster performance for different circumstances. This remains the most crucial point of future work, namely to test it out for more generalized performance, as it is possible that the solutions suggested in this thesis may need changing for more generalized use-cases.

I am especially concerned for how it behaves in terms of post-injection localization covariance, considering the need for huge inflation of positional injection covariance. I deem it plausible that this behaviour will be a recurring theme for any ownship variables which already have decent estimates, which I think warrants finding a more elegant solution.

Analysis: Estimation consistency

In terms of estimation consistency, figures 10.17 and 10.19, we observe the expected improvement in the NEES percentage of x_o, which is now near consistency at 89%. The ownship position tells a similar story, at 88%.

There are are still improvements possible, especially when looking at the ANEES values of ES-JLAT, which indicate that while the ensemble averages are reasonably consistent, the per-simulation performance is still subject to a fair bit of variation (overconfident in some simulations, under-confident in others). This is somewhat disconcerting, as it can indicate that the covariance estimates are not dynamically adjusting according to the estimation accuracy, and instead just stable at constant values. By extension, the observed consistency could be just a happy coincidence of the estimation errors stabilizing at adequate values.

This reinforces the aforementioned conclusion that the current method for post-injection update of localization covariance has some obvious weaknesses. In my eyes, this stands as

the weakest part of this algorithm, and would need some serious tweaking (or a different solution entirely) before this algorithm stands a chance against more rigorous testing.

As in the previous run, we observe that the ES-JLAT is decidedly over-confident for all target estimates x_1, x_2, p_1, p_2. As usual, this has to be caused by disproportionately large estimation errors, or conversely disproportionately small covariance estimates, or some combination of the two. This illustrates the other fundamental inconsistency in the current ES-JLAT, namely a tendency for over-confidence in target states.

From a filtering standpoint, I think this is partially caused by the 'cancel out' problem which I have eluded to previously. Namely that if two (or more) targets end up with contradicting ownship error state estimates $\tilde{x}_t$, these estimates will (at least partially) cancel out in the final error state consensus $\tilde{x}$. At this point, either target has attributed a significant amount of the measurement error to the nuisance parameters, at the expense of their own estimates, and these turned out to be wrong (or at least disregarded). This constitutes a significant amount of measurement error which **should have been used** to update target estimates instead.

This information, i.e. the difference between $\tilde{x}_t$ and $\tilde{x}$, is readily available to us during an injection step, but so far not used for anything. I think this causes some inconsistency, which the target estimates frequently get 'the short end of the stick' of. If one were to *somehow* re-inject this information into the target estimates, as an injection innovation $\tilde{\nu}_t = \tilde{x}_t - \tilde{x}$, or something along these lines, I strongly suspect that the targets estimation error would more accurately match their covariance estimates, and perhaps even achieve consistency! While some preliminary experiments have been attempted to this end, I have yet to find some way of doing this which achieves both stability and consistency. A similar (perhaps safer) effort to this end could be to inflate the target covariance based on this innovation covariance.

10.6 Analysis: summary

A summary of the overall performance is included here. We have in general seen good heading estimation performance, both for RESCV-IC and ESCV-IC. This proves that ES-JLAT can serve as the lone guarantor of estimation for heading (at least in the two-target, fully associated case). For both these motion models, we did not need any extraordinary measures to maintain consistency in the heading estimates of the localization filter.

When introducing positional error states (i.e. when using ESCV-IC), some severe problems the localization covariance estimates were observed. Although an ad-hoc solution (via innovation covariance inflation) which achieved consistency was found, the need for such a solution is troubling. For this reason I deem the 'post injection localization covariance' solution to be fundamentally flawed in the current injection models (both direct and weighted).

