Scalable magnetic field mapping and localization using Gaussian process regression

Scalable magnetic field mapping and localization using Gaussian process regression

Proefschrift

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus prof. dr. ir. T.H.J.J. van der Hagen,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen
op maandag 25 november om 12:30

door

Frida Marie VISET

Dit proefschrift is goedgekeurd door de promotoren.

Samenstelling promotiecommissie:

Rector Magnificus,	voorzitter
Dr. M. Kok,	Technische Universiteit Delft, promotor
Dr. ir. R.L.J. Helmons,	Technische Universiteit Delft, promotor
Prof. dr. ir. J.W. van Wingerden,	Technische Universiteit Delft
Prof. dr. E. Cross,	University of Sheffield, UK
Prof. dr. ir. G. J. T. Leus,	Technische Universiteit Delft
Prof. dr. E. F. Brekke,	Norwegian University of Science and Technology
Prof. dr. ir. R. Toth,	Technische Universiteit Delft

Delft University of Technology

Not all those who wander are lost

Tolkien

Til Mamma, Pappa, Tarjei, Guro, Marius, Farmor og Mormor

Contents

Summary

This thesis scales methods for Gaussian process-based magnetic field mapping and localization in five distinct ways. Each way therefore answers one of the following questions related to scalability:

- **How can simultaneous magnetic field mapping and localisation be performed faster?** The first contribution seeks to perform simultaneous localisation and mapping faster than state of the art particle filters. The proposed solution is an extended Kalman filter. While the particle filter processes many different hypothesizes about what the position and map can be at any time instance, the extended Kalman filter only makes one estimate based on a linearization assumption. The conclusion is twofold and is based on theoretical arguments backed up with simulation results as well as estimates from real-world scenarios. Firstly, the extended Kalman filter is a faster and equally accurate option compared to state-of-the-art when the linearisation assumption holds. Secondly, the linearisation assumption holds when the initial position estimate has an estimation error which is lower than the lengthscale of the magnetic field, where the lengthscale encodes the minimum distance between any two locations such that the magnetic field values in those locations are likely to be considerably different from each other.

- **How can simultaneous magnetic field mapping and localization be performed by a multi-agent system?** Although the algorithms in the previous contribution can be run by several agents individually, the second contribution seeks to investigate how sharing information between the agents can improve the position and map estimates. This question is answered by two algorithms. The first algorithm performs centralized magnetic field simultaneous localization and mapping with an extended Kalman filter using the measurements from all agents by simply expanding the state-space of the model to encompass all agents' positions and orientations. The second algorithm is a distributed version of the first algorithm. It applies the filter across a multi-agent system by employing average consensus on the information form of the filter. Our experiments show that the centralized algorithm improves upon the individual estimates from each agent running simultaneous localization and mapping on its own. The distributed version of the algorithm requires frequent communication between agents to efficiently approximate the centralized algorithm, and thus improve upon the individual estimates. In our experimental setup with real magnetic field measurements from several agents and simulated communication, we investigate how much communication is required to efficiently approximate the centralized algorithm. We find that the distributed estimate accurately approximates the centralized estimate if all agents communicate once with each other at each timestep, and that the estimate from the distributed algorithm is more accurate than

the individual estimates when the agents communicate on average once every fifth timestep.

How can magnetic field maps be represented using less storage? The third contribution of this thesis answers this question by pointing out latent Hankel-Toeplitz structures of the maps used in previous contributions that can be used for lossless compression. We show that a reduced-rank Gaussian process representation with specific basis functions (including the ones used in the previous contributions) is equivalent to a sum of components, where each individual component can easily be compressed, because it has many repeated entries, due to its Hankel-Toeplitz structure. The result also applies to the representation of the magnetic field map shared between the agents in the previous contribution, allowing the algorithm to be implemented using a lower communication bandwidth.

How can magnetic field mapping and localization be extended to larger areas? State of the art Gaussian process mapping algorithms either require increasing computational resources as the mapped area grows, or are forced to split the area into smaller sub-maps with boundary effects and discontinuities degrading the field approximations near the edges of each sub-map. The fourth contribution of this thesis is an algorithm which keeps the computational complexity constant as the area grows larger without splitting the map into discrete sub-maps. The algorithm achieves this by combining a scalable method for including new measurements from the literature with a novel scalable method for predicting the map value. The scalable method for including new measurements from the literature uses finite-support basis functions near the measurement location to include each new measurement. The novel method for prediction uses finite-support basis functions near the prediction location. We demonstrate on real and simulated data that the algorithm reduces the required time to perform predictions compared to state of the art, and that it reduces the computational resources required to perform simultaneous magnetic field mapping and localization using measurements from a foot-mounted IMU in a large indoor area.

How can magnetic field maps be scaled to encompass repeated global patterns in addition to local variations? Where all the other contributions in this thesis for magnetic field mapping have only estimated local variations in the magnetic field, the final contribution is concerned with scaling the size of the area with confident and accurate magnetic field maps beyond the area where there are available measurements of the field. The question is answered through an algorithm for joint online learning of repeated global patterns and local variations in the magnetic field. This algorithm is implemented with a novel pattern-discovery kernel which is inherently equivalent to a parametric model. The fact that this kernel is inherently parametric means that it allows for computationally efficient online learning of the repeated global patterns without further approximation. The ability of the algorithm to create maps of globally repeated patterns and local variations online is demonstrated on simulated and real data.

Samenvatting

Deze scriptie onderzoekt methoden voor Gaussian process mapping en lokalisatie van magnetische velden op vijf verschillende manieren.

🕒 **Hoe kan gelijktijdige magnetisch mapping en lokalisatie sneller worden uitgevoerd?** De eerste bijdrage probeert gelijktijdige lokalisatie en mapping sneller uit te voeren dan de huidige state-of-the-art particle filter; een Extended Kalman Filter. Terwijl het particle filter veel verschillende hypothesen verwerkt over wat de positie en kaart op elk moment kunnen zijn, maakt de Extended Kalman Filter slechts één schatting op basis van een linearisatie. De conclusie is onderbouwd door middel van theoretische argumenten en experimentele data. Ten eerste is de Extended Kalman Filter sneller en net zo nauwkeurig vergeleken met de state-of-the-art, met de aanname dat de linearisatie aanname geldt. Ten tweede geldt de linearisatie aanname wanneer de initiële positiemeting een schattingsfout heeft die kleiner is dan de length-scale van het magnetisch veld, waarbij de length-scale de minimale afstand tussen twee locaties is, zodat de magnetische veldwaarden op die locaties waarschijnlijk aanzienlijk van elkaar verschillen.

☆ **Hoe kan gelijktijdige magnetisch mapping en lokalisatie worden uitgevoerd door een multi-agent systeem?** De vorige bijdrage onderzocht de bijdrage van algoritmen op individuele agenten. Dit hoofdstuk onderzoekt hoe het delen van informatie tussen meerdere agenten kan bijdragen, met focus of positie- en kaartmetingen. Deze vraag wordt beantwoord door twee algoritmen. Het eerste algoritme voert gecentraliseerde magnetisch veld gelijktijdige lokalisatie en mapping uit met een Extended Kalman Filter door alle metingen van alle agenten te gebruiken door simpelweg de state space van het model uit te breiden om de posities en oriëntaties van alle agenten mee te nemen. Het tweede algoritme is een gedistribueerde versie van het eerste algoritme. Het past het filter toe in een multi-agent systeem door gebruik te maken van gemiddelde consensus over de informatievorm van het filter. Onze experimenten tonen aan dat het gecentraliseerde algoritme beter presteert dan de individuele schattingen van elke agent die gelijktijdige lokalisatie en mapping op zichzelf uitvoert. De gedistribueerde versie van het algoritme vereist frequente communicatie tussen agenten om efficiënt het gecentraliseerde algoritme te benaderen. In onze experimentele opstelling met echte magnetische veldmetingen van meerdere agenten en gesimuleerde communicatie onderzoeken we hoeveel communicatie nodig is om efficiënt het gecentraliseerde algoritme te benaderen. We ontdekken dat de gedistribueerde schatting nauwkeurig het gecentraliseerde algoritme benadert als alle agenten met elkaar communiceren bij elke tijdstap, en dat de schatting van het gedistribueerde algoritme nauwkeuriger is dan de individuele schattingen wanneer de agenten gemiddeld eens in de vijf tijdstappen communiceren.

Hoe kunnen magnetische veldkaarten met minder opslag worden weergegeven? De derde bijdrage van dit proefschrift beantwoordt deze vraag door de Hankel-Toeplitz structuren van de kaarten die in eerdere bijdragen werden gebruikt, en die kunnen worden gebruikt voor compressie zonder verlies van informatie. We tonen aan dat een reduced-rank Gaussian process representatie met specifieke basis functions equivalent is aan een som van componenten, waarbij elke individuele component gemakkelijk kan worden gecomprimeerd, omdat het veel herhalingen heeft vanwege de Hankel-Toeplitz structuur. Het resultaat geldt ook voor de representatie van de magnetische veldkaart gedeeld tussen de agenten in de vorige bijdrage, waardoor het algoritme kan worden geïmplementeerd met een lagere communicatiedoorvoer.

Hoe kan magnetisch mapping en lokalisatie worden uitgebreid naar grotere gebieden? State-of-the-art Gaussian process mapping algoritmen vereisen meer rekenkracht naarmate de kaart groeit, of de algoritmen worden gedwongen om het gebied op te splitsen in kleinere sub-kaarten met rand effecten en discontinuïteiten die de veldbenaderingen in de buurt van de randen van elke sub-kaart negatief aantasten. De vierde bijdrage van dit proefschrift is een algoritme dat de rekencomplexiteit constant houdt naarmate het gebied groter wordt zonder de kaart op te splitsen in afzonderlijke sub-kaarten. Het algoritme bereikt dit door een schaalbare methode uit de literatuur te combineren voor het opnemen van nieuwe metingen met een nieuwe schaalbare methode voor het voorspellen van de kaartwaarde. De eerder voorgestelde methode voor het opnemen van nieuwe metingen maakt gebruik van basis functions met finite support in de buurt van de meetlocatie om elke nieuwe meting op te nemen. De nieuwe methode voor voorspelling maakt gebruik van basis functions met finite support in de buurt van de voorspellingslocatie om elke voorspelling te maken. We demonstreren op echte en gesimuleerde data dat het algoritme de benodigde tijd voor het maken van voorspellingen vermindert in vergelijking met de state-of-the-art, en dat het de benodigde rekenkracht vermindert om gelijktijdige magnetisch mapping en lokalisatie uit te voeren met behulp van metingen van een voetgemonteerde IMU in een grote ruimte.

Hoe kunnen magnetische veldkaarten worden opgeschaald om herhaalde globale patronen naast lokale variaties te omvatten? Waar alle eerder besproken algoritmen voor magnetisch mapping alleen lokale variaties in het magnetisch veld hebben geschat, is de laatste bijdrage in dit proefschrift gericht op het opschalen van de voorspellingen buiten het gebied waar er metingen van het veld zijn. De vraag wordt beantwoord door een algoritme voor gezamenlijke online learning van herhaalde globale patronen en lokale variaties in het magnetisch veld. Dit algoritme is geïmplementeerd met een nieuw pattern-detection kernel die inherent equivalent is aan een parametermodel. Het feit dat deze kernel inherent patameteric is, betekent dat het een computationeel efficiënte online learning van de herhaalde globale patronen zonder verdere benadering mogelijk maakt. Het vermogen van het algoritme om kaarten van wereldwijd herhaalde patronen en lokale variaties online te creëren, wordt gedemonstreerd op gesimuleerde en echte data.

Acknowledgments

When I got offered a position as a PhD candidate at the faculty of DCSC in the spring of 2020, on the last day before lockdown, I was told by prof. J. W. van Wingerden that a PhD is a journey. I nodded and signed the contract, thinking this was an exaggeration. Someone climbing Mount Everest, or recovering from severe addiction is on a journey. My plan was to show up to the office on weekdays to read some papers, talk to some people and implement some algorithms.

Well. It's been a journey. During the last four years, life has occasionally felt normal, but most of the time, it has felt like a movie filled with unbelievable plot twists and incredible characters. I would like to give my sincerest thanks to the cast.

Thank you Scarlett, since the time we shared a hiking path in the French mountains, talking to you has always made me feel as free as if I'm back there. Thank you Claudia, for being so colorful, kind, open-minded and blazingly smart. I am myself when I am with you. Thank you Leonore, the cool girl I didn't dare to talk to at first, for becoming one of my best friends in Delft. Thanks to Justus and Charlie, you crazy hippies, for great nights out, calm nights in and everything in between. Always trust your intuition Charlie! Thanks to Marieke, for sharing sunny bike rides, books and music, and for conversations about all the nice things in life, and all the hard stuff too. Thanks to Marceline for the rainy bike rides, and for showing me how exciting a journey becomes when you deliberately pick the harder route. Thanks to Marco for keeping me sane through the pandemic with our long walks, and for teaching me the single most important professional lesson I ever learned: If you can't make it work, try something simpler first. Thanks to Raoul, Jan, Jacomijn and Berna for dinner parties, camping, dancing and singing and for making life here so much fun. Thanks to my sister Silvia for phone calls, visits, trips, cooking classes and constant support. Thanks to Hanne, who has always been there on the phone, who I greatly admire for her kindness and creative talent, and who will hate that I wrote that down. Thanks to Eva and Pim, for teaching me to embrace neurodivergence rather than fight it. Thanks to Andrei, Sasha, Tadas, Phil, Hana, Zuza, Marnix, Noreen, Rocco, Zohar, Juanjo and Scar, for inviting me into your climbing cult in Switzerland – I mean family – with open hearts, chalkbags and big pots of risotto. And thanks to Jonas, Emanuel, Marcus, David and Samuel for the climbing in the Netherlands. Thanks to Diletta, Ombretta and Jonathan for good memories from Oslo, Italy and Delft. Thanks to Kia, it has been so fun running into you everywhere. And thanks to Sasan, Ilias, Lotfi, Emilio, Frederik and Alex for the karaoke energy! And thank you Alex! For the bear hugs, for great times. Thanks to Afra, Zoe and Zhixin for tea breaks and great stories. Thank you Noël, for your letters. Thanks to Zhongang, Edo and Costas for the shared music!

Thanks to my supervisors Manon and Rudy, for giving me the job in the first place, for giving me close to unlimited freedom to explore, while also giving support to make things happen whenever I needed it. I would also like to thank the three talented master students I was lucky enough to supervise, Niels, Danny and Sing-Chi. Thanks also to Mostafa Osman, Anton Kullberg, Gustaf Hendeby, Isaac Skog, Frederiek Wesel and Arno Solin for inspiring collaborations. Thanks to Lars for helpful discussions on functional analysis. A special thanks to Manon, Frederiek and Anton, who spend the most time by the whiteboard with me over these years, where the real magic happens. Thanks to Gustaf Hendeby, Johanna, Kailai and everyone else in Linköping for welcoming me to Sweden for a research visit, and for quickly including me both professionally and socially into your group. Thanks also to my group in Delft, Thomas, Clara, Haoyu and Zoe for making the best research environment one could ask for. Also thanks to my family, who I dedicated this thesis to, for everything. Finally, thanks to Fred, for sharing this journey with me, for joining me for the next one, and for fixing my bike whenever it was broken.

At the end of this journey, I would like to come home, my home being somewhere between all the people I am deeply connected with, that are all scattered from Trondheim in the north to Milan in the south. I have settled on Oslo as a geographical midpoint. The adventures continue from here!

Frida
Oslo, October 2024

1 Introduction

Outdoors, the magnetic field can help navigation because it points towards the magnetic north, with the strength of the earth magnetic field. In environments that have elements that perturb the magnetic field, the field can point in various directions, and have various strengths depending on the precise location within that environment. These invisible fields can therefore be used to improve the localization of vehicles, people and robots in environments and situations where typical means of navigation such as satellite signals or pre-employed navigation infrastructure are unavailable. Gaussian process regression enables creation of maps of the nonlinear magnetic field variations by combining measurements with prior knowledge about the nature of the field. Existing literature provides solutions for creating maps of the nonlinear magnetic field variations, and using these maps to improve position estimates. This thesis works towards scaling magnetic field mapping and localization in five distinct ways: Making it faster, distributed, require less storage, cover larger areas and enable extrapolation based on repeated patterns.

1.1 Why do we need magnetic field mapping and localization?

Most pedestrians, robots and vehicles (generally refered to in this thesis as agents) obtain position estimates outdoors through a combination of measurements from on-board sensors and GNSS (Global Navigation Satellite Systems). In a range of scenarios, GNSS signals are blocked or disturbed by various elements. Inside buildings, signals are blocked by walls or ceilings [2]. In ports or densely built areas, signals are either blocked by buildings or reflected in ways that cause interference with the measurements. Only a few centimeters underwater, GNSS signals are blocked by the water itself [3].

When the GNSS signals are no longer usable, agents are left only with onboard measurements to perform their localization. Most available sensors that do not rely upon external infrastructure give estimates either of change of velocity (for example accelerometers), change in orientation (gyroscope) or change of position (for example step-detection systems for pedestrians or legged robots, doppler velocity logs in underwater vehicles, wheel encoders, or cameras detecting optical flow). A common problem for any algorithms using either each of these sensing modes individually or in combination is that there are no absolute sources of position measurements [4]. All measurements are of the relative change in position and orientation. This means that to obtain estimates of the absolute position and orientation, these measurements have to be added up over time. Each of these measurements are typically perturbed with a small measurement error. Even though each of these errors individually are small, adding all of them up over time eventually leads to an error in the absolute position estimate that can increase without an upper bound. This phenomenon is called drift [5].

Outdoors, agents can use a compass as a simple magnetic field sensor to figure out which way north is. That is because outdoors, the magnetic field is typically pointing more or less in the same direction over a given area [6]. In indoor environments surrounded by structural steel elements in floors or walls, or underwater close to magnetic disturbances from buried ships, mooring chains, or close to metallic containers or near ports with metallic elements, the magnetic field is typically heavily perturbed, making it impossible to use the magnetic field sensor as a compass. In these environments, the magnetic field changes with position, in a complex nonlinear pattern. Some examples of the way the magnetic field varies with the position are displayed in Figure 1.1.

If this pattern would be known, it would be possible to correct the orientation estimate of an agent using the knowledge of the "true" orientation of the magnetic field. What makes these magnetic field maps truly interesting, however, is the fact that the spatial variations also make it possible to improve the position estimates of these agents using the magnetic field sensor [7].

The contributions of this thesis are concerned with creating scalable magnetic field maps from one or more agents moving through the magnetic field carrying a magnetic sensor. Some of the contributions also use the magnetic field map to improve the position estimates of these agents.

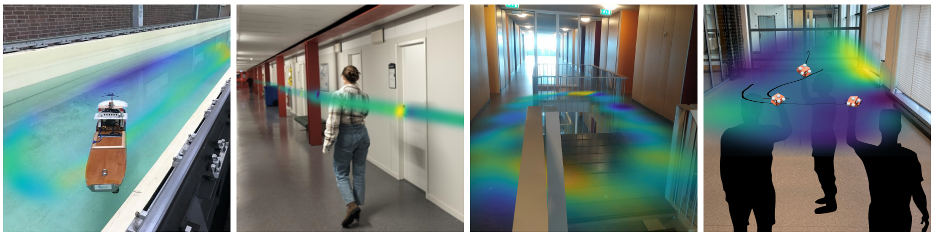


Figure 1.1: Magnetic field variations in various indoor environments.

1.2 Why do we want scalable magnetic field mapping and localization?

Current algorithms for magnetic field mapping and localization can generate magnetic field maps on the fly and use them to improve the position estimates of an agent [8]. The magnetic field map is created using measurements from a magnetometer carried by the agent. In addition, physical information about the magnetic field can be included in the map by using Gaussian process regression [9, 10]. Although Gaussian process regression typically requires an increasing amount of computational resources given an increasing amount of measurements, it is possible to perform the estimation online in a domain of finite size without requiring an ever-increasing amount of computational resources using basis-function approximations of Gaussian process regression [11].

Even with the use of basis functions, the current algorithms ([8, 12–14]) are limited by distinct scalability challenges. They have high computational requirements at each timestep, they currently only use measurements from and for a single agent, they use large amounts of storage, they can not map larger areas without increasing the computational complexity or splitting the areas into sub-maps, and they are limited to creating maps within the areas that have already been explored.

This thesis contains five distinct contributions, where each contribution is a way of improving the applicability of the currently available technology for magnetic field mapping and/or localisation using Gaussian processes by improving the scalability. The contributions respectively perform magnetic field mapping and localisation using less time at each online iteration, use multiple agents to perform magnetic field mapping and localisation using less time to collect data, create magnetic field maps using less storage, create maps covering larger areas using less computational power, and learn repeated patterns in the magnetic field using fewer measurements of the map in structured environments.

The five contributions of this thesis improve on each of these five aspects of scalability. Each of the types of scalability enables achieving more with less. The contributions enable efficient online magnetic field mapping and localisation, using a small amount of computational resources, in a collaborative manner. In all contributions, the magnetic field map is created using Gaussian process regression with basis functions. Some of the contributions apply known Gaussian processes algorithms in novel ways, while other contributions take a deep-dive into the algorithm and change the way fundamental computations are performed to improve scalability. The next part of this introductory chapter gives an introduction to map building using Gaussian process regression with basis functions.

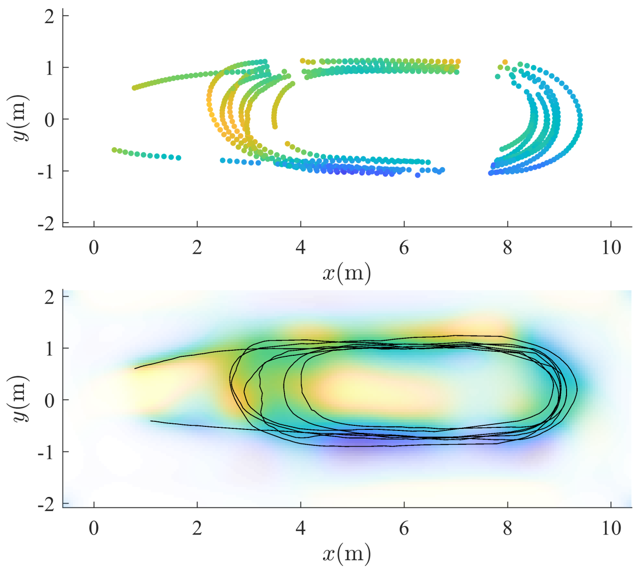


Figure 1.2: Magnetic field map based on magnetic field measurements. The color corresponds to the intensity of the magnetic field, while the opacity is proportional to the confidence of the prediction.

1.3 Building magnetic field maps with Gaussian process regression

In open air, the magnetic field is curl free, and is therefore the gradient of a potential field $f : \mathbb{R}^3 \rightarrow \mathbb{R}$ [15]. This means that magnetic field measurements $\boldsymbol{y}_{\text{mag}}$ in a position $\boldsymbol{p}$ are given by

$$\boldsymbol{y}_{\text{mag}} = \nabla_{\text{p}} f(\boldsymbol{p}) + \boldsymbol{e}_{\text{mag}}, \quad \boldsymbol{e} \sim \mathcal{N}(\boldsymbol{0}, \sigma_{\text{y}}^2 \boldsymbol{I})) \tag{1.1}$$

where $\boldsymbol{p} \in \mathbb{R}^3$ is the position of the sensor, ∇ is the gradient operator, and $\boldsymbol{e}_{\text{mag}}$ is a measurement noise with covariance $\sigma_{\text{y}}^2 \boldsymbol{I}$. An example of the norm of such measurements measured by the model ship in Figure 1.1 is displayed to the left in Figure 1.2. Each colored dot corresponds to one measurement of the magnetic field. The position of each dot in the xy-plane of the figure corresponds to the position of the ship observed by an external motion capture system when the measurement was made. The color of the dot changes gradually from blue to yellow with increasing measured magnitude of the magnetic field in that location. The ship was being steered in a repeated pattern, and therefore visited the same area multiple times. As is visible in Figure 1.2 from the fact that two dots right next to each other typically have a similar color, the magnetic field amplitude is often similar to itself in nearby locations. This property of a nonlinear function is often referred to as smoothness, and can be included as an assumption in Gaussian process regression to create a map of an area which is larger than the exact locations where we have access to measurements [16]. When there is a large amount of measurements, it improves the computational efficiency to use basis function approximations to Gaussian process regression [11]. This also enables online learning of magnetic field map by including values associated with the basis function model in the state space [10]. If we use the magnetic field measurements to perform Gaussian process regression (with basis functions) to predict $f(\boldsymbol{p})$, we obtain a magnetic field map as the one displayed to the right in Figure 1.2.

The magnetic field map displayed in the image is a visualization of the norm of the gradient of an estimate $\hat{f}(\boldsymbol{p})$ of the nonlinear potential $f(\boldsymbol{p})$. That means that the color hue changes proportionally with $\|\hat{f}(\boldsymbol{p})\|_2$, and the opacity is proportional with the confidence of this estimate in each location.

The following subsections in this chapter give an introduction to Gaussian process regression with basis functions. The advanced reader experienced in statistical estimation and Gaussian process regression is encouraged to skip forward to the last subsection of this chapter. Understanding Gaussian process regression with basis functions requires understanding least squares estimation and use of basis functions to model functions. Subsection 1.3.1 therefore first introduce least-squares estimation, subsection 1.3.2 then introduces basis function representations of nonlinear functions, and finally Gaussian process regression with basis functions are introduced in subsection 1.3.3. This is the method that has been used to create the magnetic field maps presented in Figure 1.1.

1.3.1 Linear estimation

We consider a normally distributed M-dimensional vector $\boldsymbol{w} \in \mathbb{R}^M$ with mean $\boldsymbol{\mu}$ and variance $\boldsymbol{\Lambda}$

$$\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Lambda}). \tag{1.2}$$

We use the operator Covar($\boldsymbol{w}$) to indicate the variance of a vector, which in general is defined as

$$\boldsymbol{\Lambda} = \text{Covar}(\boldsymbol{w}) = E[(\boldsymbol{w} - E[\boldsymbol{w}])(\boldsymbol{w} - E[\boldsymbol{w}])^\top], \tag{1.3}$$

where E is the expected value operator. Each entry of the covariance matrix $\boldsymbol{\Lambda}$ corresponds to the variance between each individual entries in the vector $\boldsymbol{w}$, according to

$$\Lambda_{i,j} = \text{Covar}(w_i, w_j) = E[(w_i - E[w_i])(w_j - E[w_j])^\top], \tag{1.4}$$

where w_i is the ith entry of the vector $\boldsymbol{w}$, and $\Lambda_{i,j}$ is the entry at the ith row and the jth column of the covariance-matrix $\boldsymbol{\Lambda}$.

Note that if we consider some arbitrary matrix $\boldsymbol{G} \in \mathbb{R}^{N\times M}$, then the linear combinations of the entries in vector $\boldsymbol{w}$ given by the matrix-vector product $\boldsymbol{G}\boldsymbol{w}$ is distributed according to

$$\boldsymbol{G}\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{G}\boldsymbol{w}, \boldsymbol{G}\boldsymbol{\Lambda}\boldsymbol{G}^\top), \tag{1.5}$$

which will be used in a later part of this chapter to consider stochastic linear combinations of basis functions.

Linear least squares estimate of a vector

Assume we have an unknown vector $\boldsymbol{w} \in \mathbb{R}^M$, and have access to N measurements of this vector according to

$$\boldsymbol{y} = \boldsymbol{H}\boldsymbol{w} + \boldsymbol{e}, \quad \boldsymbol{e} \sim \mathcal{N}(0, \sigma_{\text{y}}^2 \boldsymbol{I}), \tag{1.6}$$

where $\boldsymbol{e}$ is a measurement noise vector with a variance σ_{y}^2. Then, if $\boldsymbol{H}^\top \boldsymbol{H}$ is invertible, $\boldsymbol{w} = (\boldsymbol{H}^T\boldsymbol{H})^{-1}\boldsymbol{H}^\top(\boldsymbol{y}-\boldsymbol{e})$, meaning that by applying (1.5) the least squares estimate of $\boldsymbol{w}$ is given by

$$\boldsymbol{w} \sim \mathcal{N}(\hat{\boldsymbol{w}}, \boldsymbol{P}), \quad \hat{\boldsymbol{w}} = (\boldsymbol{H}^T\boldsymbol{H})^{-1}\boldsymbol{H}^\top\boldsymbol{y}, \quad \boldsymbol{P} = \sigma_y^2(\boldsymbol{H}^\top\boldsymbol{H})^{-1}, \tag{1.7}$$

where $(\boldsymbol{H}^\top\boldsymbol{H})^{-1}$ is often referred to as the Fischer information, or the information matrix of the system.

Linear least squares of a vector with prior information

If we have a vector $\boldsymbol{w} \in \mathbb{R}^M$, and have access both to the measurements in (1.6) and some prior estimate of the vector which has a distribution $\hat{\boldsymbol{w}}_0 \sim \mathcal{N}(\boldsymbol{w}, \boldsymbol{\Lambda})$, then this prior information can be used to construct an extended measurement model as

$$\begin{bmatrix} \hat{\boldsymbol{w}}_0 \\ \boldsymbol{y} \end{bmatrix} = \begin{bmatrix} \boldsymbol{I} \\ \boldsymbol{H} \end{bmatrix} \boldsymbol{w} + \begin{bmatrix} \boldsymbol{e}_0 \\ \boldsymbol{e} \end{bmatrix}, \quad \begin{bmatrix} \boldsymbol{e}_0 \\ \boldsymbol{e} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{0} \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Lambda} & \boldsymbol{0} \\ \boldsymbol{0} & \sigma_y^2 \boldsymbol{I} \end{bmatrix} \right), \tag{1.8}$$

and the least squares estimate in this case can therefore be derived to be given by

$$\boldsymbol{w} \sim \mathcal{N}(\hat{\boldsymbol{w}}, \boldsymbol{P}), \quad \hat{\boldsymbol{w}}_1 = (\boldsymbol{H}^T\boldsymbol{H} + \sigma_y^2\boldsymbol{\Lambda})^{-1}\boldsymbol{H}\boldsymbol{y}, \quad \boldsymbol{P} = \sigma_y^2(\boldsymbol{H}^\top\boldsymbol{H} + \sigma_y^2\boldsymbol{\Lambda})^{-1}, \tag{1.9}$$

where the matrix $(\boldsymbol{H}^\top\boldsymbol{H} + \sigma_y^2\boldsymbol{\Lambda}^{-1})^{-1}$ is referred to as the information matrix, and the vector $\boldsymbol{H}^\top\boldsymbol{y}$ is referred to as the information vector.

1.3.2 Representing nonlinear functions with basis functions

Various methods for Gaussian process regression use various types of basis functions. Four examples of possible basis functions are displayed in Figure 1.3. From left to right, the Gaussian process basis functions are Fourier features [17], Hilbert space basis functions [11], hat basis functions [18, 19] and squared-exponential basis functions [20, 21]. Nonlinear functions in one dimension can in general be modeled as a superposition of basis functions $\{\phi_i\}_{i=1}^m$.

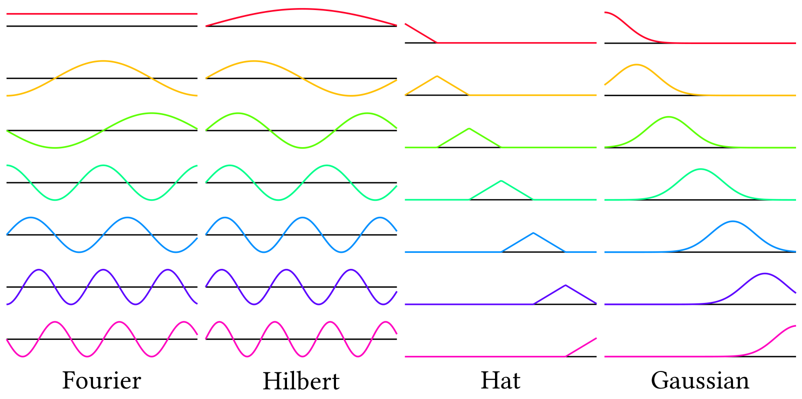


Figure 1.3: Four different examples of sets of basis functions that can be used to approximate an arbitrary nonlinear function.

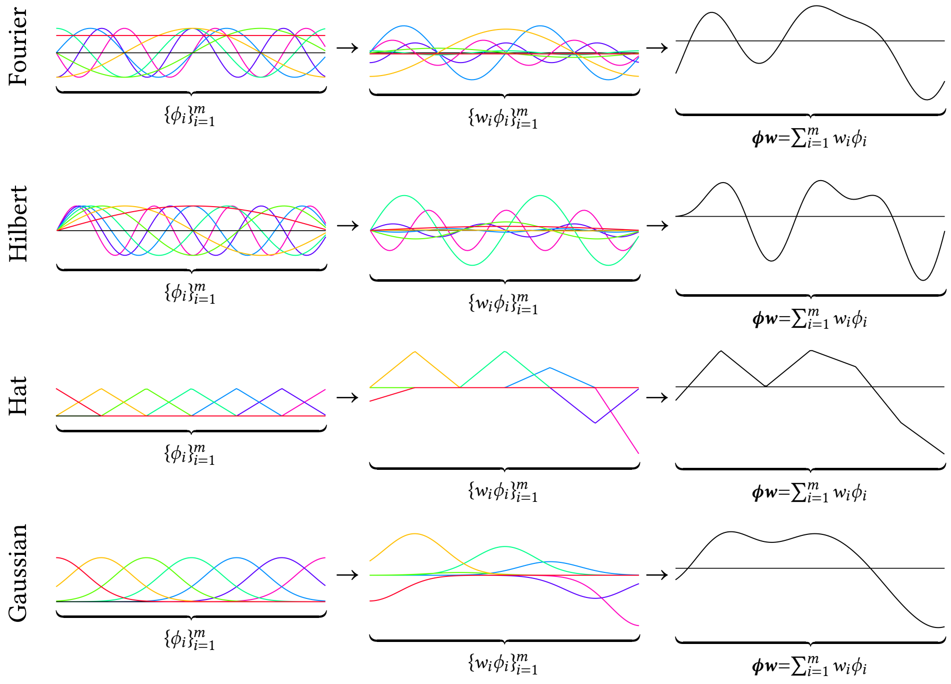


Figure 1.4: Scaling and summing together basis functions creates nonlinear functions whose shape is defined by the shape of the basis functions and the scaling factors (weights). Each row shows how scaling and summing together a different set of basis functions can create a nonlinear function with a shape that is distinct from either of the basis functions on it's own. From top to bottom, the considered basis functions are Fourier basis functions, Hilbert space basis functions, Gaussian basis functions and hat basis functions.

Consider a set of Fourier basis functions $\{\phi_i\}_{i=1}^{\infty}$, where the first 7 basis functions are visualized in Figure 1.3. Each $\phi_i : \Omega \to \mathbb{R}$, where $\Omega \subset \mathbb{R}^d$. The formulas for the basis functions (the Fourier features) are given by [17]

$$\{\phi_i\}_{i=1}^{\infty} = \{1, \cos(x), \sin(x), \cos(2x), \sin(2x), \cos(3x), \sin(3x), \ldots\} \tag{1.10}$$

A finite number of basis functions can be used to approximate any nonlinear function mapping $f : \Omega \to \mathbb{R}$, where Ω is a section of the real line between $[a,b]$, by considering a linear combination of the basis functions. The approximation defined as a weighted sum of the basis functions is therefore in general given as

$$\tilde{f}(x) = \sum_{i=1}^{m} w_i \phi_i(x) = \boldsymbol{\phi}(x)\boldsymbol{w} \tag{1.11}$$

where w_i are the weights associated with each basis function, $\boldsymbol{w} = [w_1, \cdots, w_m]^\top$, and $\boldsymbol{\phi} = [\phi_1(x), \cdots, \phi_m(x)]$. An illustration of the effect of weighting and summing together 7 basis functions to create a nonlinear function is given in Figure 1.4. If the function f is known, and the basis functions are orthonormal, the weights given by the projection

$$w_i = \int_\Omega \phi_i(x) f(x) dx \tag{1.12}$$

will minimize the cost function

$$\int_\Omega (f(x) - \tilde{f}(x))^2 dx. \tag{1.13}$$

As $m \to \infty$, the value of this cost function will go to zero. We therefore say that $f = \sum_{i=1}^{\infty} w_i \phi_i$ is a basis-function expansion of f. An example of how this property holds for all the four different basis functions approximating a nonlinear function is displayed in Figure 1.5. For the orthonormal functions, the weights in Figure 1.5 were computed using a numerical approximation to functions that are not inherently orthonormal (Hat and Gaussian), the weights were computed using (1.12). For the non-orthonormal basis functions, the weights were computed using a numerical approximation to the equation

$$\boldsymbol{w} = \left(\int_\Omega \boldsymbol{\phi}\boldsymbol{\phi}^\top \right) dx \int \boldsymbol{\phi} f dx, \tag{1.14}$$

which effectively orthogonalises the bases, and projects the nonlinear function onto the orthogonalised basis, and then maps from the weights required to scale the orthogonalised basis to the weights required to scale the original, non-orthogonal basis.

A simple way to extend basis functions to model nonlinear functions in a hypercube is given by the Kronecker-product of these basis functions, according to

$$\boldsymbol{\phi}(x) = \otimes_{d=1}^{D} \boldsymbol{\phi}^{(d)}(x^{(d)}), \tag{1.15}$$

where d is an index running through each of the D dimensions of $\boldsymbol{x} \in \mathbb{R}^D$. The resulting basis functions $\boldsymbol{\phi}(x)$ for the first 4 basis functions for Fourier, Hilbert space, Gaussian and hat basis functions is displayed in Figure 1.6. Also in higher dimensions, scaling and summing together basis functions can approximate nonlinear functions, as displayed in Figure 1.7

1.3.3 Gaussian processes explained in terms of basis functions

In the previous subsection, we looked at how we can represent a deterministic nonlinear function with basis functions, using a deterministic set of weights $\{w_i\}_{i=1}^m$. If we instead consider a set of normally distributed weights with a mean and a covariance given by

$$\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Lambda}), \tag{1.16}$$

the nonlinear function that we produce by taking the corresponding weighted average of the basis functions becomes a normally distributed nonlinear function, with a mean and covariance given by

$$\tilde{f} \sim \mathcal{N}(\boldsymbol{\phi}\boldsymbol{\mu}, \boldsymbol{\phi}\boldsymbol{P}\boldsymbol{\phi}^\top). \tag{1.17}$$

This result arises from the fact that the function evaluation at any set of locations $x_1, x_2, \ldots x_N$ is a linear combination of the normally distributed weights. As we know from (1.5), the

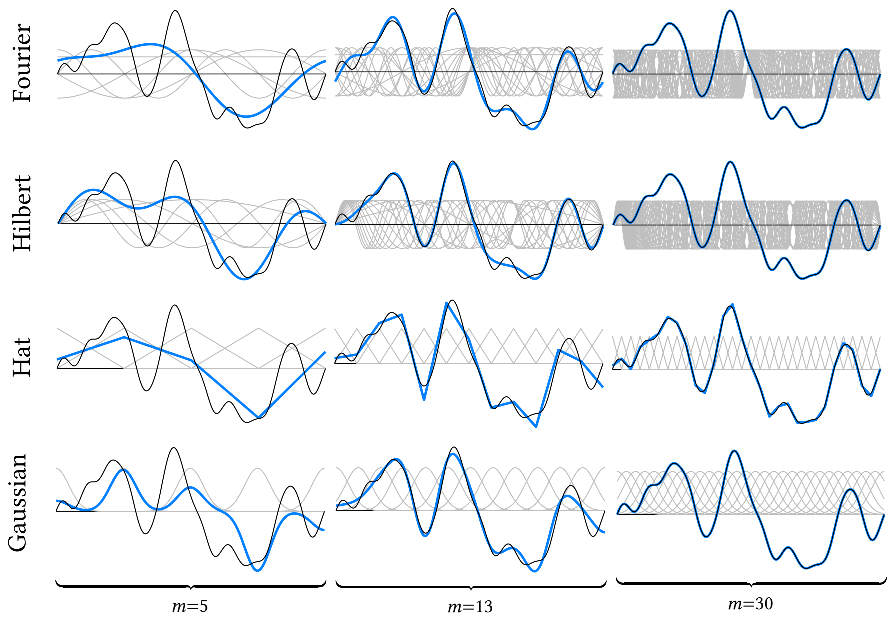


Figure 1.5: No matter which of these four basis function types you consider, an increasing amount of basis functions improves the capability of the function approximation to capture high frequency content in a nonlinear function. The black line shows the function f approximated by all methods. The blue line shows the approximate function. The gray lines show the basis functions used to construct the approximate function.

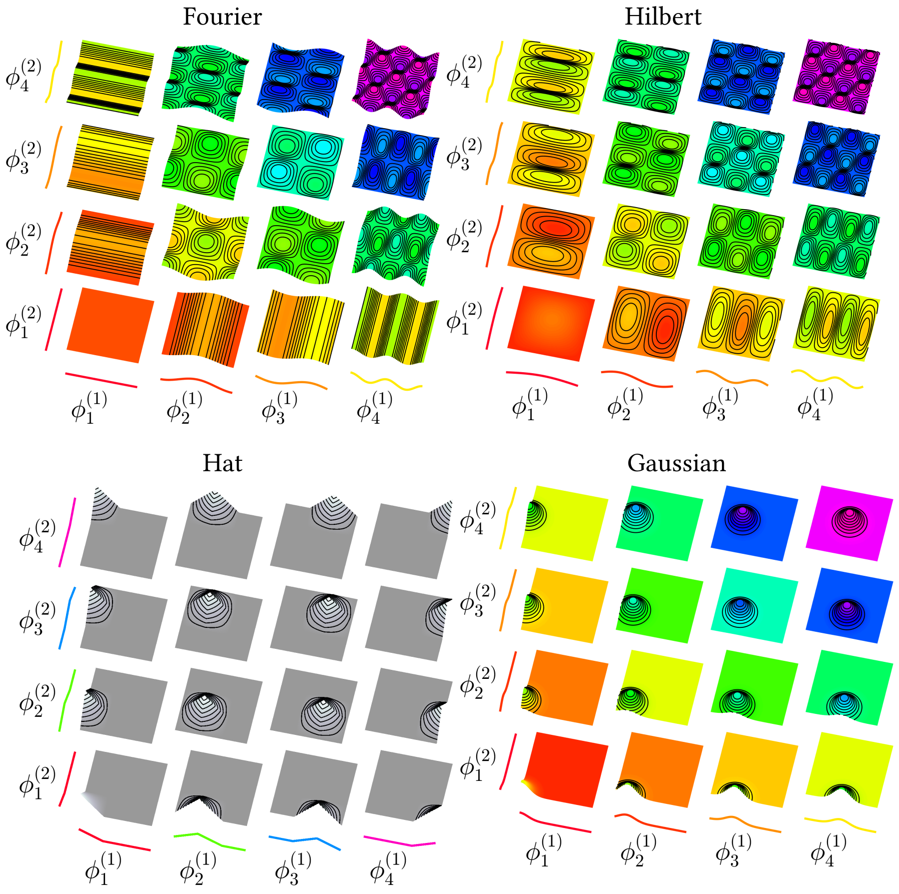


Figure 1.6: Basis functions for modeling functions in a higher dimensional hypercube can be constructed by studying the Kronecker product of basis functions along a single dimensions.

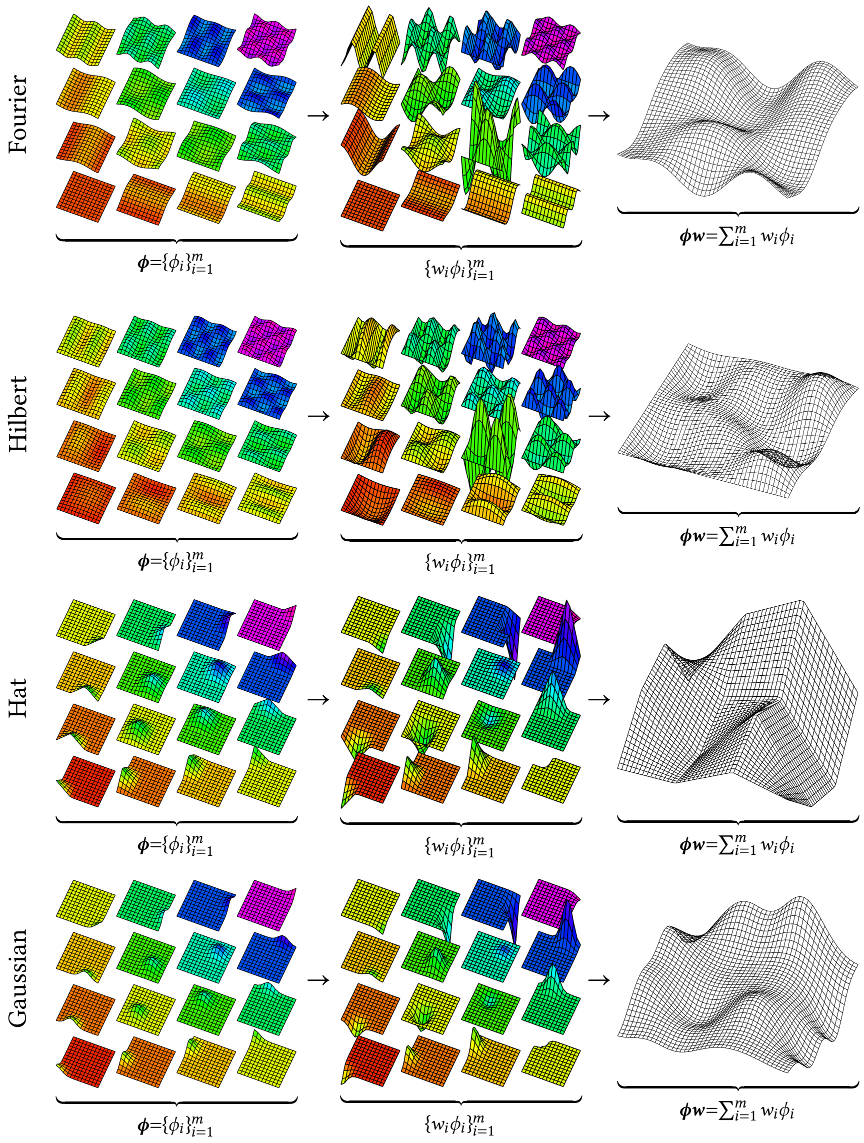

Fourier
Hilbert
Hat
Gaussian
$\boldsymbol{\phi}=\{\phi_i\}_{i=1}^m$
$\{w_i\phi_i\}_{i=1}^m$
$\boldsymbol{\phi w}=\sum_{i=1}^m w_i\phi_i$
$\rightarrow$

Figure 1.7: Scaling and summing together basis functions results in nonlinear functions with shapes that are distinct from each individual basis function also in higher dimensions. This figure displays from top to bottom the result of scaling and summing together 2-dimensional Fourier, Hilbert-Space, Gaussian and Hat basis functions.

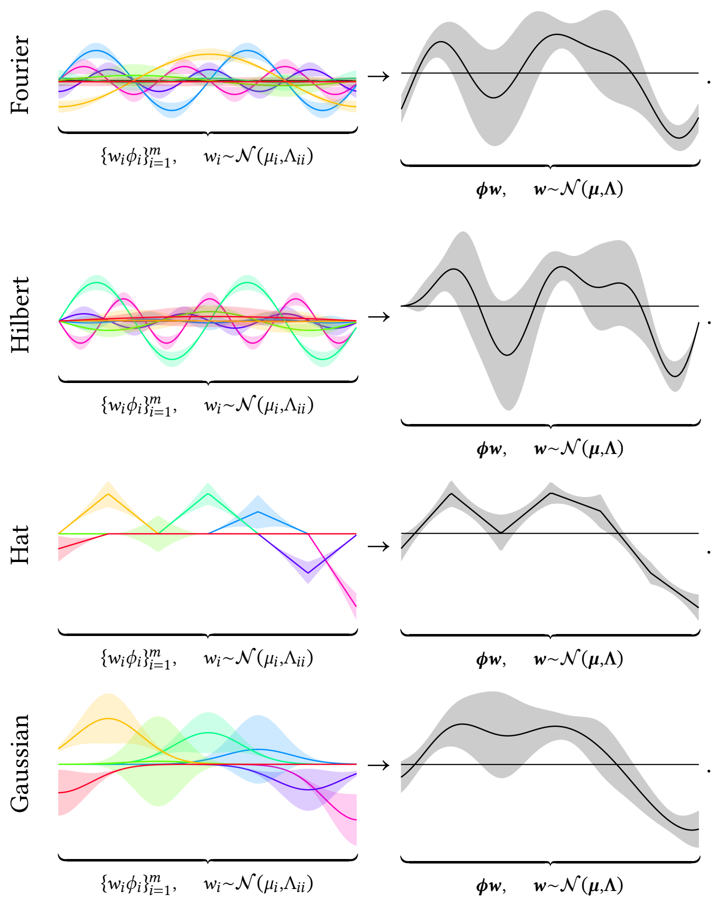


Figure 1.8: Scaling and summing together basis functions with normally distributed weights gives a normally distributed nonlinear function.

linear combination of a normal distribution is itself a normal distribution. The mean and variance of this nonlinear function are displayed in Figure 1.8. This concept extends naturally to functions in a hypercube in higher dimension, by using the Kronecker-basis functions from (1.15), as illustrated in Figure 1.9.

In general, a nonlinear function with a normal distribution is called a Gaussian process. Where a vector has a normal distribution we can define by a mean vector and a covariance matrix, a Gaussian process has a mean defined by a mean function f_μ, and a covariance defined by a covariance function $\kappa(x,x')$

$$f \sim \mathcal{N}(f_\mu, \kappa(x,x')). \tag{1.18}$$

Although linear combinations of basis functions are themselves Gaussian processes, it is common practice in literature [17, 18, 20, 22] to first define a Gaussian process in terms of the kernel function $\kappa(x,x')$, and afterwards select basis functions $\boldsymbol{\phi}(x)$ and a covariance matrix $\boldsymbol{\Lambda}$ such that this kernel matrix is approximately represented by the basis functions model, i.e. $\kappa(x,x') \approx \boldsymbol{\phi}(x)\boldsymbol{\Lambda}\boldsymbol{\phi}(x')^\top$. The way this approximation is picked varies from application to application. An example of how this approximation can be implemented is to directly apply the projection used in (1.12). This results in the prior weights being defined as

$$\boldsymbol{\Lambda} = \text{Covar}(\boldsymbol{w}) = E[(\boldsymbol{w} - E[\boldsymbol{w}])(\boldsymbol{w} - E[\boldsymbol{w}])^\top], \tag{1.19}$$

which when inserting the expression for $\boldsymbol{w}$ in terms of f in (1.12) and using the fact that $\text{Covar}(f) = \kappa$ gives that

$$\boldsymbol{\Lambda} = \int_\Omega \int_{\Omega'} \boldsymbol{\phi}(x)^\top \kappa(x,x') \boldsymbol{\phi}(x') dx dx', \tag{1.20}$$

which can be numerically approximated for any orthonormal basis functions.

