Degree Project in electrical engineering
Second cycle, 30 credits

Robust Non-Linear State Estimation for Underwater Acoustic Localization
Expanding on Gaussian Mixture Methods

DIOGO ANTUNES
Robust Non-Linear State Estimation for Underwater Acoustic Localization

Expanding on Gaussian Mixture Methods

DIOGO ANTUNES
Abstract

Robust state estimation solutions must deal with faulty measurements, called outliers, and unknown data associations, which lead to multiple feasible hypotheses. Take, for instance, the scenario of tracking two indistinguishable targets based on position measurements, where each measurement could refer to either of the targets or even be a faulty reading.

Common estimation methods model the state as having a unimodal distribution, so they are called unimodal methods. Likewise, multimodal methods model the state as a multimodal distribution. Difficult problems, such as autonomous underwater vehicle (AUV) navigation relying on acoustic localization, frequently involve recurring outliers. In these situations, the correct hypothesis only emerges as the most likely one when a substantial number of measurements are considered. Robust solutions for these problems need to consider multiple hypotheses simultaneously, which, in turn, calls for the representation of multimodal distributions.

In this work, a novel approximate inference method is presented, called the Gaussian mixture sum-product algorithm (GM-SPA), as it implements the sum-product algorithm (SPA) for Gaussian mixtures. The GM-SPA can exactly represent under-constrained linear measurements and approximate important non-linear models, such as range measurements and 2D pose kinematics.

The outlier robustness of the GM-SPA is tested and compared against the particle filter (PF) and multimodal incremental smoothing and mapping (MM-iSAM), both of which are non-parametric methods. Robustness, accuracy, and run-time are improved in simulation tests. The test problems include 1D localization with unknown data association, 3D linear target tracking with correlated outliers, and 2D range-only pose estimation with Gaussian mixture noise.

Keywords

Robust state estimation, Underwater localization, Target tracking, Gaussian mixture, AUV
ii | Abstract
Sammanfattning

Robusta lösningar för tillståndsuppskattning måste kunna hantera felaktiga mätningar, så kallade outliers, och okända dataassociationer, vilket leder till flera möjliga hypoteser. Ta till exempel scenariot att spåra två likadana mål baserat på positionsmätningar, där varje mätning kan tillhöra något av målen eller till och med vara en felaktig avläsning.


I detta arbete presenteras en ny approximativ inferensmetod, kallad Gaussian mixture sum-product algorithm (GM-SPA), eftersom den implementerar sum-product algorithm (SPA) för gaussiska blandningar. GM-SPA kan representera underbegränsade linjära mätningar exakt och approximera viktiga icke-linjära modeller, till exempel avståndsmätningar eller 2D-posekinematik.

GM-SPA:s robusthet mot outliers testas och jämförs med partikelfiltret (PF) och multimodal incremental smoothing and mapping (MM-iSAM), som båda är icke-parametriska metoder. Robusthet, noggrannhet och körtid förbättras i simulerings tester. Simulerade tester inkluderar 1D-lokalisation med okänd dataassociation, 3D linjär målföljning med korrelerade outliers och 2D-ställningsuppskattning av endast räckvidd med Gaussiskt blandningsljud.

Nyckelord

Robust tillståndsuppskattning, Undervattenslokalisering, Målsöka, Gaussisk blandning, AUV
Resumo

Soluções robustas para estimação de estado devem lidar com medidas defeituosas, chamadas de outliers, e com associações de dados desconhecidas, que levam a múltiplas hipóteses possíveis. Considere-se, por exemplo, o cenário de rastreamento de dois alvos indistinguíveis com base em medidas de posição, em que cada medida pode-se referir a qualquer um dos alvos ou até mesmo ser uma leitura defeituosa.

Métodos de estimação comuns modelam o estado como tendo uma distribuição unimodal, sendo assim chamados de métodos unimodais. Da mesma forma, métodos multimodais modelam o estado como uma distribuição multimodal. Problemas difíceis, como a navegação de veículos subaquáticos autónomos (AUVs) baseada em localização acústica, frequentemente envolvem outliers recorrentes. Nestas situações, a hipótese correta apenas surge como a mais provável quando um número substancial de medidas é considerado. Soluções robustas para estes problemas precisam de considerar múltiplas hipóteses simultaneamente, o que, por sua vez, exige a representação de distribuições multimodais.

Neste trabalho, é apresentado um novo método de inferência aproximada, chamado Gaussian mixture sum-product algorithm (GM-SPA), pois implementa o sum-product algorithm (SPA) para misturas Gaussianas. O GM-SPA pode representar exatamente medidas lineares sub-determinadas e aproximar modelos não lineares importantes, como medidas de distância e cinemática de pose 2D.

A robustez a outliers do GM-SPA é testada e comparada com o filtro de partículas (PF) e com multimodal incremental smoothing and mapping (MM-iSAM), ambos métodos não-paramétricos. A robustez, a exatidão e o tempo de execução em testes de simulação são melhorados. Os problemas de teste incluem localização 1D com associação de dados desconhecida, rastreamento linear de alvos em 3D com outliers correlacionados e estimação de pose 2D com base em medidas de distância com ruído de mistura Gaussiana.

Palavras-chave

Estimação robusta de estado, Localização subaquática, Rastreamento de alvos, Mistura Gaussiana, AUV
Acknowledgments

This thesis would not have been possible without the support and guidance of my dedicated supervisors and examiner: Aldo, Alexandre, and John. Their invaluable mentorship was essential in keeping my research on the right track.

I am grateful for their extensive experience and generous willingness to engage in fruitful discussions, which were a constant source of insight, especially during moments of research challenges. I would also like to express my sincere appreciation for the pleasant and encouraging attitudes that made the journey even more enjoyable and productive, creating an environment in which I felt supported and motivated at every step.

I want to further thank Aldo and Alexandre for their meticulous help with writing this thesis, and Aldo for initially suggesting this area of research.

Thanks should also go to the researchers and students working with the Swedish Maritime Robotics Centre who listened and commented on my research progress presentations.

Special thanks go to Gustav Bernpaintner for their contributions to the Swedish translations.

I would be remiss in not acknowledging the support of my friends, both old and new, for their emotional support and cherished companionship throughout this journey.

Finally, I want to thank my family for their love and unwavering support, and for having inspired, encouraged, and enabled my passion for studying throughout my life.

Stockholm, December 2023
Diogo Antunes
I can live with doubt and uncertainty and not knowing. I think it’s much more interesting to live not knowing than to have answers which might be wrong.

Richard Feynman
Epigraph
# Contents

1 Introduction ............................................. 1
   1.1 Motivating context .................................. 2
      1.1.1 Dead reckoning and inertial navigation ..... 2
      1.1.2 Acoustic localization ......................... 3
   1.2 Problem ............................................. 4
      1.2.1 Scientific and engineering issues .............. 4
   1.3 Purpose ............................................. 5
   1.4 Goal ............................................... 6
   1.5 Delimitations ....................................... 6
   1.6 Structure of the thesis ............................. 6

2 Background ............................................. 9
   2.1 Probability for autonomous robots ................. 9
      2.1.1 Data fusion .................................. 9
      2.1.2 Modeling outliers ............................. 10
      2.1.3 The curse of dimensionality .................. 11
   2.2 Representing distributions ......................... 12
   2.3 Factor graphs ..................................... 14
      2.3.1 The sum-product algorithm ..................... 14
      2.3.2 Cyclic factor graphs ......................... 16
   2.4 Practical inference methods ....................... 17
      2.4.1 Filtering and prediction ....................... 17
      2.4.2 Smoothing .................................... 19
      2.4.3 Inference on general factor graphs ............ 21
      2.4.4 Summary .................................... 22

3 The Gaussian mixture sum-product algorithm .......... 25
   3.1 Improper Gaussian .................................. 26
      3.1.1 Reduced dimension Gaussian .................. 27
### Contents

3.1.2 Product of improper Gaussians ........................................ 29
3.1.3 Marginal of an improper Gaussian .................................... 30
3.2 Improper Gaussian mixtures ............................................. 33
  3.2.1 Simplification methods ............................................... 33

4 Simulation tests ............................................................... 39
  4.1 Tested methods ........................................................... 40
  4.2 Performance metrics ..................................................... 41
  4.3 The 3 doors example ...................................................... 43
    4.3.1 Problem description ................................................. 43
    4.3.2 Methods description ............................................... 45
    4.3.3 Exact solution ...................................................... 46
    4.3.4 Approximate beliefs ................................................ 48
    4.3.5 Position errors ..................................................... 50
    4.3.6 Data association accuracy ........................................ 51
    4.3.7 Run-time ............................................................ 52
  4.4 Target tracking with correlated reflections .......................... 53
    4.4.1 Problem description ................................................. 53
    4.4.2 Methods description ............................................... 55
    4.4.3 Estimated trajectories .............................................. 58
  4.5 2D range-only pose estimation ....................................... 61
    4.5.1 Problem description ................................................. 62
    4.5.2 Methods description ............................................... 64
    4.5.3 Example solution ................................................... 69
    4.5.4 Statistical results .................................................. 70
  4.6 Results summary .......................................................... 72
    4.6.1 The 3 doors example ............................................... 72
    4.6.2 Target tracking with correlated reflections .................... 73
    4.6.3 2D range-only pose estimation ................................... 74

5 Conclusions and future work .............................................. 77
  5.1 Conclusions ............................................................... 77
  5.2 Limitations and future work .......................................... 78
    5.2.1 Cyclic factor graphs ............................................... 78
    5.2.2 Approximation of arbitrary likelihoods .......................... 78
    5.2.3 Considering manifold-valued variables .......................... 79
  5.3 Reflection on societal impact ........................................... 79

References ................................................................. 81
## Contents

### A Factor graphs

- **A.1 Introduction** .......................................................... 89
- **A.2 The sum-product algorithm** ..................................... 90
  - **A.2.1 The sum-marginal operator** ................................. 90
  - **A.2.2 Comparison with max-marginals** ......................... 92
  - **A.2.3 The update rule** ............................................... 93
  - **A.2.4 Derivation** ................................................... 94

### B Column pivoted QR factorizations

- **B.1 Introduction** .......................................................... 99
- **B.2 QRCP with numerical computations** ........................ 100
- **B.3 Application of the QRCP** ....................................... 101
<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AHRS</td>
<td>attitude-heading reference system</td>
</tr>
<tr>
<td>AUV</td>
<td>autonomous underwater vehicle</td>
</tr>
<tr>
<td>DVL</td>
<td>Doppler velocity log</td>
</tr>
<tr>
<td>EKF</td>
<td>extended Kalman filter</td>
</tr>
<tr>
<td>FBS</td>
<td>forward-backward smoothing</td>
</tr>
<tr>
<td>GM-SPA</td>
<td>Gaussian mixture sum-product algorithm</td>
</tr>
<tr>
<td>GPS</td>
<td>global positioning system</td>
</tr>
<tr>
<td>HMM</td>
<td>hidden Markov model</td>
</tr>
<tr>
<td>HRMSE</td>
<td>heading root mean square error</td>
</tr>
<tr>
<td>IG</td>
<td>improper Gaussian</td>
</tr>
<tr>
<td>iSAM2</td>
<td>incremental smoothing and mapping</td>
</tr>
<tr>
<td>KF</td>
<td>Kalman filter</td>
</tr>
<tr>
<td>LBL</td>
<td>long baseline</td>
</tr>
<tr>
<td>MM-iSAM</td>
<td>multimodal incremental smoothing and mapping</td>
</tr>
<tr>
<td>MM-TFS</td>
<td>multimodal two-filter smoothing</td>
</tr>
<tr>
<td>MMLV-TFS</td>
<td>multimodal two-filter smoothing with low variance sampling</td>
</tr>
<tr>
<td>NH</td>
<td>null hypothesis</td>
</tr>
<tr>
<td>PF</td>
<td>particle filter</td>
</tr>
<tr>
<td>PRMSE</td>
<td>position root mean square error</td>
</tr>
<tr>
<td>QRCP</td>
<td>column pivoted QR</td>
</tr>
<tr>
<td>RDG</td>
<td>reduced dimension Gaussian</td>
</tr>
<tr>
<td>SBL</td>
<td>short baseline</td>
</tr>
<tr>
<td>Acronym</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>SLAM</td>
<td>simultaneous localization and mapping</td>
</tr>
<tr>
<td>SMaRC</td>
<td>Swedish Maritime Robotics Centre</td>
</tr>
<tr>
<td>SPA</td>
<td>sum-product algorithm</td>
</tr>
<tr>
<td>TFS</td>
<td>two-filter smoothing</td>
</tr>
<tr>
<td>UKF</td>
<td>unscented Kalman filter</td>
</tr>
<tr>
<td>USBL</td>
<td>ultra-short baseline</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

An autonomous robot will typically have to model the environment it exists in to appropriately navigate and act upon it. Simultaneous localization and mapping (SLAM) and target tracking are two examples of this. In complex environments, sensor measurements are usually affected by noise. For example, a global positioning system (GPS) measurement might specify a position with an uncertainty on the order of 10 m [1].

Information from multiple measurements, and multiple sensors, can be fused to obtain more precise estimates. Probabilistic methods maintain a belief, which is a probability density that represents the distribution of the state given the measurements, updating it over time with each new measurement.

Probabilistic methods are classified as unimodal or multimodal according to whether they model the state as a unimodal or multimodal distribution. Unimodal methods are usually simpler and more computationally efficient. The term multimodal also has a different meaning, not used in this work, that refers to the use of multiple sensing or operating modalities. In this thesis, multimodal refers to the property of distributions having probability density functions with multiple distinct peaks/modes.

To model uncertainty about the state in the presence of faulty measurements, called outliers, and uncertain data association, multimodal state distributions can be useful. This is because they can represent multiple hypotheses for how the measurements are created. One example is the hypothesis that the measurement is entirely independent of the state it is intended to measure. This hypothesis is named the null hypothesis (NH) in this work.

If a precise enough estimate of the state is available, the hypothesis that is the most consistent with the current belief can be chosen and the rest discarded.
This approach is frequently used for outlier rejection and data association decisions because it allows the use of fast unimodal methods, but it has two problems.

Firstly, a precise enough belief is not always available. Secondly, it classifies each measurement based only on the previous information, and it does not correct previous decisions based on new information.

To avoid these problems, multimodal inference methods can be used to consider all the hypotheses simultaneously. Mixture models can be used for this goal, weighing each hypothesis with a prior probability.

This work introduces a novel application of Gaussian mixtures, called the Gaussian mixture sum-product algorithm (GM-SPA). A particularly difficult application that motivates this study is autonomous underwater vehicle (AUV) localization, which is discussed in the next section.

1.1 Motivating context

Underwater localization techniques usually rely on dead reckoning and acoustic measurements. Electromagnetic waves suffer from high attenuation underwater, so GPS signals are only available at the surface and optical visibility is limited in range. Optical homing with dedicated light sources has been demonstrated with a range of tens of meters, while acoustic measurements can be obtained at ranges of thousands of meters.

1.1.1 Dead reckoning and inertial navigation

Dead reckoning refers to the process of estimating position by updating previous position knowledge with the time integral of velocity estimates. If the seafloor is close enough for acoustic echoes to be detectable, velocity estimates in the reference frame of the AUV, called the body frame, can be obtained directly with a Doppler velocity log (DVL). They can also be obtained indirectly by integrating acceleration data from accelerometers.

A DVL works by sending multiple acoustic beams downwards in different directions and measuring the frequencies of the corresponding echoes from the seafloor. The Doppler effect allows inferring the velocity in each direction of propagation. Depending on the frequency of the DVL, the maximum working distance to the seabed ranges from tens to hundreds of meters.

Inertial navigation involves estimating orientation through the integration of rotation rate measurements from gyroscopes and determining alterations in position through double integration of acceleration data. An attitude-
heading reference system (AHRS) combines gyroscopes, accelerometers, and a heading sensor, such as a magnetometer, to estimate orientation. The earth’s gravitational field can be used as a reference for the down direction, providing a way to measure the pitch and roll angles of orientation. With orientation estimates, it becomes possible to integrate body frame referenced velocity data once and acceleration data twice into position estimates in the world frame.

Due to integration, any biases in the acceleration or velocity measurements, no matter how small, lead to position errors that grow without bounds. This justifies the need for absolute localization methods for long-term navigation.

### 1.1.2 Acoustic localization

Acoustic localization typically measures the range and/or bearing of a beacon or set of beacons (such as a moving ship or a buoy array) relative to the vehicle via acoustic pings. Typical arrangements include long baseline (LBL), short baseline (SBL), and ultra-short baseline (USBL). Baseline stands for the distance between beacons.

LBL and SBL use the time of arrival of acoustic pings to estimate the range from the AUV to multiple beacons, which place restrictions on its possible position. This method is more accurate if the vehicle is inside the area delimited by the array of beacons and at a distance that is not much bigger than the baselines. Both of these conditions apply in LBL, making LBL the most precise system of the three. LBL typically relies on sea-floor mounted beacons. In SBL, the array size is considerably shorter than the vehicle’s depth, decreasing precision. An example is when the beacons are mounted at the corners of a ship.

With USBL, a single transducer beacon is used, which consists of multiple receivers and a single transmitter. A bearing estimate can be obtained from the multiple receivers based on the difference in the phase of the received signals [8]. The range is determined from the time of arrival, as in LBL and SBL.

All mentioned acoustic localization methods require knowledge of the speed of sound to estimate range, which varies with temperature, pressure, and salinity. This variation leads to the refraction of the acoustic waves.

A layer of water with a sharp change of some physical property, such as temperature, is called a cline. A halocline, for example, is a layer of water with rapidly changing salinity. Clines can act as a barrier for sound waves and create reflections. The sea floor and sea surface also create reflections. An example of this is the ENDURANCE AUV’s 2008 exploration of the ice-
covered Lake Bonney [9], where USBL pings were reflected above on an ice sheet and below on a halocline.

Together, these factors lead to multipath propagation, which creates measurement distributions with multiple peaks, depending on the surrounding water environment. Figure 1.1 represents a USBL scenario, where the acoustic signal can take two different paths to the AUV, resulting in two different possible readings.

![Figure 1.1: Example of direct and reflected paths that an acoustic signal can take, from an underwater beacon to an AUV, in the context of USBL localization.](image)

1.2 Problem

When multiple measurement generation hypotheses are necessary, due to outliers or unknown data association, and before a precise enough state estimate is available to identify which hypothesis is correct, the multiple hypotheses must be considered simultaneously.

Acoustic reflections in AUV localization are an example of this problem, which is well-documented in [9]. The presence of clutter measurements in target tracking is another example.

The question that this thesis aims to answer is whether mixture models can be used with multimodal inference methods to obtain robust state estimation.

1.2.1 Scientific and engineering issues

Representing outliers with an uninformative component in a mixture model presents two main computational challenges.
Firstly, the number of total hypotheses increases exponentially with the number of measurements. For example, if $N$ measurements are mixture models with two components each, then there are $2^N$ hypotheses in total. Simultaneously considering all of them is only possible for small problems. Section 4.3 considers such a problem that can be solved exactly.

Secondly, robust measurement models are computationally expensive to represent with most methods. To be robust, they need to consider the possibility that the measurement does not accurately constrain the possible values of the state being measured. Therefore, they must consider many possible values for the state.

Multimodal methods are typically non-parametric methods that work directly with samples of the relevant distributions, like the particle filter (PF) [10] and multimodal incremental smoothing and mapping (MM-iSAM) [11]. The problem is that accurately representing many possible state values with this strategy requires many samples to cover all the possibilities. This can make robust measurement models prohibitively expensive to represent with sample-based methods. Section 4.4 describes a practical example of this problem in more detail.

### 1.3 Purpose

This project is done within the Swedish Maritime Robotics Centre (SMaRC), under the broader goal of underwater docking for AUVs.

Autonomous underwater docking plays an integral role in the quest to enable continuous and long-term operation of AUVs. Not only does autonomous docking greatly decrease the costly reliance on human intervention after initial deployment, but it also provides AUVs with the means for automatic charging and data off-loading. Within the SMaRC project, there is interest in the automatic recovery of an AUV by another underwater vehicle or mother ship in movement. The problem of estimating the state of both the mother ship and the AUV has been tackled using a SLAM framework, which relies on USBL acoustic measurements, dead-reckoning, and visual feedback from a set of light beacons on the mother ship’s docking station.

SMaRC has gotten preliminary results [12] that hint at the fact that assumptions on Gaussian uncertainty in the measurements might be affecting the performance of the estimators. This does not come as a surprise given the nature of the acoustic measurements. Thus, a multimodal or non-Gaussian framework is considered a potential way to improve the robustness of the estimation.
1.4 Goal

The goal of this project is to obtain a probabilistic inference method that can consider multiple hypotheses simultaneously, is robust to outliers, and improves the computational cost over existing methods. The focus is placed on problems where a small number of distinct hypotheses must be considered at the same time.

To achieve this goal, the following steps are taken:

1. Review the state of the art in multimodal probabilistic inference methods.
2. Identify methods that can efficiently represent multiple distinct hypotheses, as well as their weaknesses and limitations.
3. Improve or expand on existing work, developing a new algorithm.
4. Test the outlier robustness and computational cost of the new algorithm when using robust measurement models, compared to existing methods.

1.5 Delimitations

Although computational cost is considered an important characteristic of state estimation methods, this thesis does not include the development or validation of a real-time algorithm for any particular system. The data processing always occurs offline.

The different considered methods are evaluated with simulated results. This is because simulation experiments are faster to design and execute than real experiments, which allows more scenarios to be tested.

1.6 Structure of the thesis

Chapter 2 introduces the basics of probabilistic inference methods for robotics and the factor graph representation, before reviewing practical state estimation methods.

