Probabilistic Graphical Models for Mobile Pedestrian Localization in 3D Environments

Panagiotis Agis Oikonomou-Filandras

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

of

University College London.

Department of Electronic and Electrical Engineering

University College London

February 4, 2016

I, Panagiotis Agis Oikonomou-Filandras, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the work.

Abstract

This PhD thesis considers the problem of locating wireless nodes in indoors GPS-denied environments using probabilistic graphical models. Time-of-arrival (ToA) distance observations are assumed with Non-Line-of-Sight (NLoS) communications and a lack of adequate anchors. As a solution cooperative localization is developed using Probabilistic Graphical Models (PGMs). The nodes infer their position in an iterative message-passing algorithm, in a distributed manner, given a set of noisy distance observations and a few anchors. The focus of this thesis is to develop algorithms that decrease computational complexity, while maintaining or improving accuracy. Firstly, we develop the Hybrid Ellipsoid Variational Algorithm (HEVA) , which extends probabilistic inference in 3D localization, combining NLoS mitigation for ToA. Simulation results illustrate that HEVA significantly outperforms traditional Non-parametric Belief Propagation (NBP) methods in localization while requires only 50% of their complexity. In addition, we present a novel parametric for Belief Propagation (BP) algorithm. The proposed Grid Belief Propagation (Grid-BP) approach allows extremely fast calculations and works nicely with existing grid-based coordinate systems, e.g. NATO military grid reference system (MGRS). This allows localization using a Global Coordinate System (GCS). Simulation results demonstrate that Grid-BP achieves similar accuracy at much reduced complexity when compared to common techniques. We also present an algorithm that combines Inertial Navigation System (INS) and Pedestrian Dead Reckoning (PDR), namely Probabilistic Hybrid INS/PDR Mobility Tracking Algorithm (PHIMTA), which provides high accuracy tracking for mobile nodes. We combine it with Grid-BP and stop-and-go (SnG) algorithms, showcasing improved

accuracy, at very low computational cost. Finally, we present Stochastic Residual Belief Propagation (SR-BP). SR-BP extends the use of Residual Belief Propagation (R-BP) to distributed networks, improving the accuracy, convergence rate, and communication cost. We prove SR-BP convergence to a unique fixed point under conditions similar to those ensuring convergence of asynchronous BP. Finally, numerical results showcase the improvements in convergence speed, message overhead and detection accuracy of SR-BP.

Acknowledgements

To begin with, I would like to thank the MoDRC for giving me the opportunity to work on this exciting field of cooperative localization. In addition, BT for giving me the chance to dive into the world of PDR localization and its intricacies, as well as exposing me in a great environment to see how research works outside academia. Especially I would like to thank Dr. Santosh Kawade for all his help and support. I would also like to thank SV Life Sciences whom I work for part-time while I was doing my PhD and were always flexible enough and allowed me to give priority to my PhD, when required, especially Mr. Nick Coleman.

I would also like to thank my family, my friends, and all those close to me who stood there for me in the high and lows of doing a PhD. Especially Georgia for her help, support and proofreading skills. This PhD could have no been completed without her.

Finally and most importantly my supervisor Dr. Kai-Kit Wong, for his always positive attitude, and continuous support. He has been a source of inspiration and I will always be in his debt for all his help.

Contents

List of Figures

List of Tables

Acronyms

AoA Angle of Arrival. 15, 19, 24, 140

Box-NBP Box Nonparametric BP. 80, 81

BP Belief Propagation. 3, 4, 47, 48, 64, 70, 79, 94, 95, 98, 99, 102, 104, 105, 108–110, 112, 113, 118, 125–127, 129–133, 135

cpdf Conditional Probability Distribution Function. 63–66, 70, 71, 91

CRLB Cramer-Rao lower bound. 30, 31, 68

ECM Expectation-Conditional Maximization. 86, 89

EM Expectation-Maximization Algorithm. 53–55, 57, 58, 75

FIR Finite Impulse Response. 117

GCS Global Coordinate System. 3, 93, 94, 104, 105, 107, 114, 117, 139

GDoP Geometric Dilution of Precision. 31

GPS Global Positioning System. 93, 105–108, 124

Grid-BP Grid Belief Propagation. 3, 11, 12, 94, 95, 98, 99, 103–105, 107, 108, 111, 112, 118–123, 139