Gaussian process regression assumes to not only have access to the measurements of the nonlinear function, but also a prior distribution of the nonlinear function

$$f \sim \mathcal{N}(0, \kappa(x,x')), \tag{1.21}$$

which in terms of basis function weights is given by

$$\boldsymbol{w} \sim \mathcal{N}(0, \boldsymbol{\Lambda}), \tag{1.22}$$

where $\boldsymbol{\Lambda}$ encodes the prior assumptions about the function weights. If we have access to N measurements $\boldsymbol{y}$ of the nonlinear function $\tilde{f}$ evaluated in N locations $\boldsymbol{x}$ according to

$$\boldsymbol{y} = \boldsymbol{\phi}(\boldsymbol{x})\boldsymbol{w} + \boldsymbol{e}, \quad \boldsymbol{e} \sim \mathcal{N}(0, \sigma_{\mathrm{y}}^2 \boldsymbol{I}), \tag{1.23}$$

we can use linear least squares estimation including prior information as defined in (1.9) to obtain the estimates of the basis function weights and covariances

$$\boldsymbol{w} \sim \mathcal{N}(\hat{\boldsymbol{w}}, \boldsymbol{P}), \quad \hat{\boldsymbol{w}} = \left(\boldsymbol{\phi}(x)^\top \boldsymbol{\phi}(x) + \sigma_{\mathrm{y}}^2 \boldsymbol{\Lambda}^{-1}\right)^{-1} \boldsymbol{\phi}(x)^\top \boldsymbol{y}, \quad \boldsymbol{P} = \left(\boldsymbol{\phi}(x)^\top \boldsymbol{\phi}(x) + \sigma_y^2 \boldsymbol{\Lambda}^{-1}\right)^{-1}. \tag{1.24}$$

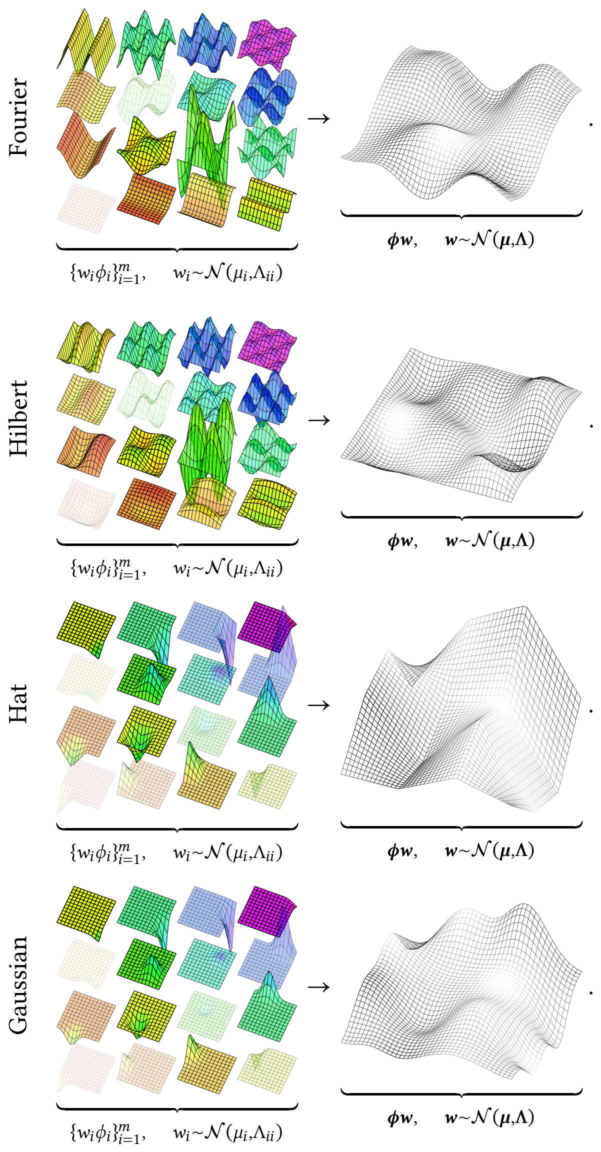

Fourier
Hilbert
Hat
Gaussian
$\{w_i\phi_i\}_{i=1}^m, \quad w_i \sim \mathcal{N}(\mu_i, \Lambda_{ii})$
$\rightarrow$
$\boldsymbol{\phi}\boldsymbol{w}, \quad \boldsymbol{w} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Lambda})$

Figure 1.9: Scaling and summing together basis functions with normally distributed weights give a normally distributed nonlinear function. The colored surfaces display the basis functions, where the color has a frequency on the light spectrum (according to the frequencies of the rainbow, so the frequency increase goes red-orange-yellow-green-blue-purple) corresponding to the product of the frequencies along each dimension. The opacity of each basis function corresponds to the variance of each basis function. The black surface indicates the value of the sum of the basis functions, and the transparency is proportional with the marginal variance of the function.

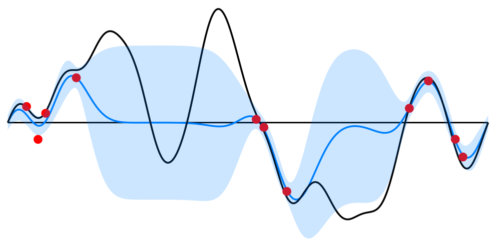


Figure 1.10: Gaussian process regression using Fourier basis functions. The black line shows an unknown nonlinear function we wish to estimate. The red dots show measurements of the functions that we have access to. The blue line shows an estimated function value using Gaussian process regression with Fourier basis functions. The light blue area indicates one standard deviation of the predicted function value.

Using these normally distributed weights, we can find the least squares estimate and marginal variance of the nonlinear function $\tilde{f}$ according to

$$E[\tilde{f}] = \boldsymbol{\phi}\hat{\boldsymbol{w}}, \quad \text{Covar}(\tilde{f}) = \boldsymbol{\phi}\boldsymbol{P}\boldsymbol{\phi}^\top, \tag{1.25}$$

which is indicated by the black line and the shaded area respectively in the right part of Figure 1.8. This estimate passes through all the measurements, and is certain close to the estimates. These equations can in other words be used to perform Gaussian process regression using basis functions, and it can be done with several types of basis functions. All of these types of basis functions can also be used to approximate the magnetic field with one small adaptation: by considering the measurement of the gradient of the nonlinear field instead of considering measurements of the field itself. Replacing the basis function approximation $\tilde{f}$ defined in (1.11) with the function f in (1.1) means that each magnetometer measurement is given by

$$\boldsymbol{y}_{\text{mag}} = \nabla_{\text{p}}\boldsymbol{\phi}(\boldsymbol{p})\boldsymbol{w} + \boldsymbol{e}, \quad \boldsymbol{e} \sim \mathcal{N}(\boldsymbol{0}, \sigma_{\text{y}}^2\boldsymbol{I}), \tag{1.26}$$

which can be stacked into a single large measurement model for several measurements. Using (1.1) instead of (1.23) simply means that $\boldsymbol{H}$ in (1.9) becomes a $3N \times M$ matrix equal to a stack of N $3 \times M$ matrices $\nabla_{\text{p}}\boldsymbol{\phi}(\boldsymbol{p})$ in (1.9). The resulting inference gives the estimated magnetic field in Figure 1.2.

1.4 Contributions of this thesis

All contributions discussed in this thesis uses basis function approximations to Gaussian process regression. The contributions consider the computational complexity of basis function approximations as a baseline. In other words, all contributions that present an improvement in computational requirements, discusses an improvement compared to the computational requirements of Gaussian process regression with basis functions. The contributions in this thesis each use different types of basis functions of the selection displayed in Figure 1.3. Some of the contributions utilize particular properties of the selected basis functions to enable scalability, while other contributions could just as well have been implemented with other basis functions. The five contributions aim to answer the following questions:

- **How can magnetic field mapping and localisation be done faster?** The second chapter uses Hilbert-space basis functions on a finite domain, and investigates how a Kalman filter can be used to efficiently learn the weights $\boldsymbol{w}$ online at the same time as learning the position of the agent carrying the sensor. This contribution can be adapted to other basis functions, and the algorithm is adapted in Chapter 2 to be used with Gaussian basis functions.

- **How can magnetic field mapping and localisation be applied to multi-agent systems?** The third chapter also uses Hilbert-space basis functions, and investigates how the information matrix for the learning of the weights $\boldsymbol{w}$ can be distributed across a multi-agent system using consensus algorithms [23].

- **How can the storage requirements of reduced-rank Gaussian process maps be reduced?** The fourth chapter reduces the storage requirements for certain classes of basis functions, including Hilbert-space basis functions and Fourier basis functions. It does this by reducing the required storage for the Fisher information of these basis functions, thereby effectively reducing the computational requirements of the multi-agent algorithm in the previous chapter without additional approximation.

- **How can magnetic field mapping and localisation be extended to larger areas?** The fifth chapter investigates how the spatial area of the magnetic field map can be scaled. It uses the Gaussian basis functions, but truncates the functions so they, like the hat basis functions, cause the information matrix to be inherently sparse.

- **How can magnetic field maps be scaled to encompass repeated global patterns in addition to local variations?** The sixth chapter is concerned with scaling the size of the area with confident and accurate magnetic field maps beyond the area where there are available measurements of the field. It does this by learning and extrapolating repeated patterns in the magnetic field, caused by repeated structural elements in indoor environments. It uses Fourier basis functions, which can utilize the algorithm in Chapter 3 to decrease storage requirements for the map.

2 An extended Kalman filter for magnetic field SLAM using Gaussian process regression

We present a computationally efficient algorithm for using variations in the ambient magnetic field to compensate for position drift in integrated odometry measurements (dead-reckoning estimates) through simultaneous localization and mapping (SLAM). When the magnetic field map is represented with a reduced-rank Gaussian process (GP) using Laplace basis functions defined in a cubical domain, analytic expressions of the gradient of the learned magnetic field become available. An existing approach for magnetic field SLAM with reduced-rank GP regression uses a Rao-Blackwellized particle filter (RBPF). For each incoming measurement, training of the magnetic field map using an RBPF has a computational complexity per time step of $O(N_p N_m^2)$, where N_p is the number of particles, and N_m is the number of basis functions used to approximate the Gaussian process. Contrary to the existing particle filter-based approach, we propose applying an extended Kalman filter based on the gradients of our learned magnetic field map for simultaneous localization and mapping. Our proposed algorithm only requires training a single map. It, therefore, has a computational complexity at each time step of $O(N_m^2)$. We demonstrate the workings of the extended Kalman filter for magnetic field SLAM on an open-source data set from a foot-mounted sensor and magnetic field measurements collected onboard a model ship in an indoor pool. We observe that the drift compensating abilities of our algorithm are comparable to what has previously been demonstrated for magnetic field SLAM with an RBPF.

This chapter is based on *Frida Viset, Rudy Helmons, Manon Kok*: An extended Kalman filter for magnetic field SLAM using Gaussian process regression. Sensors, 2022, (8), 2833.

2.1 Introduction

Autonomous navigation using only onboard sensors is a desired technology for various applications. Indoors, underground and underwater, there is no stable access to GNSS signals, as they are blocked by building elements, earth and water, respectively [24–26]. Also in other environments, the use of GNSS signals for localization can be challenging. Surface vehicles in harbours can be close to containers, bridges, or other industrial elements that can cause multi-path effects on the GNSS (Global navigation satellite system) measurements [27]. Autonomous navigation using only onboard sensors is challenging because of the drift in the position estimate obtained from sensor measurements that are independent of pre-deployed infrastructure [28]. Drift occurs when noisy measurements from, for example, gyroscopes, accelerometers, Doppler velocity logs or wheel encoders are integrated to estimate position without any absolute position measurements [5]. We will refer to the position estimates and orientation estimates obtained when integrating such noisy measurements as odometry. A range of other possible sensor readings may be available in these scenarios. The scope of our research is limited to the investigation of autonomous navigation using onboard odometry and magnetic field measurements.

Magnetic field simultaneous localization and mapping (SLAM) has been proposed to compensate for odometry drift when there is access to magnetic field measurements in a magnetic field with stationary spatial variations [10]. It has been demonstrated for indoor localization that magnetic field SLAM can be used to improve position estimates [8]. In environments with ferromagnetic structures, as for example in indoor environments, the magnetic field has rich spatial variation due to the magnetization of the metal [29]. Navigation using nonlinear variations in the ambient magnetic field has been proposed for a range of applications, such as indoor localization [7, 8, 12, 13, 24, 30–39], underwater localization [40–48], and surface and aerial navigation [49, 50]. Although [49] uses an extended Kalman filter (EKF) for localization in a learned magnetic field, the majority use a particle filter [12, 13, 24, 30–32, 34–37, 42]. A comparative study has demonstrated that the particle filter is more accurate for underwater geomagnetic navigation in the case where the initial position is not known, while the EKF is more computationally efficient [51].

Computationally tractable magnetic field SLAM in three dimensions was proposed in [8], using a Rao-Blackwellized particle filter (RBPF) to simultaneously estimate the position and orientation of a pedestrian as well as the ambient magnetic field. A set of N_p particles are used to represent the position and orientation [8]. The RBPF for magnetic field SLAM proposed by [8] uses Gaussian process (GP) regression to combine knowledge about the nature of the magnetic field from Maxwell's equations with measurements of the magnetic field to create a magnetic field map. To this end, they build the magnetic field map for each particle using reduced-rank Gaussian process regression, which represents the magnetic field map as a linear combination of N_m Laplace basis functions on hexagonal domains, and which represents the magnetic field map uncertainty as a matrix with N_m^2 entries. As each magnetic field map is represented with the weights of N_m basis functions and the corresponding covariance of these weights, all of which require updating at each time step, the computational cost of updating the magnetic field map is $O(N_p N_m^2)$ [8]. In the case where the particle filter is run on a parallelized architecture, such as FPGAs, the computation time dependence on the number of particles can be reduced dramatically [52, 53]. The scope of our research is limited to improving the speed of Magnetic field SLAM on non-parallelized architectures. Magnetic field SLAM has also been demonstrated feasible using an RBPF with Laplace basis functions defined on a single, cubic domain [13].

The contribution of this paper is an approach to magnetic field SLAM that is faster and requires less storage compared to the approach proposed in [8], inspired by the goal to run magnetic field SLAM in real-time on cheap carry-on units with low processing power. A property of using reduced-rank Gaussian process regression for magnetic field SLAM in a cubic domain is that the magnetic field map is given as a linear combination of analytically described basis functions [54]. We can therefore use the spatial derivatives of the closed-form solutions of the Laplacian to find the Jacobian of the magnetic field map with respect to the position estimate. To reduce computational expenses, we propose utilising the availability of analytical Jacobians of the reduced-rank Gaussian process magnetic field maps to perform magnetic field SLAM using an extended Kalman filter. This only requires building and updating a single copy of the magnetic field map at each time instance. Figure 2.1 shows the learned magnetic field map and estimated trajectory from our EKF algorithm for magnetic field SLAM, tested on magnetic field measurements collected onboard a model ship. The resulting computational cost is $O(N_m^2)$ at each time step, instead of $O(N_m^2 N_p)$. The use of the EKF is possible if the dynamic model and measurement model are close to linear [4]. In the case of simultaneous localization and mapping, the world frame coordinate system is defined relative to the initial body frame coordinate system [8]. As there is no uncertainty in the initial position estimate due to this definition [8], the position estimate initially has zero covariance. In cases where the growth of the uncertainty of the pose estimate that comes from odometry drift is limited by frequent enough visitations of previous areas, magnetic field SLAM remedies drift in the position estimate, which means that the position estimation error no longer grows without bounds, but stays limited [8]. When the estimated position is close to the actual position, the magnetic field linearized about the estimated position provides a good local approximation to the magnetic field itself, as the magnetic field even in environments with strong stationary disturbances can be assumed to have limited spatial variability [7–9, 12]. A key assumption for several implementations of estimating the magnetic field with GP regression is that the magnetic field in locations close to each other have a higher correlation than the magnetic field in locations that are further away [7–9, 12]. In these implementations, how rapidly the correlation diminishes with increasing distance is encoded in a hyperparameter in the GP prior describing the length scale of the spatial variations in the magnetic field [9].

We illustrate with simulations that we can expect the EKF for a localization task to give accurate estimates when the max norm of the covariance from the predictive distribution is small relative to the length scale of the magnetic field anomalies. A requirement for using GP regression to represent the magnetic field map in magnetic field SLAM is the prior knowledge of hyperparameters describing the expected distribution of the magnetic field potential [8]. These hyperparameters contain information about the expected length scale of the magnetic field spatial variations [9]. Without adding any further assumptions to the magnetic field SLAM formulation presented in [8], we can therefore assume to have information available about how rapidly we can expect our learned magnetic field to vary spatially. In SLAM, the uncertainty of the position estimate will grow when the sensor moves through unexplored areas, as there is no map information available to correct the estimated pose [8]. When the sensor re-enters an area where it has already built a map of the

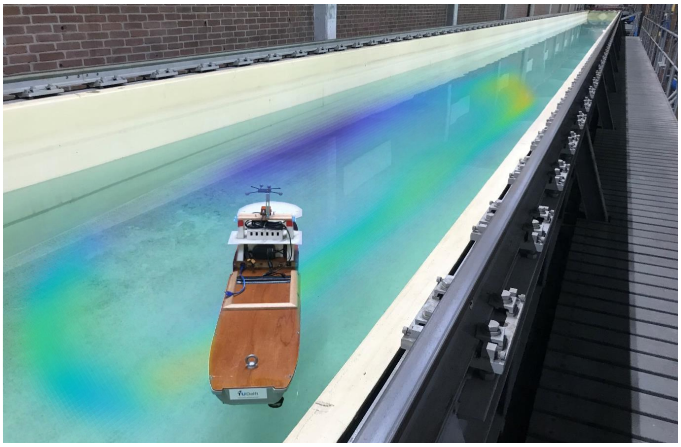


Figure 2.1: Learned magnetic field variations in test pool. The color corresponds to the estimated norm of the magnetic field map, while the opacity is inversely proportional with the variance of the estimate

local anomalies, our proposed algorithm can be expected to converge when the covariance of the position estimate is small compared to the length scale of the learned magnetic field map. We show with magnetic field data collected from a model ship in an indoor pool (see Figure 2.1) and simulated odometry that our proposed algorithm converges when the odometry noise is limited, for a trajectory when the time until the first revisitation of a mapped area is constant. We also show with magnetic field measurements and odometry obtained from an open-source implementation by [55] that our algorithm can compensate for drift in position estimates based on accelerometer and gyroscope measurements in a foot-mounted sensor.

The remainder of this paper is structured as follows. In Section 2.2, we define the model for our magnetic field SLAM estimation problem. In Section 2.3, we derive an EKF for magnetic field SLAM. In Section 2.4 we show the convergence properties of our algorithms in a simulated navigation task, where we can control the ratio of our position estimate uncertainty over the length scale of the magnetic field variations. In Section 2.5, we demonstrate the drift-compensating abilities of the EKF-SLAM algorithm on a set of data we collected with a model ship and on an open-source data-set from a foot-mounted sensor. Finally, in Section 2.6 we summarise our findings and discuss possible directions for future work. Our Matlab-implementation producing all results found in this paper can be found on https://github.com/fridaviset/EKFMagSLAM.

2.2 Modeling

Our simultaneous localization and mapping algorithm estimates the filtering distribution

$$p(x_t | y^{\mathrm{b}}_{1:t}, \Delta p^{\mathrm{w}}_{1:t}, \Delta q^{\mathrm{b}}_{1:t}), \tag{2.1}$$

where we denote the available magnetic field measurements by $y^{\mathrm{b}}_{1:t}$, the odometry describing the change in position and orientation $\Delta p^{\mathrm{w}}_{1:t}, \Delta q^{\mathrm{b}}_{1:t}$, and the state we wish to estimate at each time step t as x_t. For a vector x_t, the set $\{x_1, \cdots, x_t\}$ is denoted $x_{1:t}$ for brevity. We use the superscript b to denote the sensor's body-frame, which is aligned with its sensor axes. The superscript w refers to the world frame, which is defined as the inertial frame that shares its origin with the body frame at time zero. The gravity field in this position is aligned with the negative z-axis, and where the initial yaw-angle between the body and world-frame at $t = 0$ is zero. As we wish to estimate both position, orientation and the magnetic field map, we model our state as

$$x_t = [(p_t^{\mathrm{w}})^\top \quad (q_t^{\mathrm{wb}})^\top \quad m^\top]^\top, \tag{2.2}$$

where p_t^{w} denotes the position, q_t^{wb} denotes the orientation transformation from the world frame to the body frame, and m is a vector that describes our magnetic field map represented with reduced-rank GP regression.

2.2.1 Measurement model

We consider the case where we have access to measurements of the magnetic field in a sensor attached to the object we aim to localize. The measurement equation is given by

$$y_t^{\mathrm{b}} = R_t^{\mathrm{bw}} \nabla_{\mathrm{p}} \varphi(p_t^{\mathrm{w}}) + e_{\mathrm{m},t}^{\mathrm{b}}, \qquad e_{\mathrm{m},t}^{\mathrm{b}} \sim \mathcal{N}(0, \sigma_{\mathrm{m}}^2 \mathcal{I}_3), \tag{2.3}$$

where y_t^{b} denotes the magnetic field measurement, $e_{\mathrm{m},t}^{\mathrm{b}}$ denotes the measurement noise and R_t^{bw} denotes the rotation from world to body frame, corresponding to the conjugate of the quaternion q_t^{wb}, expressed as a rotation matrix. See [4] for definitions of the quaternion conjugate, and definitions of transformation from a quaternion to a rotation matrix. The function $\nabla_{\mathrm{p}} \varphi(p_t^{\mathrm{w}})$ is our model of the magnetic field. We model the magnetic field as in [9] as the gradient of the function $\varphi(p_t^{\mathrm{w}})$ with respect to the position p_t^{w}, where $\varphi : \mathbb{R}^3 \to \mathbb{R}$ is a scalar potential distributed as a GP with prior

$$\varphi \sim \mathcal{N}(0, \kappa_{\mathrm{SE}}(\cdot,\cdot) + \kappa_{\mathrm{lin}}(\cdot,\cdot)), \tag{2.4}$$

and where the kernel is defined by the functions

$$\kappa_{\mathrm{SE}}(p, p') = \sigma_{\mathrm{SE}}^2 \exp\left(-\frac{\|p - p'\|_2^2}{2 l_{\mathrm{SE}}^2}\right), \tag{2.5a}$$

$$\kappa_{\mathrm{lin}}(p, p') = \sigma_{\mathrm{lin}}^2 p^\top p', \tag{2.5b}$$

with σ_{SE}, σ_{lin}, l_{SE} and σ_{m} being hyperparameters. The hyperparameter l_{SE} refers to the length scale of the spatial variations in the magnetic field potential that is represented by the kernel [9]. The parameters σ_{SE}, σ_{lin} and σ_{m} define the presence of the nonlinear components, linear components and measurement noise in the magnetic field respectively [9]. The linear component modelled by the kernel component κ_{lin} represents of the constant underlying earth magnetic field, while the nonlinear disturbances caused by the modelled by the nearby ferromagnetic structures is modelled by the kernel component κ_{SE}. Modelling the magnetic field as the gradient of a scalar potential ensures that Maxwell's equations are satisfied, under the assumption that no current passes through the domain where we construct our magnetic field map [9]. We use a reduced-rank approximation to the GP similar to the one used in [8] for mapping the indoor magnetic field for localization purposes with the same kernel. Our approach is an application of the GP approximation presented in [11], which is based on conditioning the GP prediction on a set of basis functions corresponding to a subset of the eigenbasis of the negative Laplace operator in a finite domain, subject to Dirichlet boundary conditions [11]. The reduced-rank approximation models the magnetic field potential as a sum of basis functions defined as the solutions to the Laplace equations over a finite domain $\Omega \subset \mathbb{R}^3$

$$\begin{cases} -\nabla_{\mathrm{p}}^2 \phi_i(p) = \lambda_i^2 \phi_i(p), & p \in \Omega, \\ \phi_i(p) = 0, & p \in \delta\Omega, \end{cases} \tag{2.6}$$

where ϕ_i is the i'th eigenfunction, and λ_i is the i'th eigenvalue [11]. We approximate the GP with the first N_m basis functions solving the Laplace equations defined over a cubical domain $\Omega = [L_{l,1}, L_{u,1}] \times [L_{l,2}, L_{u,2}] \times [L_{l,3}, L_{u,3}]$. In this case, using the N_m first eigenfunctions to represent the potential gives the approximation

$$\varphi(p) \approx \Phi(p)m, \tag{2.7}$$

with $\Phi(p)$ being the matrix

$$\Phi(p) = \begin{bmatrix} p^\top & \phi_1(p) & \dots & \phi_{N_m}(p) \end{bmatrix}, \tag{2.8}$$

where ϕ_i is the $i'th$ eigenfunction of the Laplace basis, and $m \in \mathbb{R}^{N_m+3}$ is a vector determining the contribution of each linear components as well as each basis function to the potential. Each eigenfunction $\phi_i(p)$ has a closed form expression given by

$$\phi_i(p) = \prod_{d=1}^{3} \frac{\sqrt{2}}{\sqrt{L_{\mathrm{u},d} - L_{\mathrm{l},d}}} \sin\left(\frac{\pi n_{i,d}(p_d + L_{\mathrm{l},d})}{L_{\mathrm{u},d} - L_{\mathrm{l},d}} \right), \tag{2.9}$$

where the set $(n_{i,1}, n_{i,2}, n_{i,3})$ is the set of three natural numbers that is different from the sets $(n_{j,1}, n_{j,2}, n_{j,3})$ defined for all $j < i$, that gives the corresponding eigenvalue

$$\lambda_i = \sum_{d=1}^{D} \left(\frac{\pi n_{i,d}}{L_{\mathrm{u},d} - L_{\mathrm{l},d}} \right)^2, \tag{2.10}$$

as large as possible. The basis functions in (2.9) and eigenvalues in (2.10) are identical to those used in [13]. The vector m has a prior distribution given by

$$m \sim \mathcal{N}(0, \Lambda), \tag{2.11}$$

where Λ is defined as

$$\Lambda = \mathrm{diag}\left[\sigma_{\mathrm{lin}}^2 \mathcal{I}_3, \quad S_{\mathrm{SE}}(\sqrt{\lambda_1}), \quad \cdots, \quad S_{\mathrm{SE}}(\sqrt{\lambda_{N_m}}) \right], \tag{2.12}$$

with $S_{\mathrm{SE}}(\cdot)$ being the spectral density of the squared exponential kernel, as defined in [10]. This corresponds to the magnetic field potential $\varphi(p) \approx \Phi(p)m$ having a prior distribution given by (2.4) as N_m goes to infinity, and the size of the domain goes to infinity [11]. Inserting this approximation to the magnetic field model gives the measurement model

$$y_t^{\mathrm{b}} \approx R_t^{\mathrm{bw}} \nabla_{\mathrm{p}} \Phi(p_t^{\mathrm{w}}) m + e_{\mathrm{m},t}^{\mathrm{b}} \qquad e_{\mathrm{m},t}^{\mathrm{b}} \sim \mathcal{N}(0, \sigma_{\mathrm{m}}^2 \mathcal{I}_3), \tag{2.13}$$

with the closed form expressions for $\nabla_{\mathrm{p}} \Phi(p_t^{\mathrm{w}})$ given in Appendix 2.A. This measurement model is identical to the measurement model used in [8], with the exception of the basis functions $\Phi(p_t^{\mathrm{w}})$, which are different as they are defined with respect to different domains.

2.2.2 Dynamic model

We assume access to noisy odometry measurements Δp_t^{w} and Δq_t^{b} of the change in position and orientation at each time step. We model the change in position and orientation according to the dynamic model

$$p_{t+1}^{\mathrm{w}} = p_t^{\mathrm{w}} + \Delta p_t^{\mathrm{w}} + e_{\mathrm{p},t}^{\mathrm{w}}, \qquad e_{\mathrm{p},t}^{\mathrm{w}} \sim \mathcal{N}(0, \sigma_{\mathrm{p}}^2 \mathcal{I}_3), \tag{2.14a}$$

$$q_{t+1}^{\mathrm{wb}} = q_t^{\mathrm{wb}} \odot \Delta q_t^{\mathrm{b}} \odot \exp_{\mathrm{q}}(e_{\mathrm{q},t}^{\mathrm{b}}), \qquad e_{\mathrm{q},t}^{\mathrm{b}} \sim \mathcal{N}(0, \sigma_{\mathrm{q}}^2 \mathcal{I}_3), \tag{2.14b}$$

where $e_{\mathrm{p},t}^{\mathrm{w}}$ and $e_{\mathrm{q},t}^{\mathrm{b}}$ denote the position and orientation odometry measurement noise respectively, $\odot$ is the quaternion product and $\exp_{\mathrm{q}}$ is the operator that maps an axis-angle orientation deviation to a quaternion (see [4] for details on quaternion algebra).

2.3 EKF for magnetic field SLAM

We estimate our state with an EKF applied to the dynamic model in (2.14a)-(2.14b) and the measurement model defined in (2.13), with predictive and filtered estimates denoted $\hat{p}_{t|t-1}^{\mathrm{w}}$, $\hat{m}_{t|t-1}$, $\hat{q}_{t|t-1}^{\mathrm{wb}}$ and $\hat{p}_{t|t}^{\mathrm{w}}$, $\hat{m}_{t|t}$, $\hat{q}_{t|t}^{\mathrm{wb}}$ respectively. We initialise the magnetic field state estimate as $\hat{m}_{0|0} = 0_{N_m \times 1}$ according to the reduced-rank GP prior in (2.11). We initialise the orientation estimate according to the initial rotation $\hat{q}_{0|0}^{\mathrm{wb}} = q_0^{\mathrm{wb}}$ between the world and body frame as defined in Section 2.2. We initialise the position estimate as $\hat{p}_{0|0}^{\mathrm{w}} = 0_{3\times 1}$, also according to our definition of the world frame relative to the initial body frame from Section 2.2.

We represent the deviation between the true and estimated predictive state by an error state ξ_t defined as

$$\xi_t = [(\delta_t^{\mathrm{w}})^\top \quad (\eta_t^{\mathrm{w}})^\top \quad \nu_t^\top]^\top, \tag{2.15}$$

where $\delta_t^{\mathrm{w}} = p_t^{\mathrm{w}} - \hat{p}_{t|t-1}^{\mathrm{w}}$ denotes the position estimation error, $\nu_t = m - \hat{m}_{t|t-1}$ denotes the magnetic field state estimation error, and where η_t^{w} denotes the orientation estimation error parametrised as an axis-angle deviation according to

$$q_t^{\mathrm{wb}} = \exp_{\mathrm{q}}(\eta_t^{\mathrm{w}}) \odot \hat{q}_{t|t-1}^{\mathrm{wb}}. \tag{2.16}$$

Similarly, we represent the deviation between the true and estimated filtered state by an error state $\tilde{\xi}_t$ defined as

$$\tilde{\xi}_t = [(\tilde{\delta}_t^{\mathrm{w}})^\top \quad (\tilde{\eta}_t^{\mathrm{w}})^\top \quad \tilde{\nu}_t^\top]^\top, \tag{2.17}$$

where $\tilde{\delta}_t^{\mathrm{w}} = p_t^{\mathrm{w}} - \hat{p}_{t|t}^{\mathrm{w}}$, $\tilde{\nu}_t = m - \hat{m}_{t|t}$, and where $\tilde{\eta}_t^{\mathrm{w}}$ denotes the filtered orientation estimation error according to

$$q_t^{\mathrm{wb}} = \exp_{\mathrm{q}}(\tilde{\eta}_t^{\mathrm{w}}) \odot \hat{q}_{t|t}^{\mathrm{wb}}. \tag{2.18}$$

Since we build our map relative to our initial position and orientation, the covariance of our initial position and orientation estimates is zero. The covariance of the initial magnetic field estimate is defined in (2.11) as the magnetic field map prior Λ. Hence, our initial error state ξ_0 has a covariance

$$P_{0|0} = \begin{bmatrix} 0_{6\times 6} & 0_{6\times(N_m+3)} \\ 0_{(N_m+3)\times 6} & \Lambda \end{bmatrix}. \tag{2.19}$$

To perform the dynamic update, we propagate our filtered state estimate through the non-linear dynamic model (2.14a)-(2.14b), giving the predictive updates as described in (2.26a)-(2.26b). As the magnetic field is assumed stationary, its estimate is unchanged by the dynamic update defined in (2.26c). We derive the covariance update in the EKF by linearising about the filtered state estimate from the previous time step, with respect to the error state $\tilde{\xi}_t$. Inserting (2.15), (2.17) and (2.26a)-(2.26b) into the dynamic model (2.14a)-(2.14b) gives

$$\hat{p}^{\mathrm{w}}_{t|t-1} + \delta^{\mathrm{w}}_t = \hat{p}^{\mathrm{w}}_{t-1|t-1} + \Delta p^{\mathrm{w}}_t + \tilde{\delta}^{\mathrm{w}}_{t-1} + e^{\mathrm{w}}_{\mathrm{p},t} \tag{2.20a}$$

$$\exp_q\left(\eta^{\mathrm{w}}_t\right) = \exp_{\mathrm{q}}\left(\tilde{\eta}^{\mathrm{w}}_{t-1}\right) \odot \hat{q}^{\mathrm{wb}}_{t|t-1} \odot \exp_{\mathrm{q}}(e^{\mathrm{b}}_{\mathrm{q},t}) \odot (\hat{q}^{\mathrm{wb}}_{t|t-1})^C, \tag{2.20b}$$

where $(\hat{q}^{\mathrm{wb}}_{t|t-1})^C$ denotes the conjugate of the quaternion $\hat{q}^{\mathrm{wb}}_{t|t-1}$. The linearization of the dynamic model with respect to the error states gives the following propagation of the error states

$$\delta^{\mathrm{w}}_t = \tilde{\delta}^{\mathrm{w}}_{t-1} + e^{\mathrm{w}}_{\mathrm{p},t}, \tag{2.21a}$$

$$\eta^{\mathrm{w}}_t \approx \tilde{\eta}^{\mathrm{w}}_{t-1} + \hat{R}^{\mathrm{wb}}_{t|t-1} e^{\mathrm{b}}_{\mathrm{q},t}, \tag{2.21b}$$

$$v_t = \tilde{v}_{t-1}, \tag{2.21c}$$

where $\hat{R}^{\mathrm{wb}}_{t|t-1}$ denotes the rotation matrix corresponding to the rotation represented by the quaternion $\hat{q}^{\mathrm{wb}}_{t|t-1}$. This linearization is exact for the position, and it is a good approximation for the orientation error state in the cases where the orientation error is small [4]. Equations (2.21a)-(2.21b) can equivalently be written as

$$\xi_t \approx \tilde{\xi}_{t-1} + e_{\mathrm{dyn},t}, \qquad e_{\mathrm{dyn},t} \sim \mathcal{N}(0_{(N_m+9)\times 1}, Q), \tag{2.22}$$

where

$$Q = \begin{bmatrix} \sigma^2_{\mathrm{p}} \mathcal{I}_3 & 0_{3\times 3} & 0_{3\times(N_m+3)} \\ 0_{3\times 3} & \sigma^2_{\mathrm{q}} \mathcal{I}_3 & 0_{3\times(N_m+3)} \\ 0_{(N_m+3)\times 3} & 0_{(N_m+3)\times 3} & 0_{(N_m+3)\times(N_m+3)} \end{bmatrix}. \tag{2.23}$$

As the linearization of the error state propagation is given in (2.22), the covariance $P_{t|t-1}$ of the predictive state error ξ_t is given by (2.26d).

For the measurement update, we apply an EKF measurement update to the measurement model in (2.13). We linearise about the predictive state estimate, with respect to the error state ξ_t. The linearized measurement model is given by

$$y^b_t = \hat{R}^{\mathrm{bw}}_{t|t-1} \nabla_{\mathrm{p}} \Phi(\hat{p}^{\mathrm{w}}_{t|t-1}) \hat{m}_t + H_t \xi_t + e^{\mathrm{b}}_{\mathrm{m},t}, \quad e^{\mathrm{b}}_{\mathrm{m},t} \sim \mathcal{N}(0_{3\times 1}, \sigma^2_{\mathrm{m}} \mathcal{I}_3), \tag{2.24}$$

where

$$H_t = \begin{bmatrix} \left(\nabla_{\mathrm{pp}} \Phi(\hat{p}^{\mathrm{w}}_{t|t-1}) \hat{m}_t\right)^\top \\ \left[\left(\nabla_{\mathrm{p}} \Phi(\hat{p}^{\mathrm{w}}_{t|t-1}) \hat{m}_t\right) \times\right]^\top \\ \left(\nabla_{\mathrm{p}} \Phi(\hat{p}^{\mathrm{w}}_{t|t-1})^\top\right) \end{bmatrix}^\top \tag{2.25}$$

with $[v\times]$ being the scew-symmetric matrix representing the cross product $v \times u$ between two vectors $v, u \in \mathbb{R}^3$ as a matrix multiplication $[v\times]\,u$ (explicit expression for $[v\times]$ is given in [4]),

Algorithm 1 EKF for magnetic field SLAM

```
1: Input: $\left\{\Delta p_t^{\mathrm{w}}, \Delta q_t^{\mathrm{b}}, y_t^{\mathrm{b}}\right\}_{t=1}^{N}$
2: Output: $\left\{\hat{p}_{t|t}^{\mathrm{w}}\right\}_{t=1}^{N}, \left\{\hat{q}_{t|t}^{\mathrm{wb}}\right\}_{t=1}^{N}, \left\{\hat{m}_{t|t}\right\}_{t=1}^{N}$
3:
4: Initialisation: $\hat{p}_{0|0}^{\mathrm{w}} = 0_{3\times 1}$, $\hat{q}_{0|0}^{\mathrm{wb}} = q_0^{\mathrm{wb}}$, $\hat{m}_{0|0} = 0_{(N_m+3)\times 0}$, (2.19)
5: for $t = 1$ to $N$ do
6:     Dynamic update
           $\hat{p}_{t|t-1}^{\mathrm{w}} = \hat{p}_{t-1|t-1}^{\mathrm{w}} + \Delta p_t^{\mathrm{w}}$   (2.26a)
           $\hat{q}_{t|t-1}^{\mathrm{wb}} = \hat{q}_{t-1|t-1}^{\mathrm{wb}} \odot \Delta q_t^{\mathrm{b}}$   (2.26b)
           $\hat{m}_{t|t-1} = \hat{m}_{t-1|t-1}$   (2.26c)
           $P_{t|t-1} = P_{t-1|t-1} + Q$   (2.26d)
7:     Measurement update
           $z_t = \hat{R}_{t|t-1}^{\mathrm{wb}} y_t^{\mathrm{b}} - \nabla\Phi_{\mathrm{p}}(\hat{p}_{t|t-1}^{\mathrm{w}})\hat{m}_{t|t-1}$   (2.27a)
           $S_t = H_t P_{t|t-1} H_t^{\top} + \sigma_{\mathrm{m}}^2 \mathcal{I}_3$   (2.27b)
           $K_t = P_{t|t-1} H_t^{\top} S_t^{-1}$   (2.27c)
           $\hat{\xi}_t = K_t z_t$   (2.27d)
           $P_{t|t} = P_{t|t-1} - K_t S_t K_t^{\top}$   (2.27e)
8:     Relinearization
           $\hat{p}_{t|t}^{\mathrm{w}} = \hat{p}_{t|t-1}^{\mathrm{w}} + \hat{\delta}_t^{\mathrm{w}}$   (2.28a)
           $\hat{q}_{t|t}^{\mathrm{wb}} = \exp_{\mathrm{q}}(\hat{\eta}_t^{\mathrm{w}}) \odot \hat{q}_{t|t-1}^{\mathrm{wb}}$   (2.28b)
           $\hat{m}_{t|t} = \hat{m}_{t|t-1} + \hat{\nu}_t$   (2.28c)
9: end for
```

and $\nabla_{\mathrm{pp}}\Phi(\cdot)$ being the Jacobian of the basis functions, given in Appendix 2.A. Note that this Jacobian is a matrix with $3\times 3\times(N_m+3)$ entries, and multiplying it with the (N_m+3)-dimensional state vector m therefore gives a 3×3 matrix. Applying the Kalman filter measurement update to this linearized measurement function gives the EKF measurement update in (2.27a)-(2.27e). This gives an estimate $\hat{\xi}_t$ of the predictive state error ξ_t, with a corresponding covariance. By concatenating the estimated predictive state error to the predictive state, we obtain the filtered state estimates as defined in (2.28a)-(2.28c). The covariance of the filtered error state $\tilde{\xi}_t$ then becomes the same as the covariance of the estimated predictive error state $\hat{\xi}_t$. Recursively applying the dynamic update, measurement update and re-linearization step results in Algorithm 1.

2.4 Simulations

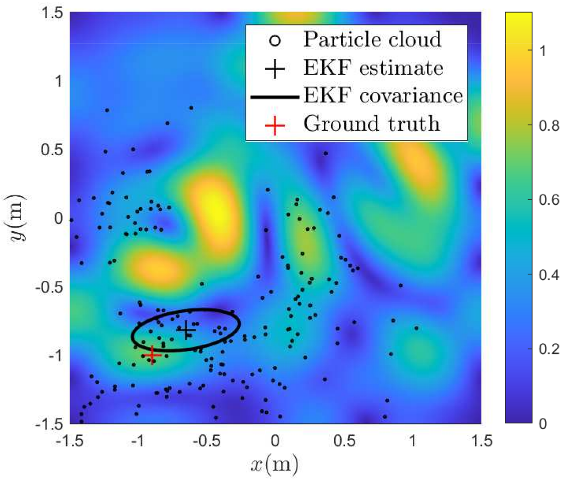
We study when Algorithm 1 for localization in a previously learned magnetic field gives a converging pose estimate in a known nonlinear field depending on the position uncertainty at time t. Since we assume the magnetic field map is know, we can replace $\hat{m}_{t|t}$ with the known m everywhere in our algorithm. As a consequence of this, we can alo skip (2.26c) and (2.28c), use $P_{0|0} = 0_{6\times 6}$ and

$$H_t = \begin{bmatrix} \left(\nabla_{\text{pp}}\Phi(\hat{p}^{\text{w}}_{t|t-1})m\right)^\top \\ \left[\left(\nabla_{\text{p}}\Phi(\hat{p}^{\text{w}}_{t|t-1})m\right)\times\right]^\top \end{bmatrix}^\top . \tag{2.29}$$

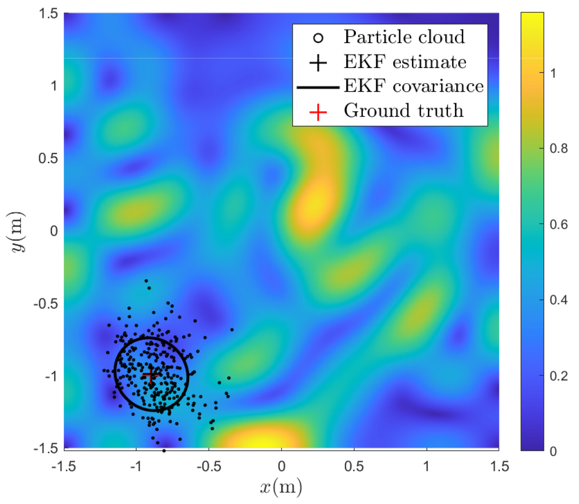
We start the simulation at time t with a varying predictive position estimation error, and set the standard deviation of our position uncertainty equal to the distance between the actual and estimated position at the beginning of the simulation. This artificially introduces a predictive position estimation error representing the estimation error that can accumulate over time in magnetic field SLAM. Position errors can, for example, accumulate when the sensor is moved for a long time through areas with no information about the magnetic field available from previous measurements [8].

We simulate positions p_t along a square trajectory moving with constant velocity for four laps, and simulate the odometry by adding sampled realisations of the white noises $e^{\text{w}}_{\text{p},t}$ and $e^{\text{b}}_{\text{q},t}$ to (2.14a) and (2.14b). We simulate a nonlinear field by drawing a sample from the reduced-rank GP prior $m_{\text{sim}} \sim \mathcal{N}(0, \Lambda)$, with $\sigma_{\text{lin}} = 0$, $\sigma_{\text{SE}} = 0.1$ and $l_{\text{SE}} = 0.2$. We used 50 basis functions to represent the magnetic field map. Using a domain that is $3\,\text{m} \times 3\,\text{m} \times 1\,\text{m}$. This number of basis functions ensures that for a GP trained with 2000 sampled measurements the root mean squared error (RMSE) between the approximate and the full GP predictions in 1000 randomly selected locations is below the measurement noise. To prevent ill effects from the boundary conditions we ensured that both the training and test data was at least 0.5 meters away from the border. We then simulate magnetic field measurements by adding white noise to the gradient of the nonlinear field in the ground truth position according to (2.13), replacing $\hat{m}_t$ with m_{sim}, and using $\sigma_{\text{m}} = 0.03$. We use the incoming magnetic field measurements and the odometry to estimate the position and orientation using an EKF and a particle filter. The EKF is implemented according to Algorithm 1, but reducing the state-space to only contain the position and orientation, and inserting m_{sim} in place of $\hat{m}_t$ in (2.27a). We implement a particle filter for navigation in a magnetic field represented by a field learned with GP regression according to Algorithm 1 in [7], with the difference that we use the simulated reduced-rank map instead of a learnt full GP map, and that we perform the prediction step using our odometry model in (2.14a)-(2.14b).

In Figure 2.2, we can see two examples of PFs' and an EKFs' estimate of the position filtered distribution, represented with a particle cloud and a mean and an uncertainty interval, respectively. In Figure 2.2(a), the initial uncertainty of the position estimate is so large that the particle cloud becomes multi-modal, making it impossible for the EKF to correctly approximate the true nature of the filtered distribution. The estimated position is, therefore, far away from the true position. In addition, the uncertainty estimate of the EKF does not reflect this, as it relies upon a linearization of the nonlinear magnetic field about

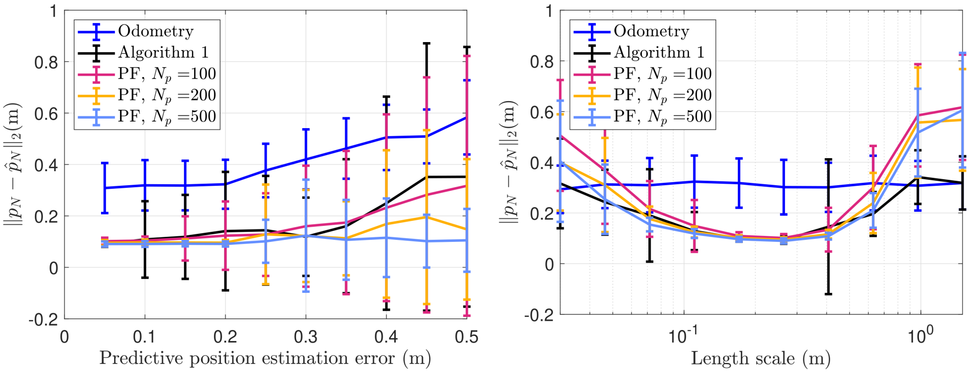


(a) Estimates of the filtered distribution based on predictive estimates with error 0.40 m.



(b) Estimates of the filtered distribution based on predictive estimates with error 0.05 m.

Figure 2.2: Comparison of approximations of the filtered position distribution given measurements from a simulated nonlinear field. The color indicates the norm of the simulated magnetic field. The covariance ellipsoids indicate the 68% confidence interval of the EKF estimate.



(a) Estimation accuracies with varying predictive position error.

(b) Estimation accuracies with varying length scales l_{SE}.

Figure 2.3: Simulation, investigating drift-compensating abilities given varying predictive position estimation errors. Comparison of position estimation error at the end of the trajectory between Algorithm 1 and a particle filter for localization in a known map with varying predictive position errors at the initialisation of the simulation. The lines connect the average results after 100 Monte Carlo repetitions with different realisations of the odometry noise, and the error bars represent one standard deviation.

the predictive estimate. In Figure 2.2(b), the position estimate at time t is still wrong, but close enough that the particle cloud representation of the filtered distribution appears unimodal, and the EKF estimate of the filtered distribution is now closer to the estimate from the particle filter. As we see in Figure 6.8(a) displaying the position estimation error at the end of the trajectory estimates across the four laps, the position estimates from the EKF are accurate across the entire trajectory if the predictive position error is lower than 0.3 meters. The particle filter on the other hand, is accurate even beyond these predictive position accuracies when using 500 particles, while it is only slightly improving the prediction accuracy over Algorithm 1 when using 100 or 200 particles. The accuracy is better for 200 particles than for 100 particles. For only 100 particles, the average prediction accuracy is worse than for Algorithm 1 for a predictive position accuracy of 0.2 meters, likely due to the fact that the particle filter is a Monte-Carlo method, meaning that there is never a guarantee for convergence [40]. In Figure 6.8(b), the average estimation accuracy for varying length scales of the simulated magnetic field is displayed. As in Figure 6.8(a), the performance of the particle filter improves with increasing amount of particles. Algorithm 1 is able to compensate for odometry drift and achieve estimation error on average below 0.2 meters for length scales between 0.1 and 0.4 meters, using a constant predictive position error of 0.1 meters. For length scales below 0.1 meters, the linearization error becomes too big for the approximation accuracy of our linearized model to give a good result. For length scales higher than 0.4 meters, the variations in the magnetic field are not rich enough to provide valuable information about the position of the sensor. Therefore, as the length scale of the field increases, even though the field becomes closer to linear and the linearization error continues to decrease, the estimation accuracy does not improve - because the signal-to-noise ratio from the magnetic field measurements also decreases. As we for the simulation results in Figure 6.8(a) use a simulated magnetic field map with spatial variations of 0.3

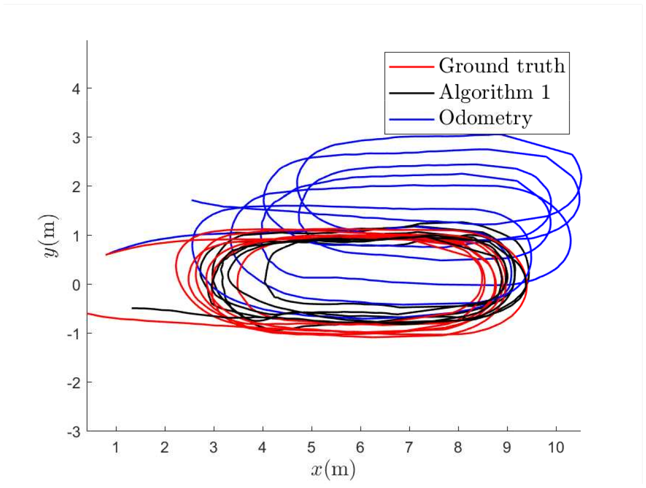
meters, this indicates that as long as the covariance of the predictive distribution does not exceed the length scale of the magnetic field, we can expect Algorithm 1 to have the same estimation accuracy as particle-based methods.