Chapter 3 describes the proposed computational methods used in the GM-SPA in detail. The GM-SPA is justified as an efficient inference method for robust measurement models.

Chapter 4 describes the simulation tests used to test and compare the GM-SPA with alternative methods, and discusses the results.
Chapter 5 contains the conclusions from the thesis, a reflection on societal impact, and describes avenues for further study.

Appendix A presents the factor graph representation in detail and derives the sum-product algorithm (SPA), the algorithm of which the proposed method is a particular implementation.

Appendix B details the column pivoted QR (QRCP), the fundamental computational tool used in the proposed method.
8 | Introduction
Chapter 2

Background

This chapter provides background information about how robots can use probability to deal with uncertainty in localization, how factor graphs provide a powerful mathematical framework for efficient inference, and about existing state estimation methods.

2.1 Probability for autonomous robots

In robotics, the usual method to represent uncertainty is probability. A robot maintains a belief function defined on some state space that represents how much it believes that each state is the true state.

Considering GPS localization as an example, a robot would represent the knowledge from a measurement $z$ by attributing a high likelihood to locations $x$ closer to $z$, according to some probabilistic model: for example, the robot might model the uncertainty from the sensor by saying that the measurement is a sample from a multivariate Gaussian distribution $[3]$ centered on the real location. This is expressed by

$$ p(z|x) = N(z; x, \Sigma), $$

where $\Sigma$ is a covariance matrix.

2.1.1 Data fusion

To maintain a probabilistic belief representation of some variables of interest $X \in \mathbb{R}^n$, a robot can fuse information from prior knowledge, $p(X)$, with information from a set of $K$ conditionally independent measurements $Z_{1:K} = \{z_1, ..., z_K\}$, via the Bayes Rule.
The result is called the posterior belief and is obtained as
\[
p(X|Z_{1:K}) = \frac{p(Z_{1:K}|X)p(X)}{p(Z_{1:K})} \propto p(Z_{1:K}|X)p(X) = p(X) \prod_{i=1}^{K} p(z_i|X).
\]

(2.1)

If there is no prior knowledge \(p(X)\) available, a uniform improper prior \(p(X) = 1\) can be used. As improper priors do not respect the unit measure axiom in the Kolmogorov probability axioms [13], they are not a classical probability. However, probability theory can be formulated without the unit measure axiom in a way that maintains the possibility for proper posterior distributions, according to [14]. If the posterior \(p(Z_{1:K}|X)p(X)\) is a normalizable function of \(X\), then it can be normalized into a probability distribution.

Sometimes, posteriors do not need to be proper probabilities because they are interpreted simply as a relative strength of belief in different states. Maximum likelihood estimation maximizes just the likelihood without the prior. Note that not using a prior is equivalent to using a uniform improper prior.

2.1.2 Modeling outliers

Unpredictable failures in sensors or changes in the environment lead to measurements that do not follow the available measurement models. This possibility can be included in the generative model of a measurement by considering that there is a probability \(P_o\) that it is an outlier. This leads to a mixture model that in this work is named the outlier component model.

To do so, we can consider two hypotheses. In the first, called \(H_i\), the measurement is obtained under normal conditions, according to the inlier model \(p(z|X, H_i)\). In the second hypothesis, called \(H_o\), it is obtained under a different outlier model \(p(z|X, H_o)\). This naturally leads to a mixture model for the total likelihood due to

\[
p(z|X) = \sum_{H \in \{H_i, H_o\}} p(z, H|X) = \sum_{H \in \{H_i, H_o\}} p(z|X, H)p(H|X).
\]

(2.2)

To avoid working with mixture models, classical outlier rejection considers that the most probable hypothesis is true, neglecting the others. This strategy can lead to wrong classifications, especially in ambiguous problems.

An alternative is to consider all the hypotheses separately, which creates many parallel problems, and choose later which one is the most probable.
This strategy can use more data for the decision but requires solving many problems in parallel, although it is possible to reuse some computations between problems [15].

In this work, we consider solving the full mixture model directly, which is possible with multimodal methods. Since, in this case, there are only two hypotheses, the notation can be simplified. The prior on the hypothesis is also restricted to be independent of the state, so \( p(H|X) = p(H) \). The result will be referred to as the outlier component model and is expressed as

\[
l_z(X) = (1 - P_o)p_i(z|X) + P_o p_o(z|X),
\]

(2.3)

where \( P_o = p(H_o) \), \( p_i(z|X) = p(z|X, H_i) \), and \( p_o(z|X) = p(z|X, H_o) \).

One problem here is that the process that generates outliers is frequently unknown, so the likelihood \( p_o(z|X) \) must be guessed. To make the outliers not affect the result of the estimation, it is desirable to choose a uniform likelihood for the outlier component. Uniform likelihood models have been used for outlier robustness in filtering [16], in smoothing [17], and in clustering [18] problems. In this work, using a constant value as the outlier component in the outlier model is called the null hypothesis (NH) model, as it amounts to saying the measurement has no information at all about the state.

A simple choice for the value of the uniform is to take the maximum likelihood from the inliers sensor model. The resulting NH model is defined as

\[
l_z(X) = (1 - P_o)p_i(z|X) + P_o \max_{X^*} p_i(z|X^*).
\]

(2.4)

With this formulation, there is a single parameter, the probability \( P_o \), to tune how much trust is placed on a measurement following the inlier model.

### 2.1.3 The curse of dimensionality

An important aspect of probabilistic methods is that, for the most part, to directly represent the full posterior (2.1), the number of parameters increases at least with the square of the dimension of \( X \). This has to do with the fact that with \( N \) scalar variables, there are \( N \cdot (N - 1)/2 \) covariances. Methods that intend to capture those relationships need at least as many parameters.

Since the robot will typically be moving in space and time, the variables to estimate will include a time-varying state \( x_t \) at various time instants, and observations from each time-step \( z_t \). The number of variables to estimate can reach thousands, for example in the case of a moving car in [19]. In many applications, the high dimensionality makes the joint posterior too complex to
maximize in useful time. Two common strategies to reduce the computational load will be discussed in detail. Most practical state estimation methods differ mainly in the decisions made in these two regards.

The first design decision is the choice of parameterization for the probability distributions used. These vary in expressive power and computational load. For simpler problems, simple models are adequate. For highly nonlinear problems with complicated noise distributions, more expressive models are necessary. However, even with a cheap parameterization, the Gaussian distribution, estimating the joint posterior directly often requires too much computational power. The second design decision has to do with marginals.

The usual marginal definition in the context of probability distributions is the marginal integral. The vector of variables to be estimated $X$ can also be understood as a set of scalar variables to be estimated. For a probability density function $p(X)$ and a subset of variables $y \subset X$, the marginal of $p(X)$ on $y$ is

$$p(y) \equiv \int p(X) dX \setminus y.$$ (2.5)

A marginal is a summary of information. It retains, from the full joint distribution, just the relevant information for the distribution of $y$. It forgets about the relationships between the variables in $y$ and the other variables in $X$.

Factor graph methods [20] allow efficient calculation of the marginal posteriors without first calculating the full posterior. Instead of a single high-dimension posterior, the result of inference is now many low-dimension marginal posteriors, effectively avoiding the curse of dimensionality.

The second design decision is how much marginalization to use. On the one hand, determining the marginal for bigger groups of variables $y$ maintains more correlation information. On the other hand, it requires more computation.

What follows is a review of the usual distribution parameterization decisions and marginalization strategies.

### 2.2 Representing distributions

As argued before, the choice of parameterization of probability distributions is one of the most important in an inference algorithm. It has a big impact on the kinds of problems that can be solved and the computational cost. Generally, parametric representations lead to faster algorithms, while non-parametric representations can solve more general problems.
The Gaussian distribution [3] is one of the most common choices of representation. It is a unimodal distribution defined completely by its mean and variance. It benefits from having simple and exact calculations of marginals, conditionals, linear transformations, and products of probability density functions, all of which are also Gaussians or scaled Gaussian densities. It suffers from not being expressive enough to approximate multimodal distributions and from having a number of parameters that grows quadratically with the dimension of the distribution.

To obtain more expressiveness while maintaining the ease of calculations on the Gaussian, a mixture of Gaussians can be used. A relevant advantage for us is that they can represent multimodal distributions. The main disadvantage of mixtures is that products of mixtures with $K$ components result in a mixture with $K^2$ components. Approximate component reduction techniques, such as merging and pruning, have been used to maintain a manageable number of components [17, 21, 22, 23]. As the products need to be computed before they are simplified, this is prohibitively expensive for big enough $K$.

Discrete approximations of distributions can be used to represent arbitrary distributions and facilitate propagation through non-linear transformations. Assuming the distribution, or a close approximation, is easily sampleable, without assuming the distribution belongs to any particular parametric family, many random and independent samples can be taken to form a discrete approximation [10].

One disadvantage of such particulate representations of probability density is that it is not possible to directly calculate their products. To address this, MM-iSAM [11] uses kernel density estimation [24] to obtain a continuous estimate from the set of samples, formed as a Gaussian mixture. Multi-scale Gibbs sampling [25] is used to sample from the product directly without calculating it, resulting again in a particle representation, because the number of components used is usually too high for direct calculation.

Particle-based representations are powerful approximate methods but generally require many particles to be accurate, resulting in a computational cost that is bigger than with parametric methods. It is common practice [26, 27, 28, 29] to use mixed representations where part of the problem is represented with particles and part of the problem is represented with parametric distributions.
2.3 Factor graphs

Since the choice of which marginal distributions are to be determined has a major impact on the information that is obtained and the computational cost of an inference problem, using a representation that facilitates reasoning about this decision is useful. The approach taken in this work is that of factor graphs [20]. Appendix A presents a more detailed formalization of factor graphs.

A factor graph is a bipartite graph that expresses how a function of many variables is factored into factors of subsets of those variables. The applicability at representing posterior distributions is apparent from (2.1), where it is expressed as a product of priors and likelihoods. From the factor graph perspective, both probabilities and likelihoods are simply factors.

In a factor graph, the nodes are either variable nodes or factor nodes. Each factor node is connected to the variable nodes that it is a function of and each variable node is connected to the factors that are defined on it.

An example of a cyclic factor graph defined on three robot poses $x_t$ and two landmark positions $l_m$ is presented on Figure 2.1. This SLAM example is from [30]. The big circles are the variable nodes and the small filled black circles are the factor nodes. There are two motion factors relating the poses of the robot, which can include odometry information or come from a motion model. The three factors that relate the robot poses to the landmarks come from landmark sightings. The single unary factor defined just on $x_1$ is a prior.

![Factor graph example](image)

Figure 2.1: Example factor graph of a SLAM problem, adapted from [30].

2.3.1 The sum-product algorithm

The sum-product algorithm (SPA) [20] determines all the marginals of the global function on each variable node exactly, assuming the factor graph has
no cycles. For this reason, altering a factor graph by grouping up variables into
different variable nodes leads the SPA into calculating different marginals.

Separating variable nodes apart allows the algorithm to better exploit
the sparsity of the problem, but determines less correlation information.
Sometimes, grouping up variable nodes eliminates cycles, enabling the use
of the algorithm.

The SPA relies on summary and product operations, as the name implies.
In our context, the summary is the marginal integral, but other choices could
be taken, such as the maximum operator. This is further discussed in Section
A.2.2.

The SPA is a message passing algorithm that is simple to implement. It just
requires that each node on the tree sends through each of its edges a message
when it receives a message on all its other edges. Let \( n(v) \) denote all the
neighbors of node \( v \), and, in particular, let \( X_f \) be the variables that factor \( f \) is
deﬁned on. The message sent from a factor \( f \) to a variable \( x \) is

\[
m_{f \rightarrow x}(x) \equiv \int \left[ f(x_f) \prod_{y \in n(f) \setminus \{x\}} m_{y \rightarrow f}(y) \right] dX_f \setminus \{x\}, \tag{2.6}
\]

and the message sent from a variable \( x \) to a factor \( f \) is

\[
m_{x \rightarrow f}(x) \equiv \prod_{h \in n(x) \setminus \{f\}} m_{h \rightarrow x}(x). \tag{2.7}
\]

Each message is a summary of all the information from the branch of the
tree it is being sent from, keeping only the information that is relevant to
calculate the marginals of the rest of the tree. The SPA executes in a ﬁnite
number of steps, as long as the factor graph is a tree. The chosen numerical
representation of the factors must support marginal integrals of products of
functions deﬁned on different variables, to calculate (2.6).

An important example of tree factor graphs is the hidden Markov model
(HMM), represented in Figure 2.2. For a tutorial on HMM see [31]. The
SPA on these graphs is a generalization of ﬁltering, prediction, and smoothing
techniques.

HMM style factor graphs are the most common type of tree factor graph,
but tree factor graphs are not limited to them. They allow for some more
flexible modeling. For example, one can model \( N \) moving targets that collide
and are fused into a single target, such as represented in Figure 2.3 for \( N = 2 \).
2.3.2 Cyclic factor graphs

As the SPA only calculates the exact variable node marginals if the graph is a tree, graphs with cycles must be converted to tree representations before they can be solved. Figure 2.1 is an example of a cyclic factor graph, due to the cycle on $x_1$, $x_2$, and $l_1$. SLAM problems are typical examples of cyclic factor graphs in robotics applications.

In some applications, the SPA is applied iteratively and approximately, called Loopy Belief Propagation, but convergence is not guaranteed in general [32].

Exact inference in the presence of cycles is still possible by changing the graph to eliminate the cycles while representing the same global product, frequently at the cost of grouping up variables together.

For the graph in Figure 2.1, it is sufficient to group the variable nodes $x_1$ and $l_1$ together, resulting in an acyclic factor graph. The graph can then be solved with the SPA, with the cost of working with beliefs of higher dimension.
Working with possibly cyclic graphs incrementally and efficiently requires clever ways of maintaining a tree representation over time.

A simple and inefficient solution for SLAM problems is to create variable nodes where the landmarks and the states are grouped together, resulting in an augmented state HMM representation. This operation is called variable node stretching in [20]. The SPA can then be applied. By grouping up all landmarks and robot state variables in a single augmented state node, this method does not exploit the sparsity in the relationships between the augmented state variables.

Automated methods to determine good groupings that lead to a tree representation with many small groups have been developed. These methods construct junction/clique [33, 34, 35] trees, like the Bayes Tree [30]. A variation of the SPA can then be applied to the tree representation.

2.4 Practical inference methods

The application of the SPA with different choices of distribution representation and variable node groupings leads to different practical algorithms. Some other methods do not have any obvious re-interpretation as particular cases of SPA. A summary of the available types of algorithms follows, split according to the kind of problem they solve.

There are three usual problems in state estimation for HMM problems, all of which are formulated as determining the posteriors $p(x_t|Z_{1:k})$: filtering when $t = k$, prediction when $t > k$ and smoothing when $t < k$. Prediction can be solved efficiently by most filtering algorithms by simply skipping the measurement steps. Also discussed are algorithms capable of solving general factor graphs.

2.4.1 Filtering and prediction

If the goal is just filtering and prediction, then the inference procedure can be simplified by optimizing only the state at the most recent time, $x_t$. Filtering can be obtained with the SPA by removing the previous states out of the factor graph at each iteration, in an operation called out marginalization.

Gaussian filters

The Kalman filter (KF) is a popular method for fusing information from sensors and dynamic models. It is only applicable for problems in the form of a HMM. It calculates the exact solution of the filtering problem when the
motion and observation models are linear with additive Gaussian noise and
the prior is Gaussian. Many other methods are formulated as extensions of the
KF.

The extended Kalman filter (EKF) and unscented Kalman filter (UKF)
adapt the KF to be applicable when the motion and/or observation models are
non-linear, but are approximate methods. Both of these extensions maintain
the property of parameterizing beliefs as Gaussian distributions.

The EKF linearizes the motion and observation models at each iteration, at
the mean of the current estimate, and applies a Kalman filter to the linearized
system. For a derivation of the KF and the EKF see [36].

The UKF instead approximates the filtered belief \( p(x_t | Z_{1:t}) \) as a discrete
distribution with a small set of points, maintaining the mean and covariance.
By representing the distribution with a finite set of particles, the nonlinear
functions can be applied exactly. The points are called sigma points and are
carefully chosen in a deterministic manner. The original UKF is presented in
[37], and [38] presents a systematic overview of multiple variations.

Compared to the EKF, the UKF is frequently reported as being more
accurate [39, 40]. It also has the advantage of being derivative-free.

Regarding complexity, all variants of the KF deal with \( n \times n \) matrices,
where \( n \) is the dimension of the state \( x \). The simplest implementations
are \( \Theta(n^3) \)* for an iteration of the filter, due to matrix multiplication for
example. There are, however, algorithms for these operations with better time
complexities of \( 2^c \), with \( c \in [2, 3] \). Naturally, all implementations that create
a new full matrix at every iteration will have a \( \Omega(n^2) \)† time complexity simply
from creating the matrix. This makes the Kalman filter, the EKF and the UKF
costly for problems with big or time-growing states, as for example in the case
of SLAM problems with many map features.

While the previous methods parameterize distributions with a single
Gaussian, it is possible to use multiple Gaussians. The Gaussian mixture filter
[42, 22] allows better approximations of non-linear systems and representation
of non-Gaussian distributions. The problem of exponential increase in
components discussed in Section 2.2 is the main drawback of this approach.

There exist implementations of the Gaussian filters that use square roots
of the covariance/information matrices [43, 44, 45]. This is for two reasons.
Firstly, these implementations have better numerical properties [46, 47].

---

*Readers unfamiliar with asymptotic notation are directed to [41], but saying that \( g \) is \( \Theta(f) \) can be interpreted vaguely as "asymptotically, \( g \) grows as fast as \( f \)."

†Saying that \( g \) is \( \Omega(f) \) can be interpreted vaguely as "asymptotically, \( g \) grows as fast or faster than \( f \)."
Secondly, it guarantees that the corresponding squares are positive semi-definite. Propagating the covariance/information matrices directly runs the risk of them becoming indefinite due to numerical errors.

**Particle filters**

Particle filtering [10] approximates the filtered posteriors \( p(x_t | Z_{1:t}) \) as discrete distributions. Unlike the UKF, there is no underlying assumption that the posteriors are Gaussian, and the sample points are chosen randomly.

Even when the relevant distributions cannot be sampled directly, importance sampling is used to create approximations. The idea is that a simpler proposal distribution can be used to create the samples and their weights adjusted with the point-wise values of the probability density functions. Resampling steps are used to forget samples in low-probability areas and add more particles to high-probability areas, focusing the approximation effort in the areas with higher probability.

The particle filter (PF) is applicable with non-linear system dynamics and non-Gaussian noise distributions. As samples are chosen randomly, many samples are needed to obtain low variance results, making this method more computationally expensive than the previous parametric alternatives.

FastSLAM [26, 27] and other Rao-Blackwellized particle filters [28, 29] are methods that can exploit the conditional independence between the landmarks in SLAM, conditioned on the robot path. They determine the full joint posterior in a factored representation. To the best of the author’s knowledge, no interpretation of these methods as factor graph operations has been published. As these methods determine the full joint posteriors, they are not equivalent to any use of the SPA. They represent the robot path with a finite set of path hypotheses/particles and have a map estimate for each particle.

The map, however, is parametric, reducing computation costs when compared to a solution using only particle filtering. For a filtering iteration, FastSLAM achieves a time complexity of \( \Theta(K \log(L)) \), where \( K \) is the number of particles and \( L \) is the number of landmarks. This allows fast approximate inference with many landmarks, which is unfeasible with augmented state methods.

### 2.4.2 Smoothing

When the goal is to use information from the relative past and future of a certain state \( x_t \), algorithms must be a little more complex.
There are two main general algorithms to perform smoothing [10]: two-filter smoothing (TFS) and forward-backward smoothing (FBS). TFS [48] is a direct application of the SPA. Its main difficulty is that it deals with improper factors. The main difficulty of FBS [49] is that it involves a division between probability densities, which does not always have a closed form, for example with Gaussian mixtures.

In order to avoid the problem of dealing with improper likelihoods in TFS, as pointed out in [21], it is possible to use uninformative proper priors for the most recent state, ensuring that messages in the backward filter will always be proper. This allows implementing TFS simply as two filters in opposite directions. An example of a proper uninformative prior is a Gaussian with a big covariance. This introduces a bias in the estimates towards the mean of that component, which can be reduced by making the covariance bigger. There is a limit to how big it can be before numerical problems arise.

**Gaussian smoothers**

Gaussian functions can be extended to include improper likelihoods by using information matrices that are positive semi-definite. The backward filter can use these functions as a parameterization, resulting in Gaussian smoothers built as extensions of the KF, EKF, and the Gaussian mixture filter [17, 23, 50].

These functions do not appear to have a standard name in the literature. They are referred to as reduced dimension Gaussian (RDG) in [17], as unnormalized Gaussian in [50], and get no proper name in [23]. In this thesis, they are called by the names improper Gaussian (IG) or RDG, depending on the parameterization used.

IGs are scaled Gaussian probability densities in a subspace of the domain and uniform in the rest of the domain. Since a uniform is also an IG, mixtures of IGs are also similar to the Gaussian and uniform mixtures used in [16], except for the fact that the uniforms in this thesis are improper instead of proper. The ability to represent uniforms makes IG mixtures efficient at representing the NH model from (2.4), requiring only one component for it.

Smoothing with different combinations of considering linear/non-linear systems and single Gaussian/Gaussian mixture noise have been explored. Linear systems with Gaussian mixture noise are considered in [17, 23, 51]. Non-linear systems with Gaussian noise are considered in [50]. Non-linear systems and Gaussian mixture noise do not seem to have been explored together yet.