On the topic of injection schemes, it was speculated that the ESCV-IC in combination with the 'direct injection' scheme gave lackluster ownship velocity estimates. This problem was seemingly solved by switching to the 'weighted injection' scheme, which I deem to be a better (and good enough) long-term solution for 'post injection localization mean'. Crucially, this scheme solves the problem of inferring 'unestimated' error states, assuming that the 'unestimated' localization states (velocity, in the case of ESCV-IC) are strongly correlated to those which are estimated. This stands as an interesting proof of concept for using only a subset of the localization state which is inherently observable in the relative measurements. By extension, this may constitute significant computational savings[4], **since the remaining (unestimated) states are now only inferred once for the consensus, and not in the update step of every tracking filter**.

Noting the aforementioned problems with localization covariance, I deem that the current form of ES-JLAT performs adequately (and is likely to be an improvement on existing systems) for the localization mean. When it comes to the localization covariance, it might be prudent to simply **not update the post-injection localization covariance**, given that all localization states are already observed by the localization filter. This may leave the localization filter in an underconfident state, but may be unnoticeable for small injections on already very precise esitmates. Even then, I deem it preferable with an under-confident filter than an overconfident one, as it does not make claims for precision which do not hold. This also matches with certain definitions of consistency, which only require that the covariance estimate is at least larger than the actual error, i.e. adequately confident or under-confident.

Finally, for target estimates, all varieties of ES-JLAT (all runs of RESCV-IC and ESCV-IC) were found to produce over-confident tracks. This is a significant problem, and unacceptable in terms of motion planning and obstacle avoidance. For the purposes of autonomy, any tracking estimates which are less precise than their covariance suggests will constitute a risk for: collisions, track loss, track merging, not to mention making data association very difficult. This of course also needs addressing, and I will outline a few options which I deem promising in the future work chapter (cha. 11).

All in all, despite these simulations giving a narrow, somewhat idealized view of the ES-JLAT performance, I think the achieved results are reasonable. If the problems related to consistency (for both localization and tracking) are solved in a meaningful way, I deem ES-JLAT to be an adequate backup in case of emergency (such as IMU/GNSS/compass failure) or indeed as an add-on to existing localization schemes, for the sake of improving localization estimates without hurting tracking performance too much.

[4] As opposed to using the full localization state for nuisance/error parameters, such as the SKF structures in [4].

Chapter 11

Future work

Being a first effort at designing a novel joint localization and tracking scheme, there are a number of challenges and problems which have been identified but not solved. This chapter presents what I deem to be the most important of these challenges and in some cases outlines for ideas on how to solve them. These are divided into two categories: first are intrinsic problems of ES-JLAT as it has been presented thus far, algorithmic weaknesses, etc.. Second are extrinsic aspects of this thesis, such as remaining work to prove ES-JLAT's worth as a full fledged localization and tracking system. In any case, I would deem it prudent to tackle the former first, as the intrinsic problems are bound to manifest themselves, especially in more rigorous (realistic) testing.

11.1 Intrinsic problems of ES-JLAT

For discussing the intrinsic problems of ES-JLAT, I will use its configuration in the final section of the analysis chapter as a baseline. i.e. ES-JLAT using the ESCV-IC model and the weighted injection scheme.

I am rather pleased with the way state injection works, i.e. it seems to be making reasonable updates to the localization state (i.e. the mean), also on the variables which are not estimated as error states (i.e. ownship velocity). So I do not think this warrants any immediate future work. With that being said, I deem the post-injection covariance update to be the decidedly the weakest point of this algorithm. The fact that positional covariances of the consensus error state needed to be inflated in order to maintain consistency (and even stability) tells me that the post-injection covariance equation does not do its job, and can (depending on the circumstances) cause the entire system to diverge. **The most cru-**

cial point of future work is to find some other way of calculating this post-injection localization covariance.

The coolest thing would of course be to compute an optimal, analytical covariance update based on the state injection, since that appears to produce sound results. However, as eluded to previously, I expect these expressions to become quite nightmarish, considering their dependence on all incoming measurements and prior target states. Nonetheless, I would deem it prudent to start by poking at the covariance equation $\text{Cov}(\hat{x}_o, \mathbf{W}_{\text{ideal}}\tilde{x})$ (as mentioned when trying to find this analytical update equation), where $\hat{x}_o$ is the pre-injection localization estimate, and $\mathbf{W}_{\text{ideal}}\tilde{x}$ is the injected consensus, as calculated with the 'weighted injection' scheme. Potentially just to search for terms of this expression which can be readily estimated or neglected.