GZD Grid Zone Designator. 95

HEVA Hybrid Ellipsoid Variational Algorithm. 3, 7, 12, 33, 58, 60, 61, 66–89, 92, 102, 104, 105, 138

IMU Inertial Navigation Unit. 8, 106–111, 113–117, 120

INS Inertial Navigation System. 3, 8, 15, 16, 106–108, 121, 123, 124, 139, 140

IPPM Iterative Parallel Projection Method. 17, 60, 81, 82, 85, 86, 89

KDE Kernel Density Estimation. 7, 50

KLD Kullback-Leibler Divergence. 79

LCS Local Coordinate System. 93, 94, 104, 114, 117, 118, 139

LDPC Low-Density Parity Check. 125, 130

LoS Line-of-Sight. 27, 29

LPF Low Pass Filter. 117, 119

MAP Maximum A Posteriori. 55

MDS multidimensional scaling. 17

MEMS microelectromechanical systems. 16, 106–108, 114, 120–123, 139

MGRS military grid reference system. 3, 94–96, 100, 102, 139

MIS Mixture Importance Sampling. 51, 66, 72, 80, 85

MSE Mean Square Error. 30

NBP Non-parametric Belief Propagation. 3, 18, 50, 51, 59, 60, 65, 80–82, 85–87, 94, 104, 121

NLoS Non-Line-of-Sight. 3, 11, 15, 22, 25, 27–30, 33, 35, 60, 61, 66–69, 83, 86, 88–90, 92, 138, 140

P-NB Parsinomius Nonparametric BP. 80, 81, 86

pdf Probability Distribution Function. 41, 51, 59, 61, 94

PDR Pedestrian Dead Reckoning. 3, 5, 8, 12, 16, 106, 107, 120–123, 139

PGM Probabilistic Graphical Model. 3, 26, 36, 38, 40, 42, 59, 138

PHEVA Parsinomius HEVA. 80, 82, 88, 89

PHIMTA Probabilistic Hybrid INS/PDR Mobility Tracking Algorithm. 3, 8, 12, 107, 108, 111, 112, 114, 119–124, 139

PL position location. 14, 15

R-BP Residual Belief Propagation. 4, 34, 125, 126, 129–133, 135, 139

RMS error Root Mean Square Error. 11, 30, 86, 89, 121–123

RSS Received Signal Strength. 15, 19, 22, 29

SDP Semidefinite Programming. 17, 30

SnG stop-and-go. 3, 12, 107, 118, 120–122

SPAWN Sum Product Algorithm For Wireless Networks. 18, 107, 118, 121, 138

SPEB Squared Position Error Bound. 31

SR-BP Stochastic Residual Belief Propagation. 4, 9, 34, 126, 130–133, 135, 136, 139, 140

TDoA Time Difference of Arrival. 15, 19, 21, 140

ToA Time-of-arrival. 3, 15, 19, 21, 26, 29, 35, 60, 61, 63, 68, 110, 138

TR-BP Tree Reweighing BP. 126, 132

URW-BP Uniformly Re-Weighted BP. 126, 132

UWB Ultra WideBand. 29

VB variational Bayes. 58, 61, 66, 67, 75, 76, 78–80, 82, 83, 85–88, 91, 92

VE Variable Elimination. 43–45, 59

VMP Variational Message Passing. 18, 60

Chapter 1

Introduction

The problem of a wireless node estimating its coordinates using either information from global reference points or from sensors attached to it as well as nearby nodes, is referred to as the position location (PL) problem, cf. [1]. As an example the nodes in Fig. 1.1, named Agents are trying to localize using information from nodes with known location, namely Anchors, as well as sharing information between themselves. The lines connecting the nodes show possible communication links.

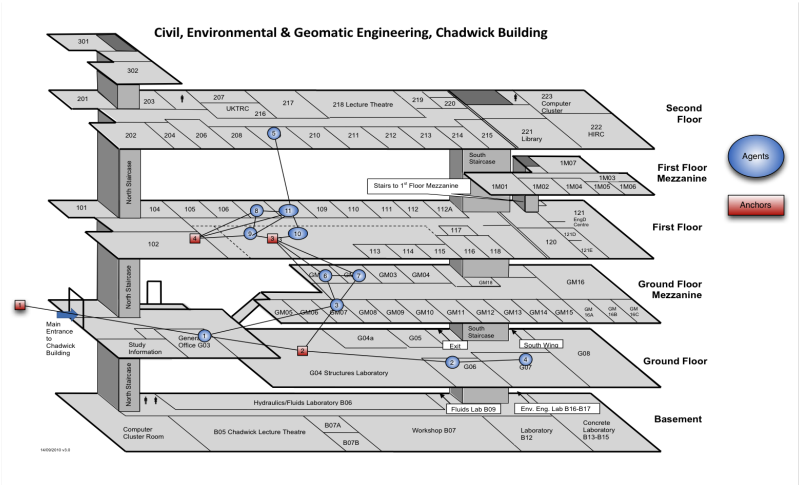

Figure 1.1: Example of agents trying to localize inside a building using cooperative localization.

The introductory chapter will present a quick overview of the PL problem and its variants, as well as the fundamentals ideas and challenges present. Then it will conclude with the contributions presented and the outline of the rest of the thesis.

1.1 Wireless Positioning Systems

To begin with, the definitions self-localisation and cooperative localisation will be given. Then basic signal parameter estimation techniques required in the PL problem, such as ToA, Time Difference of Arrival (TDoA), Angle of Arrival (AoA) and Received Signal Strength (RSS), will be presented. The effect of noise and NLoS on the distance measurements will be also discussed. Finally typical metrics for comparing position location algorithms will be given.

1.1.1 Overview

Positioning systems can be divided into two broad categories. Global and local positioning systems. Global positioning systems will provide accurate position location anywhere on the planet, by using a global reference points, the most common system being GPS satellites [2]. On the other hand local positioning systems will use local reference points, and provide position location estimates relative to these local ones [3]. Furthermore local positioning systems can be either self-localised, or use cooperative localisation. Self-localised nodes will use only information acquired by their own sensors, like acceleration and orientation, e.g. INS. In practice though these systems suffer from error accumulation over time, and therefore work better in conjunction with some remote positioning systems in order to provide consistently accurate results. In this thesis we will focus in local positioning systems, and therefore the discussion will be centred on technologies and techniques which affect GPS-denied positioning systems.

1.1.2 Self-Localisation

Self-localisation becomes highly important in scenarios, where there is no guarantee that nodes will be able to communicate to reference points, e.g. GPS satellites. Typical examples of such GPS denied environments can be dense urban and indoors areas, as well as forests and underground locations.

Obviously the issue is more profound for mobile nodes, as they move around and can stay with an out-of-date position estimate for quite some time before they can communicate with global reference points. In this case INS are used. INS uses

accelerometers and gyroscopes to track the position, velocity and orientation of the node relative to a known starting-point.

INS has been primarily used by the military to track submarines, warships, missiles, rescue teams etc. Yet, recent development of microelectromechanical systems (MEMS), allows for relatively cheap integrated circuits that contain a full suite of sensors typically including a gyroscope, an accelerometer and even a magnetometer. This has enabled the advent of a huge number of civilian applications, as MEMS can be readily found in most modern smart-phones.

The sensors can provide all the information required for determining the updated positions of the nodes, by integrating the angular velocity to obtain the orientation and doubly integrating the acceleration to obtain the position displacement. Unfortunately sensors errors will be propagated to the position and orientation due to the integration and accumulate over time. This is called integration drift and greatly increases the position error. Consequently, the INS system needs to be combined with a position system that uses external reference points in order to ameliorate the integration drift.

Finally, in an effort to avoid or minimize integration drift, many techniques have been developed that extract more information from the the sensor data avoiding the integration altogether. For example, in the case of tracking pedestrians, PDR algorithms, cf. [4, 5, 6] have been designed that take into account the periodic patterns and statistics of the human stride. PDR algorithms have been extensively researched as well as hybrid algorithms [7] that combine INS and PDR. A review of PDR algorithms can be found in [8].

1.1.3 Cooperative localisation

Another important category in the case of local positioning systems is the use of cooperation between the agents. In this case the agents achieve localisation by sharing information and this allows even agents that do not have sufficient reference points close by to self-localise. More information will be provided in Chapter 3, but also good overviews are provided in [3] and [9].

After all the information has been propagated through the network it can be

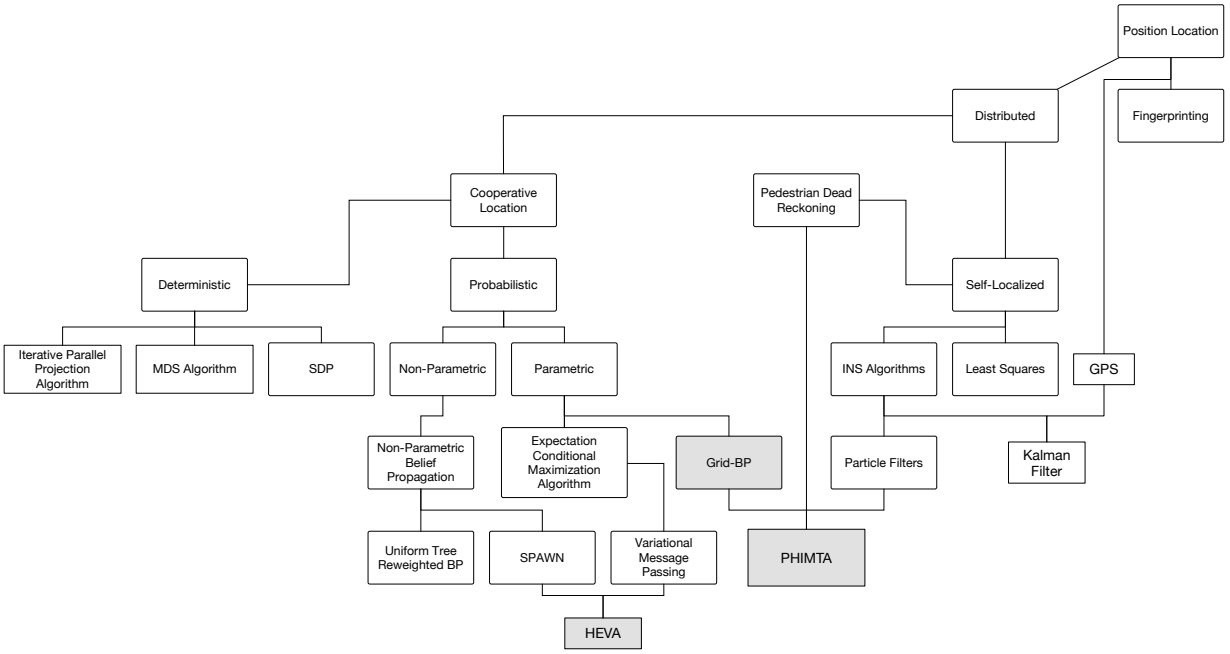

Figure 1.2: Hierarchical tree of position location algorithms. The algorithms that will be presented in this thesis are in grey boxes

combined centrally, either in a dedicated control centre, or an arbitrarily selected node, e.g. [10]. Alternatively the problem can be tackled in a distributed manner, which is much more preferable, as distributed solutions provide a more flexible and robust framework.

In the distributed case a number of algorithms have been developed to tackle the problem. Broadly they can be separated in deterministic and probabilistic algorithms. In deterministic algorithms, the nodes treat the problem as an optimization trying to find the best point estimate. An example of an optimization method is Semidefinite Programming (SDP), as in [11] and [12]. A technique which provides the best accuracy relative to optimization position location algorithm was proposed in [13] using multidimensional scaling (MDS). In [14] the authors proposed an algorithm, called Iterative Parallel Projection Method (IPPM), that treats each message from a neighbouring node as a distinct optimization problem, and then tries to find the solution that minimizes the averages of all the neighbouring optimization problems. The disadvantage of deterministic techniques is that they can very easily fall in local optima from which they can not escape, especially in ambiguous scenarios

with high noise.

On the other hand probabilistic methods assign probability distributions for the whole space and in an effort to avoid local optima. There is a large number of different probabilistic techniques that have been applied to the localisation problem. One family uses particles to approximate the localisation distribution. A representative particle based method is the NBP in [15]. The Sum Product Algorithm For Wireless Networks (SPAWN) algorithm proposed in [9] generalized the aforementioned into a general framework for cooperating localisation using message passing by introducing factor graphs. Another proposed method was the use of Monte Carlo chains in [16], or Variational Message Passing (VMP) in [17]. VMP tries to approximate the real localisation distributions with a simpler one which is tractable and easier to handle. Unfortunately this requires the use of an optimization algorithm in order to find the optimal parameters of the parametric messages, which can lead to bad approximations and increased complexity. Probabilistic techniques manage better the issue of local optima as they keep alive the possibility of other solutions, which tends to help them escape. Unfortunately this comes at much increased computational cost and many more numerical issues that need to be handled, e.g. numerical overflows, singularities [18]. A hierarchical tree with the main families of algorithms and how they fit with each other is shown in Fig. 1.2. The figure is by no means exhaustive and is simply representative of the families of algorithms in the literature on position location and offers some examples relevant to this thesis as well as the algorithms presented in the following chapters.

Despite the intrinsic differences of most algorithms, a simple general formulation that can work as a foundation for describing all different methods will be discussed below.

1.1.4 Basic Distance Measurements used in Position Location

When localizing using cooperative localisation, the nodes will localize relative to some nearby reference points, namely anchors. Besides the coordinates of the anchors, there is a number of different metrics that can be observed and used to achieve localisation. Each technique has its own merits and disadvantages. In essence, as

discussed in [1], all methods described here can be formulated mathematically in the same manner

$$\boldsymbol{r} = \underbrace{\boldsymbol{f}(\boldsymbol{\theta})}_{\boldsymbol{d}} + \boldsymbol{\eta}, \tag{1.1}$$

where $\boldsymbol{r}$ is the distance measurements vector, $\boldsymbol{\theta}$ is the agent coordinates that we are looking for, $\boldsymbol{f}$ is a non-linear function that associates $\boldsymbol{\theta}$ with $\boldsymbol{d}$, i.e. the distance vector between the agent and its neighbouring anchors and $\boldsymbol{\eta}$ is a noise vector that distorts the measurements taken. Depending on the measurement technique used function $\boldsymbol{f}$ will be different but the aim is always the same, i.e. to use an optimization algorithm to minimize the error $\varepsilon = \boldsymbol{r} - \boldsymbol{f}(\hat{\boldsymbol{\theta}})$. Accordingly, first the derivation of eq. (1.1) for various measurements techniques will be derived in the rest of this Subsection and followed by a discussion on the algorithms to minimize the error cost function in Subsection 1.1.5. The most common ways to associate $\boldsymbol{\theta}$ and $\boldsymbol{d}$ from sensor measurements are ToA, TDoA, AoA and RSS, which will presented in the following sections, and a common formulation for all of them will be presented. In all cases the final formulation derived is very similar, cf. eqs. (1.7), (1.12), (1.19) and (1.24), and as a result even though this thesis will mainly focus on ToA measurements it is straightforward to extend it to different distance measurements.

1.1.4.1 Time of Arrival Measurements

In ToA an agent measures the time a message requires to travel between an anchor and itself. This provides a distance estimate, and consequently it has a probable location situated on a circle (2D case) or a sphere (3D) with centre the anchor's coordinates, and radius the distance measured. By receiving more measurements from other anchors, the intersection of the shapes can be calculated, namely triangulated or more generally multilaterated in order to obtain the location estimate.

Mathematically, let $\boldsymbol{\theta} = [x, y, z]^T$ be the unknown agent position and $\boldsymbol{\theta}_i = [x_i, y_i, z_i]^T$ be the respective known coordinates for the i-th anchor, where $i = 1, 2, \ldots M$. Let $M \geq 4$ for the 3D case, as at least four reference points are required to remove multilateration ambiguity in three dimensions. Then the distance

between the agent and anchor i is given by

$$d_i = ||\boldsymbol{\theta} - \boldsymbol{\theta}_i||_2 = \sqrt{(x-x_i)^2 + (y-y_i)^2 + (z-z_i)^2}, \qquad i = 1,2,\ldots,M. \quad (1.2)$$

The relationship between distance and time measurement is pretty straightforward, assuming the anchor i emitted the signal at $t = 0$, and it arrived to the agent at $t = t_i$, the relationship is simply

$$d_i = t_i \times c, \quad (1.3)$$

where c is the speed of light. Let η_i be random noise then distance measured at the agent affected by noise will be

$$r_i = d_i + \eta_i, \quad (1.4)$$

or combining all anchors this can be expressed in vector form as:

$$\boldsymbol{r} = \underbrace{\boldsymbol{d}}_{\boldsymbol{f}(\boldsymbol{\theta})} + \boldsymbol{\eta}. \quad (1.5)$$

Assuming zero-mean Gaussian noise η_i, with uncorrelated variance between different anchor measurements σ_i^2, the pdf that the random variable R_i is equal to the measured (observed) distance r_i, can be written as

$$\Pr(R_i = r_i) = \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left(-\frac{1}{2\sigma_i^2}(r_i - d_i)^2\right) \quad (1.6)$$

and respectively the pdf for the vector case, including all anchor measurements will be

$$\Pr(\boldsymbol{R} = \boldsymbol{r}) = \frac{1}{(2\pi)^{M/2}|\boldsymbol{C}|^{1/2}} \exp(-\frac{1}{2}(\boldsymbol{r} - \boldsymbol{d})^T \boldsymbol{C}^{-1}(\boldsymbol{r} - \boldsymbol{d})) \quad (1.7)$$

where $\boldsymbol{C} = \text{diag}(\sigma_1^2, \ldots, \sigma_i^2, \ldots, \sigma_M^2)$, since as we previously mentioned noise between different anchor measurements is considered uncorrelated. The aforemen-

tioned, (1.7), provides a probabilistic cost function and as well shall see in Section 1.1.5.2, our aim is to get the optimal values that maximize (1.7). The main disadvantage of ToA is that it requires all nodes, both agents and anchors to be perfectly synchronised, as even a small synchronisation offset, can lead to a large localisation error. Also there is an added complexity as signals are required to have a time-stamp, and finally NLoS will add a positive bias, which can seriously distort the localisation results, necessitating some form of NLoS mitigation technique. A way to overcome the synchronisation issue, would be to use round-trip ToA, or two-way ToA.

1.1.4.2 Time Difference of Arrival Measurements

Another technique based on time measurements is the TDoA. The aim behind TDoA is to overcome the synchronisation issue of ToA. The agent uses one measurement as reference and subtracts it from all the other measurements it receives. This removes the need for the agent to be synchronised with the anchors, and creates hyperbola spaces. Typically it is easier for the anchors to be synchronised, as they have high quality clocks, e.g. atomic clocks, and can take advantage of a backbone network for accurate synchronisation. By contrast agents typically have cheaper clocks and in order for them to synchronise an added layer of complexity needs to be inserted. As a result TDoA has been deployed in many applications.

Mathematically the problem can be considered again in the form of (1.1). Let the reference measurement be for $i = 1$, without any loss of generality we have

$$r_i = d_{i,1} + \eta_i, \qquad \text{for} \quad i = 2, \dots M, \tag{1.8}$$

and

$$d_{i,1} = d_i - d_1, \tag{1.9}$$

respectively. In vector form we have as in the previous sections that

$$\boldsymbol{r} = \boldsymbol{f}(\boldsymbol{\theta}) + \boldsymbol{\eta} \tag{1.10}$$

where

$$\boldsymbol{f}(\boldsymbol{\theta}) = \boldsymbol{d}_1 = \begin{bmatrix} \sqrt{(x-x_2)^2+(y-y_2)^2+(z-z_2)^2} - \sqrt{(x-x_1)^2+(y-y_1)^2+(z-z_1)^2} \\ \vdots \\ \sqrt{(x-x_i)^2+(y-y_i)^2+(z-z_i)^2} - \sqrt{(x-x_1)^2+(y-y_1)^2+(z-z_1)^2} \\ \vdots \\ \sqrt{(x-x_M)^2+(y-y_M)^2+(z-z_M)^2} - \sqrt{(x-x_1)^2+(y-y_1)^2+(z-z_1)^2} \end{bmatrix}. \tag{1.11}$$

Making the same zero-mean Gaussian noise assumption as in the ToA case the measurement pdf can be written similarly as

$$\Pr(\boldsymbol{R} = \boldsymbol{r}) = \frac{1}{(2\pi)^{M/2}|\boldsymbol{C}|^{1/2}} \exp(-\frac{1}{2}(\boldsymbol{r}-\boldsymbol{d}_1)^T \boldsymbol{C}^{-1}(\boldsymbol{r}-\boldsymbol{d}_1)). \tag{1.12}$$

It is important to note, as all time differences are done with respect to the first measurement, the noise and correspondingly $\boldsymbol{C}$, will be correlated.

1.1.4.3 Received Signal Strength Measurements

An alternative to time measurements is the use of the RSS, in order to calculate the distance between the anchor and the node. RSS measurements are quite simple to obtain, and do not require synchronisation between nodes. Obtaining accurate distance estimators is quite difficult though and highly reliant on the environment of each localisation scenario. However NLoS is not as destructive for RSS, as for e.g., time measurements, as it only leads to a shadowing effect in power, that can be filtered out. If P_t^i is the power of the signal transmitted from the anchor i, then let P_r^i, be the power of the signal when it gets received from the agent. The path loss

attenuation can be modelled as follows [1]

$$P_r^i = K_i P_t^i d_i^{-a}, \tag{1.13}$$

where K_i is a factor containing all possible factors that affect power, e.g., height and antenna gains, and a is the path loss constant. This can vary between 2 and 5, where $a = 2$ is free space. Noise is believed to follow a lognormal distribution so accordingly we have

$$\ln(P_r^i) = \ln(K_i + \ln(P_t^i) - a\ln(d_i) + \eta_i, \qquad i = 1,2,\ldots M. \tag{1.14}$$

Assuming zero-mean uncorrelated Gaussian noise, let

$$r_i = \ln(P_r^i) - \ln(K_i) - \ln(P_t^i), \tag{1.15}$$

then we have

$$r_i = -a\ln(d_i) + \eta_i, \qquad i = 1,2,\ldots,M \tag{1.16}$$

or in vector form (1.1)

$$\boldsymbol{r} = \boldsymbol{f}(\boldsymbol{\theta}) + \boldsymbol{\eta}, \tag{1.17}$$

where

$$\boldsymbol{f}(\boldsymbol{\theta}) = \boldsymbol{p} = -a \begin{bmatrix} \ln(\sqrt{(x-x_1)^2 + (y-y_1)^2 + (z-z_1)^2}) \\ \vdots \\ \ln(\sqrt{(x-x_i)^2 + (y-y_i)^2 + (z-z_i)^2}) \\ \vdots \\ \ln(\sqrt{(x-x_M)^2 + (y-y_M)^2 + (z-z_M)^2}) \end{bmatrix}. \tag{1.18}$$

Finally as in the previous cases the measurement pdf assuming zero-mean Gaussian noise is

$$\Pr(\boldsymbol{R}=\boldsymbol{r}) = \frac{1}{(2\pi)^{M/2} \mid \boldsymbol{C} \mid^{1/2}} \exp(-\frac{1}{2}(\boldsymbol{r}-\boldsymbol{p})^T \boldsymbol{C}^{-1}(\boldsymbol{r}-\boldsymbol{p})). \tag{1.19}$$

1.1.4.4 Angle of Arrival Measurements

Previously we considered measurements in order to obtain distance estimators between the anchors and the nodes. In the case of AoA, a measurement of the angle between the anchor and the node is made. This essentially limits the possible location of an agent to a line, and consequently multilateration can be done with measurements from only two reference points. Also no synchronisation is required. Unfortunately AoA requires an antenna array in order to obtain the angle and consequently there is a cost in the size of the agent, the power requirements of the agent, and possibly a computational cost for the array processing algorithms.

If ϕ_i is the angle between the anchor and the agent, then for the 2D case we have

$$\tan\phi_i = \frac{y-y_i}{x-x_i}, \qquad i=1,2,\ldots,M \tag{1.20}$$

with $M \geq 2$. Assuming measurement errors we have

$$r_i = \phi_i + n_i = \tan^{-1}(\frac{y-y_i}{x-x_i}) + \eta_i, \quad i=1,2,\ldots,M \tag{1.21}$$

and respectively in vector form

$$\boldsymbol{r} = \underbrace{f(\theta)}_{\phi} + \boldsymbol{\eta}, \tag{1.22}$$

where

$$\boldsymbol{f}(\boldsymbol{\theta}) = \boldsymbol{\phi} = \begin{bmatrix} \tan^{-1}(\frac{y-y_1}{x-x_1}) \\ \vdots \\ \tan^{-1}(\frac{y-y_i}{x-x_i}) \\ \vdots \\ \tan^{-1}(\frac{y-y_M}{x-x_M}) \end{bmatrix}. \tag{1.23}$$

The noise between different anchor measurements can be assumed to be uncorrelated so that the pdf can be written as

$$\Pr(\boldsymbol{R} = \boldsymbol{r}) = \frac{1}{(2\pi)^{M/2} \mid \boldsymbol{C} \mid^{1/2}} \exp(-\frac{1}{2}(\boldsymbol{r}-\boldsymbol{\phi})^T \boldsymbol{C}^{-1}(\boldsymbol{r}-\boldsymbol{\phi})), \tag{1.24}$$

where $\boldsymbol{C}$ is a diagonal matrix with the variances for each measurement.

1.1.5 Position Location Algorithms

After a node has received the coordinates and the corresponding measurements from its neighbours it needs to find an optimal coordinates estimate $\boldsymbol{\theta}^*$ for (1.1). There are many reasons that make finding an optimal solution problematic. Firstly the equations are non linear, resulting in non-convex multi-modal solution spaces, where it is easy to converge to a local optimum. Also the solution spaces can be quite big, making an exhaustive search computationally intractable. Other issues are: high noise levels, NLoS, or even not enough neighbours, which might create ambiguities, divergent or oscillating behaviours in the optimisation. All the above make position location such an interesting and difficult problem, which has stimulated a huge and varied number of proposed techniques and algorithms in the literature. Two surveys with indoors position location algorithms can be found in [19, 20].

Broadly speaking some methods use linear approximations of (1.1), while others face straight on the non-linear optimisation problem. In the latter category, the problem can be formulated either as an optimisation problem, or as a probabilistic problem. Each path has its own advantages and disadvantages. Linear approx-

imation optimisations always converge to a global solution, are computationally cheap, and algorithmically simple, yet depending on the scenario can provide very low accuracy. By contrast non linear optimisation methods, as well as probabilistic modelling, are much more expensive computationally and require more complex algorithms, without necessary guaranteeing a global optimum solution, yet tend to provide higher accuracy results.

Basically, as is often the case, there is a compromise between complexity and accuracy, making the discovery of a solution that both is fast, and provides high accuracy results, as much an art, as it is science. As the focus of this thesis is on PGMs, the linear approximations case will not be presented. The interested reader is referred to [1] for an in-depth discussion and variety of references. In the following subsections, ToA will be used as an example case, and a non-linear solution derivation will be presented leading to a probabilistic analysis of the Position Location problem.

1.1.5.1 NonLinear Least Squares

Let $\varepsilon = \boldsymbol{r} - \boldsymbol{f}(\hat{\boldsymbol{\theta}})$ be the measurement error, between the real and position measurements. Then $\boldsymbol{\theta}^*$ are the coordinates that will minimise the error ε. With that in mind, a suitable cost function is

$$J(\hat{\boldsymbol{\theta}}) = \sum_{i}^{M} (r_i - \sqrt{(\hat{x} - x_i^2) + (\hat{y} - y_i^2) + (\hat{z} - z_i^2)})^2 \tag{1.25}$$

$$= (\boldsymbol{r} - \boldsymbol{f}(\hat{\boldsymbol{\theta}}))^T (\boldsymbol{r} - \boldsymbol{f}(\hat{\boldsymbol{\theta}})) \tag{1.26}$$

and $\boldsymbol{\theta}^*$ can be found as

$$\boldsymbol{\theta}^* = \arg\min_{\hat{\boldsymbol{\theta}}} J(\hat{\boldsymbol{\theta}}). \tag{1.27}$$

As was mentioned earlier, $J(\hat{\boldsymbol{\theta}})$, is a multi-modal non-convex function. Consequently convergence to a global optimum is not guaranteed and either a global search is required using techniques like genetic algorithms [21]. A computational cheaper alternative is to use an iterative algorithm to do a local search near an initial

estimate $\hat{\boldsymbol{\theta}}^0$. This is highly dependent on the initial guess, but given a good starting choice, any classic