One approach that generalizes all of the mentioned combinations is to
approximate the likelihood directly. The idea is argued for filtering in [22] and for smoothing with linear systems and Gaussian mixture noise in [23], but it does not need to be limited to those uses. In this thesis, it is also applied to smoothing with non-linear measurements and Gaussian mixture noise.

**Particle smoothers**

Particle filters and variants such as FastSLAM can trivially be used as smoothers by simply remembering the trajectory of each particle. As particles are resampled, more will share values for the previous states. This maintains only particles in the previous states that are consistent with new information. This form of smoothing suffers from severe particle depletion. For any state $x_t$, there will eventually be a $k > t$ such that the smoothed posterior $p(x_t | Z_{1:k})$ only has a single particle.

Techniques to mitigate particle depletion include backward simulation [52] and block sampling [53]. The former manages to reduce particle depletion but still uses the same set of samples that were initially proposed, adjusting only the weights. The latter aims to solve the degeneracy problem by generating new samples not just of the most current state but also of previous states.

### 2.4.3 Inference on general factor graphs

As the factor graph approach has the advantage of being a general framework for filtering, smoothing, and SLAM, it became popular in the state of the art.

Graphical SLAM [54] is a method that searches for the estimate that maximizes the full joint posterior of a factor graph built with non-linear functions and non-Gaussian noise terms. It reduces the complexity of the search by maximizing local factors iteratively instead of the full posterior directly, by doing local calculations in local frames that can be reused, and by determining automatically which nodes of the graph need to be updated. It does not determine distributions, only an estimate of the variables.

The state of the art with inference on likelihoods from Gaussian distributions and non-linear functions is incremental smoothing and mapping (iSAM2) [55], which applies a variant of the SPA on the Bayes tree [30]. The non-linear functions are relinearized automatically when necessary around the current estimates. The method is iterative but determines the maximum likelihood estimates exactly, which amounts to a non-linear least squares optimization problem. It requires good enough initial estimates to obtain convergence. Automated variable grouping builds the Bayes tree, which allows efficient inference in cyclic graphs and allows computations to be
reused in incremental inference. GTSAM [56] includes a mature open source implementation of iSAM2.

Multimodal incremental smoothing and mapping (MM-iSAM) [11] was developed to deal with multimodal distributions. It approximates arbitrary likelihoods with a specific Gaussian mixture representation. The approximation starts by sampling from the likelihood, usually hundreds of independent samples, and attributing the same diagonal covariance matrix and weight to all particles. The covariance is optimized with leave-one-out-likelihood cross-validation, a kernel density estimation technique [24]. Likelihoods that are improper can be adapted to a samplable form by grouping them with other factors or extending them with uniform noise.

To address the exponential increase in mixture components that characterizes products of mixture densities, multi-scale Gibbs sampling [25] is used to sample from the product, without determining the product directly.

Like its unimodal counterpart, MM-iSAM uses a variant of the SPA on the Bayes tree to perform efficient inference in cyclic factor graphs and reuse computations in incremental inference. One weakness of this method is that the chosen numerical representation of factors does not allow direct products between functions of different variables, which is essential in the direct implementation of the SPA and in solving the Bayes Tree. The exact step where this is necessary is called the intraclique integral, which is explained in detail in [11, Chapter 5.4]. The equivalent step in the SPA, described in detail in Section A.2, is (2.6).

An approximation method is necessary to determine the intraclique integral. The method that MM-iSAM uses is based on Gibbs sampling and it suffers from not being guaranteed to converge to the correct result. MM-iSAM is under active development in the open-source software package Caesar [57].

### 2.4.4 Summary

There is a vast literature on Bayesian inference and state estimation. Generally, the computational cost of a method is based mainly on the choice of representation for probability distributions and the marginalization strategy.

Regarding the choice of parameterization for distributions, the Gaussian distribution should be the first to be considered in any inference problem, due to its simplicity and computational ease. This is the case with the Kalman filter and iSAM2. Gaussian mixtures and particle representations are the common alternatives, useful when the Gaussian distribution is too restrictive, for example with multimodal distributions.
Regarding the choice of which marginals to compute, the total joint posterior is typically considered too complex to compute, with FastSLAM being a notable exception that suffers from particle depletion for smoothed estimates. Factor graphs provide a general and powerful framework for representing, analyzing, and solving inference problems. The SPA is a factor graph method applicable to acyclic factor graphs and determines the marginals on variables grouped together into nodes.

Traditional methods were limited to hidden Markov models, but the move to factor graph methods led to the creation of automatic methods of turning general factor graphs into Junction Trees like the Bayes Tree, where exact inference is possible. Importantly, these methods can exploit the sparse structure of the problem and allow the reuse of computations in incremental inference. Methods based on the Bayes Tree include the well-established unimodal iSAM2 and its multimodal variant MM-iSAM.

The improper Gaussian represents the likelihood from Gaussian noise in a possibly under-constrained linear measurement exactly. Mixtures of IGs are efficient representations of the outlier component mixture model (2.3) and can approximate non-linear and non-Gaussian measurement likelihoods. As such, they are the chosen factor parameterization in this thesis. The SPA is implemented for these mixtures, resulting in the GM-SPA. The method is a generalization of the ideas applied to target tracking with linear measurements, in the presence of clutter measurements, in [17].
Background
Chapter 3

The Gaussian mixture sum-product algorithm

In Chapter 1, the outlier component model (2.3), of which the NH model (2.4) is a particular case, is introduced. The NH models possibly wrong measurements as a mixture of the inlier sensor model and an improper uniform likelihood. The result of multiplying likelihoods of this form is a multimodal sum-mixture.

Sample-based representations like particle representations and MM-iSAM are capable of approximately representing multimodal/non-Gaussian distributions. However, representing improper likelihoods is impossible in these frameworks. A possible solution is to adapt the NH to use a very wide Gaussian or a truncated uniform instead, which can be sampled. This approach is used later in the simulation in Section 4.3. There are important computational cost difficulties in using wide uninformative components with particle representations and with the Gaussian mixture representation of MM-iSAM.

In a particle filter, a wide Gaussian or finite uniform is difficult to represent because many particles need to be spread across the big $n$-volume (the equivalent of volume for $n$ dimensions) it covers.

The MM-iSAM representation, detailed in Section 2.4.3, uses a mixture of Gaussians where every Gaussian has the same diagonal covariance and weight. Due to these constraints, it is expensive to approximate mixtures with wide outlier components while also accurately representing the inlier model. To accurately represent the likelihood from the inlier measurement model, each particle must be given a small covariance. In turn, this means that many particles are still necessary to approximate the wide outlier component.
A more efficient strategy is to use Gaussians with different covariances and weights. This immediately allows exactly representing the outlier component if a wide Gaussian is used. To represent the infinite uniform of the NH model, the key is to use improper Gaussian (IG) functions, where the information matrix is allowed to be positive semi-definite. This will allow the representation of the NH model likelihood as a mixture of the inlier likelihood and a single extra IG with a 0 information matrix.

IG mixtures are used to implement the SPA, described briefly in Section 2.3.1 and in detail in Appendix A. The resulting novel algorithm is called the Gaussian mixture sum-product algorithm (GM-SPA). All that remains to specify is the precise definition of an IG, and how products and marginals of IG mixtures are calculated. The IG is defined in Section 3.1, while its mixtures and operations on its mixtures are described in Section 3.2.

### 3.1 Improper Gaussian

The central function to this work is called improper Gaussian (IG) and is defined on $x \in \mathbb{R}^n$ as

$$G(x; b, A) \equiv \exp \left( -\frac{1}{2} \|Ax - b\|^2 \right),$$

(3.1)

where $A \in \mathbb{R}^{r \times n}$ has full rank $r$ and $b \in \mathbb{R}^r$. $A$ is a square root of the information matrix. A covariance parameterization is not possible since the information matrix might not be invertible. The mode parameter $\mu$ is related to this parameterization by $b = A\mu$. Its existence is guaranteed since $A$ is full rank, but it only is unique when $r = n$.

Other equivalent parameterizations are possible. There is the canonical parameterization that uses $\eta = A^Tb$ and $\Lambda = A^TA$, like in [58] and with the exception of a minus sign in [23]. Another parameterization, named RDG, is used in [17, 50]. It will also be used in this work, so it is described in detail in Section 3.1.1.

The use of the square-root information matrix $A$ as opposed to the information matrix $\Lambda$ is due to numerical stability and the fact that rank deficient information matrices $\Lambda$ permit full rank rectangular square roots $A$.

The use of $b$ instead of $\eta$ was preferred for two reasons. Firstly, the author finds its interpretation clearer: $G(x; b, A)$ is the likelihood on $x$ resulting from the measurement $b$ obtained with the measurement model $b = Ax + r$, where $r$ is Gaussian with 0 mean and identity covariance. Secondly, it has the
convenient property that the maximum likelihood is always one and achieved when $Ax = b$. This equation has at least one solution at $x = A^T(AA^T)^{-1}b$, since $A$ has linearly independent rows.

The derivation that the likelihood from linear measurement models with Gaussian noise is an IG as defined in (3.1) is simply a substitution. Consider some $z = Bx + r$, where $B$ is a full rank $k \times n$ matrix, with $k \leq n$. $z$ is an observed noisy measurement of some unobserved state $x$ and $r$ is Gaussian noise with 0 mean and full rank information matrix $\Lambda = R^TR$. The likelihood of $x$ given $z$ is

$$l_z(x) \equiv p(z|x) \propto \exp\left(-\frac{1}{2}(z - Bx)^T R^TR(z - Bx)\right)$$ \hspace{1cm} (3.2)

$$= \exp\left(-\frac{1}{2} \|RBx - Rz\|^2\right).$$ \hspace{1cm} (3.3)

$$= \mathcal{G}(x; Rz, RB).$$ \hspace{1cm} (3.4)

$RB$ is necessarily full rank, since $R$ is non-singular and $B$ is full rank.

The name improper Gaussian is chosen to emphasize that the information matrix $\Lambda = B^TB$ might not be full rank. An IG is said to be full rank if its information matrix is full rank, equivalent to $n = r$. A full rank IG has a finite integral, although the integral is not necessarily 1 like it is for a proper probability density. Examples of IGs include:

- scaled Gaussian probability density functions, for which $A$ is invertible;
- the uniform function, for which $A = 0$;
- likelihoods from under-determined linear measurement models with additive Gaussian noise, for which $A$ is rectangular.

An IG can be interpreted as the probability density of a Gaussian distribution extended to a bigger space. This interpretation is useful to merge or compare different IGs by merging or comparing their underlying Gaussian distributions. The reduced dimension Gaussian representation makes both the extension transformation and the underlying Gaussian distribution explicit.

### 3.1.1 Reduced dimension Gaussian

An improper Gaussian $\mathcal{G}(x; b, A)$, with $A \in \mathbb{R}^{r \times n}$, is uniform along some directions and behaves as a Gaussian probability density function in the others. These directions are precisely the kernel and row space of $A$. 
The reduced dimension Gaussian (RDG) representation, used in [17], is an alternative parametrization of an improper Gaussian, defined as

\[ k_G(x; b, A) = wN(Q^T x; \mu, \Sigma), \tag{3.5} \]

where \( Q \in \mathbb{R}^{n \times r} \) has orthonormal columns, so \( Q^T Q = I \), \( \mu \in \mathbb{R}^r \), and \( \Sigma \in \mathbb{R}^{r \times r} \) is positive definite. In practice, all parameterizations of the Gaussian distribution can be used. To benefit from better numerical properties, the square root covariance matrix, named \( F \), is used, so

\[ k_G(x; b, A) = wN(Q^T x; \mu, F^T F). \tag{3.6} \]

The RDG will be important to allow comparing two different RDG with the same \( Q \) by comparing their Gaussian distributions. The RDG parameterization is not used all the time because inference operations are simpler in the IG parameterization. The relation between the two is given by

\[
wN(Q^T x; \mu, F^T F) = w \left| \det(\sqrt{2\pi F}) \right|^{-1} \exp \left( -\frac{1}{2} \| F^{-T} Q^T x - F^{-T} \mu \|^2 \right)
\tag{3.7}
\]

\[
= w \left| \det(\sqrt{2\pi F}) \right|^{-1} G(x; F^{-T} \mu, F^{-T} Q^T)
\tag{3.8}
\]

\[
= k_G(x; b, A), \tag{3.9}
\]

which implies

\[ A = F^{-T} Q^T, \quad b = F^{-T} \mu \quad \text{and} \quad k = w \left| \det(\sqrt{2\pi F}) \right|^{-1}. \tag{3.10} \]

The expressions (3.10) give the IG parameters from the RDG parameters and imply that the columns of \( Q \) form a basis of the row space of \( A \). To obtain the inverse relations, a QR factorization (B.1) of \( A^T \) can be used to determine \( A = R_A^T Q_A^T \), with invertible \( R_A^T \). Then, the RDG parameters can be determined as

\[
F = R_A^{-1}, \quad Q = Q_A, \quad \mu = R_A^{-T} b \quad \text{and} \quad w = k \left| \det((2\pi)^{-\frac{1}{2}} R_A) \right|^{-1}.
\tag{3.11}
\]

A relevant question is what operations are closed for IGs, as this is one of the main advantages of Gaussian distributions. The following sections show that the products and marginals of IGs are proportional to an IG and can be
determined exactly.

### 3.1.2 Product of improper Gaussians

One of the most common operations in factor graph based inference is multiplication of likelihood functions, so it is important to check that the product of IGs can be easily determined.

This section determines the product of \( N \in \mathbb{N} \) improper Gaussians \( \mathcal{G}(x; b_i, A_i) \).

\[
f(x) = \prod_i \mathcal{G}(x; b_i, A_i) \tag{3.12}
\]

\[
= \prod_i \exp \left( -\frac{1}{2} \|A_i x - b_i\|^2 \right) \tag{3.13}
\]

\[
= \exp \left( -\frac{1}{2} \sum_i \|A_i x - b_i\|^2 \right) \tag{3.14}
\]

\[
= \exp \left( -\frac{1}{2} \left\| \begin{bmatrix} A_1^T \\ A_2^T \\ \vdots \\ A_N^T \end{bmatrix} x - \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \end{bmatrix} \right\|^2 \right) \tag{3.15}
\]

\[
= \exp \left( -\frac{1}{2} \|S x - s\|^2 \right), \tag{3.16}
\]

where \( S \) and \( s \) simply stack all \( A_i \) and \( b_i \), respectively.

While correct, this has the problem of only stacking all the measurements together, not meaningfully fusing them, leading to an unbounded increase in the rows of \( S \) and \( s \) as the number of factors increases. Furthermore, since \( S \) does not have linearly independent rows, it does not fit the definition of an IG (3.1).

To constrain the number of rows of the square root matrix back down to \( n \), we can use an \( S = QR \) factorization. Better yet, a rank revealing factorization, like column pivoting QR (QRCP), allows obtaining an \( r \times n \) matrix \( A \) of rank \( r \). Appendix B discusses the QRCP factorization and derives how the factorization is useful to reduce the dimension of \( S \). The relevant equality is (B.6) and it will be repeatedly useful in the calculations here.
To start, take the QRCP of $S$:

$$S = \begin{bmatrix} A_1 \\ A_2 \\ \vdots \\ A_N \end{bmatrix} = QRP^T. \quad (3.18)$$

The relevant properties of the QRCP are that the columns of $Q$ are orthonormal ($Q^T Q = I$) and $RP^T \in \mathbb{R}^{r \times n}$ has full rank $r$. This allows us to separate the squared norm into a variable term and its minimum value, according to (B.6). The result is

$$f(x) = \exp \left( -\frac{1}{2} \| Sx - s \|^2 \right) \quad (3.19)$$

$$= \exp \left( -\frac{1}{2} \| RP^T x - Q^T s \|^2 - \frac{1}{2} \| s - QQ^T s \|^2 \right) \quad (3.20)$$

$$= G(x; Q^T s, RP^T) \exp \left( -\frac{1}{2} \| s - QQ^T s \|^2 \right), \quad (3.21)$$

where $RP^T$ and $Q^T s$ have only $r$ rows, $r$ being the rank of $RP^T$ and $S$.

To conclude, the result is

$$\prod_i G(x; b_i, A_i) = c G(x; b, A), \quad (3.22)$$

where

$$A = RP^T, \quad b = Q^T s, \quad c = \exp \left( -\frac{1}{2} \| s - Qb \|^2 \right),$$

and

$$S = \begin{bmatrix} A_1 \\ A_2 \\ \vdots \\ A_N \end{bmatrix} = QRP^T, \quad s = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \end{bmatrix}.$$
Let \( l(x_k, x_m) \) be a likelihood function where \( x_k \in \mathbb{R}^k, x_m \in \mathbb{R}^m, n = k + m \). The marginal likelihood function, \( l(x_k) \), obtained by marginalising out \( x_m \), is

\[
l(x_k) = \int_{\mathbb{R}^m} l(x_k, x_m) dx_m. \tag{3.23}
\]

Now consider that the likelihood is an improper Gaussian

\[
l(x_k, x_m) = \mathcal{G}(x; b, A),
\]

with \( A \in \mathbb{R}^{r \times n} \), where

\[
x = \begin{bmatrix} x_k \\ x_m \end{bmatrix} \quad \text{and} \quad A = \begin{bmatrix} A_k & A_m \end{bmatrix}.
\tag{3.24}
\]

For the integral in (3.24) to be finite, \( A_m \in \mathbb{R}^{r \times m} \) must have no null vectors, meaning it must be of full rank \( m \). This is assumed to be the case. Begin by determining the QRCP factorization of \( A_m \), which gives

\[
A_m = Q_m R_m P_m^T \equiv Q_m A_m',
\tag{3.25}
\]

with \( Q_m \in \mathbb{R}^{r \times m} \) and non-singular \( A_m' \equiv R_m P_m^T \in \mathbb{R}^{m \times m} \), since the rank of \( A_m' \) is \( m \).

Expanding \( x \) into the variables to keep and the variables to marginalize results in

\[
l(x_k) = \int_{\mathbb{R}^m} \exp \left( -\frac{1}{2} \| A x - b \|^2 \right) dx_m
\]

\[
= \int_{\mathbb{R}^m} \exp \left( -\frac{1}{2} \| A_m x_m + A_k x_k - b \|^2 \right) dx_m. \tag{3.27}
\]

Using (B.6), we get

\[
l(x_k) = \int_{\mathbb{R}^m} \exp \left( -\frac{1}{2} \| A_m' x_m + Q_m^T (A_k x_k - b) \|^2 \right) dx_m
\]

\[
\cdot \exp \left( -\frac{1}{2} \| (I - Q_m Q_m^T) (A_k x_k - b) \|^2 \right). \tag{3.28}
\]

Since \( A_m' \) is non-singular, the integral that remains is known: it is the normalization constant of a proper Gaussian density \([3]\). It is known to be \( |\det(A_m')|^{-1} (2\pi)^{\frac{m}{2}} \).
Defining $D = (I - Q_m Q_m^T) A_k$ and $d = (I - Q_m Q_m^T) b$ gives
\[
l(x_k) = |\det(A'_m)|^{-1} (2\pi)^\frac{m}{2} \exp \left( -\frac{1}{2} \|D x_k - d\|^2 \right). \tag{3.29}\]

The exponential term is almost an improper Gaussian as defined in (3.1), except that $D$ might not have linearly independent rows, which is required. This is fixed with a second QRCP factorization, this time of $D$, giving
\[
D = Q_k R_k P_k^T,
\]
where $Q_k^T Q_k = I$ and $R_k P_k^T$ has linearly independent rows. Using (B.6) again, we get
\[
l(x_k) = |\det(A'_m)|^{-1} (2\pi)^\frac{m}{2} \exp \left( -\frac{1}{2} \|D x - d\|^2 \right) \tag{3.30}
= |\det(A'_m)|^{-1} (2\pi)^\frac{m}{2} \exp \left( -\frac{1}{2} \|R_k P_k^T x - Q_k^T d\|^2 \right) \tag{3.31}
\cdot \exp \left( -\frac{1}{2} \|d - Q_k Q_k^T d\|^2 \right)
= |\det(A'_m)|^{-1} (2\pi)^\frac{m}{2} G(x; Q_k^T d, R_k P_k^T) \exp \left( -\frac{1}{2} \|d - Q_k Q_k^T d\|^2 \right). \tag{3.32}
\]

To conclude, the marginal of an improper Gaussian is
\[
\int_{\mathbb{R}^m} G \left( \begin{bmatrix} x_k \\ x_m \end{bmatrix}; [A_k A_m] \right) dx_m = c G(x; b_f, A_f), \tag{3.33}
\]
where
\[
A_f = R_k P_k^T, \quad b_f = Q_k^T (I - Q_m Q_m^T) b \quad \text{and} \quad c = |\det(A'_m)|^{-1} (2\pi)^\frac{m}{2} \exp \left( -\frac{1}{2} \|(I - Q_k Q_k^T)(I - Q_m Q_m^T)b\|^2 \right).
\]
3.2 Improper Gaussian mixtures

Having formulated the improper Gaussian and determined how to calculate products and marginals, we can use it as a building block for a mixture model for likelihoods.

The likelihoods considered in this work are finite linear combinations of improper Gaussians,

\[ l(x) = \sum_i k_i G(x; b_i, A_i), \quad (3.34) \]

where \( k_i > 0 \). The marginal integral is linear, so the marginal of a mixture is just the mixture of the marginals

\[ l(x_a) = \int_{\mathbb{R}^m} l(x_a, x_b) dx_b = \int_{\mathbb{R}^m} \sum_i k_i G \left( \begin{bmatrix} x_a \\ x_b \end{bmatrix}; b_i, A_i \right) dx_b \quad (3.35) \]

\[ = \sum_i k_i \int_{\mathbb{R}^m} G \left( \begin{bmatrix} x_a \\ x_b \end{bmatrix}; b_i, A_i \right) dx_b. \quad (3.36) \]

The marginal of each individual component is obtained according to \((3.33)\).