On the topic of tracking performance, we found all varieties of ES-JLAT to provide over-confident target state estimates. **This stands as the second, crucial point of future work, namely to ensure that the compound filter structure suggested does not end up with disproportionately small covariance estimates for its target state.**

For this I have three immediate ideas:

1. Change to a SKF esque update equation for the post-reset target covariance (such as in [5]). For example by first running a regular KF style update to get some nonzero error state estimates, then afterwards running another update step according to the SKF equations. Considering that increased tracking consistency has been observed for SKF tracking filters with the localization error as nuisance parameters [4], I would think this to be achieveable here also.

2. Run an after-the-fact inflation of the target covariance estimate, according to how much its error state estimate differed from the final consensus (injection innovation). In this sense, we assume that a tracking filter probably produced an inaccurate target state estimate if it produced a weird error state estimate.

   This is very connected to the notion of an injection innovation, which I have eluded to previously. Seeing what can be achieved with this information is something I deem profoundly interesting, and would probably be my starting point if I were to revisit this project.

   This is also connected to the "spread of innovations" inflation which is added to the posterior covariance of PDA-style trackers [22], and could be a decent place to start. This effectively inflates the covariance of tracks which were updated with many contradicting measurements, as opposed to few resolute measurements.

3. Run an after-the-fact correction on the target state estimate, again based on how much its error state estimate differed from the final consensus (i.e. injection innovation). Leveraging the ambiguity demonstrated in figure 5.1, it would make sense to assume that the target has moved, if we can 'rule out' that the ownship has moved or rotated, by virtue of what the consensus has concluded.

This kind of argument is dangerous for very few targets though, since the consensus estimate will be relatively inaccurate, by extension the injection innovation must be similarly inaccurate. If we scale up to more targets though, both become more accurate, and I would deem it safer to use the innovation for retroactive state updates. (This is part of why point 2 in this list is included, namely that I deem it safer to inflate covariance estimates than it is to make retroactive state corrections.)

11.2 Extrinsic problems: testing and simulation

The simulation setup is in this thesis provides a very narrow insight into the behaviour and performance of ES-JLAT. In this sense, it would be prudent to do more testing with different localization and tracking objectives, such as different sensors being available, letting the number of targets vary, including odometry in the localization filter, etc.

Of these, the most crucial shortcoming is the lack of typical tracking challenges. The simulations done in this thesis have all been with fully associated, clutter-free sensor data, and are in this sense not representative for tracking performance at all (no data association needed). We have in fact **only demonstrated filtering performance in this thesis**. While the ES-JLAT from an abstraction standpoint is designed to support an arbitrary (single or multi target) tracking filter (such as IPDA or JIPDA), we have yet to really test how it behaves in such a configuration. **This stands as the third point of crucial further work, in my eyes. Namely to test ES-JLAT configured with JIPDA, in a set-up with sensor scans with challenging data association. E.g. due to clutter and closely grouped targets.** I would go so far as to assume that new intrinsic problems with ES-JLAT would come to light in such a test, which would be fascinating to see.

On a more generalized note, we have seen amazing estimation performance for the ownship heading, for which ES-JLAT is the lone guarantor of estimation in this set-up. On the other hand, the position and velocity estimates have been more challenging, and even requiring ad-hoc solutions (covariance inflation) to maintain consistency. I worry that this is a recurring theme, and that any localization variable which already has decent estimates (by virtue of being observed directly by the localization filter) will have these same problems. This of course bears more testing, and thus I conclude on a fourth point of future work: **Testing the ES-JLAT with a localization filter which already has decent estimates for heading, for example via compass/magnetometer, and finding out whether the ES-JLAT simply does not like providing corrections to already observed variables.** Such a test would naturally provide useful insight into the aforementioned 'post injection localization covariance' flaw of the ES-JLAT, as it might explain the root of the problem.