2.5 Experimental results

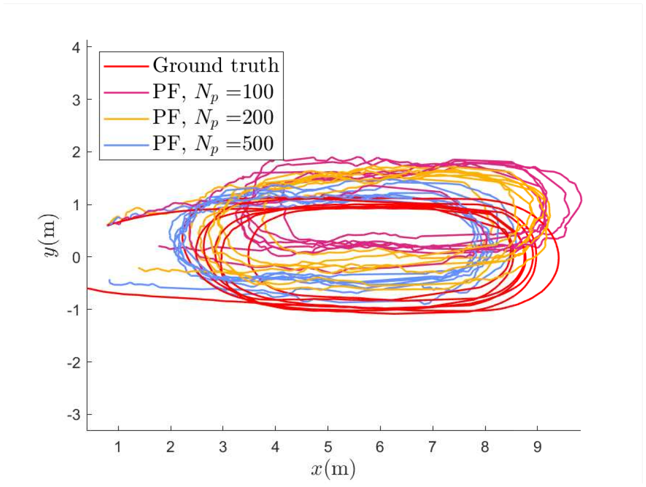
2.5.1 Model ship experiments

We performed experiments to test Algorithm 1 on a model ship in a pool. The magnetic field on the model ship was measured using an Xsens MTi-100 Inertial Measurement Unit (IMU). We recorded the ground truth position and orientation using a motion capture system with cameras and optical markers mounted on the model ship, as shown in Figure 2.1. The motion capture markers and the IMU were rigidly attached to the ship. The IMU measurements were collected on a computer onboard the ship. The magnetic field was disturbed by metal railings and building structures near the pool. We steered the model ship around in long loops in the pool, with a ground truth trajectory that is displayed in red in Figure 2.4. The magnetic field measurements were collected in the IMU at 200 Hz, and down-sampled to 5 Hz. The odometry was simulated based on the ground truth position and orientation, according to the odometry model in (2.14a)-(2.14b), also at a frequency of 5 Hz (but with some motion capture measurement dropouts due to pool reflections). We simulate drifting odometry by computing the change in position and orientation at each time step from the ground truth and adding a simulated white noise with standard deviation $\sigma_\mathrm{p} = 0.01$, $\sigma_\mathrm{q} = 0.001$. For these experiments, we use real magnetic field data and simulated odometry to investigate the effects of changing odometry noise on our algorithm. In addition, we simulate a constant position odometry bias of $[0.003 \quad 0.003 \quad 0]$ meters/time step. Our algorithm was not originally designed to compensate for constant position odometry biases. However, as this often occurs in practice (for example, when the odometry sensors are not perfectly calibrated), we chose to include it in our simulated odometry to test our algorithms' ability to compensate for a drift that consists both of integrated white noise and a constant disturbance.

In magnetic field SLAM, there is usually no possibility of optimising the hyperparameters for GPs prior to estimating the map, as there is no magnetic field data available. This motivates us to choose hyperparameters prior to running Algorithm 1 [8]. The hyperparameters were set to $l_\mathrm{SE} = 0.8$, $\sigma^2_\mathrm{SE} = 1$, $\sigma^2_\mathrm{m} = 0.01$, $\sigma^2_\mathrm{lin} = 1$ and 50 basis functions. The basis functions were defined with respect to a cubical domain Ω which is as small as possible, and whose border is at least 1 meters away from the closest ground truth position. We confirmed empirically that 50 basis functions is sufficient to ensure that the RMSE between the approximation and the full GP predictions in 1000 randomly selected locations in the domain is below the measurement noise. We based the predictions on 2000 samples from the full GP prior, sampled from randomly selected locations in the domain at least 1 meters away from the domain border. For the first set of experiments, we investigate and compare the position estimation error of Algorithm 1 with the particle filter-based approach to magnetic field SLAM [8] for odometry noise levels of $\sigma_\mathrm{p} = 0.01$, $\sigma_\mathrm{q} = 0.001$. In Figure 2.6(a), the norm of the magnetic field measurements in locations where there were no motion-capture dropouts are displayed. The norm of the measured magnetic field ranges between 0.46 and 0.83 (the Xsens MTi-100 provides unit-less measurements proportionatal



(a) Comparing Algorithm 1 and the odometry to the ground truth.



(b) Comparison of the position estimates from the RBPF with 100, 200 and 500 particles respectively to the ground truth.

Figure 2.4: Comparison of the model ship position trajectory estimates for a single realisation of simulated odometry noise from a birds-eye view.

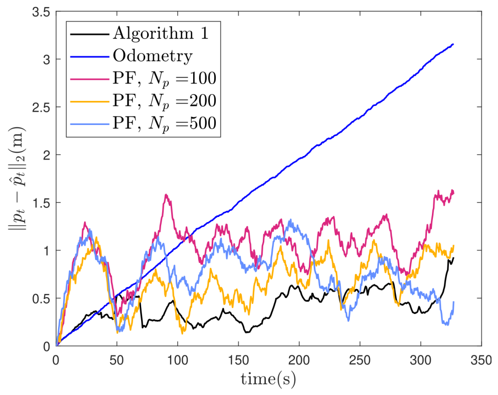
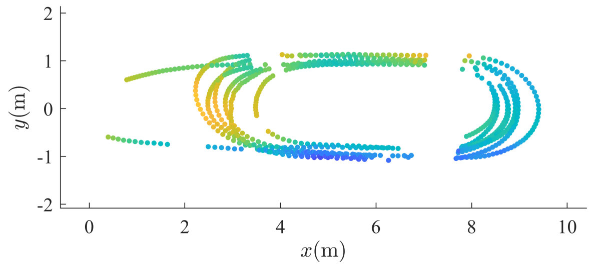


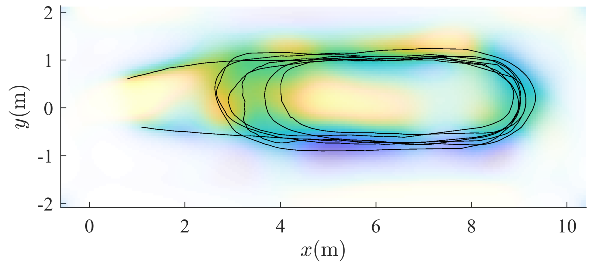
Figure 2.5: Comparison of model ship position estimation errors from Algorithm 1, drifting odometry and the PF with 100, 200 and 500 particles respectively for a single realisation of simulated odometry noise.

to the magnetic field strength). The spatial variations in the measured magnetic field norm are visible in Figure 2.6(a), and the magnetic field stays close to constant for position changes of less than 0.1 meters, while it can change as much as from 0.46 to 0.83 when the position change is more than 1 meter. As the spatial variations in the magnetic field potential are the sources of the spatial variations in the magnetic field norm [56], we expect that our assumed length scale of 0.8 meters is close enough to the actual length scale of the magnetic field variations to allow for Algorithm 1 to compensate for position estimation drift in the odometry. Figure 2.6(b) shows the norm of the magnetic field map learned by Algorithm 1, and comparing to the measured magnetic field norms in Figure 2.6(a), we can see that the norm of the magnetic field estimates and the estimated trajectory are similar, with the learned magnetic field map prediction being more certain in and near the areas where there are more magnetic field measurements available. In Figure 2.4(a), the estimated trajectory from Algorithm 1 is compared with the ground truth trajectory, as well as the dead reckoning position estimate from the simulated odometry. The position estimate from Algorithm 1 compensates visibly for the drift in the odometry.

In Figure 2.5 the position estimation error of the odometry and Algorithm 1 can be seen to increase at the beginning of the trajectory. After around 31 seconds, Algorithm 1 can use the learned magnetic field map in combination with the incoming magnetic field measurements to compensate for drift in the estimated position and orientation. In Table 2.1, the position estimation RMSE values for 4 collected data sets of a similar shape as the one displayed in Figure 2.4(a) is shown after repeated experiments with the same odometry noise and the same constant drift in the xy-plane, showing that the reduction of RMSE is comparable also for repeated experiments. The position estimation error of the dead reckoning can increase potentially unbounded, while the position estimation error of Algorithm 1 remains bounded when the ship revisits previously mapped areas. However, it will only remain bounded if the quality of the map is good enough to provide information to the position estimate. In Table 2.2, the runtime for each of the algorithms is displayed. The runtime of the PF methods grows proportionally with the number of particles used. As can be seen in Table 2.1, the PF performs on average worse than the EKF. The trajectory



(a) Measured magnetic field norm in ground truth positions.



(b) Estimated magnetic field norm and estimated trajectory.

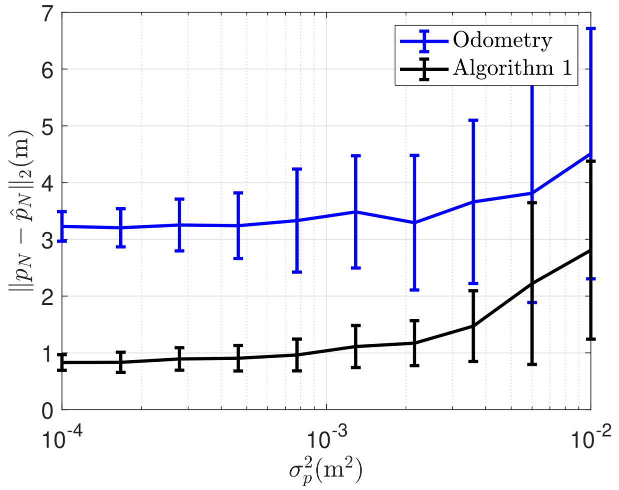
Figure 2.6: Measured and estimated magnetic field and position trajectories for the model ship. The upper plot marks with circles the locations where magnetic field measurements were successfully collected and matched with a ground truth position in the model ship, and the colors of the circles correspond to the norm of the measured magnetic field. The lower plot displays the trajectory estimate from applying Algorithm 1 in black. It also shows the learned magnetic field map, where the color corresponds to the norm of the estimated magnetic field $\|\nabla_p \Phi(p) \hat{m}_{N|N}\|_2$, and the opacity is inversely proportional with the trace of the covariance matrix of the magnetic field map estimate in each location, $\mathrm{Tr}(\nabla_p \Phi(p) P_{N|N} (\nabla_p \Phi(p))^\top)$.

of the highest-weight particle from a single run of each of the PFs is shown in Figure 2.4(b). From our simulation results, given enough particles, the particle filter compensates for drift at least as well as Algorithm 1. However, in contrast to our simulation results where we investigate how well the particle filter performs localization given a previously learned map, in practice, the particle filter has to rely upon magnetic field maps created conditionally on each particle. From these results a standard implementation of the PF in [8] with cubic domain basis functions and resampling at every timestep, with the same hyperparameter settings as the EKF performs worse on our collected model ship data, even given 500 particles. In general, the performance of the particle filter can depend on the resampling strategy, the measurement noise and the process noise [57]. Another possible explanation why the particle filter performs worse for the full SLAM scenario compared to the simulation case, is the fact that for long trajectories, the resampling step can cause loss of diversity amount the particles [58].

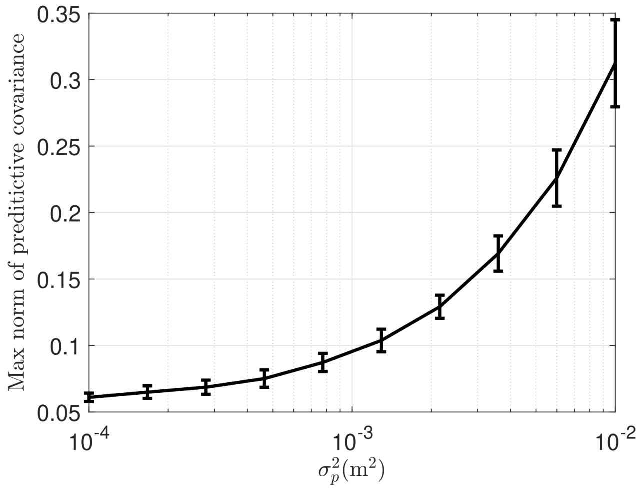
The linearization of the measurement function in (2.13) is performed around the predictive position estimate. The covariance of the estimate can grow when the sensor is moved through an area where the map is previously unknown. The growth rate will depend on the odometry noise. This is demonstrated in Figure 2.7(b), where for 100 Monte Carlo simulations with different odometry noise realisations, the max norm of the predictive estimate from Algorithm 1 can be seen to increase with increasing odometry noise for the same trajectory. In Figure 2.7(a), it can be seen that if we increase the simulated odometry noise above $\sigma_p^2 = 0.002$, the position estimation error of Algorithm 1 is no longer able to compensate for drift in the dead reckoning. A higher odometry noise means more drift is likely to accumulate to the predictive estimation error before revisiting a previously mapped area. The inability of Algorithm 1 to compensate for drift caused by odometry noises above $\sigma_p^2 = 0.002$, therefore, reflects how the assumptions of the measurement function being locally linear no longer hold when the covariance of the predictive distribution becomes large compared to the length scale of the magnetic field disturbances. For the experimental results, for odometry noises above $\sigma_p^2 = 0.002$, we observe a predictive covariance max norm of 0.15 meters in the results in Figure 2.7(b) and an accumulated drift of 0.5 meters, which is combination is comparable in magnitude to our length scale l_{SE} of 0.8 meters. These results are comparable to our simulation results in Section 2.4, where Algorithm 1 for localization only converges when the position error is 0.3 meters using a length scale of 0.2 meters. In both cases, when the order of the prediction error goes beyond the length scale of the magnetic field variations, Algorithm 1 is no longer able to compensate for drift in the position estimate.

2.5.2 Magnetic field SLAM for pedestrians with foot-mounted sensor

Using accelerometer and gyroscope measurements from an IMU mounted on the foot of a pedestrian, it is possible to estimate the position of the pedestrian with high accuracy on a short timescale using a zero-velocity-update (ZUPT) aided EKF [59]. The estimate is obtained by integrating the change in orientation and velocity. In addition, the assumption that when the foot is in the stationary part of the step, it has zero velocity is used to reduce the drift of the position and orientation estimates [59]. The position and orientation estimates are typically accurate at the beginning of a trajectory but can drift over time if



(a) Model ship position estimation error at the end of the trajectory for varying amounts of odometry noise.



(b) The max norm of the predictive covariance of the estimate from Algorithm 1 depending on varying odometry noise.

Figure 2.7: Investigation of the effect of varying odometry noise on the model ship position estimate. The lines connect the average results of the ship position estimation after 100 Monte Carlo repetitions with different realisations of the simulated odometry for varying amounts of odometry noise.

	Data set 1	**Data set 2**	**Data set** 3	**Data set 4**
Algorithm 1	0.53±0.15 (m)	0.58±0.18 (m)	0.53±0.24 (m)	0.98±0.62 (m)
RBPF with 100 particles	0.85±0.27 (m)	0.92±0.26 (m)	0.95±0.42 (m)	1.53±0.53 (m)
RBPF with 200 particles	0.85±0.22 (m)	0.89±0.27 (m)	0.98±0.47 (m)	1.48±0.46 (m)
RBPF with 500 particles	0.87±0.19 (m)	0.86±0.24 (m)	1.00±0.40 (m)	1.53±0.50 (m)
Odometry	1.98±0.54 (m)	1.52±0.48 (m)	1.76±0.52 (m)	1.65±0.47 (m)

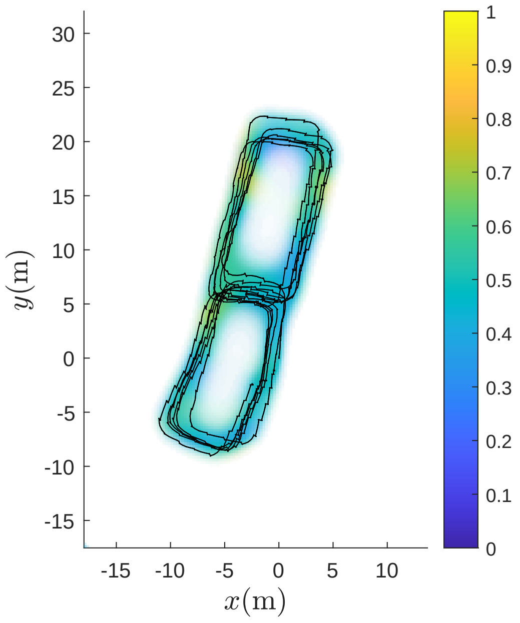
Table 2.1: **Trajectory RMSE** values in meters for the 4 collected data sets from the model ship. Values are given as averages ± one standard deviation, after 100 Monte Carlo repetitions with different realisations of the simulated odometry noise.

	Data set 1	**Data set 2**	**Data set 3**	**Data set 4**
Algorithm 1	0.06±0.01 (s)	0.05±0.00 (s)	0.14±0.01 (s)	0.05±0.00 (s)
RBPF with 100 particles	12.85±0.26 (s)	9.24±0.10 (s)	21.55±0.32 (s)	9.91±0.14 (s)
RBPF with 200 particles	25.70±0.49 (s)	18.44±0.166 (s)	42.64±0.27 (s)	19.72±0.20 (s)
RBPF with 500 particles	64.24±0.82 (s)	46.02±0.20 (s)	106.93±0.34 (s)	49.43±1.29 (s)

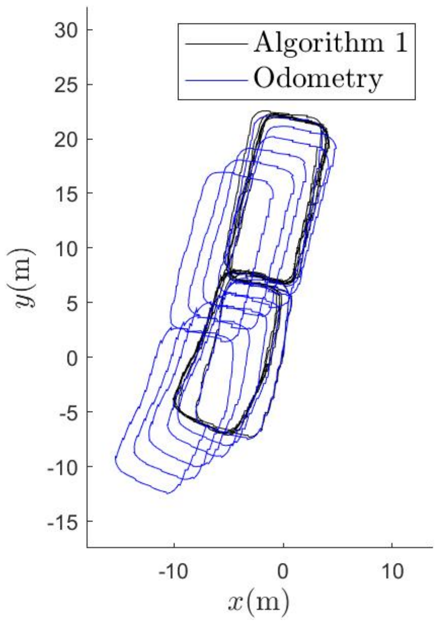
Table 2.2: **Runtime comparison.** Measured time to run the estimation algorithm (in seconds) for the 4 collected data sets from the model ship. Values are given as averages ± one standard deviation, after 100 Monte Carlo repetitions with different realisations of the simulated odometry noise.

biases and/or white noise affect the measurements [60]. This filter was implemented in an open-source implementation by [60].

To test the capabilities of Algorithm 1 on measurements from a foot-mounted sensor, we used magnetic field measurements present in the open-source data set used in [55], to remedy the drift present in the position estimates obtained form their ZUPT-aided EKF. The data set from [55] contains multiple measurement series from an IMU collected in the same hallway, walking the same trajectory. Although the implementation in [55] only used the accelerometer and gyroscope measurements from the IMU, magnetic field measurements were also collected, and are included in the published data. We first ran the open-source implementation from [55] of the ZUPT-aided EKF on the 12 available measurement sequences that were made by collected while walking in a similar trajectory. We ran the ZUPT-aided EKF independently on the 12 experiments and obtained 12 sets of position and orientation estimates. We then concatenated the 12 estimated trajectories by initialising each trajectory at the position and orientation where the previous trajectory ended. This gave a drifting odometry estimate of the position of the pedestrian. The drifting odometry is displayed in Figure 2.8(b). This odometry has an increasing error in position and orientation over time partly because the ZUPT-aided EKF will have some position and orientation drift inherently and partly because of the assumption that the foot-mounted sensor ends in the same orientation at the end of each collected data set as the beginning of the next data set may not be exactly true. However, we can see that most drift accumulates in a constant direction, and drift caused by wrong orientation initialisation should cause twisting of the trajectory. It is, therefore, likely that most of the drift visible in Figure 2.8(b) is present due to inherent drift in the ZUPT-aided EKF. To use the odometry in Algorithm 1, we down-sampled the position and orientation estimates to 10 Hz and computed the change in position and orientation between each time step. We then used the changes in position and orientation as input odometry. As SLAM is performed in real-time, we cannot know



(a) Learned magnetic field and estimated trajectory with odometry from foot-mounted sensor from a birds eye view.



(b) Trajectory estimate from Algorithm 1 compared to odometry from a birds eye view.

Figure 2.8: Trajectory and magnetic field map estimate for the foot-mounted sensor data. The estimated trajectory obtained with Algorithm 1 is compared to odometry from the foot-mounted sensor data obtained via [55] implementation of the ZUPT-aided EKF using a foot-mounted accelerometer and gyroscope. The color of the magnetic field map corresponds to the norm of the estimated magnetic field, and the opacity is inversely proportional with the sum of the marginal variance for each of the three estimated magnetic field components.

the hyperparameters a priori to running the algorithm. The magnetic field measurements available in the open-source data set from [55] were without reported units but had a norm that ranged between 0.2429 and 0.8584. We therefore selected the expected nonlinear variations $\sigma^2_{\mathrm{SE}} = 1$. We assumed that the contribution from the constant earth magnetic field had approximately the same order of magnitude and so selected $\sigma^2_{\mathrm{lin}} = 1$. We set the length scale to $l_{\mathrm{SE}} = 2$ meters, and we set the measurement noise to be $\sigma^2_{\mathrm{m}} = 0.01$. We used 1850 basis functions to approximate the magnetic field map. We selected a domain which was the smallest possible cube that was still at least 10 meters away from the first lap of the odometry. We found empirically that 1850 were a sufficient amount of basis functions using the same approach as in sections 2.4 and 2.5. The resulting position estimate from Algorithm 1 compensates for drift in the odometry, as shown in Figure 2.8(a).

2.6 Conclusion and Future Work

We proposed using an EKF for magnetic field SLAM, which is computationally more efficient and requires less memory than previously proposed methods for magnetic field SLAM. Promisingly, we demonstrated that our proposed algorithm compensates for odometry drift in a way that is comparable to previously proposed, more computationally expensive methods. Using an experiment with magnetic field measurements collected onboard a model ship and using simulated odometry, we ran Monte-Carlo simulations investigating the capabilities of our algorithm to compensate for odometry drift for varying amounts of odometry noise, illustrating that when the uncertainty of the estimate is small compared to the length scale of the magnetic field variations, our proposed algorithm will give a position estimate that compensates for drift in odometry. We also demonstrated the abilities of our proposed algorithm to compensate for drift on an open-source data-set collected with a foot-mounted sensor.

To employ our proposed algorithm in real-life applications such as indoor, surface, underground or underwater navigation, it would be necessary to incorporate sources of odometry information that are available in real-life scenarios, such as inertial sensors or visual odometry from cameras. Another possible direction of future work could be to implement an iterated EKF or another extended Kalman-filter based estimation method that can handle larger non-linearities compared to the EKF [61], and investigate if this improves the convergence of the method. Future research could also look into further reducing the computational requirements associated with reduced-rank GP regression.

2.A Analytical Jacobians

The gradient of the basis functions $\nabla_{\mathrm{p}}\Phi(p)$ used in (2.13) (the basis functions $\Phi(p)$ are defined in (2.8)) is a $3\times(N_m+3)$ matrix. The first three columns are given by an identity matrix

$$\{\nabla_{\mathrm{p}}\Phi(p)\}_{1:3} = \mathcal{I}_{3\times3}, \tag{2.30}$$

and the $j+3$rd column is given by the gradient of the j'th basis function

$$\{\nabla_{\mathrm{p}}\Phi(p)\}_{j+3} = \nabla_{\mathrm{p}}\phi_j(p) = \begin{bmatrix} \frac{\pi n_j(1)}{L_{u,1}-L_{l,1}} c_1 s_2 s_3 \\ \frac{\pi n_j(2)}{L_{u,2}-L_{l,2}} s_1 c_2 s_3 \\ \frac{\pi n_j(3)}{L_{u,3}-L_{l,3}} s_1 s_2 c_3 \end{bmatrix}, \tag{2.31}$$

with s_d and c_d defined for the three spatial dimensions $d = 1,2,3$ as

$$s_d = \sin\left(\pi n_j(d)\frac{(p_d - L_{l,d})}{(L_{u,d}-L_{l,d})}\right)\frac{1}{\sqrt{\frac{1}{2}(L_{u,d}-L_{l,d})}}, \tag{2.32a}$$

$$c_d = \cos\left(\pi n_j(d)\frac{(p_d - L_{l,d})}{(L_{u,d}-L_{l,d})}\right)\frac{1}{\sqrt{\frac{1}{2}(L_{u,d}-L_{l,d})}}. \tag{2.32b}$$

The Jacobians of the basis functions used in (2.25) is a $3\times3\times(N_m+3)$ matrix. The first 3 entries along the third dimensions are all zero-matrices

$$\{\nabla_{\mathrm{pp}}\Phi(p)\}_1 = \{\nabla_{\mathrm{pp}}\Phi(p)\}_2 = \{\nabla_{\mathrm{pp}}\Phi(p)\}_3 = 0_{3\times3}, \tag{2.33}$$

and the $(j+3)$rd entry along the third dimension is given by

$$
\begin{aligned}
&\{\nabla_{\mathrm{pp}}\Phi(p)\}_j = \nabla_{\mathrm{pp}}\phi_j(p) = \\
&\left[\begin{matrix}
-\left(\frac{\pi n_j(1)}{L_{u,1}-L_{l,1}}\right)^2 s_1 s_2 s_3 & \left(\frac{\pi n_j(1)}{L_{u,1}-L_{l,1}}\right)\left(\frac{\pi n_j(2)}{L_{u,2}-L_{l,2}}\right) c_1 c_2 s_3 \\
\left(\frac{\pi n_j(2)}{L_{u,2}-L_{l,2}}\right)\left(\frac{\pi n_j(1)}{L_{u,1}-L_{l,1}}\right) c_1 c_2 s_3 & -\left(\frac{\pi n_j(2)}{L_{u,2}-L_{l,2}}\right)^2 s_1 s_2 s_3 \\
\left(\frac{\pi n_j(3)}{L_{u,3}-L_{l,3}}\right)\left(\frac{\pi n_j(1)}{L_{u,1}-L_{l,1}}\right) c_1 s_2 c_3 & \left(\frac{\pi n_j(3)}{L_{u,3}-L_{l,3}}\right)\left(\frac{\pi n_j(2)}{L_{u,2}-L_{l,2}}\right) s_1 c_2 c_3
\end{matrix}\right. \\
&\quad \left.\cdots \begin{matrix}
\left(\frac{\pi n_j(1)}{L_{u,1}-L_{l,1}}\right)\left(\frac{\pi n_j(3)}{L_{u,3}-L_{l,3}}\right) c_1 s_2 c_3 \\
\left(\frac{\pi n_j(2)}{L_{u,2}-L_{l,2}}\right)\left(\frac{\pi n_j(3)}{L_{u,3}-L_{l,3}}\right) s_1 c_2 c_3 \\
-\left(\frac{\pi n_j(3)}{L_{u,3}-L_{l,3}}\right)^2 s_1 s_2 s_3
\end{matrix}\right].
\end{aligned}
\tag{2.34}
$$

☆3 Distributed multi-agent magnetic field norm SLAM with Gaussian processes

Accurately estimating the positions of multi-agent systems in indoor environments is challenging due to the lack of Global Navigation Satelite System (GNSS) signals. Noisy measurements of position and orientation can cause the integrated position estimate to drift without bound. Previous research has proposed using magnetic field simultaneous localization and mapping (SLAM) to compensate for position drift in a single agent. Here, we propose two novel algorithms that allow multiple agents to apply magnetic field SLAM using their own and other agents' measurements.

Our first algorithm is a centralized approach that uses all measurements collected by all agents in a single extended Kalman filter. This algorithm simultaneously estimates the agents' position and orientation and the magnetic field norm in a central unit that can communicate with all agents at all times. In cases where a central unit is not available, and there are communication drop-outs between agents, our second algorithm is a distributed approach that can be employed.

We tested both algorithms by estimating the position of magnetometers carried by three people in an optical motion capture lab with simulated odometry and simulated communication dropouts between agents. We show that both algorithms are able to compensate for drift in a case where single-agent SLAM is not. We also discuss the conditions for the estimate from our distributed algorithm to converge to the estimate from the centralized algorithm, both theoretically and experimentally.

Our experiments show that, for a communication drop-out rate of 80%, our proposed distributed algorithm, on average, provides a more accurate position estimate than single-agent SLAM. Finally, we demonstrate the drift-compensating abilities of our centralized algorithm on a real-life pedestrian localization problem with multiple agents moving inside a building.

This chapter is based on 🏆 📄 *Frida Viset, Rudy Helmons, Manon Kok*: Distributed multi-agent magnetic field norm SLAM with Gaussian processes. In Proceedings of the 26th International Conference on Information Fusion (FUSION), 2023, 1-8. Best student paper award.

3.1 Introduction

A wide range of research is being performed on multi-agent motion control and path planning algorithms [62]. For most motion control algorithms, it is crucial for each agent to know its own position [63, 64]. Collaborative pedestrian navigation can be useful for example for rescue missions or law enforcement applications [65]. Indoors, Global Navigation Satellite System (GNSS) signal availability is limited and prone to errors [66]. Current indoor navigation systems therefore often rely on integrating measurements of the change in position and orientation. For autonomous navigation in GNSS-denied environments where there are no previously deployed beacons or other structure supporting navigation, measurements of the change in position and orientation are often available from for example inertial sensors, wheel encoders or visual-inertial odometry [67]. Integrating measurements of change in position and orientation (odometry) gives accumulated position estimation errors (drift) that can increase without an upper bound [5].

To compensate for odometry drift, multi-agent simultaneous localization and mapping (SLAM) algorithms for navigation in GNSS-denied environments based on visual information have been widely studied [68]. Visual SLAM can in some applications be infeasible or prone to error due to privacy concerns, varying light conditions, or lack of distinguishable features or landmarks [69].

For several single-agent navigation tasks, magnetic field SLAM has been proposed and demonstrated to compensate for drift in the position estimate [8, 12, 14, 31, 38, 70–72]. The magnetic field indoors is affected by structural metallic elements [24]. In Figure 3.1, an example of the magnetic field norm variations that can be found indoors is displayed. The indoor magnetic field typically has significant spatial variations and stays constant over time [36, 73]. To simultaneously create and use a map of the magnetic field, most approaches use a nonlinear stochastic interpolation scheme to learn the magnetic field online based on measurements. A stochastic interpolation scheme that also gives an uncertainty measure on the predictions in every location of the map is Gaussian process regression. Several of the previous works into magnetic field SLAM use reduced-rank Gaussian process regression approximated with Hilbert space basis functions so the computational complexity does not scale with the number of measurements [8, 12, 14, 74].

The contribution of this paper is twofold. The first contribution is an algorithm that uses all information measured by multiple agents to perform magnetic field norm SLAM online with an extended Kalman filter (EKF). This EKF is obtained by augmenting the state-space of the EKF for magnetic field SLAM in [74] to contain the poses of multiple agents as opposed to just a single agent. We denote this as the centralized algorithm, as it is an algorithm that can be executed in a centralized station that receives all measurements made by all agents. Multi-agent systems do not always have access to a centralized control unit. Our second contribution is therefore a distributed version of the algorithm, where each agent uses information shared in communication between the agents to collaboratively approximate the output of the centralized algorithm. To implement the centralized EKF as a decentralized EKF, we use an approach closely related to the decentralized Kalman filter described in [23]. To the best of the author's knowledge, this is the first proposed algorithm for distributed multi-agent magnetic field SLAM with Gaussian process regression.

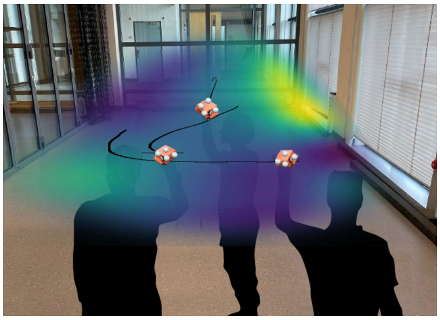


Figure 3.1: Estimated magnetic field map and trajectories of multiple agents based on measurements from magnetometers carried by three people. The position of the magnetometer was recorded in an optical motion capture lab. The color of the map reflects the magnitude of the magnetic field norm, while the opacity of the overlaid map is inversely proportional to the marginal variance of the estimate.

3.2 Connections to previous work

Previous work has applied average consensus to achieve distributed reduced-rank Gaussian process regression using measurements from multiple agents [75, 76]. Recursive stochastic least squares correspond to applying repeated Kalman filter measurement updates [77]. Magnetic field SLAM with an extended Kalman filter uses both a dynamic update and a measurement update at each timestep to jointly estimate the magnetic field map and the pose of a single agent [74]. Previous work has also demonstrated that Kalman filters with both measurement updates and dynamic updates can be implemented for multiple agents distributively with embedded consensus filters [23]. The distributed implementation in [23] is implemented by solving two consensus problems at each time step, one in the dynamic update and one in the measurement update. We also implement the distributed EKF by solving these two consensus problems at each time step. For both our distributed EKF and for the distributed Kalman filter in [23], even if each average consensus problem has not converged, the intermittent result is an approximation of the centralized solution [78].

Unlike previous work into extended Kalman filtering for magnetic field SLAM, we execute the measurement update on the information form. This allows for the measurement update to be implemented distributively by executing the average consensus algorithm at each timestep. Performing the measurement update for magnetic field SLAM on information form is closely related to the execution of the measurement updates on information form for magnetic field mapping proposed by [76]. The main difference between our work and the estimation algorithm presented in [76] is that we jointly and distributively estimate the pose of the agents and the map, while [76] only estimates the map. The main difference between our work and [74] and [13] is that we perform magnetic field SLAM for several agents instead of just one and that we propose a distributed algorithm for doing so. An additional difference between our work and the work presented in [74] is that we for simplicity consider only the magnetic field norm instead of the three-component magnetic field.

3.3 Model

We assume that each individual agent has access to noisy odometry measurements, according to a model we describe in Section 3.3.1. We also assume that each agent carries a magnetometer capable of measuring the magnetic field norm. In Section 3.3.2 we give the measurement model for the magnetometer and the model of the magnetic field norm that we use to apply Gaussian process regression to learn the magnetic field map.

3.3.1 Dynamic model

We estimate the position of a set of m agents indexed as $i = 1,\ldots,m$. The position and orientation of each agent at each timestep t are denoted by the vector $p_{i,t}$ and the unit quaternion $q_{i,t}$ respectively. The quaternion is defined as the orientation from the world frame to the body frame. The body frame has its origin in the IMU's center of mass, and its axes are aligned with the accelerometer sensor axes. The world frame is defined as the stationary inertial frame that shares its origin with the body frame at time $t = 0$, where the gravity field is aligned with the negative z-axis, and the initial yaw-angle between the body and world-frame at $t = 0$ is zero. The position is given in the world frame.

We assume that each agent has access to noisy measurements $\Delta p_{i,t}$ of the change in their position and $\Delta q_{i,t}$ of the change in their orientation from sensors mounted in the body frame. The noisy measurements are defined such that

$$\begin{aligned} p_{i,t+1} &= p_{i,t} + R(q_{i,t})(\Delta p_{i,t} + e_{i,\mathrm{p},t}), && (3.1a)\\ q_{i,t+1} &= q_{i,t} \odot \exp_{\mathrm{q}}(\Delta q_{i,t}) \odot \exp_{\mathrm{q}}(e_{i,\mathrm{q},t}), && (3.1b)\\ [e_{i,\mathrm{p},t}^{\top}, e_{i,\mathrm{q},t}^{\top}]^{\top} &\sim \mathcal{N}(0,\Sigma), && (3.1c) \end{aligned}$$

where $e_{i,\mathrm{p},t}$ is a measurement noise of the change in position, $e_{i,\mathrm{q},t}$ is a measurement noise of the change in orientation, and where Σ is a known noise covariance, $\odot$ is the quaternion product, and $\exp_{\mathrm{q}}$ is the operator that maps an axis-angle orientation deviation to a quaternion, defined as in the odometry model in [8], and where $R(\cdot)$ is an operator transforming a unit quaternion to a rotation, defined as in the odometry model in [13]. Note that we assume the odometry covariance is the same for all agents.

3.3.2 Measurement model

We assume that each agent i has access to a continuous stream of measurements from the magnetic field norm in their current position $p_{i,t}$, according to

$$y_{i,t} = f(p_{i,t}) + e_{i,t}, \qquad e_{i,t} \sim \mathcal{N}(0,\sigma_{\mathrm{y}}^2), \tag{3.2}$$

where $y_{i,t}$ is the measurement from agent i at time t, $f : \mathbb{R}^3 \to \mathbb{R}$ is a function that maps the position to the magnetic field norm, and $e_{i,t}$ is the measurement noise with a covariance σ_{y}^2. We model the function f as a stationary Gaussian process according to

$$f \sim \mathcal{GP}(0, \kappa_{\mathrm{SE}}(\cdot,\cdot)), \tag{3.3}$$

with a squared exponential kernel

$$\kappa_{\mathrm{SE}}(x,x') = \sigma_{\mathrm{SE}}^2 \exp\left(-\frac{\|x-x'\|_2}{2l_{\mathrm{SE}}^2}\right), \tag{3.4}$$

where σ_{SE}^2 and l_{SE} are hyperparameters denoting the variance and lengthscale of the magnetic field norm nonlinearities, respectively [13]. We use the same basis functions as [74] and [13] to approximate the Gaussian process regression. The basis functions are defined in Appendix 6.A.1.

3.3.3 Communication graph

We assume that the agents have the possibility to send and receive N_c messages to all other agents two times at each timestep t, once for the dynamic update and once for the measurement update.

We model the communication graph at time t and communication step t_c as an undirected graph $\mathcal{G}(t,t_c) = (\mathcal{E}(t,t_c),\mathcal{V})$ where $\mathcal{E}(t,t_c) \subset \{\{i,j\}|i,j \in \mathcal{V}\}$ denote the set of active communication edges at time t between the set $\mathcal{V} = \{1,\cdots,m\}$ of all agents. We assume the probability for two agents to be able to communicate at any timestep t at communication step t_c to be $1-\alpha$, where α is the probability of communication failure. We will refer to α as the communication failure rate or the dropout rate in the remainder of this paper. We denote the communication step at each timestep by the index t_c, where $t_c = 1,\ldots,2N_c$.

3.4 Centralized EKF for multi-agent magnetic field SLAM

Following the approach of [74], we parameterize our system in terms of an error state ξ_t linearised about the prior beliefs of the position of agents $i = 1,\ldots,m$ denoted $\tilde{p}_{i,t|t-1}$, the prior beliefs of the orientation of agent $i = 1,\ldots,m$ denoted $\tilde{q}_{i,t|t-1}$ and the prior belief of the map denoted $\tilde{w}_{i,t|t-1}$. The error state ξ_t is defined as

$$\xi_t = [\delta_{1,t}^\top \quad \eta_{1,t}^\top \quad \cdots \quad \delta_{m,t}^\top \quad \eta_{m,t}^\top \quad \nu_t^\top]^\top, \tag{3.5}$$

where $\delta_{i,t} = p_{i,t} - \tilde{p}_{i,t|t-1}$ denotes the position estimation error, $\nu_{t,i} = w - \tilde{w}_{i,t|t-1}$ denotes the magnetic field state estimation error, and $\eta_{i,t}$ denotes the orientation estimation error parameterized as an axis-angle deviation according to

$$q_{i,t} = \exp_{\text{q}}(\eta_{i,t}) \odot \tilde{q}_{i,t|t-1}. \tag{3.6}$$

For simplicity, we assume that the initial position and orientation of all agents are known. The initial error state is then distributed as $\xi_0 \sim \mathcal{N}(0, P_{0|0})$, where $P_{0|0}$ is given by

$$P_{0|0} = \begin{bmatrix} 0 & \cdots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & 0 \\ 0 & \cdots & 0 & \Lambda, \end{bmatrix}. \tag{3.7}$$

with Λ defined in (6.29).

3.4.1 Dynamic update

The posterior linearisation point is propagated to a prior linearisation point by applying the dynamic model in (3.1a)-(3.1c) through the update

$$\tilde{q}_{i,t+1|t} = \tilde{q}_{i,t|t} \odot \exp_{\mathrm{q}}(\Delta q_{i,t}), \qquad i = 1, \ldots, m \tag{3.8a}$$

$$\tilde{p}_{i,t+1|t} = \tilde{p}_{i,t|t} + R(\tilde{q}_{i,t|t})\Delta p_{i,t}, \qquad i = 1, \ldots, m \tag{3.8b}$$

$$\tilde{w}_{t+1|t} = \tilde{w}_{t|t}. \tag{3.8c}$$

The centralized dynamic update is defined as

$$P_{t+1|t} = F_t P_{t|t} F_t^\top + Q, \tag{3.9}$$

where Q is given by

$$Q = \begin{bmatrix} \Sigma & \ldots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \ldots & \Sigma & 0 \\ 0 & \ldots & 0 & 0 \end{bmatrix} \tag{3.10}$$

and F_t is defined as

$$F_t = \begin{bmatrix} F_{1,t} & \ldots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \ldots & F_{m,t} & 0 \\ 0 & \ldots & 0 & I \end{bmatrix}, \tag{3.11}$$

where the matrix $F_{j,t}$ is given by

$$F_{j,t} = \begin{bmatrix} I & R(\tilde{q}_{j,t|t})[\Delta p_{j,t} \times] \\ 0 & I \end{bmatrix}, \tag{3.12}$$

and where $[\Delta p_{i,t} \times]$ is defined as the skew-symmetric matrix such that $[\Delta p_{i,t} \times] u = \Delta p_{i,t} \times u$ gives the cross-product between $\Delta p_{i,t}$ and a vector $u \in \mathbb{R}^3$.

3.4.2 Measurement update

The measurement update is performed by linearising the measurement model in (3.2) about the prior linearisation point with respect to the error state ξ_t. We let the information vector $\iota_{t|t-1}$ and information matrix $\mathcal{I}_{t|t-1}$ denote the information form of the state estimate $\hat{\xi}_{t|t-1}$ and the corresponding covariance $P^{-1}_{t|t-1}$, according to

$$\iota_{t|t-1} = P^{-1}_{t|t-1} \hat{\xi}_{t|t-1} = 0, \tag{3.13}$$

$$\mathcal{I}_{t|t-1} = P^{-1}_{t|t-1}. \tag{3.14}$$

The Kalman filter measurement update can then be expressed as an update of the information matrix and information vector as

$$\mathcal{I}_{t|t} = \mathcal{I}_{t|t-1} + \sum_{i=1}^{m} \frac{1}{\sigma_{\mathrm{y}}^2} H_{i,t} H_{i,t}^\top, \tag{3.15a}$$

$$\iota_{t|t} = \iota_{t|t-1} + \sum_{i=1}^{m} \frac{1}{\sigma_{\mathrm{y}}^2} H_{i,t} (y_{i,t} - \Phi(\tilde{p}_{i,t-1})^\top \tilde{w}_{t|t-1}), \tag{3.15b}$$

Algorithm 2 Centralized EKF for multi-agent magnetic field SLAM

```
1: Input: $\left\{\{\Delta p_{i,t}, \Delta q_{i,t}, y_{i,t}\}_{t=1}^{N}\right\}_{i=1}^{m}$
2:
3: Output: $\left\{\left\{\tilde{p}_{i,t|t}, \tilde{q}_{i,t|t}, \tilde{w}_{t|t}\right\}_{t=1}^{N}\right\}_{i=1}^{m}$
4: Initialization: $\tilde{p}_{i,0|0} = 0_{3\times1}$, $\tilde{q}_{i,0|0} = q_{0,i}$, $\tilde{w}_{0|0} = 0_{M\times1}$, (3.7)
5: for $t = 1$ to $N$ do
6:     Dynamic update according to (3.8a), (3.8b), (3.8c) and (3.9).
7:     Measurement update according to (3.14), (3.15a), (3.15b) and (3.17). Relinearization
       according to (3.18a), (3.18b) and (3.18c).
8: end for
```

with

$$
\begin{aligned}
H_{i,t} =& [0_{1\times6(i-1)}, (\nabla\Phi(\tilde{p}_{i,t|t-1})\tilde{w}_{t|t-1})^\top, \\
& 0_{1\times(3+6(m-i))}, (\Phi(\tilde{p}_{i,t|t-1}))^\top]^\top.
\end{aligned}
\tag{3.16}
$$

The posterior error state estimate and covariance are given by

$$
\hat{\xi}_{t|t} = \mathcal{I}_{t|t}^{-1}\iota_{t|t}, \qquad P_{t|t} = \mathcal{I}_{t|t}^{-1}. \tag{3.17}
$$

The posterior linearisation point can then be calculated by propagating the estimated error state to the prior linearisation point according to

$$
\tilde{p}_{i,t|t} = \tilde{p}_{i,t|t-1} + \hat{\delta}_{i,t|t}, \qquad i = 1, \ldots, m, \tag{3.18a}
$$
$$
\tilde{q}_{i,t|t} = \exp_{\mathrm{q}}(\hat{\eta}_{i,t|t}) \odot \tilde{q}_{i,t|t-1}, \qquad i = 1, \ldots, m, \tag{3.18b}
$$
$$
\tilde{w}_{t|t} = \tilde{w}_{t|t-1} + \hat{v}_{t|t}. \tag{3.18c}
$$

Recursively applying the dynamic update and measurement update results in the centralized EKF for multi-agent magnetic field SLAM, as described in Algorithm 1.

3.5 Distributed multi-agent EKF for magnetic field SLAM

We denote agent i's approximation of a centralized term by including a superscript (i) on the approximated term. The initial posterior linearisation points are known and given as $\tilde{p}_{i,0|0}^{(i)} = p_{i,0}$, $\tilde{q}_{i,0|0}^{(i)} = q_{i,0}$ and $\tilde{w}_{0|0}^{(i)} = 0$. As in the centralized filter, we assume that the initial error centralized error state $\hat{\xi}_{t|t}^{(i)} = 0$ and the initial centralized covariance $P_{0|0}^{(i)} = P_{0|0}$ are both known.

3.5.1 Dynamic update

In the case where each agent only has access to their own measurements, the posterior linearisation point of each agent can be propagated to a prior linearisation point through the dynamic model in the same way as for the centralized EKF, using (3.9). The matrix F_t cannot be computed directly by any agent as each term $F_{j,t}$ contains the odometry measurement $\Delta p_{j,t}$ which is only available to agent j. The matrix F_t can however be approximated by the network as a whole through average consensus, if each agent initializes their belief about the matrix $F_t^{(i)}$ according to

$$F_t^{(i)} = \begin{bmatrix} F_{1,t}^{(i)} & \dots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \dots & F_{m,t}^{(i)} & 0 \\ 0 & \dots & 0 & I \end{bmatrix}, \tag{3.19}$$

where the term $F_{j,t}^{(i)}$ is defined according to

$$F_{j,t}^{(i)} = \begin{cases} mF_{j,t} - (m-1)I, & j = i \\ I, & j \neq i \end{cases} \tag{3.20}$$

The average of all the terms $\{F_t^{(i)}\}_{i=1}^m$ is F_t, so applying average consensus according to

$$F_t^{(i)} \leftarrow \sum_{j=1}^{m} W_{i,j}(t,t_c) F_t^{(j)}, \tag{3.21}$$

where the weights $W_{i,j}(t,t_c)$ are defined as in [78] as

$$W_{i,j}(t,t_c) = \begin{cases} \frac{1}{m}, & i,j \in \mathcal{E}(t,t_c) \\ 1 - \frac{d_i(t,t_c)}{m}, & i = j \\ 0, & \text{otherwise} \end{cases}, \tag{3.22}$$

and where $d_i(t,t_c)$ are the number of edges to node i in the communication graph $\mathcal{E}(t,t_c)$ at timestep t, for $t_c = 1,\dots,N_c$ causes $F_t^{(i)}$ to converge to F_t as $N_c \to \infty$ [78]. As we only apply a finite amount of average consensus steps N_c, we use $F_t^{(i)}$ at time N_c as an approximation in the dynamic update of the covariance.

3.5.2 Measurement update

The measurement update can be carried out in a distributed manner by first letting each agent update its belief about the information vector according to

$$\mathcal{I}_{t|t}^{(i)} = \mathcal{I}_{t|t-1}^{(i)} + m\frac{1}{\sigma_y^2} H_{i,t} H_{i,t}^\top, \tag{3.23a}$$

$$\iota_{t|t}^{(i)} = \iota_{t|t-1}^{(i)} + m\frac{1}{\sigma_y^2} H_{i,t}(y_{i,t} - \Phi(\tilde{p}_{i,t-1}^{(i)})^\top \tilde{w}_{t|t-1}^{(i)}), \tag{3.23b}$$

and then carry out average consensus across the network on the resulting information matrix and information vector, according to

$$\iota_{t|t}^{(i)} \leftarrow \sum_{j=1}^{m} W_{i,j}(t,t_c) \iota_{t|t}^{(j)} \tag{3.24a}$$

$$\mathcal{I}_{t|t}^{(i)} \leftarrow \sum_{j=1}^{m} W_{i,j}(t,t_c) \mathcal{I}_{t|t}^{(j)} \tag{3.24b}$$

Algorithm 3 Distributed EKF for multi-agent magnetic field SLAM for agent i

```
1: Input: ${\{\{\Delta p_{i,t}, \Delta q_{i,t}, y_{i,t}\}_{t=1}^{N}\}_{i=1}^{m}}$
2:
3: Output: $\left\{\left\{\tilde{p}_{i,t|t}^{(i)}, \tilde{q}_{i,t|t}^{(i)}, \tilde{w}_{t|t}^{(i)}\right\}_{t=1}^{N}\right\}_{i=1}^{m}$
4: Initialization: $\tilde{p}_{i,0|0}^{(i)} = 0_{3\times 1}$, $\tilde{q}_{i,0|0}^{(i)} = q_0$, $\tilde{w}_{0|0}^{(i)} = 0_{M\times 1}$, (3.7)
5: for $t = 1$ to $N$ do
6:     Dynamic update: Perform average consensus according to (3.19), (3.20) and (3.21).
       Then, propagate own belief of own state according to (3.8a), (3.8b), (3.8c) and (3.9),
       using output terms from average consensus.
7:     Measurement update: according to (3.23a), (3.23b). Average consensus according
       to (3.24a) and (3.24b). Relinearization according to (3.18a), (3.18b) and (3.18c).
8: end for
```

The result will then converge to the information matrix and information vector obtained by (3.15a)- (3.15b) as the number of communication steps goes to infinity. We use the output from the average consensus procedure as an approximation to the centralized information matrix in each agent. Each agent can therefore update their own linearization point locally by using the same update as the centralized EKF in (3.18a)-(3.18c). When there is no communication failure, the approximation will be exactly equal to the centralized solution even with $N_c = 1$. Recursively applying the dynamic update and the measurement update gives the Distributed EKF for multi-agent magnetic field SLAM described in Algorithm 2. If all m agents are running Algorithm 2, the multi-agent system will collaboratively approximate the centralized estimate of Algorithm 1.