The product is more problematic, as it causes a quadratic increase in the component number. Consider two likelihoods \( l_A(x) \) and \( l_B(x) \) of \( N_A \) and \( N_B \) components respectively. Their product is

\[ l_A(x)l_B(x) = \sum_{i=1}^{N_A} \sum_{j=1}^{N_B} k_i^A k_j^B G(x; b_i^A, A_i^A) G(x; b_j^B, A_j^B). \quad (3.37) \]

The product of each individual pair of components is obtained according to \((3.22)\). Each product results in one component, so the product has \( N_A \cdot N_B \) components.

3.2.1 Simplification methods

While performing inference with likelihoods that are IG mixtures, as a consequence of frequent products, the number of components will increase exponentially. Methods to reduce this complexity are necessary. Three methods will be discussed here: the pruning of components with small maximum likelihoods, the pruning of products from dissimilar components, and the merging of similar components.
Max pruning

The simplest method to deal with the increasing number of components is to simply remove those whose weights are small. These have the least impact on the total likelihood. To this end, a relative threshold $k_{\text{thr}}$ is defined such that components with weight $k_i$ are removed if

$$k_i \leq k_{\text{thr}} \max_j k_j.$$  

When a mixture’s components are proper probability densities, their mixture weights correspond to probability mass weights, the $L_1$ norm, of the component. This is not the case with IGs. Instead of the $L_1$ norm, the IG is normalized to have a maximum likelihood, the $L_\infty$ norm, of 1. Therefore, the mixture weights $k_i$ are the maximum likelihood of each component, and this technique will be referred to as max pruning.

However, this approach, and all that rely on the weight of the components, can be bad choices to use with the NH if the occurrence of outliers at different times is correlated. The NH model assumes that measurements are independent from each other. If this is not the case, for example if the disturbance that makes a measurement an outlier endures in time to affect the next measurements as well, the calculated weights will be wrong.

In the context of a NH measurement model, an example problematic case is if a single perturbation creates $N$ outlier measurements in a row consistent with each other, in the beginning of the estimation. An example would be a detection by a USBL transceiver of a reflected signal. Pruning components at this time might eliminate the NH component, making it impossible for the algorithm to recover in the future with inlier measurements. In Section 4.4, this scenario is simulated and discussed.

Similarity pruning

It is possible to remove some inconsistent components without using the accumulated weights. The idea here is that, in a likelihood mixture product, components that are products of inconsistent parent components can be pruned.

As a motivating example, consider two components of equal and small weight. The first got this weight from 10 slightly inconsistent measurements, whose likelihoods multiply together to an accumulated low weight. This component can be pruned with max pruning if the measurements are considered independent and multiplying them is valid. However, if the
measurements are not independent, they might all be repeating the same information, meaning that the component should not be pruned.

The second component achieves the same accumulated weight from 9 consistent measurements and a single very inconsistent measurement. The second component can be pruned based solely on the very inconsistent measurement, without assuming the measurements are independent.

This suggests a metric for pruning that is more conservative than max pruning and is a better choice when measurements might not be independent. We can check for big decreases in the maximum likelihood of the product of two components as an indication that they are inconsistent with each other.

The chosen similarity measure is the maximum value of the product. For two improper Gaussians $G_1(x) \equiv G(x; b_1, A_1)$ and $G_2(x) \equiv G(x; b_2, A_2)$, we define the similarity $P(G_1, G_2)$ as

$$0 \leq P(G_1, G_2) \equiv \max_{x^*} G_1(x^*)G_2(x^*) \leq 1.$$ (3.39)

The similarity $P$ is equal to the parameter $c$ from (3.22), so it is already being exactly determined in the process of multiplying components together, incurring no extra computation cost.

A small $P(G_1, G_2)$ similarity means that the two likelihoods are inconsistent everywhere, but a high similarity only means that they agree on at least a single point. A high $P(G_1, G_2)$ similarity does not mean that the likelihoods are similar in general. As an example, the similarity between a uniform IG and any other IG is always 1.

To conclude, similarity pruning occurs when two likelihood mixtures are multiplied together. For each component $i$ in the product, let $p_i$ be the similarity between its parent components. A relative threshold $p_{thr}$ is defined such that components with $p_i$ are removed if

$$p_i \leq p_{thr} \max_j p_j.$$ (3.40)

**RDG groups**

Component merging will use techniques that are normally only applicable to mixtures of distributions. This will be possible due to the RDG interpretation of the IG as an extension of an underlying Gaussian distribution in a specific subspace of $\mathbb{R}^n$. The RDG grouping strategy is adapted from [17].

The fundamental idea is to identify the IGs that can be parameterized into RDGs with the same $Q$ matrix. This is possible if and only if the $A_i$ matrices
have the same rowspace, as $Q$ is an orthonormal basis of that space.

All the components are grouped into RDG groups. In each group $s$, all components are parameterized as RDGs with the same $Q = Q_s$. This allows interpreting the components in each group $s$ as a mixture of Gaussian distributions defined in the subspace spanned by the columns of $Q_s$. The Gaussian distributions can be compared and merged, without changing $Q_s$, effectively applying the same operations to the corresponding RDGs. After merging, all components are converted back to IGs.

To create the groups $s$ and matrices $Q_s$, a simple algorithm can be followed. An empty set of groups $S$ is created. Let the $i$-th improper Gaussian component have parameters $k_i$, $b_i$ and $A_i$. Sequentially, the following is executed for each component. The component is compared with the existing groups $s \in S$, by checking if the columns of $Q_s$ form a basis of $A_i$. This is true if the rank of $[A_i^T \; Q_s]$ is the same as the rank of $A_i$ and $Q_s$. If this is true for no group, a new group $s$ is created with component $i$, with (3.11). If a matching group $s$ exists, then the component $i$ is added to $s$ and its RDG parameters are

$$F = (A_i Q_s)^{-T}, \quad Q = Q_s, \quad \mu = (A_i Q_s)^{-1} b_i,$$

and

$$w = k \left| \det ((2\pi)^{-\frac{1}{2}} A_i Q_s) \right|^{-1}. \quad (3.41)$$

**Component merging**

The third method used to simplify IG mixtures is component merging, applied to each Gaussian distribution mixture of each RDG group. The general algorithm consists of repeatedly comparing all Gaussians pair-wise with some distribution distance measure $D(\mathcal{N}_1, \mathcal{N}_2)$ and merging the most similar pairs. Components are merged until all distances are bigger than a threshold $d_{\text{thr}}$. This general algorithm is adapted from [59], where termination is determined instead by a goal number of components.

By using a minimum distance instead of a maximum number of components, the algorithm dynamically chooses a number of components that reflects the complexity of the mixture. This choice is preferred for experimentation and testing purposes, but real-time implementations might prefer a constant number of components to obtain more consistent calculation times.

Another alternative strategy that should reduce the computational cost is to merge components that are closer than the minimum distance immediately, instead of searching for the closest pair each merge. This was not implemented...
to avoid making the algorithm dependent on an arbitrary search order.

The two remaining design decisions are how to merge and what distance measure to use. The natural substitute for two similar Gaussians is the moment preserving Gaussian \[59\]. The moment preserving substitution of the Gaussian distribution components with parameters \(w_1, \mu_1, \Sigma_1 = F_1^T F_1\) and \(w_2, \mu_2, \Sigma_2 = F_2^T F_2\), respectively, is given by

\[
\begin{align*}
    w &= w_1 + w_2, \\
    \mu &= \frac{w_1 \mu_1 + w_2 \mu_2}{w} \quad \text{and} \\
    F &= Q R_R \begin{bmatrix}
        \sqrt{\frac{w_1}{w}} F_1 \\
        \sqrt{\frac{w_2}{w}} F_2 \\
        \frac{\sqrt{w_1 w_2}}{w} (\mu_1 - \mu_2)^T 
    \end{bmatrix},
\end{align*}
\]  

(3.42)

where \(QR_R(M)\) means the \(R\) matrix from the \(QR\) factorization of \(M\).

The expression for \(F\) in (3.42) is equivalent to

\[
F^T F = \frac{w_1 \Sigma_1 + w_2 \Sigma_2}{w} + \frac{w_1 w_2}{w^2} (\mu_1 - \mu_2)(\mu_1 - \mu_2)^T.
\]  

(3.43)

In order to make choosing a threshold \(d_{thr}\) for merging convenient, a normalized/dimensionless distance measure is desirable. Furthermore, the measure should compare both the \(\mu\) and \(\Sigma\) parameters of the two components. The Hellinger distance satisfies these properties and is the chosen distance.

For \(N_1(x) = N(x; \mu_1, \Sigma_1)\) and \(N_2(x) = N(x; \mu_2, \Sigma_2)\), with \(x \in \mathbb{R}^n\), the Hellinger distance \(0 \leq D(N_1, N_2) \leq 1\) is given by

\[
D^2(N_1, N_2) \equiv 1 - \sqrt{\frac{(8\pi)^{\frac{n}{2}} |\Sigma_1|^{\frac{1}{2}} |\Sigma_2|^{\frac{1}{2}}}{|\Sigma_1 + \Sigma_2|^{\frac{1}{2}}} N(\mu_1; \mu_2, \Sigma_1 + \Sigma_2)}.
\]  

(3.44)
The Gaussian mixture sum-product algorithm
Chapter 3 introduces the GM-SPA, a probabilistic inference method that
gains exact posteriors when the likelihoods are mixtures of improper
Gaussian functions, given unbounded computing power. The number
of mixture components used grows exponentially with the number of
measurements, so approximation methods were added to mitigate the
complexity problem.

In this chapter, three simulated tests are used to evaluate the performance of
the developed method. In Section 4.1, the methods used for comparison with
the GM-SPA are described. Section 4.2 discusses the performance metrics
used to evaluate the simulation results.

The first test, in Section 4.3, aims to determine how accurate and com-
putationally expensive the GM-SPA is at solving uncertain data associations,
with unknown outliers, using the NH model (2.4). The problem considered is
a toy problem of global localization in a 1-D world, designed to be as simple
as possible. It is adapted from the 3 doors problem in [60, Chapter 1].

The second test, in Section 4.4, evaluates the robustness of the NH model
to unmodelled correlation in the outliers. The problem is 3D linear target
tracking with correlated outliers, inspired by acoustic reflections in underwater
environments.

The third and final test, in Section 4.5, evaluates the accuracy and compu-
tational cost of the GM-SPA in a more complex problem: a simplified
AUV homing scenario. The goal is 2D pose estimation from single-beacon
range measurements, velocity measurements, heading measurements, and
angular velocity measurements.

Section 4.6 summarizes the most important results from the three tests.
4.1 Tested methods

To evaluate the proposed method, the GM-SPA, its results are compared with alternative solutions. These methods are discussed in the next paragraphs and are:

- the particle filter (PF);
- the Kalman filter (KF);
- multimodal incremental smoothing and mapping (MM-iSAM);
- multimodal two-filter smoothing (MM-TFS), not mentioned until now;
- and multimodal two-filter smoothing with low variance sampling (MMLV-TFS), a variant of MM-TFS.

The PF is implemented via sequential importance sampling and systematic (also called low-variance) resampling [10]. The full trajectories of each particle are kept for smoothed results.

The KF is a particular case of the GM-SPA, so it requires no particular implementation. The GM-SPA was implemented in code for HMM type graphs only, as the tested examples are all of this type. Expanding the implementation to general trees would be trivial. Applying the same computational techniques to inference on the Bayes Tree should also be possible, which is discussed briefly in Section 5.2.1.

For the MM-iSAM algorithm, its author’s implementation, Caesar [57], is used. This implementation is under development, so its performance is expected to improve in the future.

While MM-iSAM can solve HMM style factor graphs, it is designed to solve more general problems. By restricting the problem set to the HMM, a more accurate and less computationally expensive method can be used. In this work, this method will be called MM-TFS. This should provide a better alternative than direct MM-iSAM. Furthermore, it avoids a known issue* in the current MM-iSAM implementation, where smoothed, but not filtered, results with Gaussian mixture models fail to converge to the correct posteriors.

The fundamental difference between MM-iSAM and MM-TFS is that the latter does not need to use Gibbs sampling to approximate the intraclique integral, because, in a HMM graph, it is simply a prediction convolution. Therefore, it can be approximated directly like in a PF. See Section 2.4.3 for

*https://github.com/JuliaRobotics/IncrementalInference.jl/issues/1765
a very brief discussion of the intraclique integral and [11, Chapter 5.4] for details.

MM-iSAM and its two variants introduced here also differ in how mixture distributions are sampled. MM-iSAM samples the mixture with completely independent samples. MM-TFS and MMLV-TFS sample from mixtures by sampling each component individually with a number of samples proportional to their probability mass weight. For each component, MM-TFS takes independent samples while MMLV-TFS uses a correlated and low-variance set of samples.

The goal with MMLV-TFS is to obtain accurate particle representations with a smaller number of particles. For scalar distributions, MMLV-TFS’s low variance sampling can easily be implemented by taking equidistant quantiles as the samples. Due to time constraints, this method is not extended to multivariate distributions.

The code used to produce the results in this thesis is available online*.

### 4.2 Performance metrics

The evaluation of the simpler results from the simulation tests can be done by visual inspection of the obtained estimates and belief densities. To obtain precise conclusions, quantitative measures of performance are necessary. The state estimates, $\hat{x}_t$, are obtained by maximizing the belief densities, $\hat{p}_t(x)$, according to

$$
\hat{x}_t = \arg\max_x \hat{p}_t(x). \quad (4.1)
$$

From the beliefs and estimates, the following performance metrics are considered:

- the position root mean square error (PRMSE),
- the heading root mean square error (HRMSE),
- the run-time,
- and the data association accuracy.

The PRMSE and HRMSE take the root mean square of the errors along the trajectory to achieve a compromise between consistency and sensitivity. The errors here are the norms of the vectors between the real states and the

*https://github.com/itsDionix/gmspaMasterThesis
estimates. Using the average error would allow for estimated trajectories with mostly small errors and a single big error to obtain a low average, which is not desired. Since consistently small errors are preferred, one could consider the maximum error instead as a performance metric. However, the maximum error suffers from a lack of sensitivity, because an estimated trajectory with a single big error would be classified the same as another where all errors are as big. The root mean square error provides a compromise between these two perspectives, punishing bigger errors more while also being sensitive to all errors.

Considering $N$ real positions $p_t$ and estimates $\hat{p}_t$, each at time instants $t = t(i)$, the PRMSE is given by

$$\text{PRMSE} = \sqrt{\frac{1}{N} \sum_{i=1}^{N} \| p_{t(i)} - \hat{p}_{t(i)} \|^2}.$$  \hspace{1cm} (4.2)

Likewise, for real headings $\theta_{t(i)}$ and estimates $\hat{\theta}_{t(i)}$, the HRMSE is given by

$$\text{HRMSE} = \sqrt{\frac{1}{N} \sum_{i=1}^{N} | \theta_{t(i)} - \hat{\theta}_{t(i)} |^2}.$$  \hspace{1cm} (4.3)

The run-time is defined as the time needed to compute both the filtered and the smoothed estimates.

The data association accuracy aims to measure how accurately can data association decisions be made from the obtained state estimates. This performance metric is only considered in the 3 doors problem, in Section 4.3, where the state measurements, given by (4.8), have 4 components, one per door and one for the NH.

The methods considered do not make an explicit choice of data association. However, once a state estimate has been determined, it can be used to make a data association decision on the corresponding state measurements, by choosing the most likely hypothesis. For example, if a door component has the highest likelihood for the state estimate when compared to the other components, the measurement is classified as an inlier from that door. If the NH component has the highest likelihood, the measurement is classified as an outlier.

For each position with a state measurement, the data association accuracy is given as the number of simulations, out of the total of 101, in which the correct classification was made. For example, suppose a given method has
a smoothed data association accuracy of 87 for a measurement $z_1$. That means that the classification from the smoothed state estimate of that method classified $z_1$ correctly 87 out of 101 times.

4.3 The 3 doors example

This section considers a toy problem of global localization in a 1-D world to showcase the algorithm as simply as possible. This is an adaptation of the 3 doors example from [60, Chapter 1]. Figure 4.1 represents the robot’s initial position and the three doors.

![Figure 4.1: Representation of the 3 doors problem environment. Adapted from [60, Figure 1.1].](image)

4.3.1 Problem description

In this problem, the world is a single 1-dimensional line where a robot moves. Along this line, there are three identical doors placed in fixed, exactly known locations: $L_1 = -10, L_2 = -5, \text{and } L_3 = 10$. The robot moves along the line and can detect when it is in front of a door.

The simulated door detector has 3 imperfections. Firstly, it is noisy, so it measures

$$z_t = L - x_t + n_t,$$

(4.4)

where $L$ is a door position, $x_t$ is the robot’s position and $n_t$ is random zero-mean Gaussian noise with $\sigma = 0.5$. The second imperfection is that it cannot distinguish the doors, so it is not known which door generated the measurement. The third imperfection is that it sometimes gives an outlier detection that indicates a door where there is none.

In every run of the simulation, the outlier detection happens only in the first of the four measurements taken. The outlier measurement $z_1$ is generated as if there was a door right in front of the robot, setting $L = x_1$. The other
three measurements all correctly measure the door closest to the robot with (4.4).

The robot also has access to noisy odometry measurements. These tell it how much it moved in between two positions $x_{t+1}$ and $x_t$ by measuring

$$ s_t = x_{t+1} - x_t + v_t, $$

(4.5)

where $v_t$ is random zero-mean Gaussian noise with $\sigma = 1$.

The choices of which measurements are outliers, of which door is detected, and the movement of the robot are fixed to be the same for every simulation. Only the Gaussian noises in (4.4) and (4.5) are randomized in each run of the simulation. This ensures that the same scenario is considered in every run. In this scenario, the following events happen.

- At first, the robot is at $x_1 = -15$.
- The robot incorrectly detects a door as if there was one in its current position. This is the outlier measurement $z_1$, generated from (4.4) with $L = x_1$.
- It moves to $x_2 = -10$ and measures $s_1$.
- It correctly detects the door in front, measuring $z_2$ with $L = L_1$.
- It moves to $x_3 = -5$ and measures $s_2$.
- It correctly detects the door in front, measuring $z_3$ with $L = L_2$.
- It moves to $x_4 = 0$ and measures $s_3$.
- It moves to $x_5 = 5$ and measures $s_4$.
- It moves to $x_6 = 10$ and measures $s_5$.
- It correctly detects the door in front, measuring $z_6$ with $L = L_3$.

The final factor graph is represented in Figure 4.2.

What is interesting about the chosen trajectory is that, up to the first two door detections, all detections are consistent with the map, even though one is incorrect. The third detection, $z_3$, reveals that some detection must be wrong, but multiple possibilities explain two out of three landmarks. Only the final detection, $z_6$, reveals the real position as the only one that makes three out of four detections consistent with the map. The difficulty, then, is that multiple hypotheses must be considered simultaneously until the final measurement clears the ambiguity.
4.3.2 Methods description

This section describes the models, methods, and parameters used to solve the 3 doors problem.

To deal with the unknown data association, the likelihood on $x$ from a correct door detection has a term for each door. The inlier model that the robot uses for door detections is

$$p_{in}(z|x) = \sum_{i=1}^{3} \mathcal{N}(z; L_i - x, 0.5^2) \propto \sum_{i=1}^{3} \exp \left( -\frac{1}{2} \frac{(z + x - L_i)^2}{0.5^2} \right).$$

(4.6)

To account for failures of the sensor, the robot attributes a 0.1 probability to the null hypothesis, defined in (2.4). This results in the complete measurement likelihood for a door detection as

$$l_z(x) \propto 0.9 p_{in}(z|x) + 0.1 \max_{x^*} p_{in}(z|x^*),$$

(4.7)

which normalized to have a maximum of 1 is approximately

$$l_z(x) \approx 0.9 \sum_{i=1}^{3} \exp \left( -\frac{1}{2} \frac{(x - L_i)^2}{0.5^2} \right) + 0.1.$$  

(4.8)

The MM-iSAM based methods cannot sample from the infinite uniform component in (4.8), so instead a uniform with the same value in the range $x \in [-50, 50]$ is used.

The odometry factors model the odometry sensor exactly, resulting in the factor

$$l_{st}(x_t, x_{t+1}) = \exp \left( -\frac{1}{2} \frac{s_t + x_t - x_{t+1}}{1} \right)^2. $$

(4.9)
The proposed GM-SPA is tested both with and without its approximation methods. The exact solution can be obtained since there are only 4 multi-component factors of 4 components each, resulting in final posteriors with a total of $4^4 = 256$ components. The approximation methods used are max pruning and component merging, using $k_{thr} = 10^{-3}$ and $d_{thr} = 0.5$. When discussing the results of this simulation, these methods are referred to as Exact GM-SPA and GM-SPA, respectively.

MM-TFS, MMLV-TFS, and MM-iSAM are also used, using $N = 500$ particles.

As a limit of the ideal performance, a simpler problem is also tested. In this ideal version, the $z_t$ measurements identify which door they come from, so data association is known. Furthermore, the outlier measurements are also known, so $z_1$ is not used. The result is an inference problem that can be solved exactly with a KF. When discussing the statistical simulation results, this method is labeled as the Ideal KF.

### 4.3.3 Exact solution

We begin by looking at the marginal beliefs resulting from solving the factor graph exactly with the GM-SPA, without any heuristic to reduce the number of components. Figure 4.3 presents the belief at some key steps. Approximate beliefs obtained with MMLV-TFS are plotted in the same figure, for comparison.