Similarly, test the ES-JLAT performance for an arbitrary number of targets, and seeing which of the aforementioned problems might be specific to the 2-target tests done in this thesis, as well as identifying any new problems that might present themselves. For ex-

ample, I suspect that increasing the number of targets will have a smoothing effect on the current consensus scheme (i.e. driving the estimate close to zero, and its covariance to very low values). This could be problematic for the localization performance, especially for any variables which are not observed directly in the localization filter (such as heading, in our simulations).

Chapter 12

Conclusion

As a summary of the thesis and the project overall, this chapter is included, and divided into two parts. The first part is a systematic review of the goals set out for this project, as outlined in the 'scope' section of the introduction. The second part is a retrospective of the project overall, including any particularly good or bad choices I made along the way.

12.1 Results of project tasks

In the introduction, the following six goals for this project were outlined:

1. Develop a JLAT scheme as a number of interacting (and independent) tracking and localization filters.
2. Develop or select an existing filter structure for these tracking filters which can both handle nominal state estimation, as well as localization error state estimation.
3. Develop a scheme for calculating a single 'consensus' error state estimate, based on a number of independent error state estimates provided by each tracking filter.
4. Develop a simulator to test the performance of the new JLAT scheme.
5. Select an existing (well established) JLAT solution with which to produce grounds for comparison.
6. Analyse the results of the new scheme, and pinpoint any shortcomings which need fixing.

The first of these points was completed partially. Although I am very pleased with the overall system abstraction, resulting in a very modular JLAT system (ES-JLAT), I deem there to be some work remaining with how the injection step is performed in the current solution. Specifically, the 'post injection localization covariance' seems to struggle with inconsistency when injecting into variables which already have precise estimates. I deem this to be a devastating flaw, and would not consider ES-JLAT to be an applicable solution until this is fixed.

With that being said, if ES-JLAT is applied in a setting where all localization estimates are already decently precise, one could conceivably get away with only updating the localization mean (and not touching its covariance). This would sidestep the problem entirely, and in these cases I would expect ES-JLAT to perform admirably.

Moving along to point 2, I would deem this to be mostly completed. I concluded on a compound filter structure for tracking which behaves as a compromise between an ESKF and SKF. I am happy with this filter structure on most counts, and it certainly suits the ES-JLAT. With that being said, we have observed some tendency for overconfidence on target state estimates, I suspect this is a consequence of how the **reset_error_covariance** function is defined currently. In the future work chapter, I outlined a few ideas on how to solve this problem. If one (or more) of these solutions were implemented and proved to be effective, I would deem point 2 to be fully completed.

For point 3, I would deem it to be fully completed. The 'covariance weighted consensus' as defined in section 8.1.3 appears to do its job perfectly. There is a slight catch here though, namely that this is still only tested for 2 targets. As mentioned in the future work chapter, I suspect that this consensus model may indeed provide lackluster estimates if there are lots of targets present, although this remains to be tested.

Next, I deem the simulator to be adequate for these tests. In other words, point 4 is completed in full. There is a measure of restraint associated with this claim though. I could of course have implemented a much more elaborate simulator, with clutter, an arbitrary number of targets, reduced (or even dynamic) target visibility, ownship odometry, etc., however, I think a decent compromise was achieved. Especially considering that I wanted to end up with a manageable level of complexity in the results, which would make it easier to troubleshoot and improve the solution during the project. To this end, I deem a more complex simulator to be outside of the scope for this project.

With that being said, flip side of this simplistic simulation is a very narrow view into the performance of ES-JLAT. Most of the conclusions drawn, both good and bad, are of course within the context of this narrow testing. It stands to reason, then, that a lot more testing is warranted before bolder claims can be made for the supremacy of ES-JLAT.