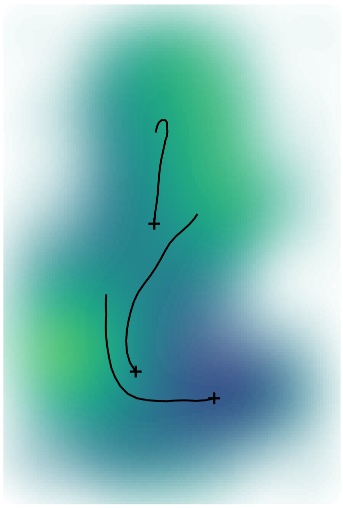
3.6 Results

3.6.1 Comparison of Algorithm 1 to Single-Agent SLAM

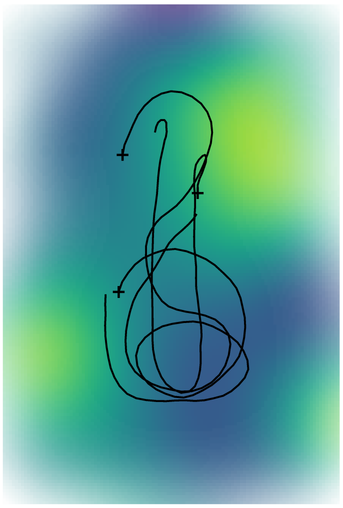
We test the ability of our algorithm to simultaneously estimate the locations of three handheld devices containing magnetometers, by testing on data collected by three test subjects in a motion capture lab. The experimental setup is illustrated in Figure 3.1. Each test subject held an Xsens MTi-100 IMU, which was used to collect magnetic field measurements. The ground truth position and orientation of the IMU were recorded with an optical motion capture system. The test subjects moved sequentially in the test area to ensure marker visibility for the optical motion capture system, but we test our algorithm on the three measured trajectories as if collected simultaneously.

The position measurements $\Delta p_{i,t}$ were simulated by first computing the difference of the recorded ground truth positions from each timestep to the next, and then adding noises of $e_{1,\mathrm{p},t} = [0.000, 0.001\ 0]$, $e_{2,\mathrm{p},t} = [-0.001, -0.0005, 0]$ and $e_{3,\mathrm{p},t} = [0.001, -0.0005, 0]$. These noise values were selected such that the position estimates of the agents would drift in different directions over a short timescale, making the dead-reckoning position estimates to other agents particularly poor.

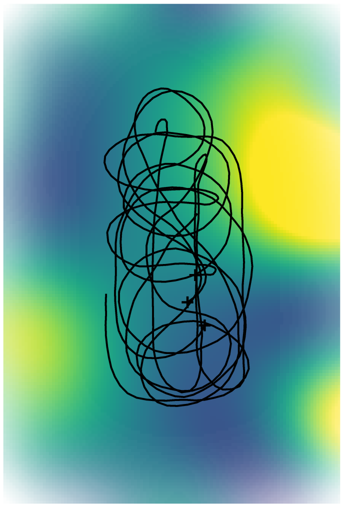
The differential orientation measurements $\Delta q_{i,t}$ were simulated by computing the



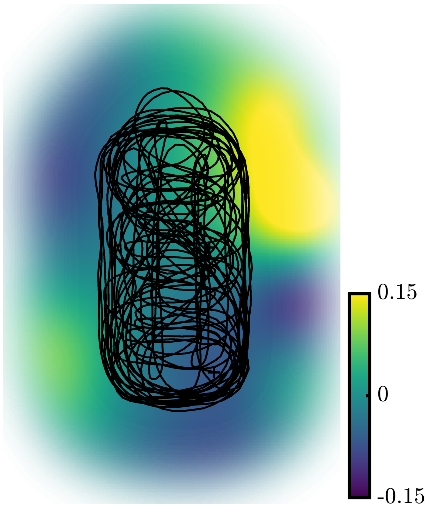
(a) t = 2 seconds



(b) t = 7 seconds



(c) t = 20 seconds



(d) t = 80 seconds

Figure 3.2: Learned magnetic field map using Algorithm 2. The intensity of the learned magnetic field norm is indicated by the color, while the marginal variance of the magnetic field map is inversely proportional to the opacity. The estimated trajectories of the agents are indicated with black lines, and the current positions at each time are indicated with black crosses.

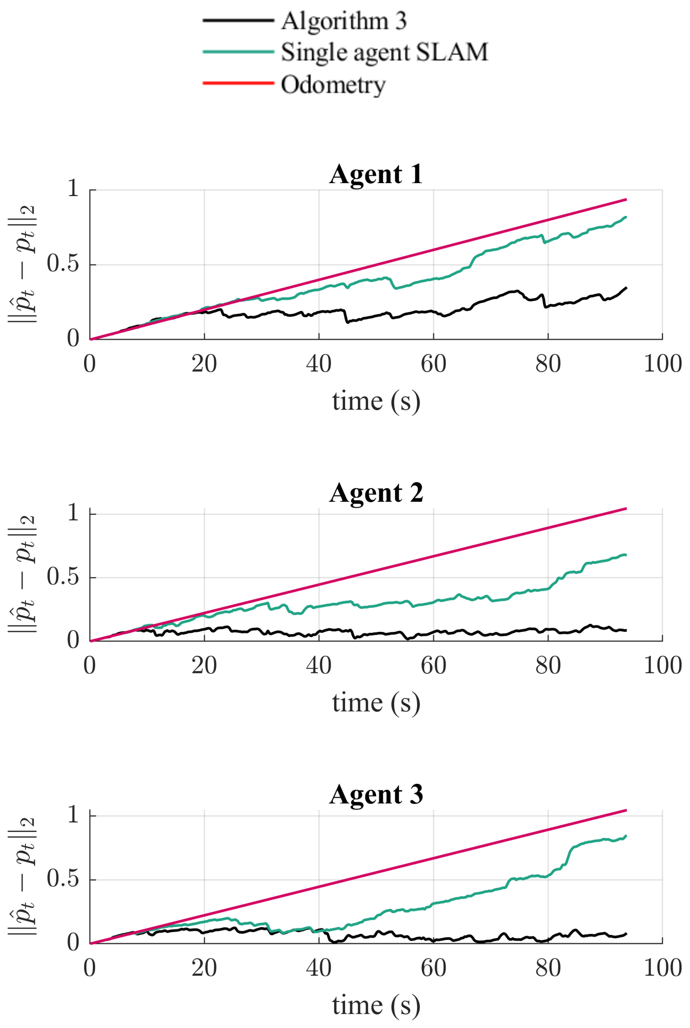


Figure 3.3: Position estimation errors for three agents, relative to known ground truth position measured with optical motion capture system. The error is given as the euclidian distance between the true position p_t and the estimated position $\hat{p}_t$ from Algorithm 3 with $N_c = 10$ and $\alpha = 0.2$, from Single agent SLAM and from integrating the pure odometry, respectively.

difference in orientation from one timestep to the next, and then adding a simulated noise sampled from a normal distribution with standard deviation $\sigma_q = 1.0e-5$. We then applied Algorithm 1 to the magnetic field measurements and the simulated odometry. The Gaussian process hyperparameters were set to $\sigma_{\mathrm{SE}} = 0.074$, $\sigma_y = 0.0042$, $l_{\mathrm{SE}} = 0.86$m. The hyperparameters were selected based on an optimization of the Gaussian process likelihood, using the recorded position for all the agents as the input locations and the magnetic field norm as the output. The parameter σ_p used in the estimation was set to 0.022, which is two times as high as the maximum norm of the simulated noise, to make sure that the Kalman filter did not put too much trust in the odometry. To approximate the Gaussian process, 100 basis functions were used in a domain Ω defined as the smallest cube that was no closer than 3 meters to the closest recorded position. This is a sufficient amount of basis functions, as the approximation error between the reduced rank and the full GP in ten test points selected in random locations sampled from a uniform distribution inside the domain given all the collected measurements is lower than one measurement noise standard deviation σ_y.

The estimated trajectories using Algorithm 1 are displayed together with the learned magnetic field in Fig. 3.2. The results in Figure 3.3 show that the EKF for a single agent improves on the position estimate for all three agents. The end-point estimation error for Single agent SLAM is 87%, 65%, and 81% of the odometry error, respectively. Over time, the position estimates for Single agent SLAM are typically bounded [74], but for this example on this timescale, each agent does not have time to collect sufficient information about the magnetic field to compensate fully for the odometry drift. Even in this challenging case for magnetic field SLAM, multi-agent SLAM is able to compensate for the odometry drift. The end-point estimation errors of the position estimates from Algorithm 1 in Figure 3.3 are 37%, 8.2% and 7.9% compared to odometry error, for the three agents respectively.

3.6.2 Testing Algorithm 2 on real magnetic field measurements with simulated odometry noise

We investigate the effects of varying communication failure rates α on the difference between the distributed estimate from Algorithm 2 and the centralized estimate from Algorithm 1. To study the most challenging case, we assume the agents have the possibility to communicate only once for each average consensus problem. By using the approximation obtained through one step of average consensus, we see in Figure 3.4 that the distributed algorithm is able to give an improved position estimate compared to single-agent magnetic field SLAM for failure rates up until 80%.

Each average consensus problem will give a solution that is exactly corresponding to the centralized solution when the communication failure rate is zero [23]. Otherwise, average consensus gives an approximation that converges to the true estimate as $N_c \to \infty$. The results in Fig. 3.4 confirm that the estimation error of Algorithm 2 is equivalent to the estimation error of Algorithm 3 when the dropout rate is zero. Furthermore, the results in Fig. 3.4 show that increasing the dropout rate α, increases the estimation error of Algorithm 2. For all dropout rates of 80% or lower, the resulting position estimate from Algorithm 2 is closer to the centralized solution compared to the Single-agent SLAM. The results in Fig. 3.5 show that for higher N_c, the distributed estimate converges more rapidly to the centralized estimate as α increases. When the communication failure rate is zero,

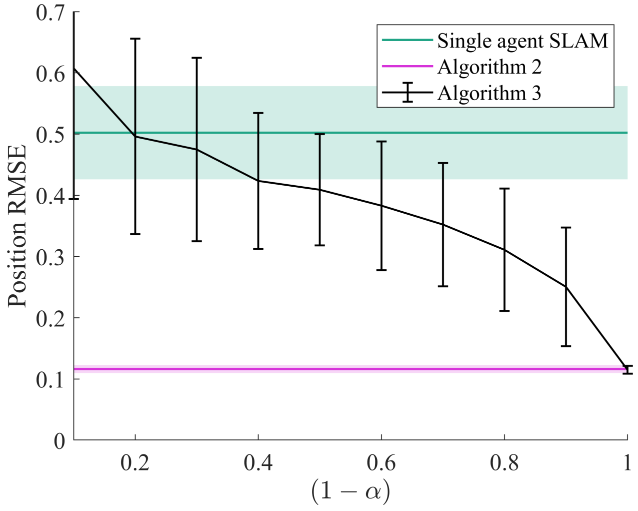


Figure 3.4: RMSE of the full trajectory estimate using measurements from the motion capture lab, for a range of communication failure rates α. The error bars indicate one standard deviation after 100 Monte-Carlo repetitions. The green line marks the average deviation in the position estimate between the single-agent SLAM solution and the centralized solution after 100 Monte-Carlo repetitions, and the light green area marks the range of one standard deviation. Algorithm 3 was run with $N_c = 1$.

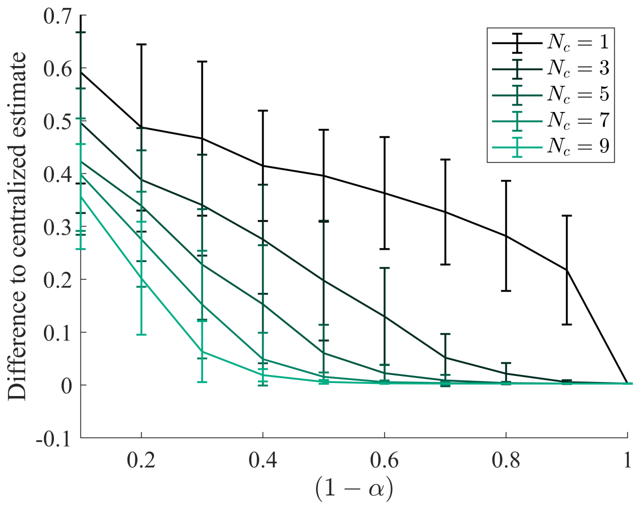


Figure 3.5: Deviation between the estimate from Algorithm 2 and Algorithm 3 using measurements from the motion capture lab, for a range of communication rates α, and a range of communication steps at each iteration N_c. The lines connect the average results after 100 MC repetitions, and the error bars indicate one standard deviation.

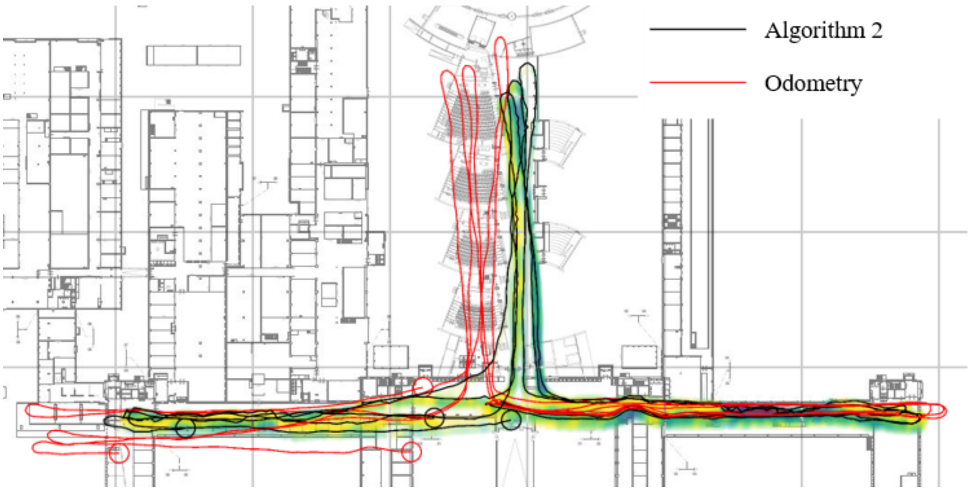


Figure 3.6: Learned magnetic field map and estimated trajectories for three agents in a large building. The black circles indicate the estimated end positions of the agents using Algorithm 2, while the red circles indicate the estimated end positions of the agents using visual-inertial odometry. The color of the map is proportional to the learned intensity of the magnetic field norm, while the opacity is inversely proportional with the marginal variance.

so for $1-\alpha = 1$, we can observe that the position estimate from the distributed EKF is equivalent to the position estimate from the centralized algorithm.

3.6.3 Indoor experiment with three smartphone measurements

To test our algorithm on a larger scale experiment with real odometry and magnetic field norm measurements, we collected three sequences of visual-inertial odometry and magnetic field norm measurements inside a building using Google Pixel smartphone. Google provides a platform primarily targeted at building augmented reality Apps called ARCore. Among other features, ARCore uses the phone's camera, accelerometer and gyroscope to compute a position and orientation estimate. Using a customized app, we simultaneously recorded this position and orientation estimate and the magnetometer measurements from the phone's built-in magnetometer at 200Hz. We subsequently computed the magnetic field norm using the three-component magnetic field measurements, and down-sampled all measurements to 10 Hz. Algorithm 1 was applied to these three sequences as if they were collected by three separate agents simultaneously. The algorithm was applied with the following hyperparameters: $\sigma_{\text{SE}} = 7.2$, $l_{\text{SE}} = 1.2$m, $\sigma_{\text{y}} = 1.2$, $\sigma_{\text{p}} = 0.15$, $\sigma_{\text{q}} = 0.0001$ and with 500 basis functions in cubic tiles of size 38m$\times$38m$\times$38m. The tiles were placed with 8 meters of overlap at the borders. The resulting visual-inertial odometry estimate of the three trajectories is displayed in Fig. 3.6. The trajectories are illustrated with respect to the floor plan of the building where they were collected. The visual-inertial odometry is initially close to the real position, but over time, it drifts away from the hallways where the measurements were collected. In the same figure, the resulting position estimate of Algorithm 1 is displayed. These estimates are closer to the hallways where the measurements were collected, and therefore likely to have higher accuracy. The magnetic field map learned collaboratively by the three agents is displayed in Fig. 3.6.

3.7 Conclusion

For multiple agents navigating in a new environment, we proposed two algorithms that allow them to collaborate about solving the simultaneous mapping and localization task. The first algorithm can be employed when a central unit has access to all measurements from all agents. The second algorithm allows for multiple agents to collaboratively approximate the estimate of the first algorithm when there is no central station that can communicate with all agents at all times. Our proposed algorithms are capable of compensating for drift also in cases where single-agent SLAM fails to do so. We presented experimental results that confirm that the centralized multi-agent SLAM algorithm obtains a higher position accuracy compared to single-agent magnetic field SLAM. For our experimental results, the second algorithm was shown to give more accurate position estimates compared to single-agent SLAM for communication drop-out rates up until 80%.

3.A Basis function definitions

The basis functions are defined over a finite-support cubical domain $\Omega \subset \mathbb{R}^d$, defined as $\Omega = [L_{l,1}, L_{u,1}] \times [L_{l,2}, L_{u,2}] \times [L_{l,3}, L_{u,3}]$. The basis functions are given as

$$\phi_i(p) = \prod_{d=1}^{3} \frac{\sqrt{2}}{\sqrt{L_{\mathrm{u},d} - L_{\mathrm{l},d}}} \sin\left(\frac{\pi n_{i,d}(p_d + L_{\mathrm{l},d})}{L_{\mathrm{u},d} - L_{\mathrm{l},d}}\right), \tag{3.25}$$

where the set $(n_{i,1}, n_{i,2}, n_{i,3})$ is the set of three natural numbers that is different from the sets $(n_{j,1}, n_{j,2}, n_{j,3})$ defined for all $j < i$, that gives the corresponding value of a parameter λ_i defined as

$$\lambda_i = \sum_{d=1}^{D} \left(\frac{\pi n_{i,d}}{L_{\mathrm{u},d} - L_{\mathrm{l},d}}\right)^2, \tag{3.26}$$

as large as possible. These basis functions are then used to approximate the Gaussian process prior with a parametric prior

$$f \approx \Phi^\top w, \qquad w \sim \mathcal{N}(0, \Lambda), \tag{3.27}$$

where Φ is a vector of M basis functions $\phi_i : \mathbb{R}^d \to \mathbb{R}$, $w \in \mathbb{R}^M$ is a vector of weights, and Λ is defined as

$$\Lambda = \operatorname{diag}\left[S_{\mathrm{SE}}(\sqrt{\lambda_1}), \quad \cdots, \quad S_{\mathrm{SE}}(\sqrt{\lambda_{N_m}})\right], \tag{3.28}$$

with $S_{\mathrm{SE}}(\cdot)$ being the spectral density of the squared exponential kernel, as defined in [9]. This means that the approximation of the magnetic field norm in (6.28) has a prior distribution that tends to (6.1) as M goes to infinity, and the size of the domain goes to infinity [22].

4 Exploiting Hankel-Toeplitz structures for fast computation of kernel precision matrices

The Hilbert-Space Gaussian Process (HGP) approach offers a hyperparameter-independent basis function approximation for speeding up Gaussian Process (GP) inference by projecting the GP onto M basis functions. These properties result in a favorable data-independent $\mathcal{O}(M^3)$ computational complexity during hyperparameter optimization but require a dominating one-time precomputation of the precision matrix costing $\mathcal{O}(NM^2)$ operations. In this paper, we lower this dominating computational complexity to $\mathcal{O}(NM)$ with no additional approximations. We can do this because we realize that the precision matrix can be split into a sum of Hankel–Toeplitz matrices, each having $\mathcal{O}(M)$ unique entries. Based on this realization we propose computing only these unique entries at $\mathcal{O}(NM)$ costs. Further, we develop two theorems that prescribe sufficient conditions for the complexity reduction to hold generally for a wide range of other approximate GP models, such as the Variational Fourier Feature approach. The two theorems do this with no assumptions on the data and no additional approximations of the GP models themselves. Thus, our contribution provides a pure speed-up of several existing, widely used, GP approximations, without further approximations.

This chapter is based on *Frida Viset, Anton Kullberg, Frederiek Wesel and Arno Solin*: Exploiting Hankel-Toeplitz Structures for Fast Computation of Kernel Precision Matrices. Accepted to TMLR (Transactions on Machine Learning Research)

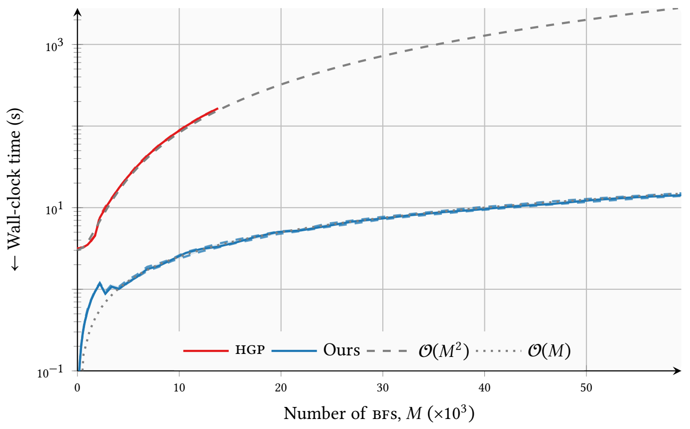


Figure 4.1: **An order of magnitude speed-up without any additional approximations:** Wall-clock time to compute the precision matrix for an increasing number M of basis functions.

4.1 Introduction

Gaussian Processes (GP) [16] provide a flexible formalism for modeling functions which naturally allows for the incorporation of prior knowledge and the production of uncertainty estimates in the form of a predictive distribution. Typically a GP is instantiated by specifying a prior mean and covariance (kernel) function, which allows for incorporation of prior knowledge. When data becomes available, the GP can then be conditioned on the observations, yielding a new GP which can be used for predictions and uncertainty quantification.

While all these operations have closed-form expressions for regression, the computations of the mean and covariance of the predictive GP require instantiating and inverting the kernel (Gram) matrix, which encodes pair-wise similarities between all data. These operations require respectively $\mathcal{O}(N^2)$ and $\mathcal{O}(N^3)$ computations, where N is the number of data points. Furthermore, if one wishes to optimize the hyperparameters of the kernel function, which in GPs is typically accomplished by optimizing the log-marginal likelihood, the kernel matrix needs to be instantiated and inverted multiple times, further hindering the applicability of GPs to large-scale datasets.

The ubiquitous strategy to reduce this computational complexity consists in approximating the kernel matrix in terms of BFs, yielding what is essentially a low-rank or "sparse" approximation of the kernel function [16, 21, 79]. Computational savings can then be achieved by means of the matrix inversion lemma, which requires instantiating and inverting the *precision matrix* (sum of the outer products of the BFs) instead of the kernel matrix at prediction, thereby lowering the computational complexity of inference to $\mathcal{O}(NM^2 + M^3)$, where M is the number of BFs. If the number of BFs is chosen smaller than the number of data points in the training set ($M < N$), computational benefits arise and the computational costs for hyperparameter optimization and inference are dominated by $\mathcal{O}(NM^2)$.

It is less widely known that this cost can be improved further. A notable BF framework is the Hilbert-space Gaussian process (HGP) [11] which projects the GP onto a dense set of orthogonal, hyperparameter-independent basis functions (BFs). This deterministic approximation is particularly attractive as it exhibits fast convergence guarantees to the

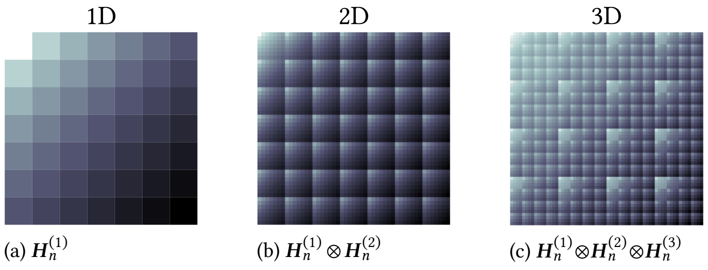


Figure 4.2: The precision matrix for polynomial basis functions has a nested Hankel structure. The visualization of the matrix is proportionally darker as the logarithm of each entry increases. The matrices are computed as the sum of all entries $\boldsymbol{H}_n$ for $n = \{1, \dots, N\}$, where the expression for $\boldsymbol{H}_n$ is given below each matrix.

full GP in terms of the number of BFs for smooth shift-invariant kernels compared to other approximations. Furthermore, the fact that the BFs are hyperparameter-independent speeds up GP hyperparameter optimization considerably, which in the HGP requires only $\mathcal{O}(M^3)$ operations after a one-time precomputation of the precision matrix, costing $\mathcal{O}(NM^2)$. These favorable properties have ensured a relatively wide usage of the HGP [see e.g. 80–82], and it is available in, e.g., PyMC [83] and Stan [84]. However, as argued by [85], a high number of BFs may be required for a faithful approximation of the full model. Thus, the HGP is typically employed in applications where forming the initial $\mathcal{O}(NM^2)$ projection does not become too heavy.

In this paper, we reduce this complexity to $\mathcal{O}(NM)$ with *no additional approximations* (see figure 4.1), through exploiting structural properties of the precision matrix. Our contributions are as follows.

- We show that each increment of the HGP precision matrix can be split in a multilevel block-Hankel and a multilevel block-Toeplitz matrix (figure 4.2), each having only $\mathcal{O}(M)$ unique entries instead of $\mathcal{O}(M^2)$. This allows us to determine the elements of the full precision matrix at $\mathcal{O}(NM)$ instead of $\mathcal{O}(NM^2)$.

- We provide sufficient conditions regarding the choice of BF for this reduction in computational complexity to hold in general. This is the case for many BFs such as polynomial, (complex) exponential and (co)sinusoidal basis functions. By doing so, we enable the speed-up of other BF-based GP approximations such as variational Fourier features [17].

In the experiments, we demonstrate in practice that our approach lowers the computational complexity of the HGP by an order of magnitude on simulated and real data.

4.2 Background

A GP is a collection of random variables, any finite number of which have a joint Gaussian distribution [16]. We denote a zero–mean GP by $f \sim \mathcal{GP}(0, \kappa(\cdot,\cdot))$, where $\kappa(\boldsymbol{x}, \boldsymbol{x}') : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$ is the kernel, representing the covariance between inputs $\boldsymbol{x}$ and $\boldsymbol{x}'$. Given a dataset of input–output pairs $\{(\boldsymbol{x}_n, y_n)\}_{n=1}^N$, GPs are used for non-parametric regression and classification by coupling the latent functions f with observations through a likelihood model $p(\boldsymbol{y} \mid f) = \prod_{i=1}^{N} p(y_i \mid f(\boldsymbol{x}_i))$.

For notational simplicity, we will in the following focus on GP models with a Gaussian (conjugate) likelihood, $y_i \sim \mathcal{N}(f(\boldsymbol{x}_i), \sigma^2)$. The posterior GP, $\mathcal{GP}(\mu_\star(\cdot), \Sigma_\star(\cdot,\cdot))$ can be written down in closed form by

$$\mu_\star(\boldsymbol{x}_\star) = \boldsymbol{k}_\star^\top(\boldsymbol{K} + \sigma^2 \boldsymbol{I})^{-1}\boldsymbol{y}, \tag{4.1a}$$

$$\Sigma_\star(\boldsymbol{x}_\star, \boldsymbol{x}_\star') = \boldsymbol{k}(\boldsymbol{x}_\star, \boldsymbol{x}_\star') - \boldsymbol{k}_\star^\top(\boldsymbol{K} + \sigma^2 \boldsymbol{I})^{-1}\boldsymbol{k}_{\star'}, \tag{4.1b}$$

where $\boldsymbol{K} \in \mathbb{R}^{N\times N}$ and $\boldsymbol{k}_\star \in \mathbb{R}^N$ are defined element-wise as $\boldsymbol{K}_{i,j} := \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)$ and $[\boldsymbol{k}_\star]_i := \kappa(\boldsymbol{x}_i, \boldsymbol{x}^\star)$ for $i, j \in 1, 2, \ldots, N$, and $\boldsymbol{k}_{\star'}$ is defined similarly. Observations are collected into $\boldsymbol{y} \in \mathbb{R}^N$. Due to the inverse of $(\boldsymbol{K} + \sigma^2 \boldsymbol{I})$ in the posterior mean and covariance, the computational cost of a standard GP scales as $\mathcal{O}(N^3)$, which hinders applicability to large datasets.

4.2.1 Basis Function Approximations

The prevailing approach in literature to circumvent the $\mathcal{O}(N^3)$ computational bottleneck is to approximate the GP with a sparse approximation, using a finite number of either inducing points or basis functions [e.g., 16, 17, 21].

The basis function (BF) representation is commonly motivated by the approximation

$$\kappa(\boldsymbol{x}, \boldsymbol{x}') \approx \boldsymbol{\phi}(\boldsymbol{x})^\top \boldsymbol{\Lambda}\, \boldsymbol{\phi}(\boldsymbol{x}'). \tag{4.2}$$

We use the notation from [11] to align with the next section. Here, $\boldsymbol{\phi}(\cdot) : \mathbb{R}^{\mathbb{D}} \to \mathbb{R}^M$ are the BFs, $\boldsymbol{\phi}(\cdot) := [\phi_1(\cdot), \phi_2(\cdot), \ldots, \phi_M(\cdot)]^\top$. Further, $\boldsymbol{\Lambda} \in \mathbb{R}^{M\times M}$ are the corresponding BF weights. Combining this approximation with the posterior GP, equation (4.1), and applying the Woodbury matrix inversion lemma yields

$$\mu_\star(\boldsymbol{x}_\star) = \boldsymbol{\phi}(\boldsymbol{x}_\star)^\top \left(\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \sigma^2\boldsymbol{\Lambda}^{-1}\right)^{-1} \boldsymbol{\Phi}^\top \boldsymbol{y}, \tag{4.3a}$$

$$\Sigma_\star(\boldsymbol{x}_\star, \boldsymbol{x}_\star') = \sigma^2 \boldsymbol{\phi}(\boldsymbol{x}_\star)^\top \left(\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \sigma^2\boldsymbol{\Lambda}^{-1}\right)^{-1} \boldsymbol{\phi}(\boldsymbol{x}_\star'). \tag{4.3b}$$

Here, $\boldsymbol{\Phi} \in \mathbb{R}^{N\times M}$ is commonly referred to as the regressor matrix and is defined as $\boldsymbol{\Phi}_{i,:} := \boldsymbol{\phi}(\boldsymbol{x}_i)^\top$. Further, $\boldsymbol{\Phi}^\top\boldsymbol{\Phi} \in \mathbb{R}^{M\times M}$ is the *precision matrix* which is a central component of the following section. With this approximation, computing the posterior mean and covariance in equation (4.3) requires instantiating and inverting $\left(\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \sigma^2\boldsymbol{\Lambda}^{-1}\right)^{-1}$ which can be performed with $\mathcal{O}(NM^2 + M^3)$ operations.

4.3 Method

Our main findings are in the form of two theorems (theorems 4.3.1 and 4.3.4). These theorems prescribe the necessary conditions that BF expansions need to fulfill to be able to reduce the computational complexity of computing the precision matrix $\boldsymbol{\Phi}^\top\boldsymbol{\Phi}$ from $\mathcal{O}(NM^2)$ to $\mathcal{O}(NM)$, applicable to multiple previous works that rely on parametric basis functions [incl. 11, 17, 86–88]. Further, both of the theorems reduce the memory scaling from $\mathcal{O}(M^2)$ to $\mathcal{O}(M)$. Note that these reductions are *without* approximations, only relying on the structural properties of the considered models.

In the following, we assume that the kernel is a tensor product kernel [16], i.e., $\kappa(x, x') = \prod_{d=1}^{D} \kappa^{(d)}(x^{(d)}, x^{(d)\prime})$, where $\kappa^{(d)}(\cdot,\cdot)$ is the kernel along the dth dimension. Then, if each component of the kernel is approximated or can be represented exactly with m_d BFs such that

$$\kappa^{(d)}(x^{(d)}, x^{(d)\prime}) \approx \boldsymbol{\phi}^{(d)}(x^{(d)})^\top \boldsymbol{\Lambda}^{(d)} \, \boldsymbol{\phi}^{(d)}(x^{(d)\prime}), \tag{4.4}$$

where $\boldsymbol{\phi}^{(d)}(\cdot) : \mathbb{R} \to \mathbb{R}^{m_d}$ are the BFs $[\phi_1^{(d)}, \phi_2^{(d)}, \ldots, \phi_{m_d}^{(d)}]^\top$ along the dth dimension and $\boldsymbol{\Lambda}^{(d)} \in \mathbb{R}^{m_d \times m_d}$ contains the associated weights. The full kernel can then be approximated as

$$\kappa(\boldsymbol{x}, \boldsymbol{x}') \approx \prod_{d=1}^{D} \boldsymbol{\phi}^{(d)}(x^{(d)})^\top \boldsymbol{\Lambda}^{(d)} \, \boldsymbol{\phi}^{(d)}(x^{(d)\prime}), \tag{4.5}$$

where in this case $\boldsymbol{\phi}(\cdot) : \mathbb{R}^D \to \mathbb{R}^M$ and $\boldsymbol{\Lambda} \in \mathbb{R}^{M \times M}$ are

$$\boldsymbol{\phi}(x) = \otimes_{d=1}^{D} \boldsymbol{\phi}^{(d)}(x^{(d)}), \tag{4.6a}$$

$$\boldsymbol{\Lambda} = \otimes_{d=1}^{D} \boldsymbol{\Lambda}^{(d)}. \tag{4.6b}$$

Here, $M := \prod_{d=1}^{D} m_d$ is the total number of BFs. Given this decomposition, the precision matrix can be expressed as

$$\boldsymbol{\Phi}^\top \boldsymbol{\Phi} = \sum_{n=1}^{N} \boldsymbol{\phi}(x_n)\boldsymbol{\phi}(x_n)^\top = \sum_{n=1}^{N} \otimes_{d=1}^{D} \boldsymbol{\phi}^{(d)}(x_n^{(d)}) \left[\boldsymbol{\phi}^{(d)}(x_n^{(d)})\right]^\top. \tag{4.7}$$

This decomposition of the precision matrix is key in the following and we will primarily study the individual products $\boldsymbol{\phi}^{(d)}(x_n^{(d)})[\boldsymbol{\phi}^{(d)}(x_n^{(d)})]^\top$ where certain structure may appear that is exploitable to our benefit. To provide some intuition, we consider the precision matrix for polynomial BFs in 1D, 2D, and 3D (see figure 4.2). The 1D case (left) has a *Hankel* structure and the 2D (middle) and 3D (right) cases have 2-level and 3-level *block Hankel* structure, respectively. It is these types of structures that allow a reduction in complexity *without* approximations. Next, we provide clear technical definitions of the matrix structures and then proceed to state our main findings. All of the discussed matrix structures are also visually explained in table 4.1 in section 4.F.

4.3.1 Hankel and Toeplitz Matrices

An $m \times m$ matrix $\boldsymbol{H}$ has Hankel structure iff it can be expressed using a vector $\boldsymbol{\gamma} \in \mathbb{R}^{(2m-1)}$ containing all the unique entries, according to

$$\boldsymbol{H} = \begin{bmatrix} \gamma_1 & \gamma_2 & \cdots & \gamma_m \\ \gamma_2 & \gamma_3 & \cdots & \gamma_{m+1} \\ \vdots & \vdots & \ddots & \vdots \\ \gamma_m & \gamma_{m+1} & \cdots & \gamma_{2m-1} \end{bmatrix}. \tag{4.8}$$

In other words, each element of the Hankel matrix on the ith row and the jth column is given by γ_{i+j-1}. Similarly, an $m \times m$ matrix has Toeplitz structure iff it can be expressed using a vector $\boldsymbol{\gamma} \in \mathbb{R}^{(2m-1)}$ such that each element on the ith row and the jth column is given by γ_{i-j+m}. See figure 4.3 for an example of what a Hankel and a Toeplitz matrix visually looks like.

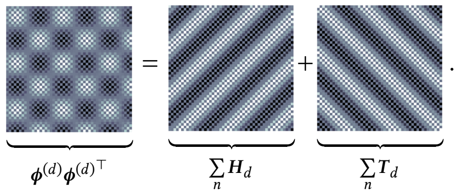


Figure 4.3: The precision matrix for sinusoidal basis functions in one dimension has neither Hankel nor Toeplitz structure. However, it can be decomposed into a sum of two matrices, where one has a Hankel structure, and one has Toeplitz structure. Here, 49 BFs are placed along one dimension.

We define a matrix $\boldsymbol{H}^{(D)}$ as a D-level block Hankel matrix if it can be expressed as

$$\boldsymbol{H}^{(D)} = \begin{bmatrix} \boldsymbol{H}_1^{(D-1)} & \boldsymbol{H}_2^{(D-1)} & \dots & \boldsymbol{H}_{m_D}^{(D-1)} \\ \boldsymbol{H}_2^{(D-1)} & \boldsymbol{H}_3^{(D-1)} & \dots & \boldsymbol{H}_{m_D+1}^{(D-1)} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{H}_{m_D}^{(D-1)} & \boldsymbol{H}_{m_D+1}^{(D-1)} & \dots & \boldsymbol{H}_{2m_D-1}^{(D-1)} \end{bmatrix}, \tag{4.9}$$

where $\boldsymbol{H}_k^{D-1}$ is a block Hankel matrix, and $\boldsymbol{H}_k^1$ is a simple Hankel matrix. Each submatrix $\boldsymbol{H}_k^{(d)}$ for $d \in \{1, \dots, D\}$ can be indexed by row and column $i_d, j_d \in \{1, \dots, m_d\}$ which again yields a submatrix $\boldsymbol{H}_{i_d+j_d-1}^{(d-1)}$ which can be indexed in the same manner. Therefore, an equivalent definition of a block Hankel matrix is that each individual entry of $\boldsymbol{H}^{(D)}$, given by a set of indices $i_1, j_1, \dots, i_D, j_D$, is given by the entry $\boldsymbol{\gamma}_{k_1,\dots,k_D}$ of a tensor $\boldsymbol{\gamma} \in \mathbb{R}^{K_1 \times K_2 \times \dots \times K_D}$, where $k_d = i_d + j_d - 1$, and $K_d = 2m_d - 1$. Similarly, $\boldsymbol{T}^{(D-1)}$ is a block Toeplitz matrix if each entry $i_1, j_1, \dots, i_D, j_D$ is given by the entry $\boldsymbol{\gamma}_{k_1,\dots,k_D}$ of a tensor $\boldsymbol{\gamma} \in \mathbb{R}^{K_1 \times K_2 \times \dots \times K_D}$, where $k_d = m_d + i_d - j_d$, and $K_d = 2m_d - 1$.

4.3.2 Block Hankel–Toeplitz Matrices

In addition to block Hankel and block Toeplitz structures, we require a slightly more general but highly related structure. We call this structure block Hankel–Toeplitz structure and define it as follows. A matrix $\boldsymbol{G}^{(D)}$ has a D-level block Hankel–Toeplitz structure, if the matrix is defined either as

$$\boldsymbol{G}^{(D)} = \begin{bmatrix} \boldsymbol{G}_1^{(D-1)} & \boldsymbol{G}_2^{(D-1)} & \dots & \boldsymbol{G}_{m_D}^{(D-1)} \\ \boldsymbol{G}_2^{(D-1)} & \boldsymbol{G}_3^{(D-1)} & \dots & \boldsymbol{G}_{m_D+1}^{(D-1)} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{G}_{m_D}^{(D-1)} & \boldsymbol{G}_{m_D+1}^{(D-1)} & \dots & \boldsymbol{G}_{2m_D-1}^{(D-1)} \end{bmatrix}, \tag{4.10}$$

if level D is Hankel, or as

$$G^{(D)} = \begin{bmatrix} G^{(D-1)}_{m_D} & \dots & G^{(D-1)}_{2} & G^{(D-1)}_{1} \\ G^{(D-1)}_{m_D+1} & \dots & G^{(D-1)}_{3} & G^{(D-1)}_{2} \\ \vdots & \iddots & \vdots & \vdots \\ G^{(D-1)}_{2m_D-1} & \dots & G^{(D-1)}_{m_D+1} & G^{(D-1)}_{m_D} \end{bmatrix}, \tag{4.11}$$

if level D is Toeplitz. Further, $G^{(D-1)}_j$ are $D-1$ level block Hankel–Toeplitz matrices if $D-1>2$, and a simple Hankel or Toeplitz matrix if $D-1=1$. Each submatrix $G^{(d)}_k$ for $d \in \{1,\dots,D\}$ can be indexed by row and column $i_d, j_d \in \{1,\dots,m_d\}$ which again yields a submatrix defined either as $G^{(d-1)}_{i_d+j_d-1}$ or $G^{(d-1)}_{m_d+i_d-j_d}$ (depending on whether the d^{th}-level has Hankel structure as in equation (4.10) or Toeplitz structure as in equation (4.11)). Each entry in a block Hankel–Toeplitz matrix can also be expressed by the entry $\boldsymbol{\gamma}_{k_1,k_2,\dots,k_D}$ of a tensor $\boldsymbol{\gamma} \in \mathbb{R}^{K_1\times K_2\times\dots\times K_D}$, where $K_d = 2m_d - 1$, and

$$k_d = \begin{cases} i_d + j_d - 1, & \text{if level } d \text{ is Hankel} \\ m_d + i_d - j_d, & \text{if level } d \text{ is Toeplitz.} \end{cases} \tag{4.12}$$

A crucial property of the block Hankel–Toeplitz structure is the preservation of structure under addition. Assume that $\boldsymbol{A}$ and $\boldsymbol{B}$ are two block Hankel–Toeplitz matrices and that they are structurally identical, in the sense that they have the same number of levels, the same number of entries in each block, and each level shares either Toeplitz or Hankel properties. Then, let each entry of $\boldsymbol{A}$ and $\boldsymbol{B}$ be given by $\alpha_{k_1,k_2,\dots,k_D}$ for a tensor $\alpha \in \mathbb{R}^{m_1,\dots,m_D}$ and $\beta_{k_1,k_2,\dots,k_D}$ for $\beta \in \mathbb{R}^{m_1,\dots,m_D}$, respectively, with k_d defined in equation (4.15). Each entry in $\boldsymbol{A}+\boldsymbol{B}$ is then given by the sum of the entries $\alpha_{k_1,\dots,k_D} + \beta_{k_1,\dots,k_D}$. Thus, the sum of two block Hankel–Toeplitz matrices with identical structure is also a block Hankel–Toeplitz matrix. By associativity of matrix addition, a sum $\sum_{n=1}^{N} G_n$ of N Hankel–Toeplitz matrices $\{G_1,\dots,G_N\}$ with identical structure is therefore itself a Hankel–Toeplitz matrix.

4.3.3 Kronecker Products of Hankel–Toeplitz Matrices and Block Hankel–Toeplitz Matrices

A special case of a class of matrices G which has block Hankel–Toeplitz structure are Kronecker products of D Hankel or Toeplitz matrices $\{G^{(1)},\dots,G^{(D)}\}$, i.e.,

$$G = \bigotimes_{d=1}^{D} G^{(d)} := G^{(1)} \otimes G^{(2)} \otimes \dots \otimes G^{(D)}. \tag{4.13}$$

An equivalent definition of the Kronecker product gives each entry on the i^{th} row and j^{th} column in the block Hankel–Toeplitz matrix G as an expression of the entries on the ${i_d}^{\text{th}}$ row and ${j_d}^{\text{th}}$ column of each matrix $G^{(d)}$ according to

$$G_{i,j} = \textstyle\prod_{d=1}^{D} G^{(d)}_{i_d,j_d}. \tag{4.14}$$

Note that for both Hankel and Toeplitz matrices there is a one-to-one map between each index i,j and the index sets $\{i_1,\dots,i_D\}$ and $\{j_1,\dots,j_D\}$. As each matrix $G^{(d)}$ has Hankel or

Toeplitz structure, the entries can equivalently be defined by a vector $\boldsymbol{\gamma}^{(d)}$ with $2m_d - 1$ entries. Each entry $\boldsymbol{G}_{i,j}$ is therefore given by

$$\boldsymbol{G}_{i,j} = \prod_{d=1}^{D} \boldsymbol{\gamma}_{k_d}^{(d)} = \boldsymbol{\gamma}_{k_1,\ldots,k_D}, \tag{4.15}$$

with k_d defined in equation (4.12), and where $\boldsymbol{\gamma} := \bigotimes_{d=1}^{D} \boldsymbol{\gamma}^{(d)}$ is a rank-1 tensor with $\prod_{d=1}^{D}(2m_d - 1)$ elements.

4.3.4 Main Results

We are now ready to state our main findings. The following two theorems rely on the product decomposition equation (4.5) and study each dimension d separately, as is evidently possible from equation (4.7). Our first theorem generalizes a result by [89] regarding complex exponential basis functions. Our first theorem establishes that if the product $\boldsymbol{\phi}^{(d)}(x_n^{(d)})[\boldsymbol{\phi}^{(d)}(x_n^{(d)})]^\top$ has Hankel or Toeplitz structure, the resulting precision matrix only has $\prod_{d=1}^{D}(2m_d - 1)$ unique entries, reducing the computational complexity of instantiating it from $\mathcal{O}(NM^2)$ to $\mathcal{O}(NM)$. We formalize this in the following theorem.

4

Theorem 4.3.1. *If the matrix*

$$\boldsymbol{G}^{(d)}(x_n^{(d)}) := \boldsymbol{\phi}^{(d)}(x^{(d)})\left[\boldsymbol{\phi}^{(d)}(x^{(d)})\right]^\top, \tag{4.16}$$

is a Hankel or Toeplitz matrix for all $x^{(d)} \in \mathbb{R}$ along each dimension d, the information matrix $\boldsymbol{\Phi}^\top\boldsymbol{\Phi}$ will be a multi-level block Hankel or Toeplitz matrix, and therefore have $\prod_{d=1}^{D}(2m_d - 1)$ unique entries.

Proof. Assume that the matrix $\boldsymbol{G}^{(d)}(x_n^{(d)}) := \boldsymbol{\phi}^{(d)}(x_n^{(d)})[\boldsymbol{\phi}^{(d)}(x_n^{(d)})]^\top$ is Hankel or Toeplitz. The precision matrix can then be expressed as

$$\boldsymbol{\Phi}^\top\boldsymbol{\Phi} = \sum_{n=1}^{N} \otimes_{d=1}^{D} \boldsymbol{\phi}^{(d)}(x_n^{(d)})\left[\boldsymbol{\phi}^{(d)}(x_n^{(d)})\right]^\top = \sum_{n=1}^{N} \otimes_{d=1}^{D} \boldsymbol{G}^{(d)}(x_n^{(d)}), \tag{4.17}$$

where the matrix $\otimes_{d=1}^{D} \boldsymbol{G}^{(d)}(x_n^{(d)})$ is multi-level Hankel or Toeplitz by definition, see section 4.3.3. Further, the sum of several D-level block Hankel–Toeplitz matrices is itself a D-level block Hankel–Toeplitz matrix, see section 4.3.2. Since each matrix $\boldsymbol{G}^{(d)}(x_n^{(d)})$ has at most $(2m_d - 1)$ unique entries, the matrix $\boldsymbol{\Phi}^\top\boldsymbol{\Phi} = \sum_{n=1}^{N} \otimes_{d=1}^{D} \boldsymbol{G}^{(d)}(x_n^{(d)})$ therefore has at most $M = \prod_{d=1}^{D}(2m_d - 1)$ unique entries. □

The preceding theorem holds true for, for instance, polynomial and complex exponential BFs, which we establish in theorems 4.3.2 and 4.3.3.

Corollary 4.3.2. *The precision matrix for polynomial BFs defined by*

$$\phi_{i_d}^{(d)}(x^{(d)}) = (x^{(d)})^{i_d - 1}, \tag{4.18}$$

can be represented by a tensor with $\prod_{d=1}^{D} 2m_d - 1$ entries.

Proof. See section 4.A for a proof. □

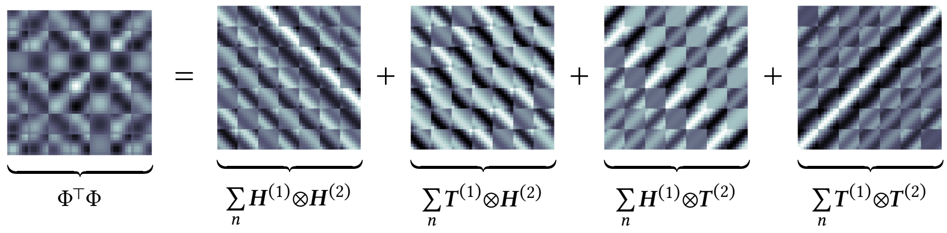

=
+
+
+
$\Phi^\top\Phi$
$\sum_n H^{(1)} \otimes H^{(2)}$
$\sum_n T^{(1)} \otimes H^{(2)}$
$\sum_n H^{(1)} \otimes T^{(2)}$
$\sum_n T^{(1)} \otimes T^{(2)}$

Figure 4.4: The precision matrix for sinusoidal BFs in two dimensions has neither Hankel nor Toeplitz structure. However, it can be decomposed into $2^D = 4$ matrices, which each have block Hankel–Toeplitz structure. Here, 7 BFs are placed along each of the two dimensions, giving a total of 49 BFs.

Corollary 4.3.3. *The precision matrix for complex exponential BFs defined by*

$$\phi_j^C(x) = \exp(i\pi j^\top x) = \prod_{d=1}^D \exp(i\pi j_d x_d), \tag{4.19}$$

can be represented by a tensor with $\prod_{d=1}^D 2m_d - 1$ entries.

Proof. See section 4.B for a proof. □

For some BFs, the structure of the product $\boldsymbol{\phi}^{(d)}(x_n^{(d)})[\boldsymbol{\phi}^{(d)}(x_n^{(d)})]^\top$ is more intricate, but is still of a favorable, exploitable nature. This is clearly evident from figure 4.4, where the precision matrix for sinusoidal BFs in two dimensions is visualized. In particular, for some BFs, the product is the sum of a Hankel and a Toeplitz matrix, such that the precision matrix only has $\prod_{d=1}^D 3m_d$ unique entries, again reducing the computational cost of computing it to $\mathcal{O}(NM)$. We formalize this in the following theorem.

Theorem 4.3.4. *If the product $\boldsymbol{\phi}^{(d)}(x_n^{(d)})[\boldsymbol{\phi}^{(d)}(x_n^{(d)})]^\top$ is the sum of a Hankel matrix denoted $\boldsymbol{G}^{(d),(1)}$ and a Toeplitz matrix $\boldsymbol{G}^{(d),(-1)}$, and there exists a function $g^{(d)}(k_d)$ such that $\boldsymbol{G}^{(d),(1)}_{i_d,j_d} = g(i_d + j_d)$ and $\boldsymbol{G}^{(d),(-1)}_{i_d,j_d} = -g(i_d - j_d)$, all entries in the precision matrix can be represented by a tensor $\boldsymbol{\gamma}(k_1, k_2, \ldots, k_D)$ with $\prod_{d=1}^D 3m_d$ entries.*

Proof. The precision matrix can in this case be expressed as

$$\begin{aligned}
\overbrace{\boldsymbol{\Phi}^\top\boldsymbol{\Phi}}^{:=C} &= \textstyle\sum_{n=1}^N \otimes_{d=1}^D \boldsymbol{\phi}^{(d)}(x_n^{(d)}) \left[\boldsymbol{\phi}^{(d)}(x_n^{(d)})\right]^\top \\
&= \textstyle\sum_{n=1}^N \otimes_{d=1}^D \left(\boldsymbol{G}^{(d),(1)} + \boldsymbol{G}^{(d),(-1)}\right) \\
&= \textstyle\sum_{p=1}^{2^D} \left(\prod_{d=1}^D e_p^{(d)}\right) \sum_{n=1}^N \otimes_{d=1}^D \boldsymbol{G}^{(d),(e_p^{(d)})},
\end{aligned} \tag{4.20}$$

where $e_p = \{e_p^{(1)}, \ldots, e_p^{(D)}\} \in S^D$ and $S^D = \{1, -1\}^D$ is a set containing 2^D elements. Each of the 2^D matrices $\left(\prod_{d=1}^D e_p^{(d)}\right) \sum_{n=1}^N \otimes_{d=1}^D \boldsymbol{G}^{(d),(e_p^{(d)})}$ is now the Kronecker product between D Hankel or Toeplitz matrices. The entries of $\boldsymbol{C}$ can be expressed element-wise as

$$C_{i,j} = \textstyle\sum_{p=1}^{2^D} \left(\prod_{d=1}^D e_p^{(d)}\right) \sum_{n=1}^N \prod_{d=1}^D g(i_d + e_p^{(d)} j_d). \tag{4.21}$$

Algorithm 4 Sketch of an algorithm for Hilbert GP learning and inference. The original approach by [11] in red, our proposed approach in blue.

```
Input: Data as input–output pairs $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$,
       test inputs $\boldsymbol{x}_\star$, number of basis functions $M$
Compute $\boldsymbol{\Phi}^\top\boldsymbol{\Phi}$ at cost $\mathcal{O}(NM^2)$          ▷ equation (4.7)
Compute $\boldsymbol{\gamma}$ at cost $\mathcal{O}(NM)$          ▷ equation (4.22)
Construct $\boldsymbol{\Phi}^\top\boldsymbol{\Phi}$ using $\boldsymbol{\gamma}$ at cost $\mathcal{O}(M^2)$
repeat
    Optimize maximum likelihood w.r.t. hyperparameters at cost $\mathcal{O}(M^3)$
until Convergence
Perform GP inference using the pre-calculated matrices
```

If we define a tensor $\boldsymbol{\gamma}$ as

$$\gamma_{k_1,\dots,k_D} = \sum_{n=1}^N \prod_{d=1}^D g(k_d). \tag{4.22}$$

for indices $k_d = 1-m_d, 2-m_d, \dots, 2m_d-1, 2m_d$, each entry of the precision matrix $C_{i,j}$ can be expressed as

$$C_{i,j} = \sum_{p=1}^{2^D} \left(\prod_{d=1}^D e_p^{(d)}\right) \gamma_{i_1+e_p^{(1)} j_1,\dots,i_D+e_p^{(D)} j_D}. \tag{4.23}$$

As each sum $k_d = i_d + e_p^{(d)} j_d$ is an integer between $1-m_d$ and $2m_d$, the tensor $\boldsymbol{\gamma}$ will have $\prod_{d=1}^D 3m_d$ entries. □

The preceding theorem applies to, for instance, the BFs in an HGP [11] defined on a rectangular domain, formalized in theorem 4.3.5. It further holds for multiple other works using similar BFs, formalized in theorem 4.3.6. We remark that for $D = 1$, we require $m_d > 3$ for any savings to take effect, whereas for $D > 1$, $m_d \geq 2$ suffices.

Corollary 4.3.5. *The precision matrix in an HGP defined on a rectangular domain* $[-L_1, L_1] \times \dots \times [-L_D, L_D]$ *can be represented by a tensor with* $\prod_{d=1}^D 3m_d$ *entries.*

Proof. See section 4.C for a proof. □

Corollary 4.3.6. *The precision matrix in a GP approximated by sinusoidal and cosine BFs with frequencies on a grid (such as the regular Fourier features described in [17] and [90], the Fourier approximations to periodic kernels described in [87], the quadrature Fourier features described in [91], the equispaced-version of sparse spectrum BFs described in [86], or the one-dimensional special case of [88]) can be represented by a tensor with* $\prod_{d=1}^D 3m_d$ *entries.*

Proof. See section 4.D for a proof. □

4.3.5 Outlook and Practical Use

Both theorems 4.3.1 and 4.3.4 reduce the computational complexity of calculating the entries of the precision matrix equation (4.7) from $\mathcal{O}(NM^2)$ to $\mathcal{O}(NM)$. This enables us to scale the number of BFs significantly more than previously, before running into computational, or storage related, problems. For clarity, the standard HGP is given in algorithm 4 with the original approach in red and our proposed approach in blue. As compared to the standard (offline) HGP, the only change we make is the computation of the precision matrix.

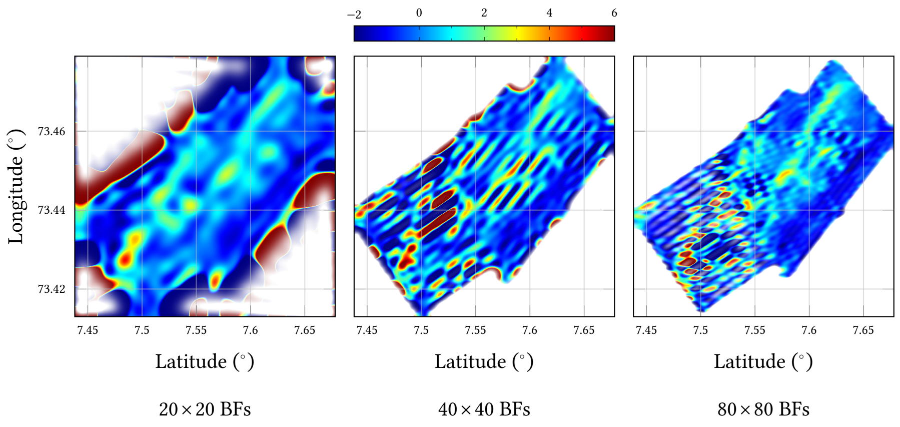
While we focus primarily on the dominating $\mathcal{O}(NM^2)$ cost of obtaining the precision matrix, it is worthwhile to mention the cost of hyperparameter optimization, which is in $\mathcal{O}(M^3)$ as we choose to optimize the maximum likelihood as in [11]. This cost could potentially be reduced through the use of efficient approximate matrix inverses and trace estimators [see, e.g., 92]. Another consequence of these theorems is that multi-agent systems that collaborate to learn the precision matrix such as [76] or [75] can do this by communicating $\mathcal{O}(M)$ bits instead of $\mathcal{O}(M^2)$ bits.

4.4 Experiments

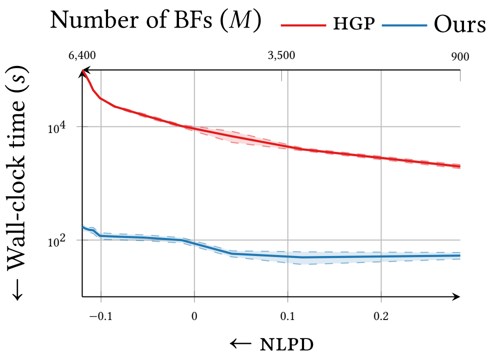
We demonstrate the storage and computational savings of our structure exploiting scheme by means of three numerical experiments. The experiments demonstrate the practical efficiency of our scheme using the HGP. We reiterate that the savings are *without* additional approximations and the posterior is therefore *exactly* equal to that of the standard HGP. Further, as HGPs adhere to theorem 4.3.4, this demonstration is a representative example of the speedups that can be expected using, e.g., regular Fourier features [17], or BF expansions of periodic kernels [88]. Since theorem 4.3.1 requires computing and storing only $2M$ components whereas theorem 4.3.4 requires $3M$, our experiments demonstrate a practical upper bound on the storage and computational savings, a “worst case”.

Our first experiment demonstrates the computational scaling of our scheme on a simulated 3D dataset. Secondly, we consider a magnetic field mapping example with data collected by an underwater vessel, as an application where the high-frequency content of the data requires a large amount of BFs to reconstruct the field. Thirdly, a precipitation dataset is used, mirroring an example in [11], improving the computational scaling in that particular application even further than the standard HGP. Our freely available HGP reference implementation along with hyperparameter optimization is written for GPJax [93]. All timing experiments are run on an HP Elitebook 840 G5 laptop (Intel i7-8550U CPU, 16GB RAM). For fair comparison, we naively loop over data points, to avoid any possible low-level optimization skewing the results.

Computational Scaling We compare the wall time necessary for computing the precision matrix for $N = 500$ data points for the standard HGP as well as for our structure exploiting scheme, which only requires the unique entries. The results are presented in figure 4.1 for an increasing number of BFs. After $M = 14000$, the HGP is no longer feasible due to memory constraints, whereas our formulation scales well above that, but stop at $M = 64000$ for clarity in the illustration. Clearly, the structure exploitation gives a significantly lower computational cost than the standard HGP, even for small quantities of BFs. Further, it drastically reduces the memory requirements, where for $M = 14000$, the HGP requires roughly 1.5 GB for storing the precision matrix, while our formulation requires roughly 2.8 MB using 64-bit floats. This makes it possible for us to scale the number of BFs significantly more before running into computational or storage restrictions. It is noteworthy that even though the implementation is in a high-level language, we still see significant computational savings.

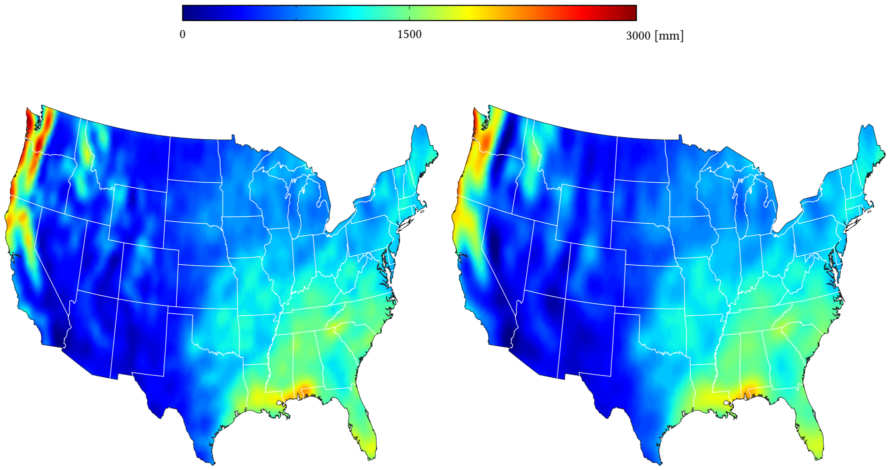


(a) Predictive means indicated by color intensity with transparency proportional to the predictive variance. Increasing number of BFs from left to right.



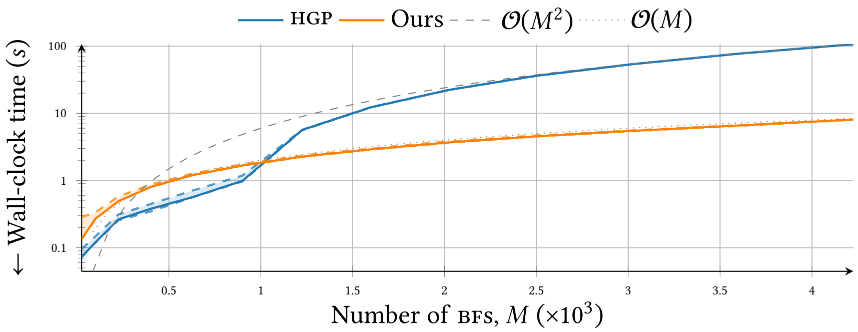
(b) Our proposed computational scheme reduces the computation time for datasets with high-frequency variations, as these require many BFs to achieve accurate reconstruction. This underwater magnetic field has lower NLPD with a large amount (6400) compared to a smaller amount (400) of BFs. For 6400 BFs, our computational scheme reduced the required time to compute the precision matrix from 2.7 hours to 1.7 minutes.

Figure 4.5: Our proposed computational scheme reduces the computation time for datasets with high-frequency variations, as these require many BFs to achieve accurate reconstruction. This underwater magnetic field has lower NLPD with a large amount (6400) compared to a smaller amount (400) of BFs. For 6400 BFs, our computational scheme reduced the required time to compute the precision matrix from 2.7 hours to 1.7 minutes.



(a) Full GP

(b) HGP / Our



(c) Time to compute the precision matrix for increasing M

Figure 4.6: These experiments recover the results from [11] exactly for predicting yearly precipitation levels across the US, and measure the wall-clock time needed by our proposed computational scheme. The HGP efficiently approximates the full GP solution using $m_1 = m_2 = 45$, totaling $M = 2025$ BFs.

Magnetic Field Mapping As HGPs are commonly used to model and estimate the magnetic field for, e.g., mapping purposes [8, 94], we consider a magnetic field mapping example and demonstrate the ability of our computational scheme to scale HGPs to spatially vast datasets. The data was gathered sequentially in a lawn-mower path in a region approximately 7×7 kilometers large by an underwater vessel outside the coast of Norway ($d = 2$ and $N = 1.39$ million). The data was split into a training set and test set with roughly a 50/50 split, deterministically split by a grid pattern, to ensure reasonable predictions in the entire data domain, see section 4.E for more details. We vary the amount of BFs and compare the time required to sequentially include each new data point in the precision matrix as well as the NLPD. As the underwater magnetic field covers a large area, a large number of BFs are required to accurately represent the field, see figure 4.5a where the details of the predicted magnetic field is captured more accurately for an increasing number of BFs. This is also apparent from the decreasing negative log predictive density (NLPD) as the number of BFs increases, see figure 4.5b. At 6400 BFs, the necessary computation time is several orders of magnitude lower for our approach compared to the standard HGP.

U.S. Precipitation Data We consider a standard precipitation data set containing US annual precipitation summaries for year 1995 ($d = 2$ and $N = 5776$) [95]. We exactly mimic the setup in [11] and primarily focus our evaluation on the calculation of the precision matrix. The time for computing the precision matrix is visualized in figure 4.6c, where our approach clearly outperforms the standard HGP. The predictions on a dense grid over the continental US can be found in figures 4.6a and 4.6b, where the HGP manages to capture both the large-scale as well as the small-scale variations well.

4.5 Conclusion

Our contribution details a computational approach for exploiting Hankel and Toeplitz structures that appear in multiple BF approximation schemes to kernels for GPs. These structures allow us to reduce the computational complexity of computing the corresponding precision matrix from $\mathcal{O}(NM^2)$ to $\mathcal{O}(NM)$ without further approximations. Further, our approach reduces the storage requirement for containing all necessary information about the posterior to make predictions from $\mathcal{O}(M^2)$ to $\mathcal{O}(M)$.

The reduced computational and storage requirements are particularly beneficial in the HGP where more BFs allow us to capture higher frequencies of the kernel spectrum, otherwise unattainable without significant computational resources. We foresee that our contribution will allow HGPs to tackle larger problems without the need for extensive specialized hardware, opening up approximate GP learning and inference for a wider audience.

The majoriy of the appendix covers proofs of the corollaries following from theorems 4.3.1 and 4.3.4. The rest is dedicated to further details on our empirical experiments as well as visual explanations of the structures that are exploited in the main body of the paper. The appendix is organized as follows.

section 4.A proves theorem 4.3.1 for polynomial BF. section 4.B proves theorem 4.3.1 for complex exponential BF. section 4.C proves theorem 4.3.4 for Hilbert space BF defined on a rectangular domain. section 4.D proves theorem 4.3.4 for ordinary Fourier features.

section 4.F contains visual representations of the structures explained in the main body of the paper. Lastly, section 4.E provides a full description of all the included experiments and the data used in them, with additional plots and results.

4.A Proof of use of theorem 4.3.1 for Polynomial Basis Functions

Polynomial BFs (as used in for example [96]) are defined along each dimension as

$$\phi_{i_d}^{(d)}(x^{(d)}) = (x^{(d)})^{i_d-1} \tag{4.24}$$

By selecting $g_{i_d+j_d}(x^{(d)}) = (x^{(d)})^{j_d+i_d-1}$, the product of two component BF becomes

$$\phi_{i_d}^{(d)}(x^{(d)})\phi_{j_d}^{(d)}(x^{(d)}) = (x^{(d)})^{i_d-1}(x^{(d)})^{j_d-1} = (x^{(d)})^{j_d+i_d-1} = g_{i_d+j_d}, \tag{4.25}$$

which shows that the conditions for theorem 4.3.1 is satisfied.

4.B Proof of use of theorem 4.3.1 for Complex Exponential Basis Functions

Complex exponential BF (as used in for example [97]) are defined along each dimension as

$$\phi_{i_d}^{(d)}(x^{(d)}) = \exp(i\pi j_d x^{(d)}) \tag{4.26}$$

By selecting $g_{i_d+j_d}(x^{(d)}) = \exp(i\pi j_d + i_d x^{(d)})$, the product of two component BF becomes

$$\phi_{i_d}^{(d)}(x^{(d)})\phi_{j_d}^{(d)}(x^{(d)}) = \exp(i\pi i_d x^{(d)})\exp(i\pi j_d x^{(d)}) = \exp(i\pi j_d + i_d x^{(d)}) = g_{i_d+j_d}, \tag{4.27}$$

which shows that the conditions for theorem 4.3.1 is satisfied.

4.C Proof of use of theorem 4.3.4 for Hilbert Space Basis Functions on a Rectangular Domain

Hilbert space BF on a rectangular domain are defined according to [11]

$$\phi_i(\boldsymbol{x}) = \prod_{d=1}^{D} \frac{1}{\sqrt{L_d}} \sin\left(\frac{\pi i_d(x^{(d)} + L_d)}{2L_d}\right) = \prod_{d=1}^{D} \frac{1}{\sqrt{L_d}} \cos\left(\frac{\pi i_d(x^{(d)} + L_d)}{2L_d} - \frac{\pi}{2}\right), \tag{4.28}$$

where the indices $i = 1, \ldots, M^D$ has a one-to-one mapping with all possible combinations of the indices $i_1, i_2, \ldots, i_D$, given that each index $i_d = \{1, \ldots, D\}$.

This corresponds to defining the BF according to equation (4.6a), with each entry of $\boldsymbol{\phi}^{(d)}(\boldsymbol{x})$ defined as

$$\phi_{i_d}^{(d)}(x^{(d)}) = \frac{1}{\sqrt{L_d}} \sin\left(\frac{\pi i_d(x^{(d)} + L_d)}{2L}\right) = \frac{1}{\sqrt{L_d}} \cos\left(\frac{\pi i_d(x^{(d)} + L_d)}{2L_d} - \frac{\pi}{2}\right),$$

where $i_d \in \{1,\dots,m\}$. Define a linear function $\theta_{i_d} : \mathbb{R} \to \mathbb{R}$ as

$$\theta_{i_d}(x^{(d)}) = \frac{\pi i_d(x^{(d)} + L_d)}{2L_d} - \frac{\pi}{2}.$$

Hence, the BF can be written as

$$\boldsymbol{\phi}^{(d)}(x^{(d)}) = \frac{1}{\sqrt{L_d}} \begin{bmatrix} \cos(\theta_1) \\ \vdots \\ \cos(\theta_M) \end{bmatrix}.$$

The product $\boldsymbol{\phi}^{(d)}(x_n^{(d)})\boldsymbol{\phi}^{(d)}(x_n^{(d)})^\top$ is then given by

$$\boldsymbol{\phi}^{(d)}(x_n^{(d)})\boldsymbol{\phi}^{(d)}(x_n^{(d)})^\top = \frac{1}{L_d}\sum_{n=1}^{N} \begin{bmatrix} \cos(\theta_1)\cos(\theta_1) & \dots & \cos(\theta_1)\cos(\theta_M) \\ \vdots & \ddots & \vdots \\ \cos(\theta_M)\cos(\theta_M) & \dots & \cos(\theta_M)\cos(\theta_M) \end{bmatrix}.$$

Then, note that

$$\cos(u)\cos(v) = \frac{\cos(u+v) + \cos(u-v)}{2}. \tag{4.29}$$

When we apply this to each entry in the matrix, we get that

$$\boldsymbol{\phi}^{(d)}(x_n^{(d)})\boldsymbol{\phi}^{(d)}(x_n^{(d)})^\top = \\ \frac{1}{2L_d}\left(\underbrace{\sum_{n=1}^{N} \begin{bmatrix} \cos(\theta_1+\theta_1) & \dots & \cos(\theta_1+\theta_M) \\ \vdots & \ddots & \vdots \\ \cos(\theta_M+\theta_1) & \dots & \cos(\theta_M+\theta_M) \end{bmatrix}}_{\triangleq G^{d,(1)}} + \underbrace{\sum_{n=1}^{N} \begin{bmatrix} \cos(\theta_1-\theta_1) & \dots & \cos(\theta_1-\theta_M) \\ \vdots & \ddots & \vdots \\ \cos(\theta_M-\theta_1) & \dots & \cos(\theta_M-\theta_M) \end{bmatrix}}_{\triangleq G^{d,(-1)}} \right),$$

where $G^{d,(1)}$ and $G^{d,(-1)}$ are a Hankel and a Toeplitz matrix, respectively. Then, since $\cos(\theta_{i_d} \pm \theta_{j_d}) = \pm\sin(\theta_{i_d \pm j_d}) \triangleq g(i_d \pm j_d) = g(k_d)$, there exists a function $g(k_d)$ that fulfills conditions for theorem 4.3.4.

4.D Proof of use of theorem 4.3.4 for Ordinary Fourier Features

Ordinary Fourier features for separable kernels are defined slightly differently from Hilbert Space BF [17], in that they consider both a sine and a cosine function for each considered frequency. The set of BF are given by

$$\begin{aligned} \boldsymbol{\phi}^{(d)}(x) &= \begin{bmatrix} \boldsymbol{\phi}^{(d)}_{\sin}(x) & \boldsymbol{\phi}^{(d)}_{\cos}(x) \end{bmatrix} \\ &= \begin{bmatrix} \sin(\Delta x) & \sin(2\Delta x) & \dots & \sin(m_d\Delta) & \cos(\Delta x) & \cos(2\Delta x) & \dots & \cos(m_d\Delta) \end{bmatrix}, \end{aligned} \tag{4.30}$$

where Δ determines the spacing of the Fourier features in the frequency domain. The precision matrix can therefore be expressed as

$$\Phi^\top \Phi = \begin{bmatrix} \Phi_{\sin}^\top \Phi_{\sin} & (\Phi_{\cos}^\top \Phi_{\sin})^\top \\ \Phi_{\cos}^\top \Phi_{\sin} & \Phi_{\cos}^\top \Phi_{\cos} \end{bmatrix}, \tag{4.31}$$

and we can apply theorem 4.3.4 directly to entries $\Phi_{\sin}^\top \Phi_{\sin}$ and $\Phi_{\sin}^\top \Phi_{\cos}$ to prove that each of these have a block Hankel–Toeplitz structure.

For the matrix $\Phi_{\sin}^\top \Phi_{\sin}$, the product $\boldsymbol{\phi}_{\sin}(x_n^{(d)})\boldsymbol{\phi}_{\sin}(x_n^{(d)})^\top$ can be expanded as

$$\{\boldsymbol{\phi}_{\sin}(x_n^{(d)})\boldsymbol{\phi}_{\sin}(x_n^{(d)})^\top\}_{i,j} = \sin(i\Delta x^{(d)})\sin(j\Delta x^{(d)}) = \{G^{(d),(-1)}\}_{i,j} + \{G^{(d),(1)}\}_{i,j}, \tag{4.32}$$

where

$$\{G^{(d),(1)}\}_{i,j} = \cos(i\Delta x^{(d)} + j\Delta x^{(d)}), \tag{4.33}$$

and

$$\{G^{(d),(-1)}\}_{i,j} = \cos(\Delta x^{(d)} - \Delta x^{(d)}). \tag{4.34}$$

The entry at row i and column j of $G^{(d),(1)}$ can therefore be defined by the function $g(i+j) = -\cos(i+j)\Delta x^{(d)}$, and the entry at row i and column j of $G^{(d),(-1)}$ is given by $-g(i-j)$, satisfying the requirements for theorem 4.3.4.

For the matrix $\Phi_{\cos}^\top \Phi_{\sin}$, the product $\boldsymbol{\phi}_{\cos}(x_n^{(d)})\boldsymbol{\phi}_{\sin}(x_n^{(d)})^\top$ can be expanded as

$$\{\boldsymbol{\phi}_{\cos}(x_n^{(d)})\boldsymbol{\phi}_{\sin}(x_n^{(d)})^\top\}_{i,j} = \cos(i\Delta x^{(d)})\sin(j\Delta x^{(d)}) = \{G^{(d),(-1)}\}_{i,j} + \{G^{(d),(1)}\}_{i,j}; \tag{4.35}$$

where

$$\{G^{(d),(1)}\}_{i,j} = \sin(i\Delta x^{(d)} + j\Delta x^{(d)}), \tag{4.36}$$

and

$$\{G^{(d),(-1)}\}_{i,j} = \sin(i\Delta x^{(d)} - j\Delta x^{(d)}) \tag{4.37}$$

The entry at row i and column j of $G^{(d),(1)}$ can therefore be defined by the function $g(i+j) = -\cos((i+j)\Delta x^{(d)})$, and the entry at row i and column j of $G^{(d),(-1)}$ is given by $-g(i-j)$, satisfying the requirements for theorem 4.3.4.

For the matrix $\Phi_{\cos}^\top \Phi_{\cos}$, the product $\boldsymbol{\phi}_{\cos}(x_n^{(d)})\boldsymbol{\phi}_{\cos}(x_n^{(d)})^\top$ can be expanded as

$$\{\boldsymbol{\phi}_{\cos}(x_n^{(d)})\boldsymbol{\phi}_{\cos}(x_n^{(d)})^\top\}_{i,j} = \cos(i\Delta x^{(d)})\cos(j\Delta x^{(d)}) = \{G^{(d),(-1)}\}_{i,j} + \{G^{(d),(1)}\}_{i,j}; \tag{4.38}$$

where

$$\{G^{(d),(1)}\}_{i,j} = \cos(i\Delta x^{(d)} + j\Delta x^{(d)}), \tag{4.39}$$

and

$$\{G^{(d),(-1)}\}_{i,j} = \cos(i\Delta x^{(d)} - j\Delta x^{(d)}). \tag{4.40}$$

The entry at row i and column j of $G^{(d),(1)}$ can therefore be defined by the function $g(i+j) = \cos((i+j)\Delta x^{(d)})$, and the entry at row i and column j of $G^{(d),(-1)}$ is given by $g(i-j)$. An important notion for this matrix which makes it different from $\Phi_{\sin}\Phi_{\sin}^\top$ and $\Phi_{\cos}\Phi_{\sin}^\top$ is that this does not exactly satisfy the criteria for theorem 4.3.4. The difference is that for the criteria to be exactly satisfied, entry $\{G^{(d),(-1)}\}_{i,j}$ would have to be equal to

$-g(i-j)$ rather than $g(i-j)$. However, by applying the proof of theorem 4.3.4, but now noticing that the entries of $\boldsymbol{C} = \boldsymbol{\Phi}_{\cos}^{\top}\boldsymbol{\Phi}_{\cos}$ can be expressed element-wise as

$$C_{i,j} = \sum_{p=1}^{2^D}\sum_{n=1}^{N}\prod_{d=1}^{D} g(i_d + e_p^{(d)} j_d), \tag{4.41}$$

which allows us to use the tensor γ as defined in equation (4.22) to express each entry of $\boldsymbol{C}$ according to

$$C_{i,j} = \sum_{p}^{2^D} \gamma_{i_1 + e_p^{(1)}, \ldots, i_D + e_p^{(D)}}. \tag{4.42}$$

4.E Experiment Details

More specific details of the numerical experiments are provided here with additional visualizations and explanations.

U.S. Precipitation Data

The precipitation data is two-dimensional with $N = 5776$ data points first considered in [95]. We perform regression in the lat/lon domain and first center the data (but do not perform scaling) and use a simple squared-exponential kernel. We optimized the maximum likelihood using GPJax [93] for both the HGP as well as the standard GP for 100 iterations using Adam [98]. The hyperparameters of the kernel and likelihood were initialized as $l = 1, \sigma_{SE}^2 = 10$ and $\sigma_e = 1$, where l is the lengthscale, σ_{SE}^2 is the kernel variance and σ_e the noise standard deviation. Purely for visualization purposes, the inputs were projected to a local coordinate system given by CRS 5070 which are used for all of the plots. The original data is plotted in figure 4.7. The timing experiments were run using $m_d = 5, 10, \ldots, 65$ BF along each dimension, totaling between $M = 25$ and $M = 4225$ BF.

Magnetic Field Mapping

The magnetic field data has $N \approx 1.4$ million data points and is collected with an underwater vehicle outside the Norwegian coast. The data used were collected by MARMINE/NTNU research cruise funded by the Research Council of Norway (Norges Forskningsråd, NFR) Project No. 247626/O30 and associated industrial partners. Ocean Floor Geophysics provided magnetometer that was used for magnetic data acquisition and pre-processed the magnetic data. The data was later split into a training set and test set, at a roughly 50% split. The nature of the data split is visualized in figure 4.9. However, in practice, we selected the width of the test squares and the training squares smaller than the one displayed in the illustration and they are merely that big for visualization purposes. The illustration displays squares that are 0.01 latitudinal degrees wide and 0.03 longitudinal degrees tall, corresponding to approximately 1 kilometer in Cartesian coordinates in this area. The split we actually used was squares which were 0.001 latitudinal degrees wide and 0.003 longitudinal degrees tall, corresponding to approximately 100 meters in both directions in Cartesian coordinates in that area. GP regression with a squared exponential kernel is able to extrapolate for approximately one lengthscale, but will not necessarily give a very informative prediction one or two lengthscales away from the nearest measurement.

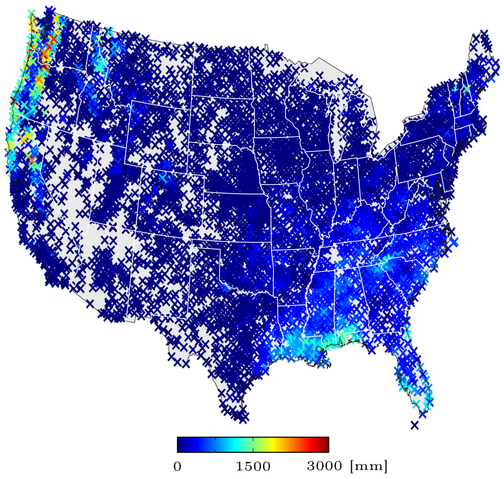


Figure 4.7: Raw data from precipitation data set. Each data point is visualized as a cross with color indicating the precipitation. The data set contains mostly low frequency content with some high frequency content apparent in the west coast as well as the south eastern parts.

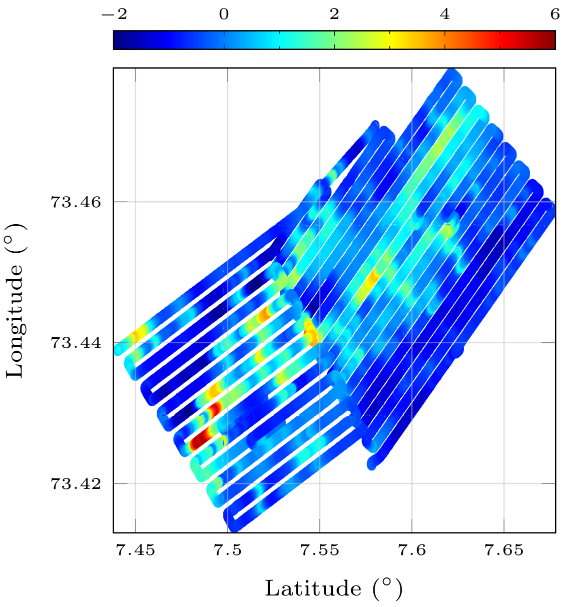


Figure 4.8: Raw magnetic field measurements for the underwater magnetic field data. The plotted data is subsampled to 100th of the data for visualization purposes.

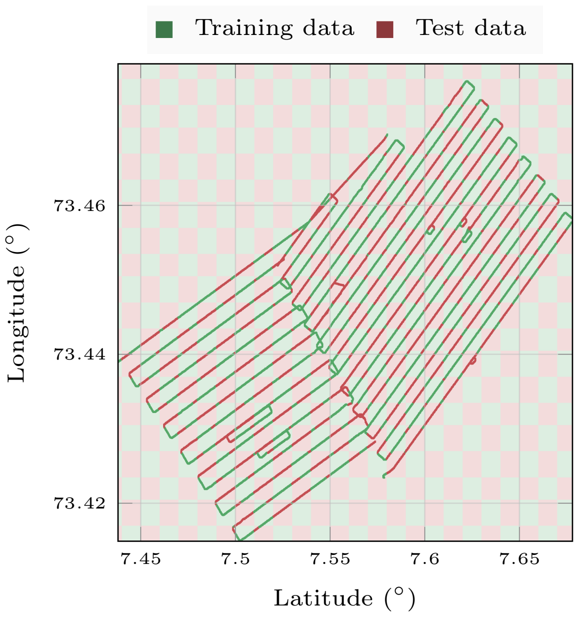
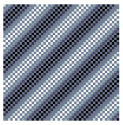
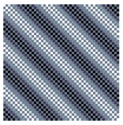
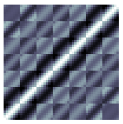
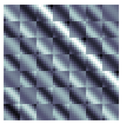
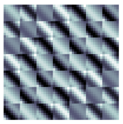
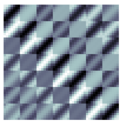


Figure 4.9: Data divided into training and test set. Deterministically split to ensure ensure that the lengthscale is captured properly in the training data. The data is roughly 50/50 split between training and test set. Both training and test data are normalized only by the mean and standard deviation of the training data.

Although we do not know the lengthscale that would optimize the likelihood of the data before using training data to find it, we see from a zoomed-in version of figure 4.8 approximately how fast the magnetic field is changing across the spatial dimension and use this to make a reasonable guess at the distance we expect a well-tuned GP to be able to extrapolate the learned magnetic field. We then project the data into a local coordinate system using WGS84 and perform regression in Cartesian coordinates. We center and standardize the data with the mean and standard deviation of the training data. A squared-exponential kernel was used and we optimize the maximum likelihood in GPJax [93] using Adam [98] for 100 iterations to find hyperparameters. The hyperparameters were initialized as $l = 200\text{meter}, \sigma^2_{\text{SE}} = 1$ and $\sigma^2_{\text{y}} = 1$. The resulting hyperparameters were $l_{SE} = 190$, $\sigma^2_{\text{y}} = 0.0675$, $\sigma^2_{\text{SE}} = 7.15$.

4.F Overview of Block Hankel–Toeplitz Matrix Structures

Table 4.1: An overview of the matrix structure and tensor representation for Hankel, Toeplitz, block Hankel, block Toeplitz and block Hankel matrices. The illustrations are examples of matrices with the property described in each row. The illustrations contain one square for each matrix entry, where the color of the square corresponds to the value.

Structure	Definition	Visual	Domain
Hankel	$\boldsymbol{H} = \begin{bmatrix} \gamma_1 & \gamma_2 & \cdots & \gamma_m \\ \gamma_2 & \gamma_3 & \cdots & \gamma_{m+1} \\ \vdots & \vdots & \ddots & \vdots \\ \gamma_m & \gamma_{m+1} & \cdots & \gamma_{2m-1} \end{bmatrix}$		$\gamma \in \mathbb{R}^K$
Toeplitz	$\boldsymbol{T} = \begin{bmatrix} \gamma_m & \cdots & \gamma_2 & \gamma_1 \\ \gamma_{m+1} & \cdots & \gamma_3 & \gamma_2 \\ \vdots & \iddots & \vdots & \vdots \\ \gamma_{2m-1} & \cdots & \gamma_{m+1} & \gamma_m \end{bmatrix}$		$\gamma \in \mathbb{R}^K$
D-level block Hankel	$\boldsymbol{H}^{(D)} = \begin{bmatrix} \boldsymbol{H}_1^{(D-1)} & \boldsymbol{H}_2^{(D-1)} & \cdots & \boldsymbol{H}_{m_D}^{(D-1)} \\ \boldsymbol{H}_2^{(D-1)} & \boldsymbol{H}_3^{(D-1)} & \cdots & \boldsymbol{H}_{m_D+1}^{(D-1)} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{H}_{m_D}^{(D-1)} & \boldsymbol{H}_{m_D+1}^{(D-1)} & \cdots & \boldsymbol{H}_{2m_D-1}^{(D-1)} \end{bmatrix}$		$\gamma \in \mathbb{R}^{K_1 \times \ldots \times K_D}$
D-level block Toeplitz	$\boldsymbol{T}^{(D)} = \begin{bmatrix} \boldsymbol{T}_{m_D}^{(D-1)} & \cdots & \boldsymbol{T}_2^{(D-1)} & \boldsymbol{T}_1^{(D-1)} \\ \boldsymbol{T}_{m_D+1}^{(D-1)} & \cdots & \boldsymbol{T}_2^{(D-1)} & \boldsymbol{T}_2^{(D-1)} \\ \vdots & \iddots & \vdots & \vdots \\ \boldsymbol{T}_{2m_D-1}^{(D-1)} & \cdots & \boldsymbol{T}_{m_D+1}^{(D-1)} & \boldsymbol{T}_{m_D}^{(D-1)} \end{bmatrix}$		$\gamma \in \mathbb{R}^{K_1 \times \ldots \times K_D}$
D-level block Hankel–Toeplitz, level D is Hankel	$\boldsymbol{G}^{(D)} = \begin{bmatrix} \boldsymbol{G}_1^{(D-1)} & \boldsymbol{G}_2^{(D-1)} & \cdots & \boldsymbol{G}_{m_D}^{(D-1)} \\ \boldsymbol{G}_2^{(D-1)} & \boldsymbol{G}_3^{(D-1)} & \cdots & \boldsymbol{G}_{m_D+1}^{(D-1)} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{G}_{m_D}^{(D-1)} & \boldsymbol{G}_{m_D+1}^{(D-1)} & \cdots & \boldsymbol{G}_{2m_D-1}^{(D-1)} \end{bmatrix}$		$\gamma \in \mathbb{R}^{K_1 \times \ldots \times K_D}$
D-level block Hankel–Toeplitz, level D is Toeplitz	$\boldsymbol{G}^{(D)} = \begin{bmatrix} \boldsymbol{G}_{m_D}^{(D-1)} & \cdots & \boldsymbol{G}_2^{(D-1)} & \boldsymbol{G}_1^{(D-1)} \\ \boldsymbol{G}_{m_D+1}^{(D-1)} & \cdots & \boldsymbol{G}_3^{(D-1)} & \boldsymbol{G}_2^{(D-1)} \\ \vdots & \iddots & \vdots & \vdots \\ \boldsymbol{G}_{2m_D-1}^{(D-1)} & \cdots & \boldsymbol{G}_{m_D+1}^{(D-1)} & \boldsymbol{G}_{m_D}^{(D-1)} \end{bmatrix}$		$\gamma \in \mathbb{R}^{K_1 \times \ldots \times K_D}$

5

Spatially scalable recursive estimation of Gaussian process terrain maps using local basis functions

When an agent, person, vehicle or robot is moving through an unknown environment without GNSS signals, online mapping of nonlinear terrains can be used to improve position estimates when the agent returns to a previously mapped area. Mapping algorithms using online Gaussian process (GP) regression are commonly integrated in algorithms for simultaneous localisation and mapping (SLAM). However, GP mapping algorithms have increasing computational demands as the mapped area expands relative to spatial field variations. This is due to the need for estimating an increasing amount of map parameters as the area of the map grows. Contrary to this, we propose a recursive GP mapping estimation algorithm which uses local basis functions in an information filter to achieve spatial scalability. Our proposed approximation employs a global grid of finite support basis functions but restricts computations to a localized subset around each prediction point. As our proposed algorithm is recursive, it can naturally be incorporated into existing algorithms that uses Gaussian process maps for SLAM. Incorporating our proposed algorithm into an extended Kalman filter (EKF) for magnetic field SLAM reduces the overall computational complexity of the algorithm. We show experimentally that our algorithm is faster than existing methods when the mapped area is large and the map is based on many measurements, both for recursive mapping tasks and for magnetic field SLAM.

This chapter is based on 🗎 *Frida Viset, Rudy Helmons, Manon Kok*: Spatially scalable recursive estimation of Gaussian process terrain maps using local basis functions. Under review.

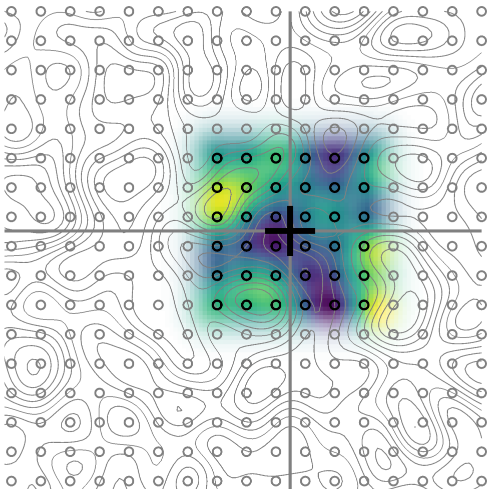


Figure 5.1: A locally reconstructed approximation (indicated by the color of the heatmap) of a simulated large, nonlinear geospatial field (indicated with gray level curves) based on a local subset (marked with the black circles) of a global grid of basis functions (marked with the gray circles).

5.1 Introduction

Navigation in large, unknown, nonlinear geospatial fields can be done through simultaneous localisation and mapping (SLAM). This requires a spatially scalable online mapping technique. SLAM using nonlinear geospatial field maps has been used to compensate for odometry drift in robots, marine vehicles, aerial vehicles, and pedestrians [8, 38, 99, 100]. Some nonlinear fields that have been used in SLAM are magnetic fields [8, 38, 72], underwater bathymetry [99, 101], and terrain fields [102, 103]. A range of this research uses Gaussian process (GP) regression [16] to create nonlinear field maps [8, 99, 101–103]. These methods suffer from the computational complexity scaling either with the number of measurements used to build the map or with the area of the map. To remedy this, we propose a temporally and spatially scalable online GP mapping algorithm that removes this bottleneck from SLAM algorithms that use GPs for terrain navigation. Our proposed approximation uses a global grid of finite-support basis functions to approximate the GP. At each timestep, our approach restricts computations to a local grid of basis functions surrounding each prediction point, as illustrated in Fig. 5.1, which results in a reduction in computational cost. Our algorithm does not only reduce the computational cost for constructing maps, its recursive nature also opens up for using it in SLAM. We extend the magnetic field SLAM algorithm from [74] to include our recursive mapping algorithm, and show that it causes a reduction in the overall computational complexity of the SLAM algorithm. The magnetic field SLAM approach in [74] consists of a measurement update and a dynamic update. Our approach directly reduces the computational complexity of the measurement update. As our approximation gives rise to sparsity patterns in the map representation, our approximation also gives rise to a reduction in complexity in the dynamic update.

The remainder of the paper is organized as follows: Section 5.2 gives an overview of some background information. Specifically, it introduces GP regression and basis function approximations to GPs. Section 5.3 gives a description of our method for approximating a large GP scale map, and shows how the map can be incorporated in an EKF for SLAM. Section 5.4 gives an overview of the relation between our proposed mapping algorithm and other online GP mapping algorithms that can be used in SLAM. Section 5.5 describes experimental results comparing our proposed approach to other mapping and SLAM algorithms. Section 5.6 gives some concluding remarks and recommendations for future work.

5.2 Background

5.2.1 GP Regression

We are interested in estimating the terrain map using a GP. GP regression allows for estimating a nonlinear function $f : \mathbb{R}^d \to \mathbb{R}$, distributed according to

$$f \sim \mathcal{GP}(0, \kappa(\cdot,\cdot)), \tag{5.1}$$

where $\kappa(x,x') : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is some known kernel function, and $\mathcal{GP}(0,\kappa(\cdot,\cdot))$ denotes the GP prior (see [16] for definition) with a mean of 0 and covariance defined by the kernel function. The kernel that is used for all of the mentioned online GP maps in section 5.4 is called the squared exponential kernel, and is defined as

$$\kappa_{\text{SE}}(x,x') = \sigma_{\text{SE}}^2 \exp\left(\frac{\|x-x'\|_2^2}{2l_{\text{SE}}^2}\right), \tag{5.2}$$

where $\|\cdot\|_2$ is the Euclidean norm, σ_{SE} is a hyperparameter indicating the magnitude of the spatial variations and l_{SE} is a hyperparameter indicating the expected lengthscale of the spatial variations [16]. GP regression also uses N noisy measurements of the function $y_{1:N} = \{y_t\}_{t=1}^N$ modelled as

$$y_t = f(x_t) + e_t, \qquad e_t \sim \mathcal{N}(0,\sigma_{\text{y}}^2), \tag{5.3}$$

where $x_{1:N} = \{x_t\}_{t=1}^N$ are known input locations where $x_t \in \mathbb{R}^d$, e_t is a measurement noise, σ_{y}^2 is the noise variance, and $\mathcal{N}(0,\sigma_{\text{y}}^2)$ denotes the normal distribution with mean 0 and covariance σ_{y}^2. The expected value and variance of the function value in any arbitrary location $x^\star \in \mathbb{R}^d$ is then given by

$$\begin{aligned} &\mathbb{E}[f(x^\star)] \\ &= K(x^\star, x_{1:N})\left(K(x_{1:N},x_{1:N}) + \sigma_{\text{y}}^2 I_N\right)^{-1} y_{1:N}, \end{aligned} \tag{5.4a}$$

$$\begin{aligned} &\text{Var}[f(x^\star)] = \\ &-K(x^\star, x_{1:N})\left(K(x_{1:N},x_{1:N}) + \sigma_{\text{y}}^2 I_N\right)^{-1} \\ &K(x_{1:N},x^\star) + K(x^\star,x^\star), \end{aligned} \tag{5.4b}$$

respectively. Here, the matrix $K(x_{1:N},x_{1:N})$ is constructed by the kernel evaluated along each possible cross-combination of the entries in the vector $x_{1:N}$, such that the entry on the ith row and the jth column of $K(x_{1:N}, x_{1:N})$ is $\kappa(x_i, x_j)$. Similarly, the row vector $K(x^\star, x_{1:N})$ is defined such that the jth column is defined by $\kappa(x^\star, x_j)$. By the same notation, $K(x^\star, x^\star)$ is a single-entry matrix with value $\kappa(x^\star, x^\star)$. We are concerned with approximating the GP posterior. To distinguish the posterior in (5.4a) and (5.4b) from any approximation of it, we will refer to this as the full GP posterior. Computing the full GP posterior has a complexity of $\mathcal{O}(N^3)$, as it requires the inversion of a $N \times N$ matrix.

5.2.2 Sparse approximations to GP Regression with Basis Functions

Sparse approximations to GP regression approximate the function f with a linear combination of m basis functions according to

$$f \approx \Phi^\top w, \qquad w \sim \mathcal{N}(0, P), \tag{5.5}$$

where $w = \{w_i\}_{i=1}^m$ is a set of m scalar weights, and Φ is a vector of m basis functions $\phi_i : \mathbb{R}^d \to \mathbb{R}$ [21]. The prior covariance on the weights $P \in \mathbb{R}^{m \times m}$ is chosen so that (5.5) approximates (5.1) [21].

There are different methods to approximate predictions using the assumption in (5.5). A commonly used one is the Deterministic Training Conditional (DTC) approximation [21]. The sparse predictions with DTC are given by

$$\begin{aligned}\mathbb{E}[f(x^\star)] \approx{}& \Phi(x^\star)^\top \left(\Phi(x_{1:N})\Phi(x_{1:N})^\top + \right. \\ & \left. \sigma_y^2 P^{-1}\right)^{-1} \Phi(x_{1:N}) y_{1:N},\end{aligned} \tag{5.6a}$$

$$\begin{aligned}\mathrm{Var}[f(x^\star)] \approx{}& \sigma_y^2 \Phi(x^\star)^\top \left(\Phi(x_{1:N})\Phi(x_{1:N})^\top + \right. \\ & \left. \sigma_y^2 P^{-1}\right)^{-1} \Phi(x^\star) \\ & + K(x^\star, x^\star) - \Phi(x^\star)^\top P \Phi(x^\star),\end{aligned} \tag{5.6b}$$

where the entry on the ith row and the jth column of the matrix $\Phi(x_{1:N})$ is defined as $\phi_i(x_j)$, and the ith row of the column vector $\Phi(x^\star)$ is defined as $\phi_i(x^\star)$. The expressions in (5.6a) and (5.6b) require $\mathcal{O}(Nm^2 + m^3)$ operations to compute, and $\mathcal{O}(Nm^2)$ storage. This greatly improves the computational complexity in the case where the number of basis functions $m \ll N$ compared to the full GP approach in Section 5.2.1. The required number of basis functions m required to accurately approximate the GP scales with the size of the input domain relative to the lengthscale (l_{SE} in (5.2)) of the kernel [11].

5.3 Method

In terrain mapping we typically have large areas and small lengthscales. The approach from Section 5.2.2 needs many basis functions in this case. To remedy this scaling, we propose training on a large grid of finite-support basis functions which we will introduce in Section 5.3.1. We incorporate each new measurement with an information filter. Section 5.3.2 describes how we include a new measurement. Including a new measurement in the way described in section 5.3.2 has a computational complexity of $\mathcal{O}(m'^2)$, where m' is the number of basis functions with overlapping support in any input location. Section 5.3.3 describes how we use this trained map to make a prediction. Making a prediction in the way described in Section 5.3.3 requires $\mathcal{O}(m''^3)$ computations. Section 5.3.4 suggests how our proposed GP map approximation can be integrated into EKF SLAM with GP maps. The algorithm given in Section 5.3.4 requires $\mathcal{O}(m)$ computations at each timestep.

5.3.1 Choice of Basis Functions

We choose a set of basis functions $\{\phi_j\}_{j=1}^m$ as truncated cross-sections with centers distributed on a uniform grid according to

$$\phi_j(x) = \begin{cases} \kappa(u_j, x), & \|x - u_j\|_\infty \leq r \\ 0, & \|x - u_j\|_\infty > r \end{cases}, \tag{5.7}$$

where $\|\cdot\|_\infty$ denotes the sup-norm, and where each basis function j is centered in a fixed location $u_j \in \mathbb{R}^d$ on a d-dimensional grid. For stationary geospatial fields, d is maximally 3. The basis functions converge to the basis functions induced by a grid of inducing inputs as the truncation limit tends to infinity. A grid of inducing inputs is known to converge to the true GP as the grid density goes to infinity [104].

5.3.2 Recursive map estimation

Since the basis function approximation in (5.5) to the full GP regression is a parametric model, the posterior can be found using stochastic least squares estimation. A stochastic least squares estimate can recursively be obtained using a information filter without any dynamics. Specifically, obtaining the posterior on information form corresponds to computing the terms $\Phi(x_{1:N})y_{1:N}$ and $\Phi(x_{1:N})\Phi(x_{1:N})^\top$ in (5.6a) and (5.6b) recursively [105]. In the remainder of this paper, we let the information vector at time t be defined as $\iota_{1:t} = \Phi(x_{1:t})y_{1:t}$, and the information matrix at time t be defined as $\mathcal{I}_{1:t} = \Phi(x_{1:t})\Phi(x_{1:t})^\top$. Updating the information vector $\iota_{1:t}$ and the information matrix $\mathcal{I}_{1:t}$ as [105]

$$\iota_{1:t} = \iota_{1:t-1} + \Phi(x_t)y_t, \tag{5.8a}$$

$$\mathcal{I}_{1:t} = \mathcal{I}_{1:t-1} + \Phi(x_t)\Phi(x_t)^\top, \tag{5.8b}$$

only requires updating a finite amount of elements $m' \ll m$ in each update step when using the basis functions from Section 5.3.1. This is because the terms $\Phi(x_t)y_t$ and $\Phi(x_t)\Phi(x_t)^\top$ only contain m' non-zero elements because only m' basis functions have overlapping support in any given location. The set $\mathcal{S}$ of basis functions (identified by their index j) that are non-zero in x_t follows from the definition of the basis functions in (5.7) as

$$\mathcal{S} = \{j \,|\, \|x_t - u_j\|_\infty \leq r\}. \tag{5.9}$$

It follows that for any single measurement y_t, the only entries of the information vector ι_j that change are the ones where $j \in \mathcal{S}$. Let l_u denote the distance between two neighboring inducing inputs along each dimension. The size of the set $\mathcal{S}$ will always be smaller than or equal to $m' = (\frac{2r}{l_u} + 1)^d$, where d is the dimension of the input vector. The recursive updates in (5.8a) and (5.8b) only need to be applied to entries $j \in \mathcal{S}(x_t, r)$ for the information vector, and $j, j' \in \mathcal{S}(x_t, r) \times \mathcal{S}(x_t, r)$ for the information matrix, as the contribution from the term $\phi_j(x_t)y_t$ and $\phi_j(x_t)\phi_{j'}(x_t)$ will be zero for all other entries. Hence this update has computational complexity $\mathcal{O}(m')$ for the information vector, and $\mathcal{O}(m'^2)$ for the information matrix.

5.3.3 Prediction

Making a prediction of the terrain value in a new location requires evaluating the GP posterior mean and covariance in that location.

We could simply plug the information vector $\iota_{1:N} = \Phi(x_{1:N})y_{1:N}$ and the information matrix $\mathcal{I}_{1:N} = \Phi(x_{1:N})\Phi(x_{1:N})^\top$ into the approximate GP posterior in (5.6a) and (5.6b). However, solving these equations would require solving a linear system with m variables, which in general requires $\mathcal{O}(m^3)$ operations. Instead, we use only a local subset of m'' basis functions to perform each prediction. We select the local subset of basis functions in a local domain surrounding the prediction point.

The entries of the information vector of any subset of the basis functions are a subset of the entries of the information vector $\iota_{1:N}$. Equivalently, the entries of the information matrix of any subset of the basis functions are a subset of the entries of the information matrix $\mathcal{I}_{1:N}$. As an illustrating example, consider a set of four basis functions $\{1,2,3,4\}$. Then, consider the subset of basis functions indicated in black $\{2,3\}$. The entry at row i and column j of the information matrix $\mathcal{I}_{1:N}$ is given by $\mathcal{I}^{i,j}_{1:N} = \sum_{t=1}^{N} \phi_i(x_t)\phi_j(x_t)$. The entry in row i of the information vector $\iota_{1:N}$ is given by the sum $\iota^{i}_{1:N} = \sum_{t=1}^{N} \phi_i(x_t)y_t$. The information matrix and information vector for the full set can therefore be written element-wise as

$$\mathcal{I}_{1:N}=\begin{bmatrix} \mathcal{I}^{1,1}_{1:N} & \mathcal{I}^{1,2}_{1:N} & \mathcal{I}^{1,3}_{1:N} & \mathcal{I}^{1,4}_{1:N} \\ \mathcal{I}^{2,1}_{1:N} & \mathcal{I}^{2,2}_{1:N} & \mathcal{I}^{2,3}_{1:N} & \mathcal{I}^{2,4}_{1:N} \\ \mathcal{I}^{3,1}_{1:N} & \mathcal{I}^{3,2}_{1:N} & \mathcal{I}^{3,3}_{1:N} & \mathcal{I}^{3,4}_{1:N} \\ \mathcal{I}^{4,1}_{1:N} & \mathcal{I}^{4,2}_{1:N} & \mathcal{I}^{4,3}_{1:N} & \mathcal{I}^{4,4}_{1:N} \end{bmatrix}, \tag{5.10a}$$

$$\iota_{1:N}=\begin{bmatrix} \iota^{1}_{1:N} & \iota^{2}_{1:N} & \iota^{3}_{1:N} & \iota^{4}_{1:N} \end{bmatrix}. \tag{5.10b}$$

The information matrix and information vector of basis functions 2 and 3 is a subset of the information matrix and information vector indicated by the black sub-entries in (5.10a)-(5.10b).

We leverage this property to define an approximation to the GP based on a local subset $S^\star$ of basis functions close to the prediction point. In Fig. 5.1 we illustrate how a local subset of basis functions can be selected from a global grid. The local subset of the corresponding information vector and information matrix defined as $\iota^\star_{1:N} = \Phi_{S^\star}(x_{1:N})y_{1:N}$ and $\mathcal{I}^\star_{1:N} = \Phi_{S^\star}(x_{1:N})\Phi_{S^\star}(x_{1:N})^\top$, respectively, can be constructed from subsets of the information vector $\iota_{1:N}$ and the information matrix $\mathcal{I}_{1:N}$ for the full grid. From acquiring the information matrix and information vector on the global grid of basis functions indicated with gray circles in Fig. 5.1, we have access to the trained system the local subsets of basis functions illustrated with black circles in Fig. 5.1.

Formally, we choose $S^\star = S(x^\star, r^\star)$ consisting of all basis functions whose center is closer than $r^\star$ along each dimension to be in this set. The size of the set $S(x^\star, r^\star)$ will always be smaller than or equal to $m'' = (\frac{2r^\star}{l_u} + 1)^d$, where d is the dimension of the input vector. Evaluating the predicted mean and predicted variance using the basis functions indexed by the set $S^\star$ in place of the basis functions $\{1,\ldots,N\}$ in (5.6a)-(5.6b) gives

$$\mathbb{E}[f(x^\star)] \approx \Phi_{\mathcal{S}^\star}(x^\star)^\top(\mathcal{I}^\star_{1:N} + \sigma_y^2 P_\star^{-1})^{-1}\iota_{\mathcal{S}^\star}, \tag{5.11a}$$

$$\begin{aligned}\operatorname{Var}[f(x^\star)] \approx{}& \sigma_y^2 \Phi_{\mathcal{S}^\star}(x^\star)^\top(\mathcal{I}^\star_{1:N} + \sigma_y^2 P_\star^{-1})^{-1} \\ & \Phi_{\mathcal{S}^\star}(x^\star) + K(x^\star,x^\star) \\ & - \Phi_{\mathcal{S}^\star}(x^\star)^\top P_\star \Phi_{\mathcal{S}^\star}(x^\star),\end{aligned} \tag{5.11b}$$

where the ith row and the jth column of $\Phi_{\mathcal{S}^\star}(x^\star)$ is defined by the evaluation of basis functions number i in the subset $\mathcal{S}^\star$ on the prediction point $x^\star$, and where $P_\star$ is the prior covariance on the basis function weights. We select $P_\star$ such that the prior is recovered exactly in the center locations $u_{\mathcal{S}^\star}$ of the basis functions, which gives $P_\star = (\Phi_{\mathcal{S}^\star}(u_{\mathcal{S}^\star}))^{-\top} K(u_{\mathcal{S}^\star}, u_{\mathcal{S}^\star})(\Phi_{\mathcal{S}^\star}(u_{\mathcal{S}^\star}))^{-1}$. A detailed derivation for the expression for $P_\star$ is given in Appendix 5.A. As the size m'' of the set $\mathcal{S}^\star$ is always smaller than or equal to $(\frac{2r^\star}{l_u}+1)^d$, the matrix inversion required in (5.11a) and (5.11b) is bounded. The prediction of our proposed GP prediction in (5.11a)-(5.11b) can therefore be computed with $\mathcal{O}(m''^3) = \mathcal{O}((\frac{2r^\star}{l_u}+1)^{3d})$ operations.

5.3.4 Integration of mapping algorithm into an EKF for magnetic field SLAM

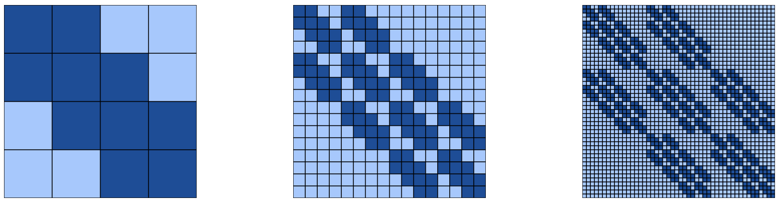
In this section, we give the algorithmic details of the implementation of a SLAM algorithm with an EKF as it is described in [74], but using our proposed map approximation instead of Hilbert space basis functions. From here on we will refer to the magnetic field SLAM algorithm with and EKF in [74] as EKF Mag-SLAM.

EKF Mag-SLAM gives estimates of the sensor's position $\hat{x}_t$, its orientation $\hat{q}_{t|t}$ and the magnetic field map $\hat{m}_{t|t}$. The magnetic field map is parameterized as the estimate of the weights w_t of a sparse GP approximated with Hilbert space basis functions, using online measurements of the magnetic field denoted y_t at each timestep t and measurements of the change in position and orientation denoted Δq_t and Δp_t at each timestep t respectively.

We use the same measurement and dynamic model as [74] (see Eqns (14a), (14b) and (13)), but we replace the basis functions denoted Φ with the finite-support basis functions described in section 5.3.1. As the computational benefits of the finite-support basis functions arise on the information form, we implement the EKF on the information form.

In the measurement update, the system is approximated using only the information of the basis functions that are closer than r^* to the prediction point. This means that the information between two prediction points that are further apart than $2r^*$ will never be needed to carry out the measurement update. This means that the only entries of the information matrix we need to compute to carry out the dynamic update are given a sparse subset of the information matrix. We name this subset of all index pairs such that the distance between the corresponding basis functions is less than $2r^*$ as $\mathcal{S}_{\forall\star} = \{i,j \| \|x_i - x_j|_\infty \leq 2r^*\}$. The location of the indices corresponding to this subset for the information matrix for Mag-SLAM in three dimensions is illustrated in Fig. 5.2. The pattern arises from the ordering of the indices in the information matrix according to the location of the finite-support basis functions along each of the three spatial dimensions.

As the information matrix for online Kalman filters with dynamic updates [106] is generally defined as the inverse of the covariance matrix P, we initialize the information



(a) 1D grid (b) 2D grid (c) 3D grid

Figure 5.2: Sparsity patterns illustrating which entries i, j in the information matrix correspond to pairs of basis function locations x_i, x_j that are closer than $2r^\star$ according to the infinity norm (dark blue) and which are not (light blue). The patterns arise from the ordering of the indexes of the basis functions, relative to their locations along each of the three dimensions.

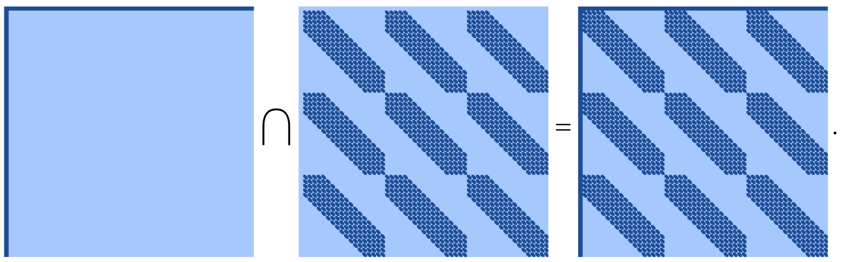


Figure 5.3: Sparsity pattern illustration of the information matrix for the full state consisting of the position, orientation and magnetic field. The first sparsity pattern has a dark blue color in the entries corresponding to the first six columns and the first six rows of the information matrix (corresponding to position and orientation), and a light blue color in the other entries. The second sparsity pattern has a dark blue color in the entries in the set $S_{\forall\star}$ of all entries that can possibly be necessary to make a map prediction in any location. The last sparsity pattern is the union of these two sets.

matrix as $I_0^{\text{EKF}} = P_0^{-1}$. The initial error state of EKF Mag-SLAM is zero and thus the initial information vector $\iota_0 = \mathcal{I}_0^{\text{EKF}} \xi_0 = 0$.

The dynamic update on information form in general is given by

$$(\mathcal{I}_{t+1}^{\text{EKF}})^{-1} = ((\mathcal{I}_t^{\text{EKF}})^{-1} + Q)^{-1}, \tag{5.12}$$

where Q is the process noise on the state-space vector, defined in equation (23) in [74]. Since we assume the map to be static, the matrix Q can be factorized according to

$$\begin{bmatrix} \tilde{Q} & 0 \\ 0 & 0 \end{bmatrix} = V^\top Q V, \qquad V = \begin{bmatrix} I & 0 \end{bmatrix}, \tag{5.13}$$

and where $\tilde{Q}$ is a 6×6 matrix expressing the process noise on the position and orientation. After applying the matrix inversion lemma to this dynamic update, it reduces to

$$\mathcal{I}_{t+1}^{\text{EKF}} = \mathcal{I}_t^{\text{EKF}} - \mathcal{I}_t^{\text{EKF}} V^\top (V \mathcal{I}_t^{\text{EKF}} V^\top + \tilde{Q})^{-1} V I^{\text{EKF}_t}. \tag{5.14}$$

5

The Kalman filter measurement update of the information matrix when applied to Mag-SLAM becomes [106]

$$\mathcal{I}_{t+1}^{\text{EKF}} = \mathcal{I}_t^{\text{EKF}} + \tfrac{1}{\sigma_y^2} H_t H_t^\top, \tag{5.15}$$

where the Jacobian of the measurement model H_t is as derived in a similar way as equation (25) in [74], but with finite-support basis functions from 5.7 replacing the Hilbert space basis functions. This update, as the one in (5.8b) is inherently sparse, and has complexity $O(m'')$. Similarly, the information vector of EKF-Mag SLAM at each timestep becomes [106]

$$\iota_t^{\text{EKF}} = \tfrac{1}{\sigma_y^2} H_t y_t, \tag{5.16}$$

with y_t being a three-component magnetic field measurement. (see [107] for a detailed derivation of the extended information filter). The re-linearisation of the estimated position, orientation and magnetic field would then in general require an inversion of the full information matrix. However, by employing the approximation that only the basis functions that are closer to the prediction point (which in this case is the estimated location) are the ones necessary to approximate the magnetic field, we can execute the re-linearisation at a computational cost of $\mathcal{O}(m''^3)$. We do this explicitly according to

$$\begin{aligned} \begin{bmatrix} \delta_t^\top & \eta_t^\top & \nu_{\star,t}^\top \end{bmatrix}^\top &= (I_{t,\star}^{\text{EKF}})^{-1} \iota_{t,\star}^{\text{EKF}}, && (5.17\text{a}) \\ \hat{p}_{t|t} &= \hat{p}_{t|t-1} + \delta_t, && (5.17\text{b}) \\ \hat{q}_{t|t} &= \hat{q}_{t|t-1} \odot \exp_{\text{q}}(\eta_t), && (5.17\text{c}) \\ \hat{m}_{\star,t|t} &= \hat{m}_{\star,t|t-1} + \nu_{\star,t}, && (5.17\text{d}) \end{aligned}$$

where we only add the correction on the linearisation point corresponding to the magnetic field map m_t in the entries in the subset $S_\star$. The orientation estimate $\hat{q}_{t|t}$ is represented as a unit quaternion. The operator $\exp_{\text{q}}(\cdot)$ is defined in [4].

What is worth noting about the relinearisation in equation (5.17) is that it only requires knowledge of a subset $S_\star$ of the entries in the information matrix. Furthermore, for all possible position estimates $\hat{p}_{t|t-1} \in \mathbb{R}^3$, the union of all possible subsets $S_\star$ is given as the set of index pairs i, j where the corresponding basis functions are closer to each other according to the infinity norm than $2r^\star$. We can formally define this union of all possible usable subsets as $S_{\star,\forall} = \{i, j \| \|p_{i,t} - p_{j,t}\|_\infty \leq 2r^\star\}$. For an information matrix I, this subset is a sparse subset of the full set of entries, with $\mathcal{O}(mm''^2)$ elements. Assuming $m >> m''$, this sparse subset contains $O(m)$ elements. Fig. 5.2 illustrates examples of sparsity patterns for an information matrix corresponding to a one-dimensional, a two-dimensional, and a three-dimensional grid of equispaced basis functions. In the one-dimensional grid, four basis functions are located along the x-axis at $[-1.5, -0.5, 0.5, 1.5]$, and the indexes are ordered correspondingly in Fig. 5.2. In the two-dimensional grid, 16 basis functions are located along the x and y-axis at

$$\begin{bmatrix} -1.5 & -1.5 & -1.5 & \dots & 1.5 & 1.5 \\ -1.5 & -0.5 & 0.5 & \dots & 0.5 & 1.5 \end{bmatrix}, \tag{5.18}$$

and also indexed chronologically.

5

The only entries of the information matrix that are used to compute the term

$$\mathcal{I}^{\text{EKF}} V^\top (V \mathcal{I}^{\text{EKF}} V^\top + \tilde{Q})^{-1} V \mathcal{I}^{\text{EKF}}, \tag{5.19}$$

are the entries in the first 6 rows and the first 6 columns of the information matrix. These are the only entries that can possibly affect the values contained in $\mathcal{S}_{\forall\star}$ during the dynamic update. The only entries of the information matrix we need to keep track of are therefore the union of the first 6 columns and the first 6 rows and the values in $\mathcal{S}_{\forall\ \star}$. Fig. 5.3 shows an example of the union of these two sparsity sets, for our estimation problem with an equispaced 3D grid of finite-support basis functions. The sparsity pattern shows a fractal-like structure that is explained by the fact that we are displaying the pattern for a 3D grid of basis functions, and the nature of the ordering of the indexes of these basis functions. It is therefore sufficient during application of the entire algorithm to only keep track of the entries in $\mathcal{S}_{\forall\ \star}$, and in the first six rows and the first six columns.

5.4 Relation to existing work

Many approximations to GP maps have been proposed to remedy the poor scalability of GP terrain maps. The particular use-case our proposed approximation does well compared to previous work is the ability to perform online GP mapping with a computational complexity that does not change over time as more measurements are gathered (temporal scalability), and that simultaneously does get slower as the size of the map increases relative to the spatial variations (spatial scalability).

5.4.1 Temporally scalable approaches which are not spatially scalable

GP regression approximated with basis functions is a temporally scalable and well-established strategy for online GP mapping [8, 13, 74, 76, 99, 108]. This is commonly used for online learning with GP regression [109]. A prediction can also be made at any time in any location with a computational complexity of $\mathcal{O}(m^2)$.

Practical examples of realistic terrains used for aiding navigation have small spatial variations relative to the size of the map [110–113]. Existing sparse approximations to GPs with basis functions require a number of computations which scale with the size of the mapped area relative to the lengthscale of the kernel [114]. This is due to the fact that GP approximations that efficiently recover the higher eigenvalues of the kernel but not the smaller eigenvalues will, in general, have poor performance when the lengthscale of the kernel is small relative to the domain [11, 114]. In practice, this manifests as a need for a larger number m of basis functions as the domain size increases [11], which in turn increases the computational requirements $\mathcal{O}(m^2)$. Another online GP mapping approximation using finite-support basis functions was proposed by [115], which uses a sparse-weight Kalman filter for this approach as described in [116], and therefore has a computational complexity of including each new measurement of $\mathcal{O}(m)$.

In both our work and a range of other work, the magnetic field map is estimated online in a Bayesian filter for SLAM by including the basis function weights in the state space [8, 74, 99]. Basis-function approximation to GPs are employed for online creation of magnetic field maps [8], underwater bathymetry maps [99] and ground elevation maps [108]. Due to the computational cost of $\mathcal{O}(m^2)$, SLAM algorithms using basis function approximations to create the map are limited to a small map area [74, 76, 99], or to breaking up the map into separate maps without sharing information between the maps [8, 117].

5.4.2 Spatially scalable approaches which are not temporally scalable

There is also some work for large-scale terrain mapping that suggests using only the k nearest measurements for applying GP regression [12, 111]. This approach uses KD-trees for efficient querying of the data. It requires storage which scales linearly in the amount of measurements. The storage requirements of this approach will therefore increase over time. Although they are spatially scalable, these approximations are therefore not spatially scalable.

5.4.3 Spatially and temporally scalable approximations that suffer from boundary effects

Local GPs avoid the issues of increased computational complexity with increasing domain size by splitting the input domain into discrete sub-domains [118, 119]. This approach suffers from discontinuities and inaccurate predictions at the domain boundaries [118]. Patched local GPs [120] and domain decomposition methods [121] both remedy this problem by introducing constraints connecting the local domains. However, this remedy to the boundary effect problem requires a number of computations that scale with the number of considered domains and thus do not truly achieve a prediction time complexity independent of the spatial size of nonlinear field [120]. Although all of the previously mentioned local GP regression approximations are formulated offline, an online approach to local GP regression was proposed by [122].

The technique of splitting the field into sub-domains have also been integrated into SLAM [8, 14, 72]. These approaches often require additional modifications in the SLAM algorithm to handle the cases where position estimates are close to borders of the domains, where the boundary effects are strong [8]. This is in contrast to our proposed approach,

Table 5.1: Comparison of computational complexities for online mapping.

Step	(SKI, [18])	Our method	HS functions
Meas update	$\mathcal{O}(m'^2)$	$\mathcal{O}(m'^2)$	$\mathcal{O}(m^2)$
Prediction	$\mathcal{O}(m\log(m))$	$\mathcal{O}(m''^3)$	$\mathcal{O}(m)$

where the local domain always surrounds the prediction point, and therefore does not suffer from boundary effects.

5.4.4 Structured Kernel Interpolation

Another approximation that performs GP regression efficiently in large maps was proposed by [18]. It applies the Woodbury matrix inversion lemma to reduce the required computational complexity of a GP approximation called Structured Kernel Interpolation (SKI) [104]. SKI uses grid-structured finite-support basis functions to perform approximate GP regression.

The novelty in our contribution in contrast to SKI is that we perform these updates online and that we perform predictions using only a local subset of m'' of the finite support basis functions. The prediction cost of [18] is dependent on the size of the input domain. Each prediction with SKI, therefore, requires $\mathcal{O}(m\log(m))$ operations, while each prediction with our method requires only $\mathcal{O}(m''^3)$ operations. SKI therefore is able to include new measurements in the map in a way which is both spatially and temporally scalable. In contrast to our work, SKI requires more computations to make predictions as the size of the map area grows. A detailed comparison of the computational complexity of SKI and our approach is given in table 5.1.

5.4.5 Integration into SLAM

The use of our approximation reduces the overall computational complexity of an existing extended Kalman filter Mag-SLAM (EKF-Mag-SLAM) algorithm with Hilbert space basis functions [74]. Mag-SLAM with an EKF or requires a measurement update and dynamic update at each timestep. Additionally, a prediction of the magnetic field is required at each timestep (this is officially part of the measurement update, but we consider it a separate step to evaluate the computational complexity required to make this prediction). Incorporating our approach in Mag-SLAM gives direct improvements to the computational requirements for the measurement update, and for making a prediction. Furthermore, our approximation of using only a local subset of the basis functions in the prediction step induces a sparsity structure which reduces the computational requirements of the dynamic update. This causes an overall improved computational complexity of the EKF Mag-SLAM algorithm in [74] to reduce from $O(m^2)$ to $O(m)$. An overview of the computational complexities of EKF Mag-SLAM with our mapping technique compared to EKF Mag-SLAM with Hilbert space methods is given in table 5.2.

Table 5.2: Comparison of computational complexities after integrating the approach into SLAM.

Step	EKF Mag-SLAM with our method	EKF Mag-SLAM with HS functions
Meas update	$\mathcal{O}(m'^2)$	$\mathcal{O}(m^2)$
Prediction	$\mathcal{O}(m''^3)$	$\mathcal{O}(m)$
Dyn update	$\mathcal{O}(m)$	$\mathcal{O}(m^2)$

Table 5.3: SMAE accuracies, combined time to include a new measurement and make a new prediction, MSLL scores on audio dataset. The results that obtain the lowest run-time while also obtaining the lowest SMAE is highlighted. SMAEs and MSLL scores are evaluated on all standardized test points inside the considered domain, and the average time plus minus one standard deviation is calculated based on 100 repetitions.

	10% of domain, $m = 800$			100% of domain, $m = 8000$		
	SMAE	Time[s]	MSLL	SMAE	Time[s]	MSLL
$r = 6l_{\text{SE}}$	0.42	$6.2 \cdot 10^{-5} \pm 2.1 \cdot 10^{-5}$	37.5	0.34	$7.3 \cdot 10^{-5} \pm 2.2 \cdot 10^{-5}$	48.0
$r = 12l_{\text{SE}}$	**0.22**	$\mathbf{9.4 \cdot 10^{-5} \pm 2.1 \cdot 10^{-5}}$	4.18	**0.20**	$\mathbf{1.1 \cdot 10^{-4} \pm 1.6 \cdot 10^{-5}}$	12.2
$r = 18l_{\text{SE}}$	0.22	$1.3 \cdot 10^{-4} \pm 2.7 \cdot 10^{-5}$	3.94	0.20	$1.6 \cdot 10^{-4} \pm 2.1 \cdot 10^{-5}$	12.2
SKI	0.22	$8.0 \cdot 10^{-4} \pm 1.2 \cdot 10^{-4}$	5.74	0.20	$1.6 \cdot 10^{-3} \pm 1.7 \cdot 10^{-4}$	18.2
Inducing inputs	0.22	$1.0 \cdot 10^{-2} \pm 7.4 \cdot 10^{-4}$	3.81	0.20	$2.1 \cdot 10^{-1} \pm 1.6 \cdot 10^{-3}$	12.2
Hilbert space	0.22	$9.5 \cdot 10^{-3} \pm 5.1 \cdot 10^{-4}$	3.80	0.20	$2.2 \cdot 10^{-1} \pm 4.6 \cdot 10^{-3}$	12.2

5.5 Results

In this section, we first compare the performance of our method with existing approaches on two low-dimensional, spatially large benchmark data sets. As the choice of kernel hyper-parameters affects the trade-off between computational complexity and the approximation accuracy, to ensure a fair comparison, we consider the same hyperparameters that are used by previous work for these data sets. See [18, 104] for the audio dataset, and [18, 123] for the precipitation dataset. Then, we measure how long it takes for our method to make online predictions using a short length scale and millions of basis functions on a bathymetry dataset that is too large for existing approaches given our hardware constraints. For the experiments we perform on the audio benchmark, like other approaches that use this dataset as a benchmark to investigate spatial scalability [18, 104], we treat the temporal axis as a spatial axis.

As the data sets considered in Sections 5.5.2 and 5.5.3 are geospatial data with a non-zero mean value, the average of the output is subtracted before training. This average is subsequently added to each prediction. All computation times reported are measured while running on a Dell XPS 15 9560 laptop, with 16 GB RAM and an Intel Core i7-7700HQ CPU running at 2.80 GHz. In all experiments we set $r = 2r^\star$, as picking $r >= 2r^\star$ gives that the expression for $P_\star^{-1}$ reduces to $P_\star^{-1} = K(u_{S^\star}, u_{S^\star})$, as derived in Appendix 5.A. This means that we have a closed-form expression for the inverse of the prior covariance which does not rely upon computing any numerical inverses, further reducing the necessary computational efforts to make each prediction. All predictions are made using the squared exponential kernel. All code and files required to reproduce the results can be found

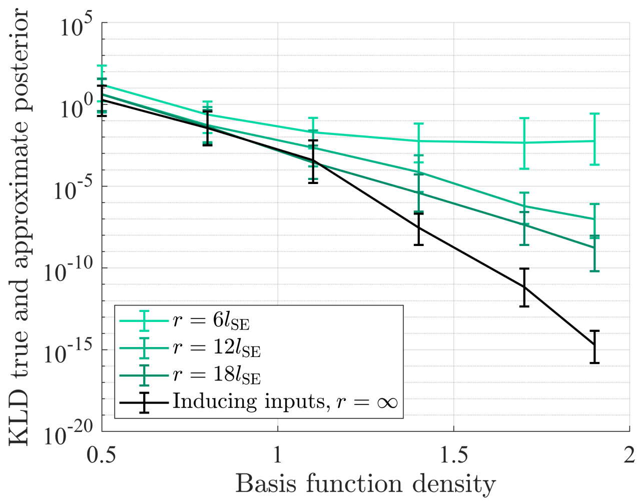


Figure 5.4: KL divergence (KLD) between the full GP posterior, and approximations with various local domain sizes r, trained on the audio dataset. The error bars indicate the average deviation above and below the mean, respectively, after 100 repeated experiments with 100 randomly sampled measurements from the training set.

on https://github.com/fridaviset/FastGPMapping.

5.5.1 One-dimensional sound map

A dataset describing natural sound was published by [124], and subsequently used by [104] and [18] to demonstrate the ability of their approximations to learn a large dataset with a large input domain relative to the size of the spatial variations in the field. This section describes various experiments that all make a one-dimensional GP map from the time axis to the amplitude of the sound wave.

We illustrate multiple properties of our approximation using this dataset. Firstly, we show on the audio dataset that our approximation converges to the inducing point approximation as the radius of the local domain increases. Secondly, we show that our approach converges to the GP posterior as both the radius of the local domain and the basis function density increase. Thirdly, we show that our approach has a lower online computation time compared to both SKI, inducing points, and Hilbert space methods on the full dataset while matching the prediction accuracy measured by SMAE.

To evaluate the accuracy and required computation time on the same dataset with the same GP prior as [18], we use the squared exponential kernel in equation 5.2 with hyperparameters set to $\sigma_{\text{SE}} = 0.009$, $l_{\text{SE}} = 10.895$, $\sigma_{\text{y}} = 0.002$. The dataset is one-dimensional and contains a training set of 59 309 measurements of sound amplitude collected at a known input time and a fixed test set of 691 points.

In Fig. 5.4 we investigate how high the grid density has to be for the basis function approximation described in section 5.2.2 using the basis functions described in section 5.3.1 with $r = \infty$ and for our proposed method in sections 5.3.2 and 5.3.3 with various local domain sizes r. We assess our results in terms of the KL divergence between the approximate inference and the full GP posterior. This is a measure that can attain values between 0 and ∞

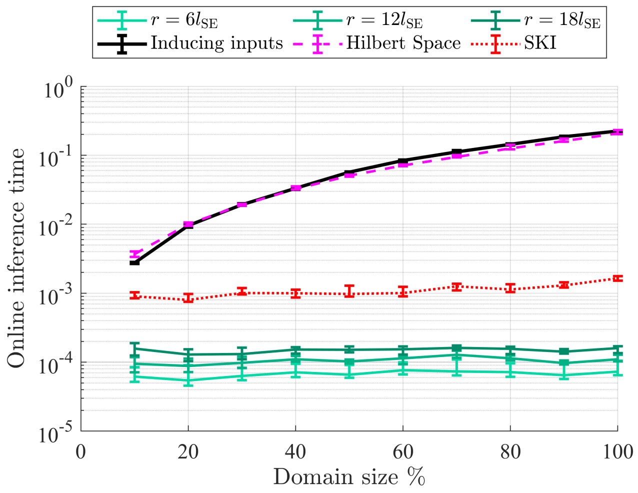


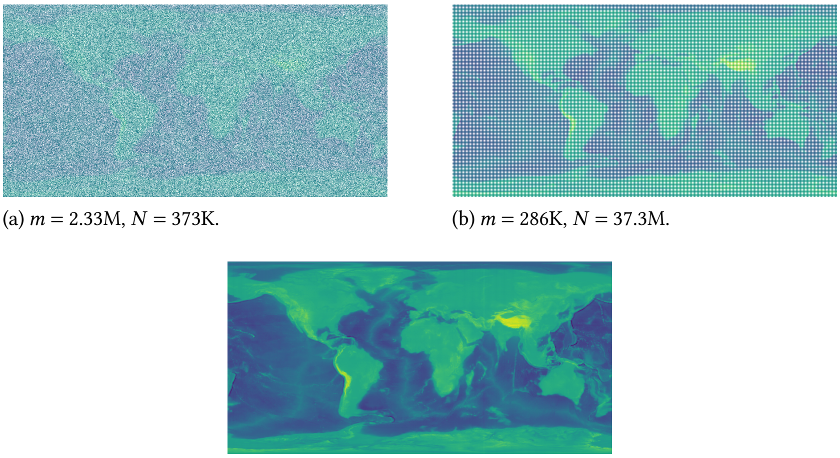
Figure 5.5: Online inference time on audio dataset for a growing domain size for our proposed method with various local domain sizes r, compared to the inducing input approximation using inducing inputs on a grid, the Hilbert space basis function approximation, and SKI. All methods were run using the same amount of basis functions. The error bars indicate the average deviation above and below the mean after 100 repeated experiments, respectively.

that compares how similar distributions are and a lower value means that the distributions are more similar. For all the investigated approximations, a higher basis functions density (measured in the number of basis functions per lengthscale) gives a lower KL divergence between the approximate inference and the full GP posterior. The results demonstrate that the KL divergence between the full GP posterior and our approximation reduces with increasing parameter choice r.

To measure how long it takes for our approach to perform online mapping, we measure the time it takes to include one additional measurement and perform one prediction step. In Fig. 5.5, we compare our proposed method to the inducing input solution implemented with an online Kalman filter [109], an online Kalman filter implementation with Hilbert space basis functions [11], and an online implementation of SKI [18], for increasing domain sizes. As the domain size increases, we keep the basis function density constant to retain the same approximation accuracy. The amount of basis functions m is therefore increasing linearly. As the online inference time of SKI is affected by how many iterations are used in the conjugate gradient solver to improve the approximation accuracy, we measure the run-time using just one iteration in the solver. To compare the prediction accuracy of SKI to our approach in Table 5.3, however, we use the exact solution that the solver is approximating. We can therefore conclude that an online evaluation of our algorithm is faster than competing approaches while being able to recover the same or better SMAE (standardized mean average error) and MSLL (mean squared log loss) scores (see Table 5.3). The online computational complexity of both Hilbert space basis functions and inducing inputs increases quadratically as the domain size increases. The computational complexity of our approach is bounded by $O(m'^3)$ independent of the increase in the domain size, which is why our online computational complexity remains lower than 10^{-3} seconds independent

Table 5.4: SMSE accuracies of daily precipitation level predictions. The predictions are obtained with a local domain with size $r^\star = 3l_{\text{SE}}$, which corresponds to using at most 144 basis functions in each local prediction for the highest $m = 800K$.

N	full GP	Inducing inputs			Local information filter		
10 000	0.823	0.957	0.905	0.957	0.906	0.824	0.823
20 000	0.766	0.946	0.861	0.947	0.862	0.770	0.766
100 000	N/A	0.907	0.782	0.907	0.786	0.561	0.545
528 474	N/A	0.894	0.746	0.895	0.751	0.468	0.435


(a) $m = 2.33\text{M}$, $N = 373\text{K}$.

(b) $m = 286\text{K}$, $N = 37.3\text{M}$.

(c) $m = 2.33\text{M}$, $N = 37.3\text{M}$.

Figure 5.6: Bathymetry dataset reconstruction with GP regression. The color corresponds to the posterior predicted elevation of the earth surface both above and below the sea, and the opacity is inversely proportional to the variance of the approximate GP prediction in each location.

of the growth of the domain size.

5.5.2 Mapping daily precipitation levels

In this section, we compare the prediction accuracy and computation time of our mapping approach to alternative methods on a large geo-spatial dataset which is used as a benchmark dataset for evaluation accuracy and computation time by [18], and [104].

The precipitation dataset contains 528 474 measurements of daily precipitations from the US in the training set, and 100 000 measurements in the test set [18]. The input dimensions are latitude, longitude, and time. We use the squared exponential kernel using the same hyperparameters as [18]. These hyperparameters are $\sigma_{\text{SE}} = 3.99$, $\sigma_{\text{y}} = 2.789$ and $l_{\text{SE}} = [3.094, 2.030, 0.189]$, where the three lengthscales l_{SE} apply to each of the three dimensions, respectively. The results are displayed in Table 5.4.

In Table 5.4, the standardized mean squared error (SMSE) of our approach with $r^\star = 3.5l_{\text{SE}}$ is compared to the full GP prediction, and to the inducing input approximation with basis functions placed on the same grid. The SMSE of our approach almost matches the inducing input approximation with the same number of basis functions. Using 800 000 basis functions, it matches the GP prediction accuracy with the same number of measurements. Using all the measurements and 800 000 basis functions gives the highest prediction accuracy, which is a combination that is computationally infeasible given our hardware constraints for both the full GP regression and the inducing input approximation to give a prediction. However, our proposed method has an online training time of 0.017 seconds per measurement and 0.016 seconds per prediction.

5.5.3 Global Bathymetry Field Mapping

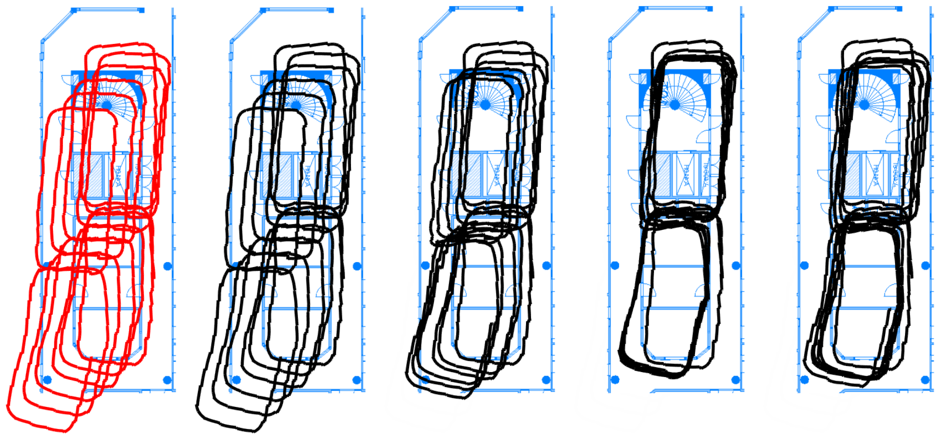
To investigate the time required for our method to include a new measurement and make a prediction in a large geospatial field with fine-scale variations, we run it on a dataset containing values of the height difference with respect to sea level across the globe [125]. The input domain of this data is huge compared to the scale of the spatial variations, and the data is therefore challenging to train on using state-of-the-art methods. We retrieve 37.5 million depth values from the database, and we consider the latitude and longitude as input locations.

We test our approach with the hyperparameters σ_{SE}^2 equal to the variance of the 37.5 million measurements, $\sigma_{\mathrm{y}} = 0.1\sigma_{\mathrm{SE}}$, and a lengthscale $l_{\mathrm{SE}} = 0.16$ degrees (which corresponds to 18.3 km on the equator). We use a local subset contained within $r^\star = 3l_{\mathrm{SE}}$ for our prediction-point dependent local approximations to the GP posterior mean. We perform predictions using our proposed algorithm for three cases: one where we use 2.33 million basis functions and 373 thousand measurements, one where we use 268 thousand basis functions and 37.3 million measurements, and one where we use both 2.33 million basis functions and 37.3 million measurements.

The result in Fig. 5.6 shows the learned bathymetry map using 10% of the measurements, and 2.33 million basis functions (Fig. 5.6a), and the learned map using 100% of the measurements and 268 thousand measurements (Fig. 5.6b) with a map learned with 100% of the measurements and a dense grid of 2.33 million basis functions (Fig. 5.6c). For the results in Fig. 5.6c, including each new measurement takes $3.7 \times 10^{-4} \pm 0.12 \times 10^{-4}$ seconds. Each prediction takes 0.0097 ± 0.017 seconds. These results demonstrate that our proposed approach can remain computationally feasible in cases where the measurement density is high, and the map area is large relative to the length scale of the spatial variations. For that particular usecase, our proposed approach does not have the disadvantage of discarding measurements which is visible in Fig. 5.6a, which is that some areas can be far from any measurements relative to the lengthscale of the inference, meaning that the prediction in some areas is less certain than in others. Similarly, our proposed approach does not have the disadvantage of using less tight grids of inducing points, which is visible in Fig. 5.6b, where the prediction points far from inducing point locations are less certain than the ones close to the inducing point locations.

5.5.4 Using local information filter for faster Mag-SLAM

To experimentally compare the local information filter with Hilbert space basis functions in EKF Mag-SLAM, we apply both algorithms to a dataset from a foot-mounted sensor that was collected by [55] and subsequently used in [74] to demonstrate the drift-compensating



(a) Odometry

(b) Our approach, $r = 1.5l_{SE}$, 2.8 ± 0.85 ms

(c) Our approach, $r = 2l_{SE}$ 4.2 ± 1.1 ms

(d) Our approach, $r = 2.5l_{SE}$ 5.6 ± 1.3 ms

(e) Baseline, 26 ± 4.6 ms

Figure 5.7: Trajectory estimates for indoor pedestrian walking laps in a hallway using only foot-mounted odometry, EKF-SLAM with the local information filter with various sizes of the local domain determined by r and EKF-SLAM with Hilbert space basis function (baseline). The average computation time in milliseconds for one iteration of each filter (dynamic update + measurement update) is written below each sub-Fig..

abilities of EKF Mag-SLAM. We replace the Hilbert space basis function approximation with the local basis function approximation in EKF Mag-SLAM as described in section 5.3.4, and measure the average computation time required to make each iteration of the algorithm on the same laptop computer. We compare this with the average computation time required to make each iteration of the original EKF Mag-SLAM using Hilbert space basis functions.

In Fig. 5.7a, the odometry obtained by using only accelerometer and gyroscope measurements from a foot-mounted sensor using the algorithm in [126] is displayed. In Fig. 5.7e, the estimated trajectory using the EKF with Hilbert space basis functions as in [74] is displayed. All estimates are overlayed the floorplan of the building where the measurements were collected. This gives a rudimentary means of evaluating the position estimation accuracy, since the subject was repeatedly walking through the same hallways in a repeated pattern 8-motion. The drift in the odometry in Fig. 5.7a therefore shows up as a slow displacement of the position estimate away from the hallway the pedestrian was walking in. In contrast, the estimates obtained from our approximate mapping in Fig. 5.7d and from the baseline EKF Mag-SLAM with Hilbert space basis functions in Fig. 5.7e compensate for the drift of the estimates from the gyroscope and accelerometer. The notable difference between our approach and the approximation using Hilbert space basis functions, is that the Hilbert space basis functions require 26 ± 4.6 ms to run at each iteration, while our equally accurate algorithm require 5.6 ± 1.3 ms to run at each iteration.

5.6 Conclusion

To improve indoor position estimates online with few computational resources, we have presented an efficient online mapping technique that gives an approximation to the GP posterior requiring a number of computations that neither scales with the temporal duration of the mapping, nor with the spatial extend of the map. The storage requirements of our presented mapping algorithm scales linearly with the spatial extent of the map, and does not scale with the temporal duration of the mapping. We have also demonstrated the ability of our mapping algorithm to match the accuracy of previously proposed approximations on benchmark datasets using a lower computation time. It is possible to replace a previously proposed GP approximation in an EKF for Mag-SLAM. Our experiments on data from a foot-mounted sensor using our proposed method achieve the same prediction accuracy as Mag-SLAM with a previously proposed mapping approximation using a shorter computation time on the same computer. Future work could investigate the existence of theoretical bounds on the approximation errors, or investigate ways to incorporate the mapping technique for SLAM in different geospatial fields, and for other applications such as navigation of robots or vehicles.

5.A Derivation of prior covariance for the prediction-point dependent basis function approximation

The prior for the parametric approximation to the GP regression using the local subset of basis functions is given by

$$\tilde{f}^\star = \Phi_{\mathcal{S}^\star}^\top w_{\mathcal{S}^\star}, \qquad w_{\mathcal{S}^\star} \sim \mathcal{N}(0, P_\star), \tag{5.20}$$

where $P_\star$ is the prior covariance on the local weights. The condition that the prior should be recovered in the center locations of the basis functions is given as

$$p(\tilde{f}^\star(u_{\mathcal{S}^\star})) = p(f(u_{\mathcal{S}^\star}))). \tag{5.21}$$

As both these are normal distributions with mean 0, and difference covariances, this conditions hold if and only if the two covariances are equal according to

$$\Phi_{\mathcal{S}^\star}(u_{\mathcal{S}^\star})^\top P_\star \Phi_{\mathcal{S}^\star}(u_{\mathcal{S}^\star}) = K(u_{\mathcal{S}^\star}, u_{\mathcal{S}^\star}). \tag{5.22}$$

This gives the closed-form expression for $P_\star$ given by

$$P_\star = (\Phi_{\mathcal{S}^\star}(u_{\mathcal{S}^\star})^\top)^{-1} K(u_{\mathcal{S}^\star}, u_{\mathcal{S}^\star}) \Phi_{\mathcal{S}^\star}(u_{\mathcal{S}^\star})^{-1}, \tag{5.23}$$

in the case where $(\Phi_{\mathcal{S}^\star}(u_{\mathcal{S}^\star})^\top)$ and $\Phi_{\mathcal{S}^\star}(u_{\mathcal{S}^\star})$ are invertible. The entry in row i and column j of the matrix $\Phi_{\mathcal{S}^\star}(u_{\mathcal{S}^\star})$ is defined as $\phi_i(u_j)$. Plugged into the definition in (6), this gives

$$\phi_i(u_j) = \begin{cases} \kappa(u_i, u_j), & \|u_i - u_j\|_\infty \leq r \\ 0, & \|u_i - u_j\|_\infty > r \end{cases} \tag{5.24}$$

For all inducing input pairs i, j in the set $\mathcal{S}(x^\star, r^\star)$, from the definition of the set, it holds that $\|u_i - u_j\|_\infty \leq 2r^\star$. If $r \geq 2r^\star$, the condition $\|u_i - u_j\|_\infty \leq r$ holds for all $i, j \in \mathcal{S}^\star$. This again means that $\phi_i(u_j) = \kappa(u_i, u_j)$ for all $i, j \in \mathcal{S}^\star$, which means that $\Phi_{\mathcal{S}^\star}(u_{\mathcal{S}^\star}) = K(u_{\mathcal{S}^\star}, u_{\mathcal{S}^\star})$. Inserting this result into (5.22) gives $P_\star = K(u_{\mathcal{S}^\star}, u_{\mathcal{S}^\star})^{-1}$.

6 Online discovery of global patterns and local variations in magnetic fields using Gaussian process regression

Variations in the magnetic field can, if accurately mapped, be used as a reliable information source for indoor localization. However, the creation of high-fidelity magnetic field maps is a time-consuming and costly process making it challenging to map large areas. In many indoor environments, the variations in the magnetic field have partially repeated patterns induced by repeated structural components in the buildings. These patterns can be utilized to create map models with better extrapolation accuracy, thus requiring smaller areas to be mapped and reducing the time and cost associated with the map creation. Therefore, an algorithm for online learning of both local variations and globally repeated patterns in magnetic fields is proposed. The algorithm is based on a Gaussian process model that uses a novel curl-free pattern-discovery kernel; the graphical abstract to the right illustrates the kind of globally repeating patterns that can be learned online using the proposed model. The proposed pattern-discovery kernel is exactly equivalent to a parametric model. Observing this fact enables online learning of the global patterns without further approximation, unlike existing local methods that require parametric approximation to enable online learning. The functionality and properties of the proposed algorithm are showcased via three experiments. They show that the proposed kernel has equal interpolation accuracy compared to existing kernels, and improved extrapolation properties in the case of globally existing patterns. They also show that if there are no significant globally repeated patterns, the proposed kernel has equal performance with existing methods.

This chapter is based on *Frida Viset, Anton Kullberg, Gustaf Hendeby, Rudy Helmons, Isaac Skog and Manon Kok*: Online discovery of global patterns and local variations in magnetic fields using Gaussian process regression. Under review.

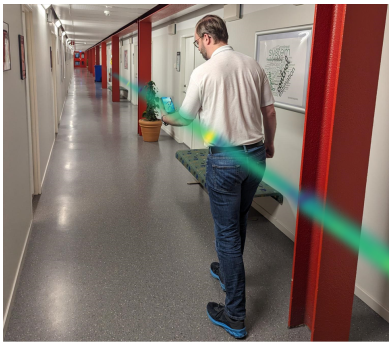
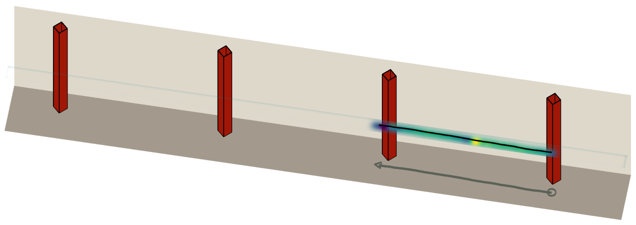


Figure 6.1: Collecting measurements of magnetic field along hallway in university building with a smartphone. The learned magnetic field measurements using the pattern-discovery kernel are displayed. The color corresponds to the learned magnetic field strength, and the opacity corresponds to the confidence of the estimate. As the metallic columns create a repetitive magnetic field pattern, our online pattern-detection algorithm is able to predict the magnetic field variations not only in the walked trajectory but also further down the hall.

6.1 Introduction

In indoor environments, magnetometer readings are typically given not only by the earth's constant magnetic field but also by spatial variations [12]. The variations arise from repeated structural components in the buildings, as well as randomly placed metallic furniture and objects [29]. An example of the magnetic field in the path of a pedestrian holding a cellular device with a magnetometer is displayed in figure 6.1. We propose an online algorithm for constructing a magnetic field map based on magnetometer measurements that not only interpolates the map in a visited area, but is also able to extrapolate to unseen areas based on expected repetitions in the magnetic field pattern, as demonstrated in figure 6.2. In recent years, a wide range of research has investigated the online creation and use of magnetic anomaly field maps to enable meaningful online interpretation of magnetometer readings in indoor robots, pedestrians, and vehicles, and for GNSS-denied navigation and mapping [7, 8, 14, 50, 72, 74, 94, 127–129]. High-accuracy magnetic field maps can also be used to improve magnetometer calibration, as demonstrated by [130]. In order to accomplish these tasks, GP regression with curl-free kernels has been widely used, because it gives a smooth interpolation between measurement points and an inherent quantification of the uncertainty of the learned magnetic field map [8, 12, 14, 74]. To enable online magnetic field mapping, all of these works use parametric approximations of the local variations that can be incorporated into any state-space model.

On floor level, the magnetic field variations are typically caused by reinforcing steel structures, which generally have an intricate structure, creating complex and chaotic distortions in the magnetic field [131]. Larger structural steel elements such as beams or reinforced columns however reoccur at predictable locations, which can give rise to variations with a repetitive pattern across larger areas. Despite this, previous work has not yet considered exploiting these patterns, which may improve the model's inherent ability to extrapolate the magnetic field map to unvisited regions. To fill this gap, we propose an algorithm for online curl-free magnetic field mapping to capture both globally repeated patterns while retaining the ability to capture local variations. This is achieved by encoding


(a) An existing algorithm using the SE kernel can not extrapolate outside, but interpolates well within, the visited region.

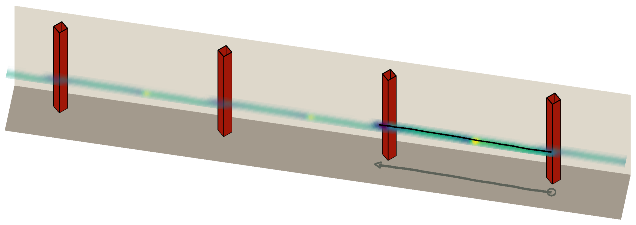

(b) Our proposed PD+SE kernel can extrapolate to unseen data regions while retaining interpolation properties.

Figure 6.2: Predicted magnetic field in a hallway of a university building. The gray line shows the measured trajectory of the pedestrian in figure 6.1 at time *t*, and the black line shows the measured trajectory of the phone at the same timestep. The predicted magnetic field norm at time *t* is illustrated by the colored surface. The opacity is proportional to the confidence of the magnetic field estimate.

this information in a GP kernel, which allows us to use magnetic field measurements from a finite region to extrapolate to unseen areas without ever visiting them, see figure 6.2b. We encode this information by adapting the kernel to be more expressive. To be precise, to encode the global patterns, we propose a curl-free and pattern-discovering kernel by considering a Fourier decomposition of any kernel in a finite, but repeated domain. We combine this kernel with a standard SE kernel to capture both local variations through the SE component as well as globally repeated patterns through the PD kernel.

To enable online magnetic field mapping with this combined kernel, we represent it using a parametric model, similar to previous works. Since the proposed kernel is a sum kernel, we approximate the SE kernel using the same parametric model for curl-free fields with local variations as proposed by [8]. See figure 6.2a for an example of how the SE kernel can be used to interpolate based on previous measurements. We then demonstrate that the PD kernel can be *exactly* rewritten to a parametric model, without approximation. By combining these two parametric representations in a single model, we enable online magnetic field mapping with excellent extrapolation properties while retaining the interpolation properties of previous works.

Our contributions are thus two-fold: (i) an *exact* parametric formulation of a PD kernel on finite, connected and repeated domains (ii) an online algorithm for learning globally repeated patterns of the magnetic field while still allowing for local variations. The paper is organized as follows: section 6.2 discusses preliminaries on GPs and the incorporation of the curl-free constraint. In section 6.3, we first present our approach to including both local variations as well as globally repeated patterns. Then, we show how the resulting GP can be approximated as a basis function expansion and detail how this results in an online algorithm for magnetic field mapping. section 6.4 presents our three experiments and showcases the capabilities of our proposed model. Lastly, section 6.5 discusses related work, section 6.6 discusses the implications of the proposed method, and in section 6.7 the conclusions are stated.

6.2 Preliminaries

6.2.1 Gaussian Processes

A Gaussian process (GP) is a collection of random variables, any finite number of which have a joint Gaussian distribution [16]. Formally, we denote a zero-mean GP by

$$f \sim \mathcal{GP}(0, \kappa(\cdot,\cdot)), \tag{6.1}$$

where $\kappa : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$ is the kernel, where, e.g., $\kappa(\boldsymbol{x}, \boldsymbol{x}')$ represents the covariance between any two pair of inputs $\boldsymbol{x}$ and $\boldsymbol{x}'$. The standard conjugate GP assumes independent observations of the latent process values, i.e.,

$$\boldsymbol{y} \sim \mathcal{N}(\boldsymbol{f}, \sigma_y^2 \boldsymbol{I}),$$

where $\boldsymbol{f} = \begin{bmatrix} f(\boldsymbol{x}_1) & \cdots & f(\boldsymbol{x}_N) \end{bmatrix}^\top$ is a vector of function values at the input locations $\{\boldsymbol{x}_i\}_{i=1}^N$. The posterior distribution of the GP is then given by $p(\boldsymbol{f}^* | \boldsymbol{y}) = \mathcal{N}(\boldsymbol{\mu}^*, \boldsymbol{V}^*)$ with [16]

$$\boldsymbol{\mu}^* = \boldsymbol{K}_{*f}(\boldsymbol{K}_{ff} + \sigma_y^2 \boldsymbol{I})^{-1} \boldsymbol{y} \tag{6.2a}$$

$$\boldsymbol{V}^* = \boldsymbol{K}_{**} - \boldsymbol{K}_{*f}(\boldsymbol{K}_{ff} + \sigma_y^2 \boldsymbol{I})^{-1} \boldsymbol{K}_{f*}, \tag{6.2b}$$

where $[\boldsymbol{K}_{ab}]_{ij} = \kappa([\boldsymbol{X}^a]_i, [\boldsymbol{X}^b]_j)$ is the kernel matrix for the inputs in $\boldsymbol{X}^a$ and $\boldsymbol{X}^b$. Thus, e.g., $[\boldsymbol{K}_{*f}]_{ij} = \kappa([\boldsymbol{X}^*]_i, [\boldsymbol{X}^f]_j)$ where $\boldsymbol{X}^*$ and $\boldsymbol{X}^f$ are the collection of test and training inputs, respectively. The kernel $\kappa(\cdot,\cdot)$ can generally be chosen freely, as long as the resulting covariance matrix $\boldsymbol{K}_{ff}$ is positive semi-definite and thus constitutes a valid covariance matrix. The *de facto* standard choice is the squared exponential (SE) kernel given by

$$\kappa_{\text{SE}}(\boldsymbol{\tau}) \triangleq \kappa_{\text{SE}}(\boldsymbol{x}, \boldsymbol{x}') = \sigma_f^2 \exp\left(-\frac{\|\boldsymbol{\tau}\|_2^2}{2l^2}\right), \tag{6.3}$$

where $\boldsymbol{\tau} = \boldsymbol{x} - \boldsymbol{x}'$ and $\|\cdot\|_2^2$ is the squared 2-norm. Further, l is commonly referred to as the characteristic lengthscale and σ_f^2 describes the signal variance. Generally, the SE kernel promotes very smooth functions that have excellent interpolation properties, but extrapolate poorly.

6.2.2 Incorporating Curl-free Constraints

When there is no free current in the space, the magnetic field is curl-free [15], i.e., the nonlinear field $\boldsymbol{f} : \mathbb{R}^3 \to \mathbb{R}^3$ describing the magnetic field fulfills

$$\nabla \times \boldsymbol{f} = \boldsymbol{0}. \tag{6.4}$$

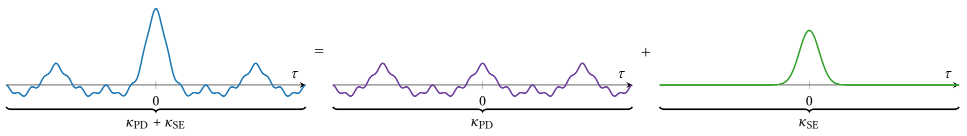


Figure 6.3: A one dimensional illustration of our stationary kernel, expressed as the sum of a pattern detection kernel with fixed interval frequencies and an SE kernel.

Note that any curl-free field can be expressed as the gradient of a potential field. It is thus possible to give an equivalent expression for the curl-free field by introducing a potential field $\varphi : \mathbb{R}^3 \to \mathbb{R}$. If the potential field is distributed according to the prior $\varphi \sim \mathcal{GP}(0, \kappa(\cdot,\cdot))$ and f is defined as

$$f(x) = \nabla_x \varphi(x), \tag{6.5}$$

the resulting GP $f(x)$ is curl-free by construction [15]. Since ∇ is a linear operator, the distribution of f is a linear transformation of a zero-mean GP with covariance [15]

$$E[f(x)f(x')] = \nabla_x \nabla_{x'} \kappa(x, x'). \tag{6.6}$$

6.3 Method

We consider the problem of identifying a model of the magnetic field in a structured environment using a GP. The environment is assumed to contain no free current, implying that the magnetic field is curl-free [15], and we will thus use the construction in equation (6.5) to define our GP. The environment is further assumed to contain both global repetitive structural elements as well as non-repetitive elements inducing local variations in the magnetic field. To learn the globally repeating patterns and the local disturbances, we combine a version of the pattern detection (PD) kernel proposed by [1], with an SE kernel. To enable online learning of the magnetic field, we rewrite the GP as a parametric model. The SE kernel is represented as a parametric model using a basis function approximation as in [11]. We then show that the PD kernel is exactly equivalent to a parametric model without further approximation. Finally, we combine the two parametric models into one joint state-space, resulting in an online algorithm for learning both local variations and globally repeating patterns.

6.3.1 Modeling Curl-free Local and Repeating Patterns

To preserve the curl-free property of the magnetic field, it is modeled by a potential field equation (6.5) and thus measures the gradient of the potential field along the x, y, and z directions, respectively. The observation model is given by

$$y_t = \nabla_x \varphi(x_t) + e_t, \quad e_t \sim \mathcal{N}(0, \sigma_y^2 I_3). \tag{6.7}$$

The vector y_t thus contains the magnetic field strength in the x, y, and z direction respectively, from here on denoted by $y_t = [y_{x,t}, y_{z,t}, y_{z,t}]^\top$. In order to capture both local variations and globally repeating patterns, the potential field is in turn modeled as

$$\varphi \sim \mathcal{GP}(0, \kappa_{PD}(\cdot,\cdot) + \kappa_{SE}(\cdot,\cdot)). \tag{6.8}$$

6

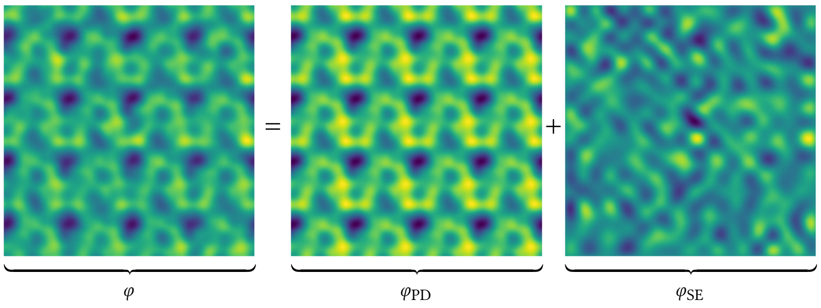


Figure 6.4: A simulated realization of a potential field from our proposed kernel, expressed as the sum of a PD kernel with fixed interval frequencies and an SE kernel. There are both distinct repeated patterns as well as local variations in the resulting potential field.

Here, the kernel κ_{SE} is given by equation (6.3), and the kernel κ_{PD} is a flexible pattern discovering kernel that will be introduced in more detail later. The shape of the SE kernel is illustrated in green to the right in figure 6.3. The shape of the PD kernel can change depending on the discovered pattern. An example of a PD kernel is illustrated in purple in figure 6.3. We consider the sum of these two kernels for modeling our magnetic field, and the resulting kernel is illustrated in blue in figure 6.3.

This construction enables extrapolation to unseen input regions through the PD kernel, but still retains the abilities of the SE kernel to capture local, non-repetitive variations. A potential function sampled from such a kernel is visualized in figure 6.4. Note that additive GP kernels correspond to an additive GP model and we can thus rewrite equation (6.8) as

$$\varphi = \varphi_{\text{PD}} + \varphi_{\text{SE}} \tag{6.9a}$$
$$\varphi_{\text{PD}} \sim \mathcal{GP}(0, \kappa_{\text{PD}}(\cdot,\cdot)) \tag{6.9b}$$
$$\varphi_{\text{SE}} \sim \mathcal{GP}(0, \kappa_{\text{SE}}(\cdot,\cdot)). \tag{6.9c}$$

To enable online learning, the two GPs in equation (6.9) are represented as basis function expansions with two different sets of basis functions, one representing the PD GP equation (6.9b) and another for the SE GP equation (6.9c), i.e.,

$$\varphi(\boldsymbol{x}) = \underbrace{\begin{bmatrix}\Phi_{\text{PD}}(\boldsymbol{x}) & \Phi_{\text{SE}}(\boldsymbol{x})\end{bmatrix}}_{\triangleq \Phi(\boldsymbol{x})} \underbrace{\begin{bmatrix}\boldsymbol{w}_{\text{PD}} \\ \boldsymbol{w}_{\text{SE}}\end{bmatrix}}_{\triangleq \boldsymbol{w}}, \quad \boldsymbol{w} \sim \mathcal{N}(0, \boldsymbol{\Lambda}), \tag{6.10}$$

where $\boldsymbol{\Lambda} = \text{diag}\begin{bmatrix}\Lambda_{\text{PD}} & \Lambda_{\text{SE}}\end{bmatrix}$. The SE GP is approximated using the Hilber-space Gaussian process (HGP) from [11], and the PD GP uses an exact parametric reformulation. These two parametric formulations are described in the following two sections.

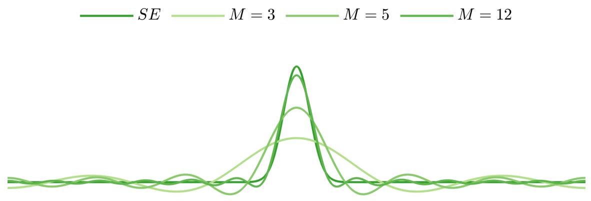


Figure 6.5: A basis function approximation to SE kernel constitutes a parametric approximation to a kernel encoding high short-range correlations and low long-range correlations. More basis functions give a more accurate approximation.

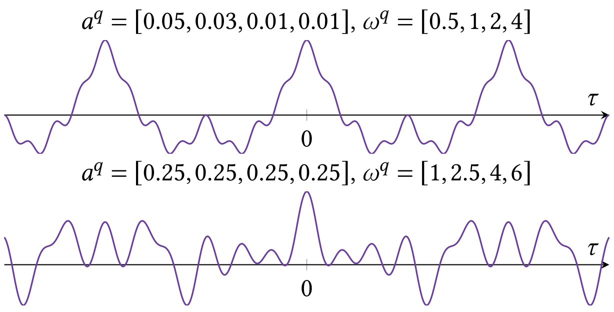


Figure 6.6: Examples of PD kernels in 1D with $Q = 4$ components and different hyperparameters. It is clearly capable of representing very distinctive patterns. The PD kernel does not require approximation with basis functions, as it can be exactly rewritten to a parametric model.

6.3.2 Parametric Approximation to the SE Kernel

To approximate the SE kernel with a parametric model, we use the HGP [11], where the GP prior is approximated with a reduced-rank basis function expansion

$$\varphi_{\text{SE}}(\boldsymbol{x}) = \Phi_{\text{SE}}(\boldsymbol{x})\boldsymbol{w}_{\text{SE}}, \qquad \boldsymbol{w}_{\text{SE}} \sim \mathcal{N}(0, \Lambda_{\text{SE}}), \tag{6.11}$$

where $\Phi_{\text{SE}}(\boldsymbol{x})$ and Λ_{SE} are found by solving a particular eigenvalue problem, see [11] for more details. The basis functions $\Phi_{\text{SE}}(\boldsymbol{x})$ are defined in section 6.A.1. The prior covariance Λ_{SE} is diagonal and is in principle given by the spectral density of the SE kernel, see [11] for details. The approximation becomes more efficient the higher the number of parameters M are included to approximate the SE kernel, as illustrated in figure 6.5.

6.3.3 Exactly Parametric Pattern Discovery Kernel

We propose a kernel that can discover repetitive patterns in a global domain. Like previous work into PD kernels [1], we propose a stationary product PD kernel. The kernel can be expressed through its spectral density according to the Wiener-Khintchine theorem [16, 17]

$$\kappa(\boldsymbol{\tau}) = \frac{1}{2\pi}\int_{-\infty}^{\infty} S(\boldsymbol{\omega})e^{i\omega\tau}d\omega, \tag{6.12}$$

where $S(\boldsymbol{\omega})$ is the spectral density of the kernel. In general, it is possible to represent any kernel by considering the spectral density as an infinite sum of Dirac functions at fixed frequencies sampled from the spectral density [17]. To obtain a flexible kernel that can span any stationary kernel, we consider the case where the spectral density can be represented exactly by Q weighted Dirac-delta functions as

$$S(\boldsymbol{\omega}) = \sum_{q=1}^{Q} a^q \delta(\boldsymbol{\omega} - \boldsymbol{\omega}^q). \tag{6.13}$$

Computing the closed-form solution to the integral in (6.12) with the spectral density in (6.13) results in

$$\kappa_{\text{PD}}(\boldsymbol{\tau}) = \sum_{q=1}^{Q} a^q \prod_{d=1}^{D} \cos(2\pi\tau_d \omega_d^q), \tag{6.14}$$

consisting of Q components, where each component consists of a Fourier feature along each dimension. This gives a kernel capable of encoding repeated patterns in space, as each kernel component itself repeats with regular intervals across each dimension, see figure 6.6 for examples of the kernel with different hyperparameters. The parameters a^q and ω_d^q are hyperparameters, allowing for the discovery of any Fourier components encoding spatially repeating patterns in the global domain. We provide some further intuition and discussion about the kernel equation (6.14) in sections 6.A.3 and 6.A.4.

The PD kernel equation (6.14) leads to a GP model that is *exactly* representable as a parametric model, which we formalize in the following proposition.

Proposition 6.3.1. *A GP prior with the kernel equation* (6.14) *can equivalently be represented by a parametric model with* 2Q *parameters, given by*

$$\varphi_{PD}(\boldsymbol{x}) = \Phi_{PD}(\boldsymbol{x})\boldsymbol{w}_{PD}, \qquad \boldsymbol{w}_{PD} \sim \mathcal{N}(0, \Lambda_{PD}), \tag{6.15}$$

where Φ_{PD} *is defined as*

$$\Phi_{PD}(\boldsymbol{x}) = \begin{bmatrix} \prod_{d=1}^{D} \phi_d^1(\boldsymbol{x}) \\ \vdots \\ \prod_{d=1}^{D} \phi_d^Q(\boldsymbol{x}) \end{bmatrix}, \phi_d^q(\boldsymbol{x}) = \begin{bmatrix} \cos(2\pi x_d \omega_d^q) \\ \sin(2\pi x_d \omega_d^q), \end{bmatrix} \tag{6.16}$$

Further, the block-diagonal matrix Λ_{PD} *is defined as*

$$\Lambda_{PD} = \text{diag}\begin{bmatrix} \Lambda_{PD}^1 & \cdots & \Lambda_{PD}^Q \end{bmatrix}, \tag{6.17}$$

with $\Lambda_{PD}^q = \text{diag}\begin{bmatrix} a^q & a^q \end{bmatrix}$.

Proof. See section 6.A.2. □

Similarly to the HGP, see section 6.3.2, the parametric model equation (6.15) allows us to perform online inference with a computational complexity that scales with the number of basis functions M and not with the number of measurements N.

6.3.4 Estimation

For both online and offline estimation, we assume access to N measurements, which can be stacked in a vector $\boldsymbol{y}_{1:N} \in \mathbb{R}^{3N}$, defined as

$$\boldsymbol{y}_{1:N} = \begin{bmatrix} \boldsymbol{y}_1^\top & \cdots & \boldsymbol{y}_N^\top \end{bmatrix}^\top, \tag{6.18}$$

measured in N locations $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$. This means that the observation model for all time steps can be expressed as

$$\boldsymbol{y}_{1:N} = \nabla_x \boldsymbol{\Phi}_{1:N} \boldsymbol{w} + \boldsymbol{e}_{1:N}, \quad \boldsymbol{e}_{1:N} \sim \mathcal{N}(\boldsymbol{0}, \sigma_{\mathrm{y}}^2 \boldsymbol{I}_{3N}) \tag{6.19}$$

where $\nabla_x \boldsymbol{\Phi}_{1:N} \in \mathbb{R}^{3N \times (M+Q)}$ is shorthand for the gradient of the basis functions evaluated in all measurement locations.

The learned magnetic field map based on the measurements $\boldsymbol{y}_{1:N}$ can then be expressed at locations $\boldsymbol{x}^*$ as

$$\boldsymbol{\mu}^* \triangleq \mathbb{E}[f^*] = \nabla_x \boldsymbol{\Phi}(\boldsymbol{x}^*) \hat{\boldsymbol{w}}_N \tag{6.20a}$$

$$\Sigma^* \triangleq \operatorname{cov}[f^*] = \nabla_x \boldsymbol{\Phi}(\boldsymbol{x}^*) \boldsymbol{P}_N [\nabla_x \boldsymbol{\Phi}(\boldsymbol{x}^*)]^\top. \tag{6.20b}$$

Here, $\hat{\boldsymbol{w}}_N$ are the posterior estimates of the model weights and $\boldsymbol{P}_N$ is the corresponding covariance given by

$$\hat{\boldsymbol{w}}_N = (\nabla_x \boldsymbol{\Phi}_{1:N}^\top \nabla_x \boldsymbol{\Phi}_{1:N} + \sigma_{\mathrm{y}}^2 \boldsymbol{\Lambda}^{-1})^{-1} \nabla_x \boldsymbol{\Phi}_{1:N}^\top \boldsymbol{y}_{1:N}, \tag{6.21a}$$

$$\boldsymbol{P}_N = \sigma_{\mathrm{y}}^2 (\nabla_x \boldsymbol{\Phi}(\boldsymbol{x})^\top \nabla_x \boldsymbol{\Phi}_{1:N} + \sigma_{\mathrm{y}}^2 \boldsymbol{\Lambda}^{-1})^{-1}. \tag{6.21b}$$

Estimation of the magnetic field map can also be performed online. This is done by initializing the estimate $\hat{\boldsymbol{w}}_t$ and covariance $\boldsymbol{P}_t$ of the magnetic field map according to the prior, $\hat{\boldsymbol{w}}_0 = 0, \boldsymbol{P}_0 = \boldsymbol{\Lambda}$, and then performing the recursion [8]

$$\boldsymbol{H}_t = \nabla_x \boldsymbol{\Phi}(\boldsymbol{x}_t) \tag{6.22a}$$

$$\boldsymbol{S}_t = (\sigma_{\mathrm{y}}^2 \boldsymbol{I} + \boldsymbol{H}_t^\top \boldsymbol{P}_t \boldsymbol{H}_t) \tag{6.22b}$$

$$\boldsymbol{K}_t = \boldsymbol{P}_t \boldsymbol{H}_t \boldsymbol{S}_t^{-1} \tag{6.22c}$$

$$\hat{\boldsymbol{w}}_{t+1} = \hat{\boldsymbol{w}}_t + \boldsymbol{K}_t (\boldsymbol{y}_t - \nabla_x \boldsymbol{\Phi}(\boldsymbol{x}_t) \hat{\boldsymbol{w}}_t) \tag{6.22d}$$

$$\boldsymbol{P}_{t+1} = \boldsymbol{P}_t - \boldsymbol{K}_t \boldsymbol{S}_t \boldsymbol{K}_t^\top \tag{6.22e}$$

Since our basis $\nabla_x \boldsymbol{\Phi}$ contains both SE and PD components, this enables not only interpolation but also extrapolation to unseen parts of the magnetic field online. Note, however, that this assumes that the hyperparameters of the two kernels $\kappa_{\mathrm{SE}}(\cdot,\cdot)$ and $\kappa_{\mathrm{PD}}(\cdot,\cdot)$, are known a priori. In our experiments, we perform offline hyperparameter optimization on a subset of the data and then proceed with online estimation.

6.3.5 Hyperparameter Optimization

The posterior given by equation (6.20) depends on the hyperparameters of the kernel $\kappa(\cdot,\cdot)$ through the posterior over the weights equation (6.21). In GPs, these hyperparameters are typically found through optimizing the maximum log likelihood (MLL) [16], which in our case is given by

$$\mathcal{L}(\theta) = -\frac{1}{2}\left(N\log 2\pi + \log\left|\boldsymbol{S}\right| + \boldsymbol{y}_{1:N}^{\top}\boldsymbol{S}^{-1}\boldsymbol{y}_{1:N}\right) \tag{6.23a}$$

$$\boldsymbol{S} \triangleq \sigma_{\mathrm{y}}^2\boldsymbol{I}_{3N} + \nabla\boldsymbol{\Phi}_{1:N}\boldsymbol{\Lambda}\nabla\boldsymbol{\Phi}_{1:N}^{\top} \tag{6.23b}$$

$$\log\left|\boldsymbol{S}\right| = (N-M)\log\sigma_{\mathrm{y}}^2 + \log|\boldsymbol{Z}| + \log|\boldsymbol{\Lambda}| \tag{6.23c}$$

$$\boldsymbol{Z} \triangleq \nabla\boldsymbol{\Phi}^{\top}\nabla\boldsymbol{\Phi} + \sigma_{\mathrm{y}}^2\boldsymbol{\Lambda}^{-1}. \tag{6.23d}$$

Here, $\theta = \{\{\{\omega_d^q\}_{q=1}^Q\}_{d=1}^3, \{a^q\}_{q=1}^Q, l, \sigma_{\mathrm{SE}}, \sigma_{\mathrm{y}}\}$ are the hyperparameters, and N and M are the number of data points and basis functions, respectively.

For the SE component, the hyperparameters enter in the prior on the weights, i.e., only in Λ_{SE}. This is exploited in the standard HGP to enable optimizing the MLL at cost $\mathcal{O}(M^3)$. However, for the periodic PD components we propose here, the hyperparameters enter both Λ_{PD} as well as the basis functions Φ_{PD} themselves. In particular, the mixture frequencies ω_d^q enter the basis functions whereas the mixture weights a^q are contained in Λ_{PD}, see theorem 6.3.1. Thus, if the mixture frequencies ω_d^q are to be optimized, the basis functions need to be recomputed each iteration of the MLL optimization, which costs $\mathcal{O}(NM^2)$ per iteration in general.

However, given prior knowledge about the environment, e.g., through maps or plans of the indoor environment, the frequencies ω_d^q can be determined a priori and need not be optimized. This can for instance be the locations of steel beams, or as we show in our experiments, the placement of columns in a parking basement.

Hence, we can opt to specify the frequencies manually and then solely optimize over the lengthscale l, variance σ_f^2, observation standard deviation σ_{y}, and mixture weights a^q. With this restriction, the optimization costs $\mathcal{O}(M^3)$ per iteration, a cost identical to the HGP.

6.4 Experiments

To demonstrate the proposed model, we consider three numerical experiments. The first experiment considers a real dataset of magnetic field disturbances collected in a corridor at Linköping University, Sweden. It demonstrates the online estimation of the magnetic field and the extrapolation properties one can expect. Secondly, we consider a simulated warehouse, with storage racks constituting the structurally repeated patterns, to show the extrapolation properties of the model in a structured environment. Thirdly, we consider a real dataset of a non-structured environment where the PD prior should not be applicable, to confirm that the model reverts to its SE kernel component without losing fidelity. The code to recover the results presented in this section is available at https://github.com/fridaviset/PatternMagMaps.

6.4.1 Hallway

We tested the online capabilities of our model on a dataset collected in a hallway using a smartphone.

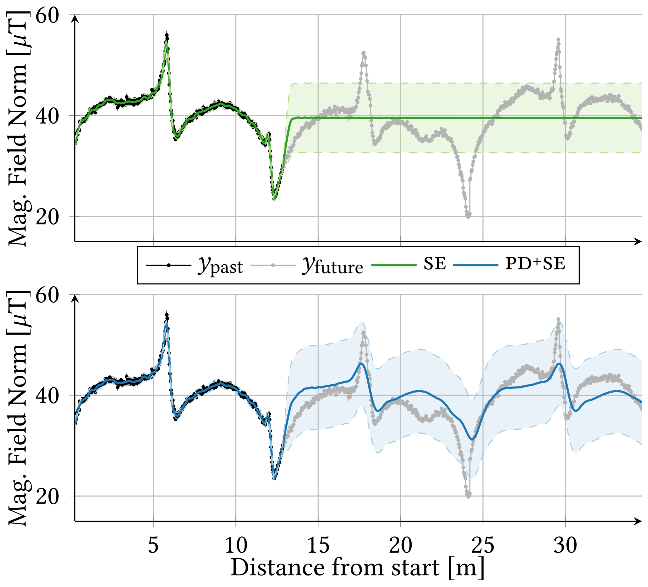


Figure 6.7: Measured and estimated magnetic field norm as a function of the distance moved by the pedestrian from the starting location. The confidence interval illustrates $\pm 3\sigma$. Clearly, the PD+SE kernel (bottom) extrapolates better than the pure SE (top).

Using a customized app, we simultaneously recorded the phone's internal position and orientation estimate and the magnetometer measurements from the phone's built-in magnetometer at 200Hz. The norm of the resulting measured magnetic field norm is visualized in figure 6.7. The visualization represents the online estimate at a time t when the pedestrian has walked the distance between two red columns in the hallway. The results in figure 6.7 indicate the ability of our sum kernel to represent the magnetic field patterns discovered in the past in a comparable way to the SE kernel, while additionally being able to make extrapolated predictions of the magnetic field down the hall. The predicted magnetic field norm at the same timestep is illustrated in a three-dimensional reconstructed environment in figure 6.2.

6.4.2 Dipole model-based simulation virtual parking basement

We consider a virtual parking basement, see figure 6.9, with 5×16 steel-reinforced columns indexed by j in a regular grid and simulate the magnetic influence of each column as a dipole. The magnetic field from dipole j is given by [132]

$$\boldsymbol{b}_j(\boldsymbol{x}) = \frac{\mu_0}{4\pi}\left(\frac{3(\boldsymbol{m}_j \cdot \boldsymbol{r})\boldsymbol{r}}{|r|^5} - \frac{\boldsymbol{m}_j}{|r|^3}\right), \quad \boldsymbol{r} = \boldsymbol{x} - \boldsymbol{x}_j \tag{6.24}$$

where μ_0 is the permeability of open air, $\boldsymbol{m}_j$ is the magnetic field moment of the dipole, $\boldsymbol{r}$ is the distance from the dipole to the location where the magnetic field $\boldsymbol{b}$ is evaluated, $\boldsymbol{x}$ is the location the magnetic field is evaluated and $\boldsymbol{x}_j$ is the location of dipole j. The red top half of the displayed columns in figure 6.9 correspond to the magnetic north and the gray

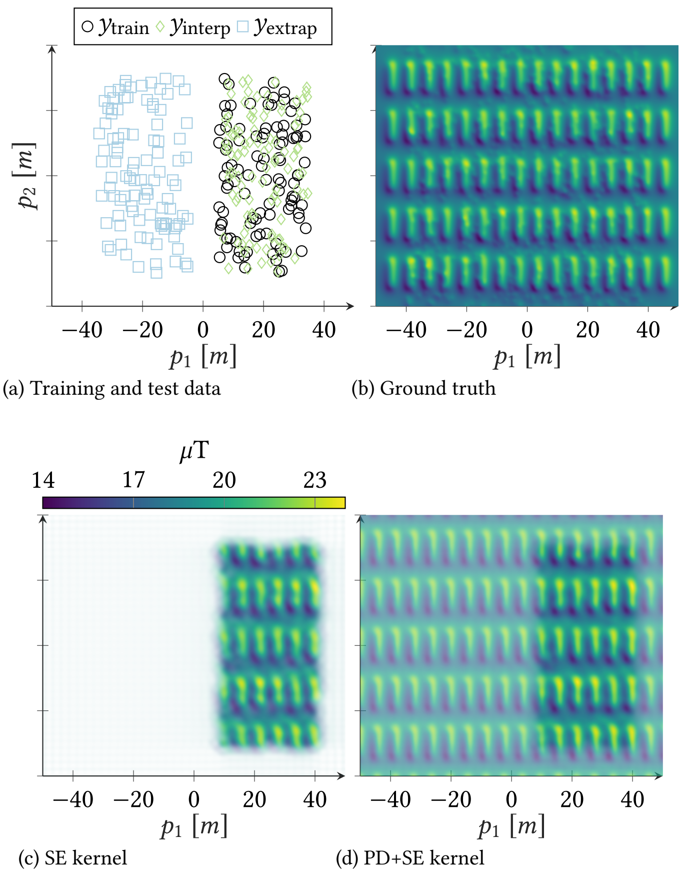


Figure 6.8: Comparison of learned magnetic field maps using a standard recursive basis function approximation to the SE kernel and the learned magnetic field map using our proposed pattern discovering mapping algorithm. The color corresponds to the strength of the magnetic field measured in μT, and the opacity is proportional to the confidence of the prediction.

bottom half to the magnetic south. To additionally include some random perturbations, we also included the magnetic field of 1000 smaller, randomly scattered dipoles. The resulting magnetic field in 3D is therefore given by the following curl-free vector field

$$f(x) = \sum_{j=1}^{5\times16+1000} b_j(x) \tag{6.25}$$

We display a local 2D projection of this magnetic field and its norm in figure 6.9, and we display a 2D projection of the magnetic norm for the full basement in figure 6.8b.

To compare the extrapolation and interpolation accuracy of the PD+SE kernel with that of the SE kernel, we simulate measurements from the right half of the basement, and sample both a set of interpolation test locations in the right half, and a set of extrapolation test locations in the left half, see figure 6.8. The predicted magnetic field based on the

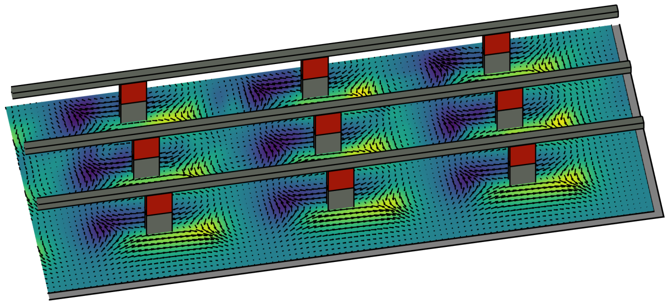


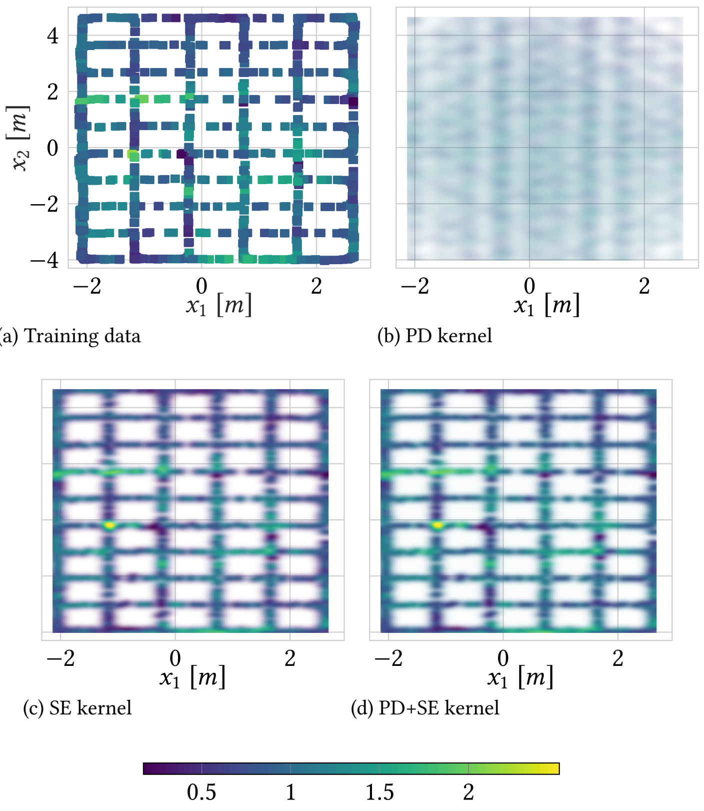
Figure 6.9: Simulation setup for virtual parking basement. The magnetic field is simulated as the resulting magnetic field summing the contribution of each column, modeled as a dipole with a magnetic moment magnitude of 100Am^2. In addition, 1000 smaller dipole disturbances with moment magnitudes of 25Am^2 are randomly scattered in the simulation space.

measurements with the SE kernel is displayed in figure 6.8c, where the opacity indicates predictive uncertainty. The predicted magnetic field reflects the ground truth in figure 6.8b close to the measurement locations in the right half of the basement, while in the right half of the basement, the predictions are not confident enough to be visible. In contrast, the PD+SE kernel, see figure 6.8d, extrapolates the observed periodic repetitions in the field beyond the region with the observed measurements.

In figure 6.11b, we display the prediction error in the extrapolated area for both the SE and the PD+SE kernel. When the number of measurements increases, the prediction error of the SE kernel in this area remains fairly constant, while the PD+SE kernel can learn and generalize a pattern, reducing the average root-mean squared error (RMSE) after 10 experiments from 1.5 to 0.5. Interestingly, for the interpolation experiments in figure 6.11a, we see the same drop from 1.5 meters to 0.5 meters RMSE from 10 to 200 measurements. This drop is caused by the fact that when there are few measurements spaced in the right half of the basement, there is an increased distance between the measurements, meaning that the extrapolation properties of the PD+SE improves estimation accuracy compared to the SE kernel also in this case. As the number of measurements increases for the interpolation experiments, the performance of the SE kernel matches the PD+SE kernel, as expected.

6.4.3 Non-structured Environment

Lastly, we consider a dataset that is *not* representative of the scenarios we expect the proposed model to work well, to illustrate that the model falls back to the SE performance in the absence of global patterns. The dataset was gathered in a non-structured environment *without* repeating structures that induce a periodic behavior of the observed magnetic field. We train three GPs, respectively with a kernel given by equation (6.14), a standard SE kernel, and a sum of the two, the last of which constitutes the proposed kernel. We plot the predictive mean over a regular grid of test points, where the opacity of the predictions is proportional to the predictive variance. The results are plotted in figure 6.10. It is immediately clear that the PD kernel does not discover any repetitive pattern in this data, as expected. Thus, as expected, the PD+SE kernel more or less mirrors the performance



(a) Training data (b) PD kernel

(c) SE kernel (d) PD+SE kernel

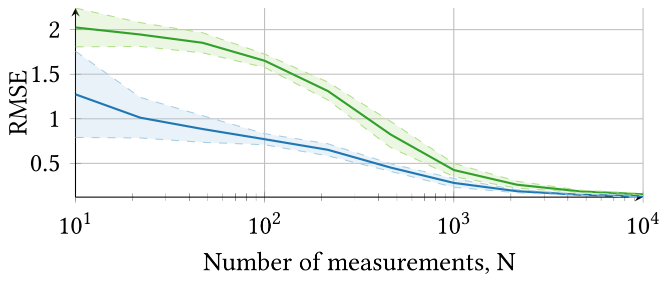
Figure 6.10: Magnetic field maps in a non-structured environment with different kernel choices. The PD+SE kernel matches the performance of the sole SE kernel as the PD component does not discover any repeated patterns in the data. The color indicates the magnetic field strength.

of the SE kernel. The average negative log predictive density (NLPD) was also calculated for a set of test inputs disjoint from the training data. The NLPD was 320.8 for the PD+SE kernel, 348.5 for the SE kernel and 243515 for the PD kernel. This clearly demonstrates that the PD+SE construction does not suffer from having the PD component, even when it is not contributing to the predictive capabilities.

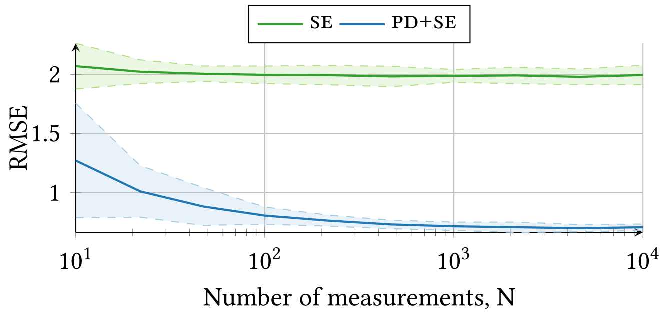
6.5 Related work

Reduced-rank GP regression is widely used for online estimation of nonlinear magnetic field variations [8, 9, 74, 94]. Previous works have primarily used the standard SE kernel as a prior on the potential field. We also model the potential field, but propose a different kernel. By modeling the potential field, we preserve the curl-free property, meaning that we implicitly capture a different type of curl-free kernel than what has been considered previously by [9].

Our proposed kernel for discovering global patterns have strong links to many previ-



(a) Interpolation RMSE



(b) Extrapolation RMSE

Figure 6.11: Comparison of RMSE of magnetic field predictions. Training data was located in the right half of the basement, see figure 6.8a. For interpolation, test data was also located in the right half, while for extrapolation the test data was located in the left half, see figure 6.8a.

ously proposed kernels in literature. Specifically, our PD kernel, like the PD kernel proposed by [1] aims to span the space of all stationary kernels, and the exact link to their kernel is explained in section 6.A.3. In the special case where $D = 1$, the frequencies ω_d^q are integer multiples of each other, and the weights a^q are the Fourier weights of a periodic kernel, this PD kernel can also express the *de facto* periodic kernel [16], as explained in section 6.A.3. In the special case where the number of coefficents $Q \to \infty$, and the frequencies are sampled randomly from the spectral density of the SE kernel, this kernel corresponds exactly to the random Fourier feature approximation of a SE kernel [17].

A wide range of work in the GP community has noted the convenience of expressing periodic kernels in terms of integer multiples of a base frequency, as this corresponds to the Fourier series expansion of any periodic kernel [87, 133**?**]. We generalize this to arbitrary frequencies, which can be identified through regular hyperparameter optimization. Unlike these works, we do not consider variational inference on the Fourier features resulting from this expansion, but rather use the Fourier representation of the kernel directly in the conjugate posterior, similarly to [11]. The result is that our estimation algorithm corresponds to a parametric estimation problem, that can be easily implemented online using recursive stochastic least squares. Further, by using the parametric expansion of the PD kernel to approximate the periodic part of the potential field, we get a magnetic field model that combines the benefits of using a potential field model [9] as opposed to modeling each of the three components independently (as in for example [12, 34, 38, 134]), with the benefits of taking advantage of periodic repetitions for improved accuracy in extrapolation.

A range of other approaches have also been considered for mapping magnetic fields indoors. Notably, averaging values over discrete, hexagonal tiles [72] has been used for creating maps of indoor magnetic fields. Indoor magnetic fields can also be mapped by dipole-models [135], which are inherently curl-free. The work of [136] interestingly also uses orthogonal basis functions to map the magnetic field, but in contrast to our work, places basis functions $\phi : \mathbb{R} \rightarrow \mathbb{R}$ along a one-dimensional trajectory passing by a single dipole source and uses this to identify the location of the dipole. In contrast, our basis functions $\nabla\phi : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ encode a map for the entire magnetic field in all three dimensions. Some algorithms have also been developed for computationally efficient creation and storage of large-scale magnetic field maps [137]. All of these approaches focus on interpolation of the magnetic field close to visited areas. Our work differs qualitatively from this research in that it considers extrapolation to previously unseen areas.

Similarly to a range of previous work on magnetic field mapping, our work provides uncertainty quantification in unseen areas. This can be beneficial for integration into algorithms for simultaneous localisation and mapping [8], or for implementing uncertainty-decreasing exploration strategies [32]. Since it is a parametric approximation, it also integrates effortlessly into frameworks for distributed mapping across several platforms [76, 117]. Several other works have noted that a parametric reformulation of a GP allows for online learning without the computational complexity inflating over time [138].

Creating new kernels by combining existing ones to give a more flexible fit to the underlying data has been considered before [1, 86, 139–141]. Our work can be viewed as a generalization of the work in [86] that allows for detecting a fixed number of Fourier frequency components approximating the kernel, but that unlike [86] does not constrain the variance of these components to be the same. This gives our kernel the flexibility to approximate any kernel on a finite domain, just like Fourier basis functions can approximate any continuous function on a finite domain. Our final proposed kernel is a sum of this periodic PD kernel and an SE kernel, which is comparable to how [140] adds a periodic kernel to a non-periodic component to capture a function that has a clear additive linear component on top of a periodic component.

6.6 Discussion

The difference in assumptions between previously proposed curl-free reduced-rank GPs and our proposed kernel is that we assume the underlying potential field might possess globally repeating patterns in addition to non-repeated variations. Since the magnetic potential field obtains periodic repetitions when dipoles are placed in a periodically repeated pattern, this condition can arise naturally in structured indoor environments. In the cases when there are repeated patterns, our algorithm can improve upon existing methods. In the case where there are no repeated periodic patterns, our algorithm matches the performance of previous work.

An interesting property of our proposed kernel is that depending on the chosen length-scales and the number of basis functions, it can end up capturing a combination of high-frequency and low-frequency content in the learned magnetic field. Similarly to [?], it can learn correlations at discretely different resolutions. In cases where the majority of the signal strength has a large-scale and a low-scale frequency component, this can have the result that our kernel gives better predictions not because there is an underlying periodic signal, but simply because it has the flexibility to place some basis functions at arbitrary frequencies.

Although our work uses Fourier basis functions to express the repeated patterns of the magnetic field, this would not be the only way to estimate repeated patterns using a parametric formulation. Both inducing point approximations, as well as Hilbert-space approximations, could be used to approximate the PD kernel itself, just like they can be used to approximate the SE curl-free kernel. As is explained in Section 5.1 in [11], the SoR approximation of the inducing point approximation can be interpreted as a complete Nyström approximation to the full GP problem in the same way as the Hilbert space basis functions, which effectively enables online parametric estimation in the same way as is done for the magnetic field in [8]. The advantage of using the analytical equivalence between each component of the PD kernel and the parametric reformulation in section 6.3, is that the recursive least-squares online learning of the magnetic field gives exactly the prediction from the PD kernel, rather than an approximation of it.

Although our kernel allows for online learning of the posterior mean and variance, it does not allow for online learning of all hyperparameters without using simplifying tricks such as for example batch-training [81]. We have therefore carried out all hyperparameter optimization offline. The computational cost of carrying out the hyperparameter optimization depends on which hyperparameters are known from previous knowledge about the building structures, and which parameters need to be learned from magnetic field measurements. For reduced-rank GPs that do not require the knowledge of the hyperparameters to compute the precision matrix $\Phi^\top\Phi$, hyperparameter optimization can be carried out using only $\mathcal{O}(M^3)$ operations at each iteration, rather than the $\mathcal{O}(NM^2)$ operations that are normally required [11].

Our parametric reformulation results in basis functions that are dependent on only one hyperparameter, namely the frequency of the expected patterns. Our proposed method will therefore require re-instantiating the precision matrix $\Phi^\top\Phi$ at each iteration of the hyperparameter optimization if there is no prior knowledge available about the expected periodic repetitions in the environment. Thus, hyperparameter optimization generally will cost $\mathcal{O}(NM^2)$, but if knowledge of the environment is available a priori, hyperparameter optimization only costs $\mathcal{O}(M^3)$.

6.7 Conclusion

We have proposed a GP-based model for magnetic field modeling. In contrast to previous work, we explicitly modeled possible repeated patterns in the underlying magnetic field, enabling excellent extrapolation to unseen data regions in structured environments. The parametric representation of the periodic patterns that we developed may also prove useful in other domains where such patterns exist. The results showed that only a small amount of data was needed to capture the general characteristics of the magnetic field over a larger region. This enables the creation of large magnetic field maps of structured indoor environments without needing to collect data in the entire structure.

6.A Appendix

6.A.1 Basis function definitions

The basis functions are defined over a finite-support cubical domain $\Omega \subset \mathbb{R}^d$, defined as $\Omega = [L_{l,1}, L_{u,1}] \times [L_{l,2}, L_{u,2}] \times [L_{l,3}, L_{u,3}]$. The basis functions are given as

$$\phi_{\text{SE},i}(p) = \prod_{d=1}^{3} \frac{\sqrt{2}}{\sqrt{L_{\text{u},d} - L_{\text{l},d}}} \sin\left(\frac{\pi n_{i,d}(p_d + L_{\text{l},d})}{L_{\text{u},d} - L_{\text{l},d}}\right), \tag{6.26}$$

where the set $(n_{i,1}, n_{i,2}, n_{i,3})$ is the set of three natural numbers that are different from the sets $(n_{j,1}, n_{j,2}, n_{j,3})$ defined for all $j < i$, that gives the corresponding value of a parameter λ_i defined as

$$\lambda_i = \sum_{d=1}^{D} \left(\frac{\pi n_{i,d}}{L_{\text{u},d} - L_{\text{l},d}}\right)^2, \tag{6.27}$$

as large as possible. In our case, these basis functions are used to approximate the SE–GP prior as a parametric model

$$f \approx \Phi_{\text{SE}}^{\top} \boldsymbol{w}_{\text{SE}}, \qquad \boldsymbol{w}_{\text{SE}} \sim \mathcal{N}(0, \Lambda_{\text{SE}}), \tag{6.28}$$

6

where Φ is a vector of M basis functions $\phi_i : \mathbb{R}^d \to \mathbb{R}$, $\boldsymbol{w} \in \mathbb{R}^M$ is a vector of weights, and Λ_{SE} is defined as

$$\Lambda_{\text{SE}} = \operatorname{diag}\left[S_{\text{SE}}(\sqrt{\lambda_1}) \quad \cdots \quad S_{\text{SE}}(\sqrt{\lambda_{N_m}})\right], \tag{6.29}$$

where $S_{\text{SE}}(\cdot)$ is the spectral density of the SE kernel, as defined in [9]. This means that the approximation of the magnetic field norm in equation (6.28) has a prior distribution that tends to equation (6.1) as M goes to infinity, and the size of the domain goes to infinity [22].

6.A.2 Proof of proposition 1

Let φ be given by

$$\varphi(\boldsymbol{x}) = \underbrace{\left[\prod_{d=1}^{D} \cos(2\pi\mu_d x_d) \quad \prod_{d=1}^{D} \sin(2\pi\mu_d x_d)\right]}_{\Phi(\boldsymbol{x})} \boldsymbol{w} \tag{6.30a}$$

$$\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{0}, \lambda \boldsymbol{I}). \tag{6.30b}$$

Clearly, $\mathbb{E}[\varphi] = 0$ and thus

$$\begin{aligned} \operatorname{cov}[\varphi(\boldsymbol{x}), \varphi(\boldsymbol{x}')] &= \mathbb{E}\left[\varphi(\boldsymbol{x})[\varphi(\boldsymbol{x}')]^{\top}\right] = \lambda \Phi(\boldsymbol{x})[\Phi(\boldsymbol{x}')]^{\top} \\ &= \lambda\Bigg(\prod_{d=1}^{D} \cos(2\pi\mu_d x_d)\cos(2\pi\mu_d x_d') \\ &\qquad + \sin(2\pi\mu_d x_d)\sin(2\pi\mu_d x_d')\Bigg). \end{aligned} \tag{6.31}$$

Now, note that $\cos(\theta)\cos(\phi) = \frac{1}{2}(\cos(\theta-\phi)+\cos(\theta+\phi))$ and that $\sin(\theta)\sin(\phi) = \frac{1}{2}(\cos(\theta-\phi)-\cos(\theta+\phi))$. Hence,

$$\kappa(\boldsymbol{x},\boldsymbol{x}') \triangleq \operatorname{cov}[\varphi(\boldsymbol{x}),\varphi(\boldsymbol{x}')] = \lambda \prod_{d=1}^{D} \cos(2\pi\mu_d\tau_d), \tag{6.32}$$

where $\tau_d = x_d - x'_d$. The proof for multiple components Q follows by assuming a diagonal covariance matrix for the weights $\boldsymbol{w}$.

6.A.3 The PD kernel as a Fourier series of a periodic kernel

To obtain a periodic kernel from any (one-dimensional) kernel, a “warping” of $[\sin(x),\cos(x)]$ is applied to the input x. The kernel is then applied to $[\sin(x),\cos(x)]$. As is clearly explained in [133], this warping applied to the SE kernel

$$\kappa_{\text{SE}}(x,x') = \sigma_{\text{SE}}^2 \exp\left(\frac{\|x-x'\|_2^2}{l^2}\right) \tag{6.33}$$

gives

$$\begin{aligned}\kappa_{\text{SE,per}}(x,x') &= \kappa_{\text{SE}}([\sin(x),\cos(x)],[\sin(x'),\cos(x')]) \\ &= \sigma_{\text{SE}}^2 \exp\left(\frac{-2\sin^2(\frac{1}{2}w\tau)}{l^2}\right).\end{aligned} \tag{6.34}$$

For the Matern-kernel with degrees of freedom $\nu = \frac{1}{2}$

$$\kappa_{\text{Matern}} = \sigma_{\text{Matern}}^2 \exp\left(\frac{-\|x-x'\|_2}{l}\right), \tag{6.35}$$

the corresponding periodic Matern-kernel is given by

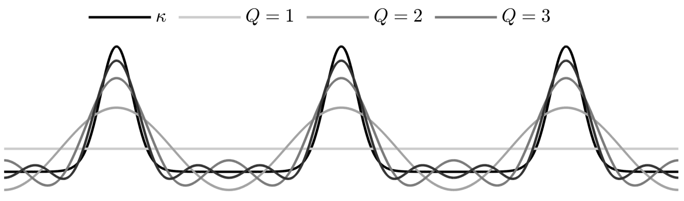
$$\begin{aligned}&\kappa_{\text{Matern,per}}(x,x') \\ &= \kappa_{\text{Matern}}([\sin(x),\cos(x)],[\sin(x'),\cos(x')]) \\ &= \sigma_{\text{Matern}}^2 \exp\left(\frac{-2|\sin(\frac{1}{2}w\tau)|}{l^2}\right)\end{aligned} \tag{6.36}$$

Our proposed kernel equation (6.14) can in general span any stationary kernel on a finite domain. If the mixture frequencies ω_d^q are constrained to be integer multiples of each other, the kernel can span any periodic kernel. In fact, it is then interpretable as a Fourier series of a periodic kernel. This Fourier series interpretation is visualized in figure 6.12, where it is used to approximate two different periodic kernels. However, notice that the mixture weights a^q, and frequencies ω_d^q, are included in the hyperparameters of our kernel. Thus, this kernel is more general than a standard periodic kernel.

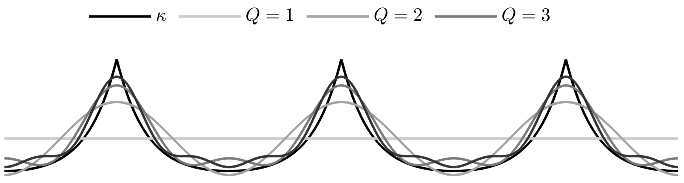
6.A.4 A special case of [1]

The PD kernel proposed by [1] is given by

$$\kappa_{\text{PD}}(\tau) = \sum_{q=1}^{Q} a^q \prod_{d=1}^{D} \exp(-2\pi^2\tau_d v_d^q)\cos(2\pi\tau_d w_d^q), \tag{6.37}$$



(a) Periodic SE kernel



(b) Periodic Matern kernel, $\nu = \frac{1}{2}$.

Figure 6.12: The adaptability of our proposed pattern-detection kernel allows for arbitrarily precise approximation of any periodic kernel on a finite domain. This figure has two examples of how our pattern detection kernel with an increasing amount of parameters can approximate equations (6.34) and (6.36), respectively.

where v_d^q are hyperparameters indicating the inverse of the lengthscale of the decay of the cosine-shaped pattern detected by component q along dimension d of this kernel. In the case where this hyperparameter v_d^q goes to zero, this corresponds to the decay lengthscale going to infinity, which corresponds to encoding the expectation that all detected patterns will prevail over the entire considered domain.

Our PD kernel is therefore a special case of the PD kernel proposed by [1], letting $v_d^q \to 0$. In contrast to the PD kernel proposed by [1], our PD kernel can be exactly rewritten as a parametric model, enabling online learning without further approximation.

7 Conclusion and Recommendations

This thesis addresses scalability challenges in current algorithms for magnetic field mapping and localization. It improved the scalability of five distinct aspects of these algorithms. Some of the contributions are explicitly used by each other during the thesis, while other contributions can easily be combined in future work for applications where multiple types of scalability are required at the same time.

7.1 Conclusion

This thesis has focused on magnetic field mapping and localization. The magnetic field maps can be used to give non-drifting position estimates independent of external infrastructure in environments where there are no GNSS signals, and where the magnetic field has significant spatial variations. Examples of such environments are most indoor environments, some underwater environments, GNSS-denied airspace, and mines. The maps were created using reduced-rank approximations to Gaussian process regression. The thesis has answered five research questions in five chapters, each one involving a particular type of scalability of magnetic field mapping, or magnetic field mapping and localization. The questions with their corresponding conclusions are:

How can magnetic field mapping and localization be done faster?

Magnetic field mapping and localization can be done faster by applying an extended Kalman filter in place of a particle filter. The reason the extended Kalman filter requires less computational resources is that it only requires storing and updating one copy of the Gaussian process magnetic field map. The map update is computationally expensive, and it is therefore faster to update just a single map compared to updating several maps. This is in contrast to the particle filter, that requires storing and updating as many copies of the magnetic field map as the filter has particles, which is typically a number in order of magnitude 10^2. The extended Kalman filter is a viable option in cases where the error magnitude of the position estimate does not exceed the lengthscale of the magnetic field variations.

How can magnetic field mapping and localization be applied to multi-agent systems?

Magnetic field mapping and localization can be applied to multi-agent systems through using the information matrix for the learning of the weights $\boldsymbol{w}$ of the Gaussian process approximation, and distributing the information matrix across a multi-agent system using consensus algorithms [23]. We find that the distributed estimate accurately approximates the centralized estimate if all agents communicate once with each other at each timestep, and that the estimate from the distributed algorithm is more accurate than the individual estimates when the agents communicate on average once every fifth timestep.

How can the storage requirements of reduced-rank Gaussian process maps be reduced?

The storage requirements of reduced-rank Gaussian process maps can be reduced for certain classes of basis functions, including Hilbert-space basis functions and Fourier basis functions, using latent Hankel and Toeplitz structures in the information matrix. This reduction in storage comes with no approximations on the discussed methods. The observed structure also means that one needs a reduced amount of computation time to create an offline map, in the case where all measurements are available at once. Without taking advantage of this structure, this class of basis-function approximations to Gaussian process regression requires $\mathcal{O}(NM^2)$ computations to find the posterior mean and covariance, where N is the number of measurements, and M^2 is the number of basis functions. Taking advantage of this structure, this class of basis-function approximations to Gaussian process regression requires $\mathcal{O}(NM)$ computations to find the posterior mean and covariance. Furthermore, it can be directly employed to reduce the communication requirements in the multi-agent algorithm presented in Chapter 3.

How can magnetic field mapping and localization be extended to larger areas, while keeping the computational complexity low?

In state of the art large-scale Gaussian processs mapping with basis functions, when the map is extended to larger areas, the amount of basis functions required increases with the area, and the computational complexity increases with the amount of basis functions. Therefore, the computational complexity increases with increasing area. Magnetic field mapping and localization can be extended to larger areas while keeping the computational complexity low, by use of a subset of a larger grid of finite-support basis functions close to the current location of the agent. Using a finite amount of finite-support basis functions induces a sparsity in the largest matrices that are required to be computed, meaning that only a finite amount of computational resources are required to update the magnetic field map independent of the size of the mapped domain.

How can magnetic field maps be scaled to encompass repeated global patterns in addition to local variations?

Magnetic field maps can be scaled to encompass repeated global patterns in addition to local variations through the use of a combination of a kernel encoding local spatial variations with a kernel encoding globally repeated patterns. By directly learning the presence of higher-dimensional repeated Fourier features, we implement a pattern-detecting Gaussian process built from basis functions that effectively learns periodic patterns at the same time as being inherently parametric and therefore suitable for online recursive learning. The benefit of this approach is that we can make predictions in map locations where there are no measurements of the magnetic field, by extrapolating the repetitive patterns.

7.2 Recommendations

This thesis has presented contributions to improving various aspects of scalability in magnetic field localization and mapping. This subsection outlines possible directions for future research, building on the work of the thesis.

The spatially scalable mapping technique from Chapter 5 integrated into the extended Kalman filter (EKF) for localization and mapping from Chapter 2 causes a change in the computational complexity at each timestep from $\mathcal{O}(M^2)$ to $\mathcal{O}(M)$ operations, where M is the number of basis functions, as was shown in Chapter 5). The spatially scalable mapping technique from Chapter 5 could also be used in a particle filter. This would reduce in even more computational gain, with a reduction from $\mathcal{O}(N_\mathrm{p}M^2)$ down to $\mathcal{O}(N_\mathrm{p})$. The particle filter would see a much more drastic reduction in computational complexity compared to the EKF. Future work could therefore explore when using this mapping technique in a particle filter yields a beneficial accuracy/computational expense trade-off.

One of the remaining challenges of using the EKF from Chapter 2 is that it is an unimodal representation of the posterior, and will therefore struggle to represent the posterior in cases where it is multimodal. As was discussed in Chapter 2, this case specifically arises when the position uncertainty is larger than the lenghtscale of the magnetic field variations. Another potential advancement of the contribution presented in Chapter 2 is the use of the Gaussian sum filter in place of the Extended Kalman filter [142]. A Gaussian sum filter can potentially provide more accurate position estimates than the EKF in scenarios where the posterior is multi-modal, while being more computationally efficient than the particle filter.

The algorithms in Chapters 2, 3 and 5 assume a priori knowledge of Gaussian process hyperparameters. The exploration of online hyperparameter optimization in the case where the hyperparameters cannot be reasonably estimated a priori could be beneficial. In this case, hyperparameters could be treated as additional parameters, expanding the state space of the estimation algorithm. Future work could also explore the feasibility of simultaneous magnetometer calibration, mapping, and localization for multiple agents. A multi-agent setup generally increases the amount of information one has access to about the magnetic field map, making it a suitable setup for online hyperparameter optimization and magnetometer calibration, which benefits from knowledge about the strength and direction of the magnetic field.

Another promising direction is to continue research on utilizing the Hankel-Toeplitz structure present in the reduced-rank Gaussian process equations in more applications. The Hankel-Toeplitz structure presented in Chapter 4 is present in the information matrix shared between multiple agents in Chapter 3, meaning that the application of the results from Chapter 4 to the communication setup in Chapter 3 reduces the required communication bandwidth between agents from $\mathcal{O}(M^2)$ to $\mathcal{O}(M)$ where M is the number of basis functions required to represent the map. The Hankel-Toeplitz structure is also present in the pattern-detection kernel presented in Chapter 6, and can therefore be used to reduce storage requirements and speed up offline learning of magnetic field maps also with globally repeated patterns. Implementing the pattern-detection mapping algorithm from Chapter 6 in an EKF for SLAM from Chapter 6 could potentially increase accuracy of position estimates in trajectories without loop closures. This algorithm could also be applied in multi-agent systems, allowing collaborative mapping and localization where agents can benefit from globally repeating patterns in the environment. In other words, the pattern-detection kernel presented in Chapter 6 can be used as an alternative map-representation learned by a multi-agent system as presented in Chapter 3. The Hankel-Toeplitz structure discussed in Chapter 4 would also be present in this case, and could therefore also reduce the required communication bandwidth in this case. This could potentially allow one agent mapping one corner of a room to improve the position estimate of an agent in the opposite corner, as long as the room has globally repeating patterns. Furthermore, there are many solvers for linear equations that can speed up the solution to the linear system using Hankel-Toeplitz structures, this could be a natural extension of the work in Chapter 4, which could possibly also yield computational benefits for Chapters 2, 3, and 6, as all of these chapters work with reduced-rank Gaussian process maps that possess the Hankel-Toeplitz structure described in Chapter 4.

The mapping and localization techniques in all chapters can also be adapted to diverse environments including underground settings, the moon, GNSS-denied airspace, underwater, ports, and harbors. Finally, future work can extend the applicability of all chapters to other nonlinear fields apart from the magnetic field such as air quality and pollution, temperature and climate data, soil topography and elevation, seismic activity, gravity and geomagnetic anomalies, ocean currents, atmospheric quantities, ecological population distributions, neural activity, financial time series, spatial economics, disease spread and epidemiology, biomedical signals, acoustic and electromagnetic fields, and sound propagation. Potential societal applications include search and rescue operations, gaming, warehouse operations, greenhouse management, shopping, environmental modeling, healthcare equipment tracking, and assisted living for the visually impaired.

Bibliography

References

[1] A. Wilson and R. Adams, "Gaussian Process Covariance Kernels for Pattern Discovery and Extrapolation," *Proceedings of the 30th International Conference on Machine Learning, ICML 2013*, Feb. 2013.

[2] R. Harle, "A Survey of Indoor Inertial Positioning Systems for Pedestrians," *Communications Surveys & Tutorials, IEEE*, vol. 15, pp. 1281–1293, Jan. 2013.

[3] J. J. Leonard and A. Bahr, "Autonomous Underwater Vehicle Navigation," in *Springer Handbook of Ocean Engineering* (M. R. Dhanak and N. I. Xiros, eds.), pp. 341–358, Cham: Springer International Publishing, 2016.

[4] M. Kok, J. Hol, and T. Schön, *Using Inertial Sensors for Position and Orientation Estimation*. Now Foundations and Trends, Jan. 2017.

[5] O. J. Woodman, "An introduction to inertial navigation," 2007. Technical Report. University of Cambridge.

[6] R. Lanza and A. Meloni, *The Earth's Magnetic Field*. Springer, 2006.

[7] A. Solin, S. Särkkä, J. Kannala, and E. Rahtu, "Terrain navigation in the magnetic landscape: Particle filtering for indoor positioning," in *Proceedings of the European Navigation Conference (ENC)*, pp. 1–9, May 2016.

[8] M. Kok and A. Solin, "Scalable magnetic field SLAM in 3D using Gaussian process maps," in *Proceedings of the 21st International Conference on Information Fusion (FUSION)*, pp. 1353–1360, 2018.

[9] N. Wahlström, M. Kok, T. B. Schön, and F. Gustafsson, "Modeling magnetic fields using Gaussian processes," in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, pp. 3522–3526, May 2013. ISSN: 2379-190X.

[10] A. Solin, M. Kok, N. Wahlström, T. B. Schön, and S. Särkkä, "Modeling and Interpolation of the Ambient Magnetic Field by Gaussian Processes," *IEEE Transactions on Robotics*, vol. 34, pp. 1112–1127, Aug. 2018.

[11] A. Solin and S. Särkkä, "Hilbert space methods for reduced-rank Gaussian process regression," *Statistics and Computing*, vol. 30, pp. 419–446, 2014.

[12] I. Vallivaara, J. Haverinen, A. Kemppainen, and J. Röning, "Magnetic field-based SLAM method for solving the localization problem in mobile robot floor-cleaning task," in *Proceedings of the 15th International Conference on Advanced Robotics (ICAR)*, pp. 198–203, June 2011.

[13] F. Viset, J. T. Gravdahl, and M. Kok, “Magnetic field norm SLAM using Gaussian process regression in foot-mounted sensors,” in *proceedings of the European Control Conference (ECC)*, (Delft, Netherlands), pp. 392–398, IEEE, June 2021.

[14] M. Osman, F. Viset, and M. Kok, “Indoor SLAM using a foot-mounted IMU and the local magnetic field,” in *Proceedings of the 25th International Conference on Information Fusion (FUSION)*, pp. 1–7, July 2022.

[15] N. Wahlström, *Modeling of Magnetic Fields and Extended Object for Localization Applications.* PhD Thesis, Linköping University, Dec. 2015.

[16] C. Rasmussen and C. Williams, *Gaussian Processes for Machine Learning.* Adaptive Computation and Machine Learning series, MIT Press, 2005.

[17] J. Hensman, N. Durrande, and A. Solin, “Variational Fourier Features for Gaussian Processes,” *The Journal of Machine Learning Research*, vol. 18, pp. 5537–5588, Jan. 2017.

[18] M. Yadav, D. Sheldon, and C. Musco, “Faster Kernel Interpolation for Gaussian Processes,” in *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, vol. 130 of *Proceedings of Machine Learning Research*, pp. 2971–2979, PMLR, 2021.

[19] W. Fang, H. Li, H. Huang, S. Dang, Z. Huang, and Z. Wang, “Sparse Gaussian Process Based On Hat Basis Functions,” in *proceedings of the International Conference on Electrical, Communication, and Computer Engineering (ICECCE)*, pp. 1–5, June 2020.

[20] J. Hensman, N. Fusi, and N. D. Lawrence, “Gaussian Processes for Big Data,” in *Proceedings of the 29th Conference on Uncertainty in Artificial Intelligence*, UAI, pp. 282–290, AUAI Press, 2013.

[21] J. Quiñonero-Candela and C. E. Rasmussen, “A Unifying View of Sparse Approximate Gaussian Process Regression,” *Journal of Machine Learning Research*, vol. 6, no. 65, pp. 1939–1959, 2005.

[22] A. Solin and S. Särkkä, “Explicit Link Between Periodic Covariance Functions and State Space Models,” in *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics*, pp. 904–912, PMLR, Apr. 2014. ISSN: 1938-7228.

[23] R. Olfati-Saber, “Distributed Kalman Filter with Embedded Consensus Filters,” in *Proceedings of the 44th IEEE Conference on Decision and Control*, pp. 8179–8184, Dec. 2005. ISSN: 0191-2216.

[24] W. Storms, J. Shockley, and J. Raquet, “Magnetic field navigation in an indoor environment,” in *Ubiquitous Positioning Indoor Navigation and Location Based Service*, pp. 1–10, Oct. 2010. ISSN: null.

[25] L. Paull, S. Saeedi, M. Seto, and H. Li, “AUV Navigation and Localization: A Review,” *IEEE Journal of Oceanic Engineering*, vol. 39, pp. 131–149, Jan. 2014.

[26] L. Dong, D. Sun, G. Han, X. Li, Q. Hu, and L. Shu, "Velocity-Free Localization of Autonomous Driverless Vehicles in Underground Intelligent Mines," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 9, pp. 9292–9303, 2020.

[27] C. Mekik and O. Can, "Multipath effects in RTK GPS and a case study," *Journal of Aeronautics, Astronautics and Aviation, Series A*, vol. 42, pp. 231–240, Dec. 2010.

[28] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1309–1332, 2016.

[29] K. Yamazaki, K. Kato, K. Ono, H. Saegusa, K. Tokunaga, Y. Iida, S. Yamamoto, K. Ashiho, K. Fujiwara, and N. Takahashi, "Analysis of magnetic disturbance due to buildings," *IEEE Transactions on Magnetics*, vol. 39, pp. 3226–3228, Sept. 2003. Conference Name: IEEE Transactions on Magnetics.

[30] J. Haverinen and A. Kemppainen, "A global self-localization technique utilizing local anomalies of the ambient magnetic field," in *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 3142–3147, May 2009. ISSN: 1050-4729.

[31] I. Vallivaara, J. Haverinen, A. Kemppainen, and J. Röning, "Simultaneous localization and mapping using ambient magnetic field," in *Proceedings of the IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, pp. 14–19, 2010.

[32] A. Kemppainen, J. Haverinen, I. Vallivaara, and J. Röning, "Near-optimal SLAM exploration in Gaussian processes," in *Proceedings of the Conference on Multisensor Fusion and Integration*, pp. 7–13, Sept. 2010.

[33] J. Chung, M. Donahoe, C. Schmandt, I.-J. Kim, P. Razavi, and M. Wiseman, "Indoor location sensing using geo-magnetism," in *Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services (MobiSys 2011)*, pp. 141–154, Jan. 2011.

[34] E. L. Grand and S. Thrun, "3-Axis magnetic field mapping and fusion for indoor localization," in *Proceedings of the IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, pp. 358–364, Sept. 2012. ISSN: null.

[35] Seong-Eun Kim, Yong Kim, Jihyun Yoon, and Eung Sun Kim, "Indoor positioning system using geomagnetic anomalies for smartphones," in *Proceedings of the International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pp. 1–5, Nov. 2012. ISSN: null.

[36] M. Frassl, M. Angermann, M. Lichtenstern, P. Robertson, B. J. Julian, and M. Doniec, "Magnetic maps of indoor environments for precise localization of legged and non-legged locomotion," in *Proceedings of the International Conference on Intelligent Robots and Systems*, pp. 913–920, Nov. 2013. ISSN: 2153-0866.

[37] S. Roberts, M. Osborne, M. Ebden, S. Reece, N. Gibson, and S. Aigrain, “Gaussian processes for time-series modelling,” *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 371, p. 20110550, Feb. 2013.

[38] J. Jung, S.-M. Lee, and H. Myung, “Indoor Mobile Robot Localization and Mapping Based on Ambient Magnetic Fields and Aiding Radio Sources,” *IEEE Transactions on Instrumentation and Measurement*, vol. 64, no. 7, pp. 1922–1934, 2015.

[39] H.-S. Kim, W. Seo, and K.-R. Baek, “Indoor Positioning System Using Magnetic Field Map Navigation and an Encoder System,” *Sensors*, vol. 17, p. 651, Mar. 2017.

[40] H. Zheng, H. Wang, L. Wu, H. Chai, and Y. Wang, “Simulation research on gravity-geomagnetism combined aided underwater navigation,” *Journal of Navigation*, vol. 66, no. 1, pp. 83–98, 2013.

[41] I. Nygren, “Robust and efficient terrain navigation of underwater vehicles,” in *Proceedings of the IEEE/ION Position, Location and Navigation Symposium*, pp. 923–932, 2008.

[42] M. Lager, E. A. Topp, and J. Malec, “Underwater Terrain Navigation Using Standard Sea Charts and Magnetic Field Maps,” in *In proceedings of the IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, pp. 78–84, Nov. 2017.

[43] A. Tal, I. Klein, and R. Katz, “Inertial Navigation System/Doppler Velocity Log (INS/DVL) Fusion with Partial DVL Measurements,” *Sensors*, vol. 17, p. 415, Feb. 2017.

[44] C. Tyrén, “Magnetic terrain navigation,” in *Proceedings of the 1987 5th International Symposium on Unmanned Untethered Submersible Technology, UUST 1987*, pp. 245–256, 1987.

[45] Z. Wu, X. Hu, M. Wu, H. mu, J. Cao, K. Zhang, and Z. Tuo, “An experimental evaluation of autonomous underwater vehicle localization on geomagnetic map,” *Applied Physics Letters*, vol. 103, pp. 4102–, Sept. 2013.

[46] M. Wu and Jian Yao, “Adaptive UKF-SLAM based on magnetic gradient inversion method for underwater navigation,” in *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 839–843, 2015.

[47] Y. Huang and Y. Hao, “Method of separating dipole magnetic anomaly from geomagnetic field and application in underwater vehicle localization,” in *Proceedings of the IEEE International Conference on Information and Automation*, pp. 1357–1362, 2010.

[48] F. C. Teixeira, J. Quintas, and A. Pascoal, “Robust methods of magnetic navigation of marine robotic vehicles,” *IFAC-PapersOnLine*, vol. 50, pp. 3470–3475, July 2017.

[49] H. Mu, M. Wu, X. Hu, and M. Hongxu, “Geomagnetic surface navigation using adaptive EKF,” in *Proceedings of the Second IEEE Conference on Industrial Electronics and Applications*, pp. 2821–2825, 2007.

[50] A. Canciani and J. Raquet, "Airborne Magnetic Anomaly Navigation," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 53, no. 1, pp. 67–80, 2017.

[51] J. Quintas, J. Cruz, A. Pascoal, and F. C. Teixeira, "A Comparison of Nonlinear Filters for Underwater Geomagnetic Navigation," in *Proceedings of IEEE/OES Autonomous Underwater Vehicles Symposium (AUV)*, pp. 1–6, 2020.

[52] S. Liu, G. Mingas, and C.-S. Bouganis, "Parallel resampling for particle filters on FPGAs," in *Proceedings of the International Conference on Field-Programmable Technology (FPT)*, pp. 191–198, 2014.

[53] A. Varsi, S. Maskell, and P. G. Spirakis, "An O(log2N) Fully-Balanced Resampling Algorithm for Particle Filters on Distributed Memory Architectures," *Algorithms*, vol. 14, no. 12, 2021.

[54] A. Solin and S. Särkkä, "Gaussian quadratures for state space approximation of scale mixtures of squared exponential covariance functions," in *Proceedings of the IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, pp. 1–6, Sept. 2014. ISSN: 2378-928X.

[55] I. Skog, "OpenShoe Matlab Framework," 2012. KTH Royal Institute of Technology, Stockholm, Sweden and Indian Institutet of Science, Bangalore, India. https://www.openshoe.org/.

[56] N. Wahlström and F. Gustafsson, "Magnetometer Modeling and Validation for Tracking Metallic Targets," *IEEE Transactions on Signal Processing*, vol. 62, no. 3, pp. 545–556, 2014.

[57] F. Gustafsson, *Statistical Sensor Fusion.* Studentlitteratur AB, 2013.

[58] M. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 174–188, 2002.

[59] E. Foxlin, "Pedestrian tracking with shoe-mounted inertial sensors," *IEEE Computer Graphics and Applications*, vol. 25, pp. 38–46, Nov. 2005.

[60] J. Nilsson, I. Skog, P. Händel, and K. V. S. Hari, "Foot-mounted INS for everybody - an open-source embedded implementation," in *Proceedings of the 2012 IEEE/ION Position, Location and Navigation Symposium*, pp. 140–145, Apr. 2012.

[61] M. A. Skoglund, G. Hendeby, and D. Axehill, "Extended Kalman filter modifications based on an optimization view point," in *Proceedings of the 18th International Conference on Information Fusion (Fusion)*, pp. 1856–1861, 2015.

[62] K.-K. Oh, M.-C. Park, and H.-S. Ahn, "A survey of multi-agent formation control," *Automatica*, vol. 53, pp. 424–440, Mar. 2015.

[63] M. Egerstedt and X. Hu, "Formation constrained multi-agent control," *IEEE Transactions on Robotics and Automation*, vol. 17, pp. 947–951, Dec. 2001. Conference Name: IEEE Transactions on Robotics and Automation.

[64] S. Sandeep, B. Fidan, and C. Yu, “Decentralized Cohesive Motion Control of Multi-Agent Formations,” in *Proceedings of the 14th Mediterranean Conference on Control and Automation*, pp. 1–6, June 2006.

[65] P. Fankhauser, M. Bloesch, P. Krüsi, R. Diethelm, M. Wermelinger, T. Schneider, M. Dymczyk, M. Hutter, and R. Siegwart, “Collaborative navigation for flying and walking robots,” in *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*, pp. 2859–2866, Oct. 2016. ISSN: 2153-0866.

[66] P. Puricer and P. Kovar, “Technical Limitations of GNSS Receivers in Indoor Positioning,” in *Proceedings of the 17th International Conference Radioelektronika*, pp. 1–5, Apr. 2007.

[67] S. A. S. Mohamed, M.-H. Haghbayan, T. Westerlund, J. Heikkonen, H. Tenhunen, and J. Plosila, “A Survey on Odometry for Autonomous Navigation Systems,” *IEEE Access*, vol. 7, pp. 97466–97486, 2019. Conference Name: IEEE Access.

[68] D. Zou, P. Tan, and W. Yu, “Collaborative visual SLAM for multiple agents:A brief survey,” *Virtual Reality & Intelligent Hardware*, vol. 1, pp. 461–482, Oct. 2019.

[69] A. Tourani, H. Bavle, J. L. Sanchez-Lopez, and H. Voos, “Visual SLAM: What Are the Current Trends and What to Expect?,” *Sensors*, vol. 22, p. 9297, Jan. 2022. Number: 23 Publisher: Multidisciplinary Digital Publishing Institute.

[70] J. Coulin, R. Guillemard, V. Gay-Bellile, C. Joly, and A. de La Fortelle, “Online Magnetometer Calibration in Indoor Environments for Magnetic field-based SLAM,” in *Proceedings of the 12th International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pp. 1–8, Sept. 2022. ISSN: 2471-917X.

[71] P. Robertson, M. Angermann, and B. Krach, “Simultaneous Localization and Mapping for Pedestrians using only Foot-Mounted Inertial Sensors,” in *ACM International Conference Proceeding Series*, Sept. 2009.

[72] P. Robertson, M. Frassl, M. Angermann, M. Doniec, B. J. Julian, M. Garcia Puyol, M. Khider, M. Lichtenstern, and L. Bruno, “Simultaneous Localization and Mapping for pedestrians using distortions of the local magnetic field intensity in large indoor environments,” in *Proceedings of the International Conference on Indoor Positioning and Indoor Navigation*, pp. 1–10, Oct. 2013.

[73] G. Ouyang and K. Abed-Meraim, “A Survey of Magnetic-Field-Based Indoor Localization,” *MDPI Electronics*, vol. 11, p. 864, Jan. 2022. Number: 6 Publisher: Multidisciplinary Digital Publishing Institute.

[74] F. Viset, R. Helmons, and M. Kok, “An Extended Kalman Filter for Magnetic Field SLAM Using Gaussian Process Regression,” *Sensors*, vol. 22, p. 2833, Jan. 2022. Number: 8 Publisher: Multidisciplinary Digital Publishing Institute.

[75] G. Pillonetto, L. Schenato, and D. Varagnolo, “Distributed Multi-Agent Gaussian Regression via Finite-Dimensional Approximations,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, pp. 2098–2111, Sept. 2019.

[76] D. Jang, J. Yoo, C. Y. Son, D. Kim, and H. J. Kim, "Multi-Robot Active Sensing and Environmental Model Learning With Distributed Gaussian Process," *IEEE Robotics and Automation Letters*, vol. 5, pp. 5905–5912, Oct. 2020. Conference Name: IEEE Robotics and Automation Letters.

[77] S. Särkkä, *Bayesian Filtering and Smoothing.* Institute of Mathematical Statistics Textbooks, Cambridge: Cambridge University Press, 2013.

[78] L. Xiao, S. Boyd, and S. Lall, "A scheme for robust distributed sensor fusion based on average consensus," in *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks, 2005.*, pp. 63–70, Apr. 2005.

[79] E. Snelson and Z. Ghahramani, "Sparse Gaussian Processes using Pseudo-inputs," in *Advances in Neural Information Processing Systems*, vol. 18, MIT Press, 2005.

[80] A. Svensson, A. Solin, S. Särkkä, and T. B. Schön, "Computationally Efficient Bayesian Learning of Gaussian Process State Space Models," in *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, vol. 51, pp. 213–221, PMLR, 2016.

[81] K. Berntorp, "Online Bayesian inference and learning of Gaussian-process state–space models," *Automatica*, vol. 129, p. 109613, July 2021.

[82] M. Kok, "Magnetic Field SLAM," 2018. Published: Guest Lecture.

[83] O. Abril-Pla, V. Andreani, C. Carroll, L. Dong, C. J. Fonnesbeck, M. Kochurov, R. Kumar, J. Lao, C. C. Luhmann, O. A. Martin, and others, "PyMC: a modern, and comprehensive probabilistic programming framework in Python," *PeerJ Computer Science*, vol. 9, p. e1516, 2023. Publisher: PeerJ Inc.

[84] G. Riutort-Mayol, P.-C. Bürkner, M. R. Andersen, A. Solin, and A. Vehtari, "Practical Hilbert space approximate Bayesian Gaussian processes for probabilistic programming," *Statistics and Computing*, vol. 33, p. 17, Dec. 2022. Publisher: Springer.

[85] F. Lindgren, D. Bolin, and H. Rue, "The SPDE approach for Gaussian and non-Gaussian fields: 10 Years and still running," *Spatial Statistics*, vol. 50, p. 100599, Aug. 2022.

[86] M. Lázaro-Gredilla, J. Quiñonero-Candela, C. E. Rasmussen, and A. R. Figueiras-Vidal, "Sparse spectrum Gaussian process regression," *Journal of Machine Learning Research*, vol. 11, no. 63, pp. 1865–1881, 2010.

[87] A. Tompkins and F. Ramos, "Fourier Feature Approximations for Periodic Kernels in Time-Series Modelling," *in Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[88] V. Dutordoir, N. Durrande, and J. Hensman, "Sparse Gaussian processes with spherical harmonic features," in *Proceedings of the 37th International Conference on Machine Learning*, vol. 119 of *Proceedings of Machine Learning Research*, pp. 2793–2802, PMLR, 2020.

[89] P. Greengard, M. Rachh, and A. Barnett, “Equispaced Fourier representations for efficient Gaussian process regression from a billion data points,” 2023. arXiv:2210.10210v2.

[90] S. Wahls, V. Koivunen, H. V. Poor, and M. Verhaegen, “Learning multidimensional Fourier series with tensor trains,” in *Proceedings of the IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pp. 394–398, IEEE, 2014.

[91] M. Mutný and A. Krause, “Efficient High Dimensional Bayesian Optimization with Additivity and Quadrature Fourier Features,” in *Advances in Neural Information Processing Systems*, vol. 32, pp. 9019–9030, 2018.

[92] A. J. Davies, *Effective Implementation of Gaussian Process Regression for Machine Learning*. PhD Thesis, University of Cambridge, 2015.

[93] T. Pinder and D. Dodd, “GPJax: A Gaussian Process Framework in JAX,” *Journal of Open Source Software*, vol. 7, no. 75, p. 4455, 2022. Publisher: The Open Journal.

[94] A. Solin, M. Kok, N. Wahlström, T. B. Schön, and S. Särkkä, “Modeling and Interpolation of the Ambient Magnetic Field by Gaussian Processes,” *IEEE Transactions on Robotics*, vol. 34, pp. 1112–1127, Sept. 2015.

[95] J. Vanhatalo and A. Vehtari, “Modelling local and global phenomena with sparse Gaussian processes,” in *Proceedings of the 24th Conference on Uncertainty in Artificial Intelligence (UAI)*, pp. 571 – 578, AUAI Press, 2008.

[96] Z. Chen, K. Batselier, J. A. K. Suykens, and N. Wong, “Parallelized Tensor Train Learning of Polynomial Classifiers,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 10, pp. 4621–4632, 2018. Publisher: IEEE.

[97] A. Novikov, M. Trofimov, and I. Oseledets, “Exponential machines,” *Bulletin of the Polish Academy of Sciences Technical Sciences*, vol. 66, no. 6 (Special Section on Deep Learning: Theory and Practice), pp. 789–797, 2018.

[98] D. P. Kingma and J. B. Adam, “Adam: A method for stochastic optimization,” in *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 2015.

[99] I. Torroba, M. Cella, A. Terán, N. Rolleberg, and J. Folkesson, “Online Stochastic Variational Gaussian Process Mapping for Large-Scale Bathymetric SLAM in Real Time,” *IEEE Robotics and Automation Letters*, vol. 8, pp. 3150–3157, June 2023.

[100] Y. Kim and H. Bang, “Utilization of terrain elevation map in SLAM for unmanned aircraft,” in *Proceedings of the 6th International Conference on Automation, Robotics and Applications (ICARA)*, pp. 57–62, Feb. 2015.

[101] S. Barkby, S. B. Williams, O. Pizarro, and M. V. Jakuba, “Bathymetric SLAM with no map overlap using Gaussian Processes,” *Proceedings of the International Conference on Intelligent Robots and Systems*, pp. 1242–1248, Sept. 2011.

[102] M. Kjaergaard, E. Bayramoglu, A. S. Massaro, and K. Jensen, “Terrain Mapping and Obstacle Detection Using Gaussian Processes,” in *Proceedings of the 10th International Conference on Machine Learning and Applications and Workshops*, vol. 1, pp. 118–123, Dec. 2011.

[103] H. Yu and B. Lee, “Terrain field SLAM and Uncertainty Mapping using Gaussian Process,” in *Proceedings of the 18th International Conference on Control, Automation and Systems (ICCAS)*, pp. 1077–1080, Oct. 2018.

[104] A. Wilson and H. Nickisch, “Kernel interpolation for scalable structured gaussian processes (KISS-GP),” in *Proceedings of the 32nd International Conference on Machine Learning* (F. Bach and D. Blei, eds.), vol. 37 of *Proceedings of machine learning research*, (Lille, France), pp. 1775–1784, PMLR, July 2015.

[105] A. G. O. Mutambara, *Decentralized Estimation and Control for Multisensor Systems*. CRC Press, Jan. 1998. Google-Books-ID: Z1YfUGkG8poC.

[106] M. R. Walter, R. M. Eustice, and J. J. Leonard, “Exactly Sparse Extended Information Filters for Feature-based SLAM,” *The International Journal of Robotics Research*, vol. 26, pp. 335–359, Apr. 2007.

[107] S. Thrun, D. Koller, Z. Ghahramani, H. Durrant-Whyte, and A. Y. Ng, “Simultaneous Mapping and Localization with Sparse Extended Information Filters: Theory and Initial Results,” in *Algorithmic Foundations of Robotics V* (J.-D. Boissonnat, J. Burdick, K. Goldberg, and S. Hutchinson, eds.), Springer Tracts in Advanced Robotics, pp. 363–380, Berlin, Heidelberg: Springer, 2004.

[108] A. Viseras, T. Wiedemann, C. Manss, L. Magel, J. Mueller, D. Shutin, and L. Merino, “Decentralized multi-agent exploration with online-learning of Gaussian processes,” in *proceedings of the International Conference on Robotics and Automation (ICRA)*, pp. 4222–4229, May 2016.

[109] H. Bijl, J.-W. van Wingerden, T. B. Schön, and M. Verhaegen, “Online sparse Gaussian process regression using FITC and PITC approximations,” *IFAC-PapersOnLine*, vol. 48, pp. 703–708, Jan. 2015.

[110] R. Karlsson and F. Gustafsson, “Bayesian Surface and Underwater Navigation,” *IEEE Transactions on Signal Processing*, vol. 54, pp. 4204–4213, Nov. 2006. Conference Name: IEEE Transactions on Signal Processing.

[111] S. Vasudevan, F. Ramos, E. Nettleton, H. Durrant-Whyte, and A. Blair, “Gaussian Process modeling of large scale terrain,” in *Proceedings of International Conference on Robotics and Automation*, pp. 1047–1053, May 2009. ISSN: 1050-4729.

[112] P. Tichavský, O. Straka, and J. Duník, “Grid-Based Bayesian Filters With Functional Decomposition of Transient Density,” *IEEE Transactions on Signal Processing*, vol. 71, pp. 92–104, 2023. Conference Name: IEEE Transactions on Signal Processing.

[113] F. Gustafsson, F. Gunnarsson, N. Bergman, U. Forssell, J. Jansson, R. Karlsson, and P.-J. Nordlund, “Particle filters for positioning, navigation, and tracking,” *IEEE Transactions on Signal Processing*, vol. 50, pp. 425–437, Feb. 2002.

[114] Y. Ding, R. Kondor, and J. Eskreis-Winkler, “Multiresolution Kernel Approximation for Gaussian Process Regression,” in *Advances in Neural Information Processing Systems*, vol. 30, Curran Associates, Inc., 2017.

[115] A. Kullberg, I. Skog, and G. Hendeby, “Online Joint State Inference and Learning of Partially Unknown State-Space Models,” *IEEE Transactions on Signal Processing*, vol. 69, pp. 4149–4161, 2021.

[116] S. Julier and L. Jr, “On Kalman Filter With Nonlinear Equality Constraints,” *Signal Processing, IEEE Transactions on*, vol. 55, pp. 2774 – 2784, July 2007.

[117] F. Viset, R. Helmons, and M. Kok, “Distributed multi-agent magnetic field norm SLAM with Gaussian processes,” in *Proceedings of the 26th International Conference on Information Fusion (FUSION)*, pp. 1–8, June 2023.

[118] R. B. Gramacy and D. W. Apley, “Local Gaussian Process Approximation for Large Computer Experiments,” *Journal of Computational and Graphical Statistics*, vol. 24, no. 2, pp. 561–578, 2015.

[119] E. Snelson and Z. Ghahramani, “Local and global sparse Gaussian process approximations,” in *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics*, pp. 524–531, PMLR, Mar. 2007.

[120] C. Park and J. Z. Huang, “Efficient Computation of Gaussian Process Regression for Large Spatial Data Sets by Patching Local Gaussian Processes,” *Journal of Machine Learning Research*, vol. 17, no. 174, pp. 1–29, 2016.

[121] C. Park, J. Z. Huang, and Y. Ding, “Domain Decomposition Approach for Fast Gaussian Process Regression of Large Spatial Data Sets,” *Journal of Machine Learning Research*, vol. 12, no. 47, pp. 1697–1728, 2011.

[122] D. Nguyen-Tuong, M. Seeger, J. Peters, D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, “Local Gaussian Process Regression for Real Time Online Model Learning and Control,” *Advances in Neural Information Processing Systems*, Jan. 2008.

[123] K. Dong, D. Eriksson, H. Nickisch, D. Bindel, and A. G. Wilson, “Scalable Log Determinants for Gaussian Process Kernel Learning,” in *Advances in Neural Information Processing Systems*, vol. 30, Curran Associates, Inc., 2017.

[124] R. E. Turner, *Statistical Models for Natural Sounds.* PhD thesis, Gatsby Computational Neuroscience Unit, UCL, 2010.

[125] G. C. Group, “Gridded bathymetry data GEBCO (General Bathymetric Chart of the Oceans) 2021 Grid.,” 2021. gebco.net/data_and_products/gridded_bathymetry_data/.

[126] I. Skog, J.-O. Nilsson, P. Händel, and A. Nehorai, "Inertial Sensor Arrays, Maximum Likelihood, and Cramer-Rao Bound," *IEEE Transactions on Signal Processing*, vol. 64, pp. 1–1, Aug. 2016.

[127] X. Wang, P. Wang, X. Zhang, Y. Wan, H. Shi, and W. Liu, "Target Electromagnetic Detection Method in Underground Environment: A Review," *IEEE Sensors Journal*, vol. 22, pp. 13835–13852, July 2022.

[128] G.-X. Liu, L.-F. Shi, S. Chen, and Z.-G. Wu, "Focusing Matching Localization Method Based on Indoor Magnetic Map," *IEEE Sensors Journal*, vol. 20, no. 17, pp. 10012–10020, 2020.

[129] M. N. Nabighian, V. J. S. Grauch, R. O. Hansen, T. R. LaFehr, Y. Li, J. W. Peirce, J. D. Phillips, and M. E. Ruder, "The historical development of the magnetic method in exploration," *Geophysics*, vol. 70, no. 6, pp. 33ND–61ND, 2005.

[130] B. Siebler, S. Sand, and U. D. Hanebeck, "Localization With Magnetic Field Distortions and Simultaneous Magnetometer Calibration," *IEEE Sensors Journal*, vol. 21, pp. 3388–3397, Feb. 2021. Conference Name: IEEE Sensors Journal.

[131] M. Angermann, M. Frassl, M. Doniec, B. J. Julian, and P. Robertson, "Characterization of the indoor magnetic field for applications in Localization and Mapping," in *In proceedings of the International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pp. 1–9, Nov. 2012. ISSN: null.

[132] K. Seleznyova, M. Strugatsky, and J. Kliava, "Modelling the magnetic dipole," *European Journal of Physics*, vol. 37, pp. 025203 (1–14), Mar. 2016. Publisher: European Physical Society.

[133] A. Tompkins and F. Ramos, "Periodic Kernel Approximation by Index Set Fourier Series Features," in *Proceedings of The 35th Uncertainty in Artificial Intelligence Conference* (R. P. Adams and V. Gogate, eds.), vol. 115 of *Proceedings of Machine Learning Research*, pp. 486–496, PMLR, July 2020.

[134] A. V. Ruiz and C. Olariu, "A general algorithm for exploration with Gaussian processes in complex, unknown environments," in *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pp. 3388–3393, May 2015. ISSN: 1050-4729.

[135] I. Skog, G. Hendeby, and F. Trulsson, "Magnetic-field Based Odometry – An Optical Flow Inspired Approach," in *Proceedings of the International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pp. 1–8, Nov. 2021. ISSN: 2471-917X.

[136] S. L. de Gijsel, A. R. P. J. Vijn, and R. G. Tan, "A Compressed Sensing Algorithm for Magnetic Dipole Localization," *IEEE Sensors Journal*, vol. 22, no. 15, pp. 14825–14833, 2022.

[137] F. M. Viset, R. Helmons, and M. Kok, "Fast Gaussian Process Predictions on Large Geospatial Fields with Prediction-Point Dependent Basis Functions," Oct. 2022. arXiv:2210.09168 [cs, stat].

[138] P. Mattsson, D. Zachariah, and P. Stoica, “Recursive nonlinear-system identification using latent variables,” *Automatica*, vol. 93, June 2016.

[139] C. Archambeau and F. Bach, “Multiple Gaussian Process Models,” Oct. 2011. arXiv:1110.5238 [stat].

[140] N. Durrande, J. Hensman, M. Rattray, and N. Lawrence, “Detecting periodicites with Gaussian processes,” *Peerj Computer Science*, vol. 2, Apr. 2014.

[141] M. Gönen and E. Alpaydin, “Multiple Kernel Learning Algorithms,” *Journal of Machine Learning Research*, vol. 12, no. 64, pp. 2211–2268, 2011.

[142] S.-C. Hsu, “Application of the Gaussian sum filter to magnetic field localization for achieving both accurate and efficient estimates in the case of multimodality,” 2024. Master thesis, TU Delft.

List of publications

Conference Publications

1. ***Frida Viset**, Rudy Helmons, Manon Kok*: Magnetic field norm SLAM using Gaussian process regression in foot-mounted sensors. In Proceedings of the European Control Conference (ECC), 2021, 392-398.

2. *Mostafa Osman, **Frida Viset**, Manon Kok*: Indoor SLAM using a foot-mounted IMU and the local magnetic field. In Proceedings of the 25th International Conference on Information Fusion (FUSION), 1-7

3. *Manon Kok, Mostafa Osman, **Frida Viset***: A framework for indoor localization using the magnetic field. In Proceedings of the 23rd IEEE International Conference on Mobile Data Management (MDM), 385-387

🏆 📄 4. ***Frida Viset**, Rudy Helmons, Manon Kok*: Distributed multi-agent magnetic field norm SLAM with Gaussian processes. In Proceedings of the 26th International Conference on Information Fusion (FUSION), 2023, 1-8. Best student paper award.

Journal Publications

📄 1. ***Frida Viset**, Rudy Helmons, Manon Kok*: An extended Kalman filter for magnetic field SLAM using Gaussian process regression. Sensors, 2022, (8), 2833.

📄 2. ***Frida Viset**, Rudy Helmons, Manon Kok*: Spatially scalable recursive estimation of Gaussian process terrain maps using local basis functions. Under review.

📄 3. ***Frida Viset**, Anton Kullberg, Gustaf Hendeby, Rudy Helmons, Isaac Skog and Manon Kok*: Online discovery of global patterns and local variations in magnetic fields using Gaussian process regression. Under review.

📄 4. ***Frida Viset**, Anton Kullberg, Frederiek Wesel and Arno Solin*: Exploiting Hankel-Toeplitz Structures for Fast Computation of Kernel Precision Matrices. Accepted to TMLR (Transactions on Machine Learning Research)

5. *Anton Kullberg, **Frida Viset**, Isaac Skog, Gustaf Hendeby* Adaptive Basis Function Selection for Computationally Efficient Predictions. Under review.

📄 Included in this thesis.
🏆 Won a best paper, tool demonstration, or proposal award.