Figure 4.3a presents the belief from the first, incorrect, door detection. Notice the uniform component of the likelihood, which comes from the null hypothesis. Compared to the exact posterior, MMLV-TFS’s modes have bigger variances.

Figure 4.3b shows the result of applying a prediction step, where the belief is simply moved to the right and the components become less precise, due to the odometry uncertainty. The uniform component in the MMLV-TFS belief becomes visibly noisy, due to the stochastic approximation of the prediction convolution.

Figure 4.3c represents the densities after adding the information from a new door detection, which complicates the belief and increases the number of components visibly. Note that the dominant node is not the true position. $x_2 \approx -5$ is the most likely position with the information available so far, because it explains both door detections, while the real position only explains one. Importantly, both methods preserve the secondary modes, including the one corresponding to the real position at $x_2 \approx -10$. 
Figure 4.3: Exact GM-SPA’s and MMLV-TFS’s filtered and predicted beliefs for selected positions. Also shows the positions of the robot, the doors, and the door sightings until that point.

**Figure 4.3d** presents the belief of the final position with all the information. The proposed method recovers the correct position with confidence, as the secondary modes are roughly 10 times smaller. The final mixture has a total of 256 components, but it clearly can be well approximated with fewer Gaussians.

MMLV-TFS’s solution also recovers the correct mode as the most likely. Some secondary modes are also present but roughly approximated. The difference between the dominating and the secondary modes is much smaller than in the exact belief. This smaller margin indicates that in other runs of the simulation, there is a higher chance that a wrong mode becomes dominant.
4.3.4 Approximate beliefs

This section presents beliefs obtained with the approximated methods. Approximation is necessary for bigger problems, due to the exponential growth of mixture components.

The effect of component reduction approximations in the GM-SPA from Section 3.2.1 is presented first. Figure 4.4 is a plot of the belief over the initial position $x_1$, given all the information. It shows the exact solution and the approximated solution with the GM-SPA. The approximation used max pruning with $k_{thr} = 10^{-3}$ and component merging with $d_{thr} = 0.5$.

The approximation techniques slightly reduced the likelihood of the principal modes and left the visible smaller modes unaffected. Despite being a close approximation, max pruning and component merging reduced the number of components by a factor of 25.6, from 256 to 10 components.

The filtered and smoothed beliefs from MM-iSAM, MM-TFS, and MMLV-TFS for $x_1$ are represented in Figure 4.5 and Figure 4.6.

![Smoothed $x_1$ belief](image)

Figure 4.4: Smoothed $x_1$ belief obtained with the exact and approximated GM-SPA. Also shows the positions of the robot, the doors, and the door sightings until that point.
Figure 4.5: Filtered $x_1$ belief obtained with the three methods based on MM-iSAM. Also shows the positions of the robot, the doors, and the door sightings until that point.

Figure 4.6: Smoothed $x_1$ belief obtained with the three methods based on MM-iSAM. Also shows the positions of the robot, the doors, and the door sightings until that point.
In Figure 4.5, MMLV-TFS is the only method with a constant uniform component, due to the low variance sampling. The other two methods are visibly noisy. MMLV-TFS also has the desired property of representing the three modes from the three doors with the same maximum likelihood. Both MM-TFS and MM-iSAM give different maximum likelihoods to each door, due to stochastic sampling. The difference is bigger for MM-iSAM because its samples are completely independent.

In Figure 4.6, MMLV-TFS would recover the correct position with a maximum likelihood estimate, while the other two methods would not. Even MMLV-TFS has a small margin between the maximum of the correct mode and the second biggest mode. MM-iSAM’s belief stands out because it only preserves a single mode, which is incorrect. This problem is present here and not in the filtered belief in Figure 4.5.

While MMLV-TFS presents the best performance out of the three MM-iSAM based methods, none of them get as close to the true posteriors as the GM-SPA, plotted in Figure 4.4.

### 4.3.5 Position errors

This section analyses the statistical distribution of the position root mean square error (PRMSE), obtained from 101 simulations. Box plots of the PRMSE of the filtered and smoothed beliefs from each method are present in Figure 4.7.

In the filtered case, the ideal KF’s median error is roughly 5 times smaller than the median error of all the other methods, which do not have very different results from each other. This is due to it using known data correspondences, while the other methods do not. The higher error of the other methods is a result of the ambiguity in the data. When there is not enough data, no method performs well.

In the smoothed case, all methods except MM-iSAM improve the PRMSE, especially the exact and approximate GM-SPA. These two methods achieve a median error comparable with that of the ideal KF. As such, it is to be expected that they correctly solve the data ambiguity most of the time. They still have worse worst-case errors than the ideal KF, which suggests that the data ambiguity is not correctly solved every time.

MM-TFS and MMLV-TFS also improve when compared to the filtered results. MMLV-TFS has half of MM-TFS’s median error, and double of the ideal KF’s median error.
The median error of MM-iSAM is the worst of all methods. It is important to note that it does not significantly improve when compared to the filtered case, despite being based on more information. This is explained by the fact that, as previously discussed in Section 4.1, the Caesar implementation of MM-iSAM currently often fails to converge to the correct posteriors when smoothing with Gaussian mixture noise.

### 4.3.6 Data association accuracy

Figure 4.8 presents the number of simulations, in the total of 101, that resulted in a correct implicit data association, for each measurement and method.
In the filtered case, $z_1$ is not considered because it is the first measurement and there is no other information to judge $z_1$ with. In the smoothed case, $z_6$ is omitted because it is the final measurement, so the smoothed results are the same as the filtered results.

The results for the filtered accuracy are as expected from the design of the simulation. The accuracy at $z_2$ is near 0 for all methods, much worse than random, since at that point the correct hypothesis is not the one that explains the most measurements as inliers. For $z_3$, the accuracy is between 50% and 90% for all methods, which is a consequence of there being two hypotheses that explain 2 out of the 3 measurements so far as inliers. With the last measurement, the ambiguity is removed and all methods result in a near-perfect accuracy.

Regarding the accuracy obtained from smoothed beliefs, the exact and approximated GM-SPA obtain a high accuracy for all measurements. The accuracy is lower for $z_1$ because it is furthest away in time from the measurement that eliminates ambiguity, $z_6$. The odometry uncertainty accumulates at each movement, leading to some wrong state estimates for $x_1$. MMLV-TFS and MM-TFS are the next best two methods. They suffer more from the decrease in accuracy in earlier measurements but still improve when compared to the filtered accuracy.

As with the PRMSE, the smoothed accuracies from MM-iSAM are significantly worse than all other methods, and at $z_3$ it is even worse than the filtered MM-iSAM accuracy.

In this test, the GM-SPA obtains the most accurate results, especially when smoothing is considered. The component reduction techniques do not affect the accuracy negatively. The three MM-iSAM based methods perform similarly in the case of filtering, but MM-iSAM is much less accurate in the case of smoothing.

### 4.3.7 Run-time

Figure 4.9 represents the run-time that each method took to compute the filtered and smoothed posterior beliefs.

The biggest divide is between the MM-iSAM based methods and other methods. The former are three orders of magnitude slower than the latter. Between the MM-iSAM based methods, MM-iSAM is the slowest. The median run-times are 35 seconds for both MMLV-TFS and MM-TFS, and 162 seconds for MM-iSAM.

The KF is, naturally, the fastest, with a median run-time of 4 ms. The
exact and approximated GM-SPA have median run-times of 32 ms and 18 ms respectively.

![Figure 4.9: Box plots of run-time for the ideal KF and the GM-SPA methods in milliseconds (a), and for the MM-iSAM based methods in seconds (b).](image)

### 4.4 Target tracking with correlated reflections

This section considers the problem of tracking a target moving in a straight line in a 3D underwater environment, from USBL position measurements. The USBL measurements are occasionally disturbed with a reflection across a horizontal plane, which represents a cline†.

Figure 4.10 represents the direct and indirect paths between the target and the beacon with the USBL transceiver. The disturbances last in time through multiple measurements, which means that the property of measurements being outliers is correlated between consecutive measurements. This is meant to test the NH in the presence of unmodelled correlation in the outliers.

#### 4.4.1 Problem description

A target AUV is moving in a straight line in 3D space, from $(-10, 2, 3)$ to $(10, 2, 3)$, with a constant velocity $v \equiv (v_x, v_y, v_z) = (0.4, 0, 0)$. A USBL beacon is located at the origin $(0, 0, 0)$. Every second, a USBL measurement $\tilde{p}_t \equiv (\tilde{x}_t, \tilde{y}_t, \tilde{z}_t)$ measures the 3D position $p_t \equiv (x_t, y_t, z_t)$.

†In underwater environments, a cline is a layer of water with a sharp change of some physical property, such as temperature, which can act as a barrier for sound waves and create reflections.
Figure 4.10: Diagram (not to scale) of direct and reflected paths that an acoustic signal can take, between the underwater beacon and the AUV target, in the simulation. Also represents real (opaque) and apparent (translucid) positions of the AUV.

The measurements $\tilde{p}_t$ are affected by Gaussian noise in spherical coordinates, centered at the origin, and occasionally by a reflection disturbance.

The transformation to spherical coordinates is $S$, defined as

$$S(p) \equiv \begin{bmatrix} r \\ \theta \\ \varphi \end{bmatrix} = \begin{bmatrix} \|p\| \\ \arctan2(y, x) \\ \arccos(z/\|p\|) \end{bmatrix}.$$ (4.10)

These coordinates are not defined in the $z$-axis, but the trajectory never crosses the $z$-axis so there is no problem.

The measurement before being disturbed is written as $\tilde{p}_t^{\text{inlier}}$ and determined by

$$\tilde{p}_t^{\text{inlier}} = S^{-1}(S(p_t) + \eta_t),$$ (4.11)

where $\eta_t$ is Gaussian noise with independent components. The standard deviations are $\sigma_r = 0.1$, $\sigma_\theta = \sigma_\varphi = 1 \text{ deg} = \pi/180 \text{ rad}$.

The final measurements $\hat{p}_t$ are determined from $\tilde{p}_t^{\text{inlier}}$ by possibly reflecting across a horizontal plane cline at $z = 5$. What determines if each $\hat{p}_t$ is reflected or not is a discrete variable $d_t$. Each $d_t$ is either true (1) or false (0). When it is true, the corresponding $\hat{p}_t$ is reflected. Measurements for which $d_t = 1$ are outliers and the others are inliers.
What makes the measurements correlated is that $d_t$ evolves over time according to the probabilities

$$
p(d_{t+1} = 1|d_t = 0) = 0.075, \quad p(d_{t+1} = 0|d_t = 1) = 0.125. \quad (4.12)
$$

These probabilities lead to an average of 8 consecutive measurements with outliers and an average of approximately 13.3 consecutive measurements without outliers. The resulting trajectory and measurements are plotted in Figure 4.11.

There are 29 inlier measurements and 22 outlier measurements, for a total of 51 measurements, amounting to 43 % of outliers. This particular set of measurements is difficult because, in the first quarter of the trajectory, outliers are more common than inliers. Approximate inference methods can make the mistake of prematurely deciding that the correct depth is $z \approx 7$.

---

### 4.4.2 Methods description

The target’s state is modeled as a time-varying $x_t = \begin{bmatrix} p_t \\ v_t \end{bmatrix}$, to allow a tree factor graph representation. The fact that the velocity is constant over time is approximately modeled with a small covariance in the prediction model.

The prediction model assumes a constant acceleration between $x_{t+1}$ and $x_t$, resulting in

$$
\begin{bmatrix}
    p_{t+1} \\
    v_{t+1}
\end{bmatrix} = \begin{bmatrix} p_t + \Delta t \frac{v_{t+1} + v_t}{2} + w_t \\ v_t + \Delta t a_t \end{bmatrix}, \quad (4.13)
$$
where $\Delta t = 1$, $w_t$ is Gaussian noise with covariance $I \cdot 10^{-4}$ and $a_t$ is the unknown acceleration. The acceleration is modeled as a zero mean Gaussian variable with covariance $I \cdot 10^{-6}$. $I$ is the $3 \times 3$ identity matrix.

Two methods are used in this problem, the GM-SPA and MM-TFS, and they model the position measurements differently. MM-TFS is chosen because the results from the 3 doors example, in Section 4.3, indicate it is faster and more accurate than MM-iSAM when smoothing. MMLV-TFS is not tested because it is only implemented for scalar problems.

The noise in (4.11) is not Gaussian in Cartesian coordinates. For the GM-SPA, it has to be approximated. This is done with a single Gaussian, obtained by linearizing the coordinate transformation. The linearization technique is commonly used to approximate non-linear measurements of Gaussian noise with a Gaussian likelihood, for example in the EKF.

Given a Gaussian variable $X$ with covariance $\Sigma$, the output of a linear transformation $M X$ is also Gaussian, and it has covariance $M \Sigma M^T$. If $A = \sqrt{\Sigma^{-1}}$ is the square-root information matrix of $X$, the square-root information matrix of the output is $A M^{-1}$.

In our case, the linear transformation is the linearization of the inverse of $S (4.10)$, at $S(\tilde{p}_t)$. Instead of calculating the Jacobian of $S^{-1}$ and then inverting it, we can calculate the Jacobian of $S$ directly, which gives the same result. In terms of $p$ and $\rho = \| [x, y]^T \|$, the Jacobian of $S$ is given as

$$J(p) = \begin{bmatrix}
x/\|p\| & y/\|p\| & z/\|p\| \\
y/\rho^2 - x/\rho^2 & 0 \\
xz/\rho - yz/\rho & \rho/\|p\|^2 - \rho/\|p\|^2 \\
xz/\rho - yz/\rho & \rho/\|p\|^2 - \rho/\|p\|^2
\end{bmatrix}.$$ (4.14)

Since $\tilde{p}_t$ is never on the $z$ axis, $S$ is continuously differentiable locally around $\tilde{p}_t$ and $J(\tilde{p}_t)$ is invertible. In these conditions, the inverse function theorem [61, p. 223] guarantees that the Jacobian of the inverse is the inverse of the Jacobian. Therefore, $M^{-1} = J(\tilde{p}_t)$.

Consider that the noise $\eta_t$ in (4.11) has a square-root information matrix $R$. Propagating this noise to Cartesian coordinates with a linearized $S^{-1}$ at $S(\tilde{p}_t)$ results in a Gaussian distribution for $\tilde{p}^\text{inlier}_t$. The resulting distribution $p(\tilde{p}^\text{inlier}_t|p_t)$ is a Gaussian with mean $\tilde{p}_t$ and square-root information matrix $R J(\tilde{p}_t)$.

To deal with the possibility of outliers, the NH is used, with a probability of outliers of $P_o = 0.1$. Based on the linearized inlier model and the NH, the
The measurement likelihood used by the GM-SPA is

\[ l_{\hat{p}}(x_t) = 0.9G(p_t; RJ(\hat{p}_t)p_t, RJ(\hat{p}_t)) + 0.1 \]

(4.15)

\[ = 0.9G(x_t; RJ(\hat{p}_t)p_t, RJ(\hat{p}_t)B) + 0.1 \]

(4.16)

\[ = 0.9 \exp \left( -\frac{1}{2} \| RJ(\hat{p}_t)Bx_t - RJ(\hat{p}_t)p_t \|^2 \right) + 0.1, \]

(4.17)

where \( B \equiv [I \ 0_{3 \times 3}] \). \( I \) is the identity matrix and \( 0_{3 \times 3} \) a matrix of zeros, so \( p_t = Bx_t \).

MM-TFS can sample the Gaussian term in (4.11) in spherical coordinates and then convert each sample to Cartesian coordinates. There is no need to approximate the noise distribution like it is done for the GM-SPA.

The uniform component of the NH cannot be used because it cannot sampled. An alternative would be to restrict the infinite uniform to a finite uniform, but this also leads to a problem.

Consider for example replacing the infinite uniform component of (4.15) with a finite uniform that is 0.1 when \( z \in [2.5, 7.5] \), \( x \in [\hat{x}_t - 2.5, \hat{x}_t + 2.5] \) and \( y \in [\hat{y}_t - 2.5, \hat{y}_t + 2.5] \), and 0 elsewhere. This leads to a uniform component with a probability mass weight of \( w_{\text{nh}} = 0.1 \cdot 5^3 = 12.5 \). The probability mass weight of the Gaussian component in (4.15) is \( w_{\text{inlier}} = \sqrt{2\pi^3} |\det(RJ(\hat{p}_t))|^{-1} \).

Note that \( |\det(J(\hat{p}_t))|^{-1} = \rho r \) and \( |\det(R)|^{-1} = \sigma_r \sigma_\theta \sigma_\phi \). For \( r \approx 10 \) and \( \rho \approx 10 \), the weight is

\[ w_{\text{inlier}} = \rho r 2^{3/2} \pi^{3/2} \cdot 0.1 \cdot 180^{-2} \approx 0.05. \]

(4.18)

The weights \( w_{\text{nh}} = 12.5 \) and \( w_{\text{inlier}} = 0.05 \) mean that the inlier component has \( w_{\text{inlier}} / (w_{\text{nh}} + w_{\text{inlier}}) \approx 0.4 \% \) of the probability. This is problematic because in MM-iSAM’s representation, all samples have the same weight. Consequently, using 500 particles, only around 2 are used in the inlier model, while all the rest are used for the uniform component. The uniform NH is very expensive to represent in the MM-iSAM based methods.

To simplify the computations, MM-TFS uses a model with the knowledge of the reflections. The outlier component is replaced with a copy of the inlier model likelihood reflected across the cline plane.

The complete model for \( \hat{p}_t \) in MM-TFS starts by sampling \( \hat{p}_{t,\text{inlier}} \) exactly from (4.11) and then applying the reflection for \( P_o = 10 \% \) of the samples.

To conclude, both the GM-SPA and MM-TFS use a mixture likelihood model with inlier and outlier components, normalized such that their maximum likelihoods are 0.9 and 0.1 respectively. The inlier model in the
GM-SPA is an EKF style linearized approximation of (4.11) in Cartesian coordinates, while MM-TFS uses the exact model. The outlier model in the GM-SPA is the infinite uniform component while for MM-TFS it is a duplicate of the inlier model, reflected across $z = 5$.

MM-TFS benefits from knowing how the outliers are reflected, while the GM-SPA has no information on how the outliers are generated. Both methods assume that measurements are independent, which is wrong due to (4.12).

### 4.4.3 Estimated trajectories

This section analyses the robustness of the considered methods by evaluating their estimates visually, to check if the methods can determine the correct depth of the target.

In this problem, the GM-SPA is always run with component merging, with $d_{\text{thr}} = 0.5$. MM-TFS is run with $N = 500$ particles.

Figure 4.12 shows the estimated trajectory based on maximizing the filtered beliefs obtained with the GM-SPA, using component merging and max pruning, with $k_{\text{thr}} = 10^{-3}$.

![Figure 4.12: Filtered trajectory from the GM-SPA with component merging and max pruning.](image_url)

In the beginning, the filtered estimate oscillates up and down as new measurements are obtained, as expected, but then settles in the incorrect depth, despite most measurements supporting the correct depth. This is explained by the components with the correct $z$ and the uniform component being pruned.

Manual inspection of the beliefs, not visible in Figure 4.12, confirms that after measurement 16 at $x = -4$, the filtered mixture no longer has any uniform component or component with $z \approx 3$. The smoothed estimates all have the incorrect $z$, as plotted in Figure 4.13.
The behavior of pruning all components without \( z \approx 7 \) can be interpreted as the algorithm deciding that it received enough measurements with \( z \approx 7 \) to exclude other hypotheses. This happens because the outliers are modeled as being independent, so having many consecutive consistent outliers would be very unlikely.

To mitigate the problem, one can increase the outlier probability used in the NH model or decrease the threshold \( k_{\text{thr}} \). Changing the threshold to \( 10^{-6} \) fixes the problem, resulting in filtered beliefs with components in both hypotheses throughout the entire simulation and a correct smoothed trajectory.

The downside of decreasing \( k_{\text{thr}} \) is that the number of components increases. To address this, similarity pruning can be used. Since it does not use the accumulated weights, it should not be negatively affected by the correlation in the outliers. The resulting filtered estimates are displayed in Figure 4.14 and the smoothed estimates in Figure 4.15.
Figure 4.15: Smoothed trajectory from the GM-SPA with component merging, max pruning, and similarity pruning.

The filtered estimate oscillates up and down as new measurements are obtained, as expected, according to which hypothesis has the most supporting evidence. The filtered estimates stay responsive to the new information throughout the entire simulation. The smoothed estimates correctly estimate every position at $z \approx 3$.

The median run-time from 11 runs of each of the three sets of GM-SPA parameters are: 1.1 s for $k_{\text{thr}} = 10^{-3}$; 6.7 s for $k_{\text{thr}} = 10^{-6}$; and 5.5 s for $k_{\text{thr}} = 10^{-3}, p_{\text{thr}} = 0.1$. All three cases use $d_{\text{thr}} = 0.5$.

Figure 4.16 shows the estimated trajectory based on maximizing the filtered beliefs obtained from MM-TFS with 500 particles. The estimates from the smoothed beliefs are plotted in Figure 4.17.

Figure 4.16: Filtered trajectory from MM-TFS with 500 particles.

Like the GM-SPA with $k_{\text{thr}} = 10^{-3}$, MM-TFS makes a premature definitive decision about $z$, although it happens to choose the right hypothesis. It fails to update the choice to new information, for example, at $x = -4$ when most measurements until that point indicate that $z \approx 7$. 
The smoothed estimates are noisy and occupy mainly the space in between the two depths with measurements. This is explained by the forward-filtered belief committing to $z \approx 3$ and the back-propagated (not plotted for brevity) belief committing to $z \approx 7$. Both of these beliefs have abandoned the possibility that they might have chosen the wrong depth, so they don’t keep any particles on the opposite depth. Therefore, their product is highest in the middle.