Next, I deem point 5 to be completed in full. Although the elected solution, the full state EKF, has some problems with under-confidence, its estimation accuracy seems to be stable and repeatable, and thus makes good grounds for comparison. The EKF is often used as

a baseline for performance in filtering literature, and thus a somewhat expected choice as well.

Finally, we arrive at point 6, the analysis. I deem this point to be completed, at least within the context of the developed simulator. Given access to more varied and elaborate simulations, or even just additional performance metrics for the current simulator, I could of course do a more detailed analysis. With that being said, I think the thesis is long enough, and I defer any more detailed analysis to the future work section.

On that topic, a very obvious shortcoming for point 6 is that no experimental data was used to test the solution. I counter this with the fact that experimental data is liable to introduce many uncertainties and challenges. In the prototyping stage of development, the complexity it introduces is simply not worth it. To that end, I leave experimental tests as a point of future work which can be done after the more fundamental issues (see intrinsic issues in the future work chapter) of ES-JLAT have been resolved.

12.2 Retrospective

As for the challenges underway, and the overall workflow, I have some regrets. The main one being that I spent a long time trying to get good results for the 'decoupled covariance' motion models (i.e. ESCV and RESCV). In retrospect, I wish I had looked at SKF theory earlier, and thus arrived at the 'inherited covariance' solution sooner.

Secondarily, I spent a decent amount of time implementing ownship odometry in the simulator (as well as a IMU-preintegration scheme). Despite this, I did not include IMU behaviour in the motion models in any meaningful way. The result of this being that these models ended up having terrible consistency, thus I finally opted to omit any odometry results from the thesis. In retrospect, I would have preferred to not have spent time on odometry at all, thus leaving more time to solve the other (more pressing) problems of ES-JLAT.

Third, I wish I had looked at Brink's intermittent SKF (ISKF) and fixed partial SKF (FP-SKF) a lot sooner. I deem both of these to be very promising for this application, which potentially could replace the 'compound filter' which is suggested in this thesis. Were I to do this over, I would probably start with one of these as tracking filters, and after the fact see what changes would be necessary to fit ES-JLAT (or potentially even design ES-JLAT from the ground up to use FPSKF for its tracking filters).

With that being said, it has been an overall satisfying project to work on. I am particularly pleased with the process, i.e. the workflow of testing a solution, discovering its problems, and finding meaningful solutions for them. Although there is certainly a lot of work left before ES-JLAT is applicable for real-world applications, I am still happy with the progress made.

Bibliography

[1] M. Sirnes, "Joint localization and tracking: experiments and preliminary work on active sensors," 2020.

[2] E. F. Brekke and E. Wilthil, "Suboptimal kalman filters for target tracking with navigation uncertainty in one dimension," in *Proceedings of IEEE Aerospace Conference*, Big Sky, MT, USA, Mar. 2017.

[3] S. Julier and A. Gning, "Bernoulli filtering on a moving platform," in *Proceedings of Fusion 2015*, Washington, D.C., USA, July 2015, pp. 1511–1518.

[4] E. F. Wilthil and E. F. Brekke, "Compensation of navigation uncertainty for target tracking on a moving platform," in *Proceedings of Fusion 2016*, Heidelberg, Germany, Jul. 2016.

[5] R. Novoselov, S. Herman, S. Gadaleta, and A. Poore, "Mitigating the effects of residual biases with schmidt-kalman filtering," in *2005 7th International Conference on Information Fusion*, vol. 1, 2005, pp. 8 pp.–.

[6] C. Yang, E. Blasch, and P. Douville, "Design of schmidt-kalman filter for target tracking with navigation errors," in *2010 IEEE Aerospace Conference*, 2010, pp. 1–12.

[7] K. M. Brink, "Partial-update schmidt–kalman filter," *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 9, pp. 2214–2228, 2017.

[8] S. Roumeliotis, G. Sukhatme, and G. Bekey, "Circumventing dynamic modeling: evaluation of the error-state kalman filter applied to mobile robot localization," in *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, vol. 2, 1999, pp. 1656–1663 vol.2.