The MM-TFS run-time of the run that produced the plotted results is 25 minutes.

### 4.5 2D range-only pose estimation

This section considers a simplified AUV homing scenario, where the AUV would want to navigate to a beacon, for example for docking. Figure 4.18 represents the AUV’s initial position and the beacon, viewed from above.

The control aspect is not considered, so the control inputs of the AUV are fixed. The problem is estimating the vehicle’s 2D pose, meaning the position and heading. The simplification of the problem as two-dimensional is possible under the following assumptions.

It is assumed, as is common in practice, that the depth of the AUV is accurately measured with pressure sensors [7, Chapter 14]. Further assuming that the beacon’s depth is fixed and known, the depth dimension can be removed from the estimation problem.

Regarding the 3D orientation, it is common for an attitude-heading reference system (AHRS) to determine the roll and pitch angles, since gravity provides a reference for the down direction [7, Chapter 14]. Assuming these angles are accurately determined, they can also be removed from the
estimation problem, resulting in the simplified 2D pose estimation problem.

Figure 4.18: Diagram (not to scale) of the AUV in the initial position and the range beacon, viewed from above. A range measurement is represented with the dashed line.

The sensors used in the simulation give linear and angular velocity measurements in the body frame of the vehicle, heading measurements, and range measurements to the beacon landmark. No prior information on the location of the AUV is available.

The noise is Gaussian for all measurements except for the range measurements. This noise is a Gaussian mixture of an inlier component and an outlier component.

The difficult aspects of this problem are the non-linear range measurements and the fact that the velocity measurements are in the body frame. As the range measurements are under-constrained, a precise location estimate is only possible to obtain by fusing information from multiple measurements from different positions.

### 4.5.1 Problem description

An AUV is simulated moving in 2D space. Its state at time $t$ can be naturally expressed as

$$
\mathbf{x}_t \equiv \begin{bmatrix} p_t \\ \theta_t \\ v_t \\ \omega_t \end{bmatrix},
$$

(4.19)
where \( \mathbf{p} \equiv \begin{bmatrix} x \\ y \end{bmatrix} \in \mathbb{R}^2 \) is the position, \( \theta \in SO(2) \) is the heading, \( \mathbf{v} \equiv \begin{bmatrix} v_x \\ v_y \end{bmatrix} \in \mathbb{R}^2 \), and \( \omega \in \mathbb{R} \).

To simplify the notation and simulation, the use of the \( SO(2) \) manifold can be replaced by instead defining the heading \( \theta \) as an angle in radians that corresponds to the same orientation. The total state is then a 6-dimensional real vector. The initial state \( x_0 \) is zero for all variables except \( \theta_0 = \pi/2 \) rad.

Unless specified otherwise, angular quantities are assumed to be in radians. The other simulated quantities are in an arbitrary normalized system of units, and so their units are omitted.

The vehicle is controlled with a thrust acceleration \( a_t \), applied in the heading direction

\[
\mathbf{e}_{\theta_t} = \begin{bmatrix} \cos(\theta_t) \\ \sin(\theta_t) \end{bmatrix},
\]

and with a steering angular acceleration \( \alpha_t \). The dynamics also include drag and small Gaussian disturbances. They are simulated with Euler’s method for a time step of \( \delta t = 0.01 \) as

\[
\mathbf{x}_{t+\delta t} - \mathbf{x}_t = \delta t \begin{bmatrix} \mathbf{v}_t \\ \omega_t \\ a_t \mathbf{e}_{\theta_t} - \mathbf{v}_t + \mathbf{d}_t^a \\ \alpha_t - \omega_t + \mathbf{d}_t^\alpha \end{bmatrix},
\]

where \( \mathbf{d}_t^a \) and \( \mathbf{d}_t^\alpha \) are Gaussian disturbances, the first with covariance matrix \( I \cdot 0.25 \) and the second with standard deviation of \( \frac{\pi}{180} \). The estimation algorithms do not have access to the control inputs, so the relatively small disturbances make the trajectory slightly less smooth, but not any less predictable.

The input thrust and angular accelerations are fixed to be the same every simulation run and have maximum absolute values of 10 and \( \pi \), much bigger than the disturbances. The resulting trajectory is plotted in Figure 4.19 and takes 20 units of time.

The velocities are measured every 0.2 units of time. The linear velocity is measured in the body frame. Let \( R(\theta) \) be the rotation matrix that rotates by \( \theta \) in the positive direction. Each velocities measurement \( \mathbf{s}_t \) is given by

\[
\mathbf{s}_t = \begin{bmatrix} R(-\theta_t)\mathbf{v}_t + \mathbf{n}_t^v \\ \omega_t + \mathbf{n}_t^\omega \end{bmatrix},
\]

where \( \mathbf{n}_t^v \) and \( \mathbf{n}_t^\omega \) are Gaussian noises, the first with covariance \( I \) and the
second with standard deviation $0.1 \cdot |\omega_t| + \frac{5\pi}{180}$.

Figure 4.19: Example vehicle trajectory and range measurements (left) and vehicle heading and heading measurements (right).

The heading measurements $\tilde{\theta}_t$ are obtained every 2 units of time as

$$\tilde{\theta}_t = \theta_t + n^\theta_t,$$

(4.23)

where $n^\theta_t$ is Gaussian with a standard deviation of $\frac{5\pi}{180}$.

Every 1 unit of time, the range measurements $\tilde{r}_t$ are measured from the beacon at $p_B = [-10 \ 10]^T$ and given by

$$\tilde{r}_t = \|p_t - p_B\| + n^r_t,$$

(4.24)

where $n^r_t$ is Gaussian mixture noise. It has an inlier component, with a probability of 0.8 and a standard deviation of 0.5, and an outlier component, with a probability of 0.2 and a standard deviation of 5.

**4.5.2 Methods description**

This problem was used to test the GM-SPA, MM-TFS, and the PF. With the GM-SPA, the vehicle state is modeled as a vector $x_t \in \mathbb{R}^6$, while with
MM-TFS and the PF, the heading is represented as $\theta \in SO(2)$, since these methods can work with this manifold directly.

A state $x_t$ is created and added to the factor graph for inference whenever it has a measurement on it. As all the sensors are synchronized, this results in a new pose $x_t$ every 0.2 units of time. All poses have velocity measurements $\tilde{s}_t$, a fraction also have range measurements $\tilde{r}_t$, and a smaller fraction have all three measurements.

For all methods, the control inputs and the drag dynamics are considered unknown. The prediction model assumes a constant second derivative model, with $\Delta t = 0.2$ being the time between consecutive estimation poses. For the GM-SPA, the model is

$$x_{t+\Delta t} - x_t = \Delta t \begin{bmatrix} \frac{v_{t+\Delta t} + v_t}{2} \\ \frac{\omega_{t+\Delta t} + \omega_t}{2} \\ 0 \\ 0 \end{bmatrix} + \Delta t d_t, \quad (4.25)$$

where $d_t$ is a Gaussian random variable with covariance equal to

$$\Sigma = \text{diagonal}(\begin{bmatrix} 0.1 & 0.1 & 0.1 \cdot \frac{\pi}{180} & 10 & 10 & \pi \end{bmatrix}^2).$$

MM-TFS and the PF also use this model, except that the variation in $\theta$ is converted to the manifold representation. The PF takes advantage of the fact that every measurement has $\tilde{s}_t$ measurements, including them analytically in the proposal distribution for the prediction steps.

MM-TFS and the PF model all measurements exactly as they are generated. For the GM-SPA, this is not possible due to the non-linearities. The rest of this section describes the approximated models used with the GM-SPA.

Difficulties can arise when using a real number to represent the 2D heading $\theta$ and working with big uncertainties of angles. For example, given measurements at $\tilde{\theta}_1 = 170^\circ$ and at $\tilde{\theta}_2 = -170^\circ$, both with Gaussian noise with the same standard deviation, the resulting Gaussian posterior belief would be centered at $\theta = 0$, instead of the correct $\theta = 180^\circ$.

To avoid this problem, for each $x_t$, the measurement likelihoods of the heading and velocities, which depend on $\theta_t$, are shifted by a multiple of $2\pi$ to be in the same range as the maximizer of the current belief. For the first pose, there is no prior belief, but since the first pose has all three measurements, the center is taken to be the measured heading. Since the heading measurements are precise enough, all components (that are not pruned by max pruning for
being too inconsistent) end up in the same $180^\circ$ wide interval, avoiding the problems of fusing very different measurements.

The model for the heading measurement is exactly the generative model from (4.23). The only question is that the measured angle must be shifted into the range that is being used by the current pose by adding a multiple of $2\pi$.

The range measurement likelihood is highly non-linear, so it cannot be approximated with a single IG. Instead of using a proposal distribution to build a local approximation, like the PF does, a globally valid approximation with multiple IGs is used, and the algorithm decides which components to keep based on the component reduction techniques from Section 3.2.1.

To approximate the measurement likelihood from a range measurement with Gaussian mixture noise, each noise component is approximated separately. Let each noise component have standard deviation $\sigma_j$ and weight $w_j$. For each noise component, $N_j$ Gaussian components are spread in a circle of radius $r = \tilde{r}_t$ around the range beacon. The number of Gaussians used is chosen dynamically, depending on the ratio of the range and the standard deviation of the measurement, $h = r/\sigma_j$. It is chosen as

$$N_j = \text{ceil} \left( \frac{\pi}{\arccos \left( \frac{k_N h}{k_N h + 1} \right)} \right),$$

(4.26)

where $k_N = 3$ is a complexity parameter. This heuristic is meant for $k_N h \geq 1$, where $N_j \geq 5$. More components are used if $k_N h$ is bigger.

The intuition behind this heuristic comes from counting how many sides a regular polygon needs to circumscribe the circle with radius $r$ while staying inside the circle with radius $r + \sigma_j/k_N$, as represented in Figure 4.20. The arccos term determines half the angle that one side occupies. Therefore, (4.26) calculates how many sides the polygon would need to cover $2\pi$ radians, rounded up.

Let $\alpha$ be the angle that parameterizes the circle of radius $r$ around the beacon. $N_j$ points $p_j^i$ are spread uniformly on the circle, separated by $\Delta\alpha_j = 2\pi/N_j$. For each of them, a locally valid approximation is used, based on the linearization of the range function and the Gaussian noise.

To exemplify, consider the linearization of the range function at $p_j^i$ is $r(p) \approx r + J_j^i (p - p_j^i)$, where $J_j^i = e_{rj}^T$ is the unit row vector pointing away from the beacon at $p_j^i$. 
As there are 2 Gaussian noise components in our case,

\[ p(p|r) = \sum_{j=1}^{2} w_j \mathcal{N}(r(p); r, \sigma_j^2) \] (4.27)

\[ \approx \sum_{j=1}^{2} \sum_{i=1}^{N_j} w_j \mathcal{N}(r + J_j^i(p - p_i^j); r, \sigma_j^2) \] (4.28)

\[ = \sum_{j=1}^{2} \sum_{i=1}^{N_j} w_j \mathcal{N}(J_j^i(p - p_i^j); 0, \sigma_j^2) \] (4.29)

\[ \propto \sum_{j=1}^{2} \sum_{i=1}^{N_j} w_j \sigma_j^{-1} \mathcal{G}(J_j^i(p - p_i^j); 0, \sigma_j^{-1}) \] (4.30)

\[ = \sum_{j=1}^{2} \sum_{i=1}^{N_j} w_j \sigma_j^{-1} \mathcal{G}(p; J_j^i p_i^j, \sigma_j^{-1} J_j^i). \] (4.31)
Each obtained IG component constrains \( r \), but not \( \alpha \), so it is uniform in the direction tangential to the circle. To ensure that the approximation is only used for angles close to their respective \( \alpha_j^i \), a constraint on \( \alpha \) is added to each IG. This is done by adding the row \( e^T_{\alpha_j^i} / (0.5 \Delta \alpha_j r) \) to the IG’s square-root information matrix \( \sigma^{-1}_{j} J_i^j \). This constraint is the one that would be obtained from the linearization if \( \alpha_j^i \) was measured with additive Gaussian noise with a standard deviation of \( 0.5 \Delta \alpha_j \). Figure 4.20 represents the Gaussian components obtained with this method, for \( r = 3 \) and \( \sigma_j = 1 \).

A similar approach is taken to approximate the \( \tilde{s}_i \) measurement. The linear velocity part depends on the heading angle. \( N = 15 \) components are created with multiple \( \theta_i \) values. For each \( \theta_i \), the function (4.22) is locally linearized at the point of maximal likelihood, and an IG component is created. To each IG, a constraint of measuring \( \theta_i \) with standard deviation \( 0.5 \Delta \theta \) is added.

Different combinations of component reduction techniques are considered with the GM-SPA. All twelve combinations of \( d_{\text{thr}} = 0.5 \), \( d_{\text{thr}} = 0.3 \), \( d_{\text{thr}} = 0.7 \), \( p_{\text{thr}} = 0.0 \), \( p_{\text{thr}} = 0.1 \), \( k_{\text{thr}} = 10^{-3} \), and \( k_{\text{thr}} = 10^{-2} \) are tested. They are named with the rule that “XYZ” corresponds to \( d_{\text{thr}} = 0.X \), \( p_{\text{thr}} = 0.Y \), and \( k_{\text{thr}} = 10^{-Z} \). For example, the parameter combination used in previous tests of \( d_{\text{thr}} = 0.5 \), no similarity pruning via \( p_{\text{thr}} = 0.0 \), and \( k_{\text{thr}} = 10^{-3} \) is called 503.

Each combination is executed on data from 101 runs, for statistical analysis. In each simulation, the process disturbances and measurement noises are different.

In the case of MM-TFS, to help the inference process by making the filtered and back-filtered likelihoods proper, a Gaussian prior on the first and last measurements was added with a covariance matrix of

\[
\Sigma = \text{diagonal}([100 \hspace{1em} 100 \hspace{1em} 100 \cdot \frac{\pi}{180} \hspace{1em} 20 \hspace{1em} 20 \hspace{1em} 100 \cdot \frac{\pi}{180}])^2.
\]

Unfortunately, the MM-TFS results are completely noisy, having no resemblance with the real trajectory. This is the case both with \( N = 500 \) and \( N = 2000 \). The bad performance should be a consequence of the measurement factors only constraining part of the state. The PF can handle the partial measurements without a problem and can produce meaningful results with as little as 200 particles.
4.5.3 Example solution

In Figure 4.21, example smoothed trajectory and heading estimates from the GM-SPA with parameter code 502 are plotted with the real values. In Figure 4.22, the filtered results from the first quarter of the simulation are presented, with a range measurement plotted as well. The plotted colored sets are confidence sets with the most probable states. Animations of the filtered\(^*\) and smoothed\(^†\) belief over time are available online.

![Figure 4.21: Real and smoothed trajectories, and confidence sets of final position (left). Heading measurements, real and smoothed heading, and confidence sets of each heading (right). Obtained from the GM-SPA with \(d_{\text{thr}} = 0.5, p_{\text{thr}} = 0\) and \(k_{\text{thr}} = 10^{-2}\). The differently colored sets are a nested sequence of confidence sets of 95%, 90%, 70%, 50%, 30%, and 10%.

From Figure 4.21, it is clear that, despite the range measurements being inherently under-constrained, the GM-SPA is capable of fusing information from different measurements to produce an accurate smoothed estimate. A sign of consistency is that the final position is inside the 95% confidence set.

From Figure 4.22, it is clear that the initial filtered position beliefs are multi-modal. At \(t = 5\), two distinct hypotheses exist. It is only with future measurements that ambiguity is solved. The curvature of the two modes is also noticeable, they are not simply two Gaussians. Manual inspection reveals that each of the modes is composed of four Gaussians. The plotted range measurement is clearly an outlier, and the method recognizes so, as the confidence set does not include the positions indicated by the measurement.

\(^*\)[http://web.tecnico.ulisboa.pt/ist193042/Pose2DPoseVideo502Filtered.m4v]
\(^†\)[http://web.tecnico.ulisboa.pt/ist193042/Pose2DPoseVideo502Smoothed.m4v]
From the heading measurements, which show the confidence sets for all estimates, it can be seen that the uncertainty increases over time and is reduced at each heading measurement, as expected.

![Parameter code 502](image)

**Figure 4.22:** Partial real trajectory, confidence sets of most recent position, and most recent range measurement (left). Heading measurements, real and filtered heading, and confidence sets of each heading (right). Obtained from the GM-SPA with $d_{thr} = 0.5$, $p_{thr} = 0$ and $k_{thr} = 10^{-2}$. The differently colored sets are a nested sequence of confidence sets of 95\%, 90\%, 70\%, 50\%, 30\%, and 10\%.

### 4.5.4 Statistical results

The previous results are from a single run of a single parameter combination. The same is repeated 101 times for each parameter combination. Remember that each combination is described with a 3-digit code “XYZ”. The code means that $d_{thr} = 0.X$, $p_{thr} = 0.Y$, and $k_{thr} = 10^{-Z}$.

The resulting distributions of the position root mean square error, heading root mean square error, and the run-times are represented in Figure 4.23, Figure 4.24, and Figure 4.25, respectively.

In the position RMS error, the median values are in the range [1, 1.7] for all parameter combinations, which is 2 to 3.4 times the standard deviation of the inlier range measurements. This indicates that the trajectory is well estimated most of the time with any parameter combination.

The differences between the parameter combinations are noticed in the tails of the distributions. The biggest distinction is between using $k_{thr} = 10^{-2}$
or \( k_{\text{thr}} = 10^{-3} \). The first results in a PRMSE 90th percentile more than 3 times worse than the second, indicating that it is pruning important components.

The effects of \( d_{\text{thr}} \) and \( p_{\text{thr}} \) are smaller for the tested values, with a trend of less approximation giving slightly better results, as expected. The exception to this trend is that the method 702 has a better PRMSE distribution than 502 and 302. The methods with worse error distributions are 512 and 502. It might be that more component merging is saving important components from being pruned from max merging.

![GM-SPA PRMSE](image1.png)

![GM-SPA PRMSE 90th percentile](image2.png)

(a) (b)

Figure 4.23: Box plots (a) and 90th percentile (b) of position RMS error for each GM-SPA parameter combination, from 101 runs.

![GM-SPA HRMSE](image3.png)

![GM-SPA HRMSE 90th percentile](image4.png)

(a) (b)

Figure 4.24: Box plots (a) and 90th percentile (b) of heading RMS error for each GM-SPA parameter combination, from 101 runs.

Inspecting the heading RMS error from Figure 4.24 allows similar interpretations. The median values are in the range \([3.1, 4.1]^{\circ}\) for all parameter combinations. The biggest influence is that of \( k_{\text{thr}} \) and the worst results are from 512 and 502. The other two parameters have a smaller effect.
Figure 4.25: Box plots (a) and 90th percentile (b) of run-time for each GM-SPA parameter combination, from 101 runs.

The run-time distributions, represented in Figure 4.25, indicate that $k_{thr}$ does not have as much of a dominant impact on the computational cost. The combination 303, which has the least approximations in all three parameters, is by far the slowest. It is 4 to 10 times slower than the others. Among the other combinations, $d_{thr}$ has the biggest impact.

Considering all performance metrics, 513 has the best results, being simultaneously one of the fastest and most accurate parameter combinations.

Considering the performance at the 90th percentile, the PRMSE, HRMSE, and run-time of the 513 combination are 1.8, 4.9°, and 39 s, respectively. For comparison, a PF with 2000 particles achieves a comparable run-time of 63 s, with errors roughly twice as big. Increasing the number of particles to 10000 with the PF still does not achieve errors as good as the GM-SPA, and increases the run-time to more than 300 s. The exact numbers are displayed in Section 4.6.3.

4.6 Results summary

This section summarizes the most important results from each of the three simulation tests.

4.6.1 The 3 doors example

In the 3 doors example from Section 4.3, despite the data association ambiguity and outliers, smoothing with the GM-SPA recovers the correct data association with an accuracy above 90% for all measurements. The 90th percentiles of
the position root mean square error (PRMSE) and run-time obtained from smoothing with each method are presented in Table 4.1. The three MM-iSAM based methods use \( N = 500 \) particles. The GM-SPA uses \( d_{\text{thr}} = 0.5, p_{\text{thr}} = 0, \) and \( k_{\text{thr}} = 10^{-3}. \)

The GM-SPA results in smaller position errors, better implicit data association accuracies, and better run-times than the three MM-iSAM based methods in filtering and smoothing.

Table 4.1: Smoothing 90th percentile PRMSE and run-time from 101 simulations of the tested methods in the 3 doors problem.

<table>
<thead>
<tr>
<th>Method</th>
<th>Position RMS error</th>
<th>Run-time [ms]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ideal KF</td>
<td>0.67</td>
<td>4</td>
</tr>
<tr>
<td>Exact GM-SPA</td>
<td>1.00</td>
<td>53</td>
</tr>
<tr>
<td>GM-SPA</td>
<td>0.83</td>
<td>20</td>
</tr>
<tr>
<td>MMLV-TFS</td>
<td>2.87</td>
<td>(36 \cdot 10^3)</td>
</tr>
<tr>
<td>MM-TFS</td>
<td>3.25</td>
<td>(35 \cdot 10^3)</td>
</tr>
<tr>
<td>MM-iSAM</td>
<td>5.41</td>
<td>(168 \cdot 10^3)</td>
</tr>
</tbody>
</table>

The better results are not surprising, as the GM-SPA is a method designed specifically for problems like this one.

The component reduction methods reduce the run-time by more than half and do not lead to a decrease in performance. They even lead to more accurate estimates in this test, but there is no reason to expect that to be true for other problems as well.

MM-iSAM is significantly less accurate in smoothing problems than the other methods, as expected from the discussion in Section 4.1 of the current implementation.

MMLV-TFS produces less noisy kernel density estimates of the measurement likelihoods and obtains smaller errors than MM-TFS, with equivalent run-time, which validates low-variance sampling as beneficial.

The three MM-iSAM based methods are three orders of magnitude slower than the other methods, and MM-iSAM is 4.7 times slower than MM-TFS and MMLV-TFS.

### 4.6.2 Target tracking with correlated reflections

In the linear target tracking problem from Section 4.4, the outlier component model (2.3) is used to model outlier measurements.

The null hypothesis (NH) is found to be too expensive to express with sample-based methods, due to the difficulty of spreading particles over a big
space of possibilities. Instead, by using information on how the outliers are generated, a more specific outlier component is used with MM-TFS. The GM-SPA uses the NH, which assumes less about the problem.

Before parameter tuning, both MM-TFS and the GM-SPA fail to identify the correct $z$ for all positions. For the GM-SPA, the problem is eliminated by reducing $k_{\text{thr}}$, increasing the computational cost.

Similarity pruning is then introduced, reducing the run-time by 18% while still identifying the correct trajectory. The results support the conclusion that to avoid premature pruning in the presence of correlated outliers, it is safer to add similarity pruning than to increase max pruning.

The run-time of MM-TFS is two orders of magnitude bigger than the GM-SPA’s, despite not identifying the correct trajectory and using an outlier model with more information.

These results exemplify the computational benefits, in terms of robustness and time, of using the GM-SPA over MM-TFS for target tracking in the presence of correlated outliers.

### 4.6.3 2D range-only pose estimation

In the 2D range-only problem from Section 4.5, the GM-SPA is capable of producing an accurate smoothed estimate more often than not with all tested parameter combinations.

All parameter combinations have good median error metrics, with median $\text{PRMSE}$ no bigger than 1.7 and median heading root mean square error (HRMSE) of at most $4.1^\circ$.

For comparison, MM-TFS did not converge to meaningful results, and the PF with 10000 particles achieves median errors of 2.0 and 5.7$^\circ$, while being significantly slower than all but the slowest GM-SPA parameter combination.

It is important to note that no tested method achieves good estimates in all 101 simulation runs.

Considering all performance metrics, parameter combination 513, with $d_{\text{thr}} = 0.5$, $p_{\text{thr}} = 0.1$, and $k_{\text{thr}} = 10^{-3}$, has the overall best results, being simultaneously one of the fastest and most accurate parameter combinations.

Table 4.2 presents the 90th percentile of the error and run-time metrics for the overall best GM-SPA parameter combination, 513, and the PF, with $N = 2000$ and $N = 10000$ particles. The results validate the GM-SPA as simultaneously more accurate and more computationally efficient than the non-parametric methods, for this problem.
Table 4.2: Performance metrics at the 90th percentile, from 101 simulations, of the overall best GM-SPA parameter combination 513, and of the PF.

<table>
<thead>
<tr>
<th>Method</th>
<th>PRMSE</th>
<th>HRMSE [°]</th>
<th>Run-time [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>GM-SPA 513</td>
<td>1.8</td>
<td>4.9</td>
<td>39</td>
</tr>
<tr>
<td>Particle Filter $N = 2000$</td>
<td>4.6</td>
<td>8.2</td>
<td>63</td>
</tr>
<tr>
<td>Particle Filter $N = 10000$</td>
<td>3.7</td>
<td>8.0</td>
<td>309</td>
</tr>
</tbody>
</table>

The differences in the results of the tested GM-SPA parameter combinations are relevant in the run-time and the 90th percentile error metrics. The max pruning parameter $k_{thr}$ has the biggest impact on the error metrics, where $k_{thr} = 10^{-3}$ is necessary to get low 90th percentile errors. $p_{thr}$ and $d_{thr}$ have a comparably negligible effect in the tested range.

On the run-time, it is $d_{thr}$ that has the most significant impact.
Simulation tests
Chapter 5

Conclusions and future work

This section presents the conclusions of the thesis, discusses the limitations of the results obtained, considers avenues for further work, and reflects on the societal impact of robust state estimation.

5.1 Conclusions

In Chapter 2, the state of the art in multimodal inference methods is reviewed. Fundamental computational ideas based on improper Gaussian mixture two-filter smoothing are identified as an efficient strategy for doing inference with robust mixture measurement models.

In Chapter 3, those ideas are adapted from the original linear target tracking context [17] to the more general factor graph framework. The resulting state estimation method for tree factor graphs, called Gaussian mixture sum-product algorithm (GM-SPA), is proposed as an approximate, efficient, and robust multimodal method. It can solve under-constrained problems without requiring an initialization.

The GM-SPA is a promising candidate solution for difficult-to-model underwater scenarios, such as acoustic underwater target tracking in outlier-prone environments and AUV acoustic localization. It is also well suited for under-constrained problems where a good initial solution is not available to use with local solvers.

In Chapter 4, simulation tests measure the performance of the GM-SPA. The simulated problems are: 1D linear localization with ambiguous data association and outliers; 3D linear target tracking with correlated outliers; and 2D range-only pose estimation with Gaussian mixture noise.

The GM-SPA outperforms non-parametric methods such as MM-iSAM.
and the PF in the above problems when it comes to accuracy, robustness, and computational cost. In the tested problems, despite data association and outlier detection decisions being left ambiguous, the GM-SPA recovers the correct results through consistency checks more than 90% of the time.

The outlier component model is validated as a robust estimation strategy that can be efficiently solved with the GM-SPA. The null hypothesis model (2.4) in particular provides robustness to unmodelled outliers.

In the presence of unmodelled correlation in the outliers, similarity pruning was found to be more reliable than max pruning, which can lead to prematurely forgetting important hypotheses.

5.2 Limitations and future work

The proposed method has 3 main limitations that can benefit from further study and development. They are presented next in order of increasing expected difficulty.

5.2.1 Cyclic factor graphs

The proposed GM-SPA is limited to tree factor graphs, which excludes, for example, efficient representations of the SLAM problem. Exact inference for these problems is possible with Junction Trees like the Bayes Tree, which was not considered for the sake of simplicity.

Implementing an IG mixture solver for the Bayes Tree is a clear path for expanding the applicability of the methods discussed. The same computational aspects apply and no fundamentally new problem or difficulty is expected, beyond the difficulty of building the Bayes Tree itself, which is non-trivial but has already been studied in [30]. A practical open question is how efficient are IG mixtures when the dimension of the intraclique integral, the equivalent of (A.12) for the Bayes Tree, increases, due to the big clique sizes that can arise when solving factor graphs with cycles.

5.2.2 Approximation of arbitrary likelihoods

The GM-SPA assumes that likelihood models are available as IG mixture models. This was the case for the considered problems, including simultaneously non-Gaussian and non-linear measurements.

However, it isn’t clear where the limits for the measurement models that can be efficiently approximated are. For example, while range-only
measurements as used in trilateration are well approximated, it doesn’t seem possible to obtain a globally valid approximation of a bearing-only measurement as used in triangulation.

On the topic of approximating arbitrary likelihoods as IG mixtures, a strategy that was not explored due to time constraints was that of automatically and dynamically building IG mixture approximations that are valid just on the range where they are necessary. The trade-off here is the same as in a proposal distribution for importance sampling: increased approximation accuracy in the proposed region at the risk of poor approximation in other parts of the likelihood that might also be important. In this work, static and globally valid approximations were used, limiting accuracy but ensuring a globally valid minimum approximation quality.

5.2.3 Considering manifold-valued variables

A final and important limitation of the GM-SPA is that it only considers vector-valued variables, while important estimation problems rely on manifolds, particularly the 2D and 3D orientation manifolds $SO(2)$ and $SO(3)$ [62].

In Section 4.5, $SO(2)$ was successfully approximated as a real number, which is a valid approximation because, for each orientation, components were limited to a $180^\circ$ interval.

How to efficiently consider manifold-valued variables with the techniques followed in this work, in a way that handles arbitrarily large uncertainties, remains an open question.

For the $SO(3)$ manifold, for example, there doesn’t seem to exist a known distribution for which the necessary operations, such as convolutions, are closed and can easily be determined, as is the case for the Gaussian. A commonly used distribution seems to be the von Mises-Fisher distribution [63].

Another difficulty is how to represent correlation with other variables, for which the Fisher-Gaussian distribution might be relevant [64].

5.3 Reflection on societal impact

Outlier robustness in state estimation has a range of potential benefits for society and the environment in various domains. Advances in robustness for AUV navigation can extend the range of AUVs and improve the accuracy of the information they collect. This is essential for automated underwater monitoring, for example of pollution levels and coral reefs [65]. By preventing
navigation failures, it also helps to improve safety and reduce the risk of accidental environmental damage. In autonomous driving, robustness is necessary for safety. In the agricultural domain, outlier robust methods have been validated for automated and efficient irrigation management [66]. In the renewable energy domain, outlier robust methods are important for accurate forecasting of energy production [67], reducing reliance on fossil fuels.

The predictable impacts of more reliable state estimation can be categorized according to the Sustainable Development Goals adopted by the United Nations as contributing to the following goals: 2, Zero Hunger; 7, Affordable and Clean Energy; 9, Industry, Innovation, and Infrastructure; and 14, Life Below Water.
References


[18] P. Coretto and C. Hennig, “Robust improper maximum likelihood: Tuning, computation, and a comparison with other methods


Appendix A

Factor graphs

This appendix formalizes the factor graph framework and derives the sum-product algorithm (SPA), following [20].

A.1 Introduction

Consider a real-valued global function $g(X)$ of a set of variables $X \equiv \{x_1, x_2, \ldots\}$. Let $g(X)$ allow a factorization as a product of local factors. The set of local factors is $F$ and each $f \in F$ is defined on the subset $X_f \subseteq X$. The resulting factorization is

$$g(X) = \prod_{f \in F} f(X_f). \quad (A.1)$$

To represent (A.1) as a factor graph $G$, each variable $x_i \in X$ is represented with a variable node and each factor with a factor node. From now on, factors and the respective nodes will be considered equivalent and the expressions used interchangeably. The same applies to variable and variable node. Each factor is adjacent (meaning connected by a path of length 1) to the variables it is defined on and each variable is adjacent to the factors defined on it. The neighbors of node $v$ are represented as $n(v)$.

As a motivating example, a hidden Markov model (HMM) problem is represented as a factor graph in Figure A.1. Variable nodes are represented with bigger clear circles, on which the name of the variable is usually written, while factor nodes are represented with smaller black circles. Factor nodes can be left unnamed or named with labels next to the nodes.

In the case of Figure A.1, the set of variables is $X = \{x_1, x_2, x_3\}$,
the global function is the posterior $g(X) = p(X|z_1, z_2, z_3)$ and the factors are: a prior probability $f_0(x_1) = p(x_1)$; two odometry conditional probabilities $f_{12}(x_1, x_2) = p(x_2|x_1)$ and $f_{23}(x_2, x_3) = p(x_3|x_2)$; and three state measurement likelihoods $f_i(x_i) = p(z_i|x_i)$. The $z_i$ are observed measurements.

Figure A.1: Factor graph showcasing the factorization of a HMM into a prior, measurement likelihoods, and odometry conditional probabilities.

A.2 The sum-product algorithm

The sum-product algorithm can be executed on a finite tree factor graph $G$ to calculate the marginals of the global function $g(X)$ on each variable $x \in X$.

A.2.1 The sum-marginal operator

In [20], the sum-product algorithm (SPA) is derived for a general summary operator notated $\downarrow$, sometimes also called marginal. In our context, the summary is the marginal integral, but other choices could be taken, as discussed in Section A.2.2.

Given a set of variables $A$ and a real function $f$ defined on variables $X_f$, the marginal of $f$ on $A$ is notated as $f(X_f) \downarrow A$ and it is a function of $X_f \cap A$. It marginalizes out all the variables of the domain that are not in $A$. The definition is

$$
(f(X_f) \downarrow A)(X_f \cap A) \equiv \begin{cases} 
\int f(X_f) dX_f \setminus A & \text{if } X_f \cap A \neq X_f \\
\int f(X_f) & \text{otherwise.}
\end{cases} \quad (A.2)
$$

The marginal operator is defined trivially on a single variable $x$ by $f(X_f) \downarrow x \equiv f(X_f) \downarrow \{x\}$. 


In general, when dealing with unnormalizable/improper functions, the marginal integral is not guaranteed to be finite. For all of the following results, it is assumed that all integrals converge to a real value.

An important property of the marginal operator that results directly from the definition is that marginalizing on variables that are not part of the domain of the factor does not change anything. Formally, for a real function \( f(X_f) \) and variable sets \( A \) and \( B \) where \( X_f \cap A = X_f \cap B \), it is true that

\[
f(X_f) \downarrow A = f(X_f) \downarrow B.
\] (A.3)

A second important property of marginals is that the order of sequential marginals does not matter. For a real function \( f(X_f) \) and variable sets \( A \) and \( B \), marginalizing on \( A \) and then \( B \) is the same as marginalizing on \( A \cap B \). The case when both marginals are non-trivial is proved first, by

\[
[f(X_f) \downarrow A] \downarrow B = \left[ \int f(X_f) dX_f \setminus A \right] \downarrow B
= \int \int f(X_f) d[X_f \setminus A] d[(X_f \cap A) \setminus B]
= \int f(X_f) d[(X_f \setminus A) \cup ((X_f \cap A) \setminus B)]
= \int f(X_f) dX_f \setminus (A \cap B)
= f(X_f) \downarrow (A \cap B).
\] (A.4)

For the cases for which the first marginal is trivial because \( X_f \cap A = X_f \), then

\[
[f(X_f) \downarrow A] \downarrow B = f(X_f) \downarrow B = f(X_f) \downarrow (A \cap B)
\] (A.5)

where the last equality follows from (A.3) since \( X_f \cap A = X_f \). The cases where the second marginal is trivial are proved with the same argument.

A third and final property of the marginal operator that is relevant is a situation in which the marginal of a product is the product of the marginals. Consider five variable sets \( M_1, M_2, K_1, K_2, \) and \( K \), which stand for variables to marginalize out and variables to keep. Let \( M_1, M_2 \) and \( K \) be disjoint to each other and let \( K_1 \) and \( K_2 \) be subsets of \( K \). For two real functions \( f_1(K_1, M_1) \) and \( f_2(K_2, M_2) \), the \( K \) marginal of their product
is equal to the product of the marginals, since
\[
[f_1(K_1, M_1) \cdot f_2(K_2, M_2)] \downarrow K
= \int \int f_1(K_1, M_1) \cdot f_2(K_2, M_2) dM_2 dM_1
= \int f_1(K_1, M_1) \int f_2(K_2, M_2) dM_2 dM_1
= \int f_1(K_1, M_1) dM_1 \cdot \int f_2(K_2, M_2) dM_2
= [f_1(K_1, M_1) \downarrow K] \cdot [f_2(K_2, M_2) \downarrow K]
= [f_1(K_1, M_1) \downarrow K_1] \cdot [f_2(K_2, M_2) \downarrow K_2]. \quad (A.6)
\]

A.2.2 Comparison with max-marginals

The marginal integral has a straightforward interpretation in the context of probabilistic inference. The joint posterior distribution is given as the product of the prior and the measurement likelihoods (2.1). In this case, the marginal integral computes the marginal distributions. The SPA can be used to calculate only the marginal posterior distributions, instead of the joint posterior distribution. This reduces the computational cost but also leaves out possibly important correlation information. Grouping variables together into bigger variables results in less marginalization, and therefore more correlation information and more computations.

The marginal integral is a possible marginal/summary operator, which is defined axiomatically in [20]. For a function \( f(x, y, z) \), any summary on \( x \) is a function just of \( x \). Given a specific value of \( x = x_0 \), it summarizes all the possible values of \( f(x_0, y, z) \) with a single number. Another relevant summary is the max-marginal. To compare both summary operators, let the sum-marginal, defined in (A.2), be notated in this section as
\[
f_x^{\text{sum}}(x) = \int f(x, y, z) dy dz, \quad (A.7)
\]
where the integral is over the domain of \( f \) for \( y \) and \( z \). Likewise, let the max-marginal be defined as
\[
f_x^{\text{max}}(x) = \max_{y,z} f(x, y, z). \quad (A.8)
\]

Estimating only summaries is essential for efficient inference, but it has a cost, as a summary necessarily forgets some information. Consider
Appendix A: Factor graphs

trajectory estimation, where each state is \( x_t \), the whole trajectory is \( X \), and the measurements are \( Z_{1:T} \). The most likely trajectory, obtained from \( \text{argmax}_X p(X | Z_{1:T}) \), can be different from the trajectory made up of the most likely states, where each state is given from \( \text{argmax}_{x_t} p(x_t | Z_{1:T}) \). Maximizing the max-marginals gives the former while maximizing the sum-marginals gives the latter.

As a trivial example, consider a discrete joint distribution \( f(x, y) \) over \( x, y \in \mathbb{Z} \) defined as

\[
f(1, 1) = 0.4, \quad \forall i \in \{1, 2, ..., 6\}, \quad f(i, 0) = 0.1. \tag{A.9}
\]

The most likely pair is \( (1, 1) = \text{argmax}_{x, y} f(x, y) \). The marginal integral for \( x \) is

\[
f_{\text{sum}}^x(1) = 0.4 + 0.1 = 0.5, \quad \forall i \in \{2, ..., 6\}, \quad f_{\text{sum}}^x(i) = 0.1, \tag{A.10}
\]

making the most likely \( x \) value \( 1 = \text{argmax}_x f_{\text{sum}}^x(x) \). The marginal for \( y \) is

\[
f_{\text{sum}}^y(1) = 0.4, \quad f_{\text{sum}}^y(0) = 0.1 \cdot 6 = 0.6, \tag{A.11}
\]

making the most likely \( y \) value \( 0 = \text{argmax}_y f_{\text{sum}}^y(y) \). The pair made up of joining the maximum probability estimates of each marginal is \( (1, 0) \), which is not the pair that maximizes the joint distribution. Note that this difference does not exist if, for both the joint distribution and the marginals, the most probable value is the mean, as is the case with Gaussian distributions. The mean is always the same in the joint distribution and in the marginal distributions.

A.2.3 The update rule

The SPA is executed by message passing between the nodes of the factor graph. Each message is a function of the variable it is being sent to or from. A message can only be computed and sent after all the incoming messages from other edges have been received. The message sent from a factor \( f \) to a variable \( x \) is

\[
m_{f \rightarrow x}(x) \equiv \left[ f(X_f) \prod_{y \in n(f) \setminus \{x\}} m_{y \rightarrow f}(y) \right] \downarrow x, \tag{A.12}
\]

and the message sent from a variable \( x \) to a factor \( f \) is
\[ m_{x \rightarrow f}(x) \equiv \prod_{h \in n(x) \backslash \{f\}} m_{h \rightarrow x}(x). \]  
(A.13)

In (A.12) and (A.13), the sequence products notated with \( \prod \) are equal to 1 when they have no terms, which is the case for leaf nodes. This message-passing scheme has to start by calculating the first messages at the leaf nodes. It proceeds until all messages are calculated. The calculations can be distributed on the nodes, each node calculating its outward messages.

The purpose of these messages is to carry just the information from the corresponding branch of the factor graph that is relevant to the node to which it is sent. The message \( m_{v \rightarrow w} \) is a summary of all the factors that connect to \( w \) through \( v \).

The outward messages from the factors, in (A.12), are the most difficult to compute. They require calculating a product of functions defined on different variables and then calculating the marginal integral on the receiving variable.

The marginals of the global function on each \( x \in X \) can be determined as the product of the incoming messages on the variable, with the expression

\[ g_x(x) \equiv g(X) \downarrow x = \prod_{f \in n(x)} m_{f \rightarrow x}(x). \]  
(A.14)

### A.2.4 Derivation

Next follows the derivation of SPA, namely (A.14), adapted from [20]. The goal of SPA is to calculate the marginals

\[ g_x(x) \equiv g(X) \downarrow x = \left( \prod_{f \in F} f(X_f) \right) \downarrow x. \]  
(A.15)

Since the factor graph \( G \) is a tree, there is a single path between any two nodes. Removing an edge \( e = (f, x) \) breaks \( G \) into two disconnected components, one containing \( f \) and one containing \( x \). Let \( G_{f \rightarrow x} \) be the one that contains \( f \) and \( G_{x \rightarrow f} \) contain \( x \). Note that the inclusion notation choice here is the opposite of that in [20]. For any two adjacent nodes \( v \) and \( w \), let \( X_{v \rightarrow w} \) be the variables in \( G_{v \rightarrow w} \), \( F_{v \rightarrow w} \) be the factors in \( G_{v \rightarrow w} \) and \( g_{v \rightarrow w} \) be a function that is equal to the product of all the factors in \( G_{v \rightarrow w} \).

If a node \( v \) were removed from the graph, the remaining graph would be composed of components \( G_{w \rightarrow v} \), one for each \( w \in n(v) \). Note that for two different nodes \( w, u \in n(v) \), \( G_{w \rightarrow v} \) and \( G_{u \rightarrow v} \) are disconnected.
Consequently, \( g_{f \to x} \) is a function of \( X_{f \to x} \cup \{ x \} \) and \( g_{x \to f} \) is a function of \( X_{x \to f} \). Furthermore, \( F = \bigcup_{f \in n(x)} F_{f \to x} \).

The factors in (A.15) can be grouped according to the neighbor of \( x \) with which they connect to \( x \), as

\[
g_x(x) = \left( \prod_{f \in n(x)} \prod_{h \in F_{f \to x}} h(X_h) \right) \downarrow x = \left( \prod_{f \in n(x)} g_{f \to x}(x, X_{f \to x}) \right) \downarrow x. \tag{A.16}
\]

The result from (A.6) can be used to separate the marginal of the product into the product of the marginals, resulting in

\[
g_x(x) = \prod_{f \in n(x)} (g_{f \to x}(x, X_{f \to x}) \downarrow x). \tag{A.17}
\]

What is left to prove is that

\[
g_{f \to x}(x, X_{f \to x}) \downarrow x = m_{f \to x}(x), \tag{A.18}
\]

which will be done through induction. The proof will also prove

\[
g_{x \to f}(X_{x \to f}) \downarrow x = m_{x \to f}(x). \tag{A.19}
\]

For a leaf factor \( f \), let the adjacent variable be \( x \). Since \( X_f \equiv n(f) = \{ x \} \) and \( F_{f \to x} = \{ f \} \), simple substitution confirms (A.18) by

\[
m_{f \to x}(x) \equiv \left[ f(X_f) \prod_{y \in n(f) \setminus \{ x \}} m_{y \to f}(y) \right] \downarrow x
\]

\[
= f(x) \downarrow x
\]

\[
= g_{f \to x}(x, X_{f \to x}) \downarrow x. \tag{A.20}
\]

Likewise, for a leaf variable \( x \), let the adjacent factor be \( f \). Since \( n(x) = \{ f \} \) and \( g_{x \to f}(X_{x \to f}) = 1 \), simple substitution confirms (A.19) by

\[
m_{x \to f}(x) \equiv \prod_{h \in n(x) \setminus \{ f \}} m_{h \to x}(x)
\]

\[
= 1
\]

\[
= g_{x \to f}(X_{x \to f}) \downarrow x. \tag{A.21}
\]

For a non-leaf factor \( f \), we can prove that (A.18) follows if (A.19) is
assumed for the incoming messages. The assumption is

$$m_{f \rightarrow x}(x) \equiv \left[ f(X_f) \prod_{y \in n(f) \setminus \{x\}} m_{y \rightarrow f}(y) \right] \downarrow x$$

$$= \left[ f(X_f) \prod_{y \in n(f) \setminus \{x\}} g_{y \rightarrow f}(X_{y \rightarrow f}) \downarrow y \right] \downarrow x. \quad (A.22)$$

$f$ is defined on $X_f$, so it is trivially equal to its marginal on $X_f$. (A.6) allows the passing of the products to inside the marginal with $K = X_f$. Finally, the two marginals are equal to a single marginal in $X_f \cap x = x$ due to (A.4). These operations give

$$m_{f \rightarrow x}(x) = \left[ f(X_f) \downarrow X_f \prod_{y \in n(f) \setminus \{x\}} g_{y \rightarrow f}(X_{y \rightarrow f}) \downarrow y \right] \downarrow x$$

$$= \left[ f(X_f) \prod_{y \in n(f) \setminus \{x\}} g_{y \rightarrow f}(X_{y \rightarrow f}) \downarrow X_f \right] \downarrow x$$

$$= \left[ f(X_f) \prod_{y \in n(f) \setminus \{x\}} g_{y \rightarrow f}(X_{y \rightarrow f}) \right] \downarrow x. \quad (A.23)$$

Since $F_{f \rightarrow x} = \{f\} \cup \bigcup_{y \in n(f) \setminus \{x\}} F_{y \rightarrow f}$, then $g_{f \rightarrow x}(x, X_{f \rightarrow x}) = f(X_f) \prod_{y \in n(f) \setminus \{x\}} g_{y \rightarrow f}(X_{y \rightarrow f})$, which concludes the proof that (A.18) follows if (A.19) holds. The induction can be written as

$$\forall y \in n(f) \setminus \{x\} \quad m_{y \rightarrow f}(y) = g_{y \rightarrow f}(X_{y \rightarrow f}) \downarrow y$$

$$\Rightarrow m_{f \rightarrow x}(x) = g_{f \rightarrow x}(x, X_{f \rightarrow x}) \downarrow x. \quad (A.24)$$

Likewise, for a non-leaf variable $x$, we can prove that (A.19) follows if (A.18) is assumed for the incoming messages. The assumption is

$$m_{x \rightarrow f}(x) \equiv \prod_{h \in n(x) \setminus \{f\}} m_{h \rightarrow x}(x)$$

$$= \prod_{h \in n(x) \setminus \{f\}} g_{h \rightarrow x}(x, X_{h \rightarrow x}) \downarrow x. \quad (A.25)$$

With (A.6), setting $K = \{x\}$, the order of the product and marginal
can be switched. Since \( F_{x \rightarrow f} = \bigcup_{h \in n(x) \setminus \{f\}} F_{h \rightarrow x} \), then \( g_{x \rightarrow f}(X_{x \rightarrow f}) = \prod_{h \in n(x) \setminus \{f\}} g_{h \rightarrow x}(x, X_{h \rightarrow x}) \). Consequently,

\[
\forall h \in n(x) \setminus \{f\} \quad m_{h \rightarrow x}(x) = g_{h \rightarrow x}(x, X_{h \rightarrow x}) \downarrow x \\
\Rightarrow m_{x \rightarrow f}(x) = g_{x \rightarrow f}(X_{x \rightarrow f}) \downarrow x. \tag{A.26}
\]

This finishes the proof that (A.18) and (A.19) are true for all computed messages, since they are true for the initial messages computed at leaf nodes and are true for any subsequent messages computed from received messages. (A.14) follows directly from (A.17) and (A.18).

All that remains to prove is that all nodes actually receive all messages. Remember that messages are only computed and sent when all other messages are received. Using this rule in a graph with cycles, some messages are never computed.

Consider a node \( v \). Since the factor graph \( G \) is a finite tree, it can be interpreted with \( v \) as the root at the top, branching down into multiple paths that all lead to leaf nodes. The nodes can be ordered with the usual parent and child relationships of rooted trees, where the child nodes of a parent node are one step further away from \( v \) than the parent. Note that the SPA runs without considering any particular node \( v \) as a root, it simply applies the update rules wherever possible, the order of which is not specified and not relevant. The SPA will always be able to compute the messages from the leaf nodes. When all of these are calculated, as the bottom level of the tree is only made of leaf nodes, all of the nodes in that level will have their upward messages computed.

At this point, all the nodes in the level above the bottom have received all the messages from their child nodes. Since each node has a single parent in the rooted tree, all of their upward messages can now be calculated. When this is done, it in turn guarantees that the nodes in the level above can calculate the messages to send to their parents. This induction guarantees that it will always be possible to calculate all the messages up the tree until the root \( v \), which receives all the messages. Since the node \( v \) is an arbitrary node that did not affect the choice of which messages to calculate, it can be taken to be any node in the tree, proving that every node will receive all its messages eventually. Alternatively, similar arguments could now be made to justify that all the downward messages down the tree can be calculated, meaning all nodes receive all messages.
Appendix A: Factor graphs
Appendix B

Column pivoted QR factorizations

This appendix discusses the column pivoted QR (QRCP) factorization and its use in simplifying linear least-square problems with singular matrices.

B.1 Introduction

A matrix \( X \in \mathbb{R}^{m \times n} \) of rank \( r \) can be factored with a QRCP factorization [68, p. 276] as

\[
X = QR P^T,
\]  

(B.1)

where:

- The columns of \( Q \in \mathbb{R}^{m \times r} \) form an orthonormal basis of the column space of \( X \).
- \( R \in \mathbb{R}^{r \times n} \) has full rank \( r \) and is upper trapezoidal, the latter meaning \( \forall_{i,j} R_{ij} = 0 \). The rows of \( R P^T \) form a (non-orthonormal) basis of the row space of \( X \).
- \( P \in \mathbb{R}^{n \times n} \) is a permutation matrix.

To see that \( Q \) forms a basis of the column space of \( X \), note that the column space of \( X \) is contained in the column space of \( Q \), and both have dimension \( r \), so they must be the same. The same argument applies to the row space of \( X \) and \( R P^T \).
B.2 QRCP with numerical computations

The situation is complicated if $X$ is instead of approximate rank $r$, due to noise or computational error. Using (B.1) for a matrix with full exact rank, QRCP gives

$$X = QRP^T = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \begin{bmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{bmatrix} P^T,$$

(B.2)

where $Q_1 \in \mathbb{R}^{m \times k}$, $R_{11} \in \mathbb{R}^{k \times k}$ and $R_{22} \in \mathbb{R}^{(n-k) \times (n-k)}$ form a block view for some arbitrary $k$, to help with notation for the next expressions. Also, let the notation $c_j$ refer to the $j$-th column of $R_{22}$.

The commonly used LAPACK implementation [69] ensures that the diagonal entries of $R$ have decreasing absolute values

$$|r_{11}| \geq |r_{22}| \geq \ldots \geq |r_{nn}|.$$

(B.3)

Furthermore, for any $k$ and corresponding submatrix $R_{22}$, it guarantees that its first column has the biggest norm, so

$$\forall j \quad |r_{(k+1)(k+1)}| = \|c_1\| \geq \|c_j\|.$$

(B.4)

The fact that $|r_{(k+1)(k+1)}| = \|c_1\|$ is simply a consequence of $R$ being upper trapezoidal, so $c_1$ only has one non-zero entry.

If for some $k$ it is the case that $|r_{kk}| \gg |r_{(k+1)(k+1)}|$, then $\|R_{22}\|$ is small too, as its columns have a norm no greater than $|r_{(k+1)(k+1)}|$. This means that the numerical rank of $X$ and $R$ is at most $k$. Furthermore, a rank $k$ approximation can be constructed by

$$X = \begin{bmatrix} Q_1 R_{11} & Q_1 R_{12} + Q_2 R_{22} \end{bmatrix} P^T \approx Q_1 \begin{bmatrix} R_{11} & R_{12} \end{bmatrix} P^T.$$

(B.5)

This strategy usually reveals the real approximate rank of $X$. However, it is not as reliable at measuring the rank as the singular value decomposition, another matrix factorization [68, p. 76]. Specific counter-examples are known where there are no small values in the diagonal entries of $R$ to reveal the rank deficiency, such as the Kahan matrix [68, p. 279]. Nonetheless, the QRCP is used in this work because it is less computationally expensive and found to be good enough in practice.
B.3 Application of the QRCP

In this work, the QRCP is used to separate functions of $x$ of the form $\|Ax - b\|^2$ into the sum of their minimum value and a variable term, obtaining

$$\|Ax - b\|^2 = \|RPTx - QTb\|^2 + \|b - QQ^Tb\|^2; \quad (B.6)$$

where $A = QRPT$ is the QRCP of $A$.

The form (B.6) is useful because it separates the variable part of the norm from its minimum value. Since $RPT$ is full rank, the minimum of the left term is 0 and is achieved. Consequently, the right term is the minimum squared distance. Furthermore, the minimizer is the solution of $RPTx = QTb$. Another useful result of this manipulation is that the dimensions of the resulting matrices are minimal. Let $x \in \mathbb{R}^n$ and $A$ be $N \times n$, with $N > n$. The result can be stored with the $r \times n$ matrix $RPT$, the $r$-dimensional vector $QTb$ and the scalar $\|b - QQ^Tb\|^2$. $r \leq n$ is the rank of $A$.

The proof of (B.6) stems from the understanding that $QQ^T$ is the projection matrix that projects to the column space of $Q$, which is also the column space of $A$. So $b$ can be separated into the projected and orthogonal components

$$b = QQ^Tb + (I - QQ^T)b. \quad (B.7)$$

Since $(I - QQ^T)b$ is orthogonal to the column space of $A$, it is orthogonal to $Ax$ and

$$\|Ax - b\|^2 = \|Ax - QQ^Tb\|^2 + \|b - QQ^Tb\|^2 \quad (B.8)$$

$$= \|RPTx - QQ^Tb\|^2 + \|b - QQ^Tb\|^2 \quad (B.9)$$

$$= \|RPTx - QTb\|^2 + \|b - QQ^Tb\|^2. \quad (B.10)$$

The last step uses the fact that since $Q$ has orthonormal columns, for any $v \in \mathbb{R}^r$, then $\|Qv\| = \|v\|$. 
Robust state estimation solutions must deal with faulty measurements, called outliers, and unknown data associations, which lead to multiple feasible hypotheses. Take, for instance, the scenario of tracking two indistinguishable targets based on position measurements, where each measurement could refer to either of the targets or even be a faulty reading.

Common estimation methods model the state as having a unimodal distribution, so they are called unimodal methods. Likewise, multimodal methods model the state as a multimodal distribution. Difficult problems, such as autonomous underwater vehicle (AUV) navigation relying on acoustic localization, frequently involve recurring outliers. In these situations, the correct hypothesis only emerges as the most likely one when a substantial number of measurements are considered. Robust solutions for these problems need to consider multiple hypotheses simultaneously, which, in turn, calls for the representation of multimodal distributions.

In this work, a novel approximate inference method is presented, called the Gaussian mixture sum-product algorithm (GM-SPA), as it implements the sum-product algorithm (SPA) for Gaussian mixtures. The GM-SPA can exactly represent under-constrained linear measurements and approximate important non-linear models, such as range measurements and 2D pose kinematics.

The outlier robustness of the GM-SPA is tested and compared against the particle filter (PF) and multimodal incremental smoothing and mapping (MM-iSAM), both of which are non-parametric methods.
Robustness, accuracy, and run-time are improved in simulation tests. The test problems include 1D localization with unknown data association, 3D linear target tracking with correlated outliers, and 2D range-only pose estimation with Gaussian mixture noise.

“Keywords[eng ]”: €€€€
Robust state estimation, Underwater localization, Target tracking, Gaussian mixture, AUV €€€€,

“Abstract[eng ]”: €€€€
Robust solutions for tillståndsskattning måste kunna hantera felaktiga mätningar, så kallade \\textit{(outliers)}, och okända dataassociationer, vilket leder till flera möjliga hypoteser. Ta till exempel scenariot att spåra två likadana mål baserat på positionsmätningar, där varje mätning kan tillhöra något av målen eller till och med vara en felaktig avläsning.

Vanliga skattningstekniker modellerar tillståndet som en unimodal fördelning, och kallas därför unimodala metoder. På samma sätt modellerar multimodala metoder tillståndet som en multimodal fördelning. Svåra problem, som navigering av autonom undervattensfarkoster (AUV) med hjälp av akustisk lokaliserings, involverar ofta upprepad \textit{(outliers)}. I dessa situationer framstår den korrekt hypotesen som den mest sannolika först när ett stort antal mätningar beaktas. Robusta lösningar för dessa problem måste ta hänsyn till flera hypoteser samtidigt, vilket i sin tur kräver representation av multimodala fördelningar.


GM-SPA:s robusthet mot \textit{(outliers)} testas och jämförs med partikelfiltret (PF) och \textit{multimodal incremental smoothing and mapping} (MM-iSAM), båda är icke-parametriska metoder. Robusthet, noggrannhet och körtid förbättras i simuleringstester. Simulerade tester inkluderar 1D-lokalisering med okänd dataassociation, 3D linjär målföljning med korrelerade \textit{(outliers)} och 2D-ställningsuppskattning av endast räckvidd med Gaussiskt blandningsljud.

“Keywords[swe ]”: €€€€
Robust tillståndsskattning, Undervattenslokalisering, Målspårning, Gaussisk blandning, AUV €€€€,

“Abstract[swe ]”: €€€€
Robusta lösningar för tillståndsskattning måste kunna hantera felaktiga mätningar, så kallade \textit{(outliers)}, och okända dataassociationer, vilket leder till flera möjliga hypoteser. Ta till exempel scenariot att spåra två likadana mål baserat på positionsmätningar, där varje mätning kan tillhöra något av målen eller till och med vara en felaktig avläsning.

Vanliga skattningstekniker modellerar tillståndet som en unimodal fördelning, och kallas därför unimodala metoder. På samma sätt modellerar multimodala metoder tillståndet som en multimodal fördelning. Svåra problem, som navigering av autonom undervattensfarkoster (AUV) med hjälp av akustisk lokaliserings, involverar ofta upprepad \textit{(outliers)}. I dessa situationer framstår den korrekt hypotesen som den mest sannolika först när ett stort antal mätningar beaktas. Robusta lösningar för dessa problem måste ta hänsyn till flera hypoteser samtidigt, vilket i sin tur kräver representation av multimodala fördelningar.


GM-SPA:s robusthet mot \textit{(outliers)} testas och jämförs med partikelfiltret (PF) och \textit{multimodal incremental smoothing and mapping} (MM-iSAM), båda är icke-parametriska metoder. Robusthet, noggrannhet och körtid förbättras i simuleringstester. Simulerade tester inkluderar 1D-lokalisering med okänd dataassociation, 3D linjär målföljning med korrelerade \textit{(outliers)} och 2D-ställningsuppskattning av endast räckvidd med Gaussiskt blandningsljud.

“Keywords[por ]”: €€€€
Soluções robustas para estimativa de estado devem lidar com medidas defeituosas, chamadas de \textit{(outliers)}, e com associações de dados desconhecidas, que levam a múltiplas hipóteses possíveis. Considere, por exemplo, o cenário de rastreamento de dois alvos indistinguíveis com base em medidas de posição, em que cada medida pode-se referir a qualquer um dos alvos ou até mesmo ser uma leitura defeituosa.

Métodos de estimativa comuns modelam o estado como tendo uma distribuição unimodal, sendo assim chamados de métodos unimodais. Da mesma forma, métodos multimodais modelam o estado como uma distribuição multimodal. Problemas difíceis, como a navegação de veículos subaquáticos autônomos (AUV) baseada em localização acústica, frequentemente envolvem \textit{(outliers)} recurrentes. Nestas situações, a hipótese correta apenas surge como a mais provável quando um número substancial de medidas é considerado. Soluções robustas para estes problemas precisam de considerar múltiplas hipóteses simultaneamente, o que, por sua vez, exige a representação de distribuições multimodais.

Neste trabalho, é apresentado um novo método de inferência aproximada, chamado \textit{Gaussian mixture sum-product algorithm} (GM-SPA), pois implementa o \textit{sum-product algorithm} (SPA) para misturas Gaussianas. O GM-SPA pode representar exatamente medidas lineares sub-determinadas e aproximar modelos não lineares importantes, como medidas de distância e cinemática de pose 2D.

A robustez a \textit{(outliers)} do GM-SPA é testada e comparada com o filtro de partículas (PF) e com \textit{multimodal incremental smoothing and mapping} (MM-iSAM), ambos métodos não-paramétricos. A robustez, a exatidão e o tempo de execução em testes de simulação são melhorados. Os problemas de teste incluem localização 1D com associação de dados desconhecida, rastreamento linear de alvos em 3D com \textit{(outliers)} correlacionados e estimativa de pose 2D com base em medidas de distância com ruído de mistura Gaussiana.

“Keywords[por ]”: €€€€
Estimativa robusta de estado, Localização subaquática, Rastreamento de alvos, Mistura Gaussiana, AUV €€€€,
Local Variables:
mode: LaTeX

The following command is used with glossaries\Hyphdash extra
\setabbreviationstyle[acronym]{long-short}
The form of the entries in this file is \newacronym{label}{acronym}{phrase}
or \newacronym[options]{label}{acronym}{phrase}
see "User Manual for glossaries.sty" for the details about the options, one example is shown below
\begin{itemize}
  \item \newacronym{smarc}{SMaRC}{Swedish Maritime Robotics Centre}
  \item \newacronym{auv}{AUV}{autonomous underwater vehicle}
  \item \newacronym{slam}{SLAM}{simultaneous localization and mapping}
  \item \newacronym{dvl}{DVL}{Doppler velocity log}
  \item \newacronym{ahrs}{AHRS}{attitude\Hyphdash heading reference system}
  \item \newacronym{usbl}{USBL}{ultra\Hyphdash short baseline}
  \item \newacronym{lbl}{LBL}{long baseline}
  \item \newacronym{ig}{IG}{improper Gaussian}
  \item \newacronym{rdg}{RDG}{reduced dimension Gaussian}
  \item \newacronym{iisam}{iSAM}{incremental smoothing and mapping}
  \item \newacronym{isa}{iSAM}{improved iSAM}
  \item \newacronym{isam2}{iSAM2}{incremental smoothing and mapping}
  \item \newacronym{mmisam}{MM\Hyphdash iSAM}{multimodal incremental smoothing and mapping}
  \item \newacronym{mmisp}{MM\Hyphdash ISPM}{multimodal sum\Hyphdash product algorithm}
  \item \newacronym{mmtf}{MM\Hyphdash TFS}{multimodal two\Hyphdash filter smoothing}
  \item \newacronym{mmmtf}{MM\Hyphdash TFS}{multimodal two\Hyphdash filter smoothing with low variance sampling}
  \item \newacronym{rosl}{RSL}{Robot Operating System}
  \item \newacronym{prm}{PRM}{probabilistic road map}
  \item \newacronym{hh}{HMM}{hidden Markov model}
  \item \newacronym{rf}{RF}{random forest}
  \item \newacronym{ekf}{EKF}{extended Kalman filter}
  \item \newacronym{ukf}{UKF}{unscented Kalman filter}
  \item \newacronym{pf}{PF}{particle filter}
  \item \newacronym{smc}{SMC}{sequential Monte Carlo}
  \item \newacronym{fbs}{FBS}{forward\Hyphdash backward smoothing}
  \item \newacronym{tfs}{TFS}{two\Hyphdash filter smoothing}
  \item \newacronym{nh}{NH}{null hypothesis}
  \item \newacronym{pe}{PRMSE}{position root mean square error}
  \item \newacronym{he}{HRMSE}{heading root mean square error}
  \item \newacronym{qrcp}{QRCP}{column pivoted QR} \end{itemize}