[9] J. Sola, "Quaternion kinematics for the error-state kf," *Laboratoire d'Analyse et d'Architecture des Systemes-Centre national de la recherche scientifique (LAAS-CNRS), Toulouse, France, Tech. Rep*, 2012.

[10] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert, "iSAM2: Incremental smoothing and mapping using the Bayes tree," *The International Journal of Robotics Research*, vol. 31, no. 2, pp. 216–235, 2012.

[11] J. Mullane, B.-N. Vo, M. D. Adams, and B.-T. Vo, "A random-finite-set approach to Bayesian SLAM," *IEEE transactions on robotics*, vol. 27, no. 2, pp. 268–282, 2011.

[12] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: An open-source slam system for monocular, stereo, and rgb-d cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[13] B. Bescos, J. M. FÃ¡cil, J. Civera, and J. Neira, "DynaSLAM: Tracking, mapping, and inpainting in dynamic scenes," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 4076–4083, Oct 2018.

[14] D. Moratuwage, B.-N. Vo, and D. Wang, "Collaborative multi-vehicle SLAM with moving object tracking," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, 2013, pp. 5702–5708.

[15] A. K. Nasir, "Cooperive simultaneous localization and mapping framework," Ph.D. dissertation, Universitat Siegen, 2014.

[16] E. Brekke, B. Kalyan, and M. Chitre, "A novel formulation of the Bayes recursion for single-cluster filtering," in *Proceedings of IEEE Aerospace Conference*, Big Sky, MT, USA, Mar. 2014.

[17] C. S. Lee, D. E. Clark, and J. Salvi, "SLAM with dynamic targets via single-cluster PHD filtering," *IEEE Journal of Selected Topics in Signal Processing*, vol. 7, no. 3, pp. 543 – 552, Jun. 2013.

[18] D. Musicki, R. Evans, and S. Stankovic, "Integrated probabilistic data association," *IEEE Transactions on automatic control*, vol. 39, no. 6, pp. 1237–1241, 1994.

[19] D. Musicki and R. Evans, "Joint integrated probabilistic data association: Jipda," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 40, no. 3, pp. 1093–1099, 2004.

[20] Y. Bar-Shalom and E. Tse, "Tracking in a cluttered environment with probabilistic data association," *Automatica*, vol. 11, no. 5, pp. 451 – 460, 1975. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0005109875900217

[21] T. Fortmann, Y. Bar-Shalom, and M. Scheffe, "Sonar tracking of multiple targets using joint probabilistic data association," *IEEE Journal of Oceanic Engineering*, vol. 8, no. 3, pp. 173–184, 1983.

[22] Y. Bar-Shalom, F. Daum, and J. Huang, "The probabilistic data association filter," *IEEE Control Systems Magazine*, vol. 29, no. 6, pp. 82–100, 2009.

[23] G. Welch, G. Bishop *et al.*, "An introduction to the kalman filter," 1995.

[24] G. P. Huang, A. I. Mourikis, and S. I. Roumeliotis, “Observability-based rules for designing consistent EKF SLAM estimators,” *The International Journal of Robotics Research*, vol. 29, no. 5, pp. 502–528, 2010.

[25] S. Thrun, Y. Liu, D. Koller, A. Y. Ng, Z. Ghahramani, and H. Durrant-Whyte, “Simultaneous localization and mapping with sparse extended information filters,” *The International Journal of Robotics Research*, vol. 23, no. 7-8, pp. 693–716, 2004. [Online]. Available: https://doi.org/10.1177/0278364904045479

[26] C.-Y. Chong, S. Mori, W. Barker, and K.-C. Chang, “Architectures and algorithms for track association and fusion,” *IEEE Aerospace and Electronic Systems Magazine*, vol. 15, no. 1, pp. 5–13, 2000.

Magne Sirnes

Joint localization and tracking: a per-track error state approach

NTNU
Norwegian University of
Science and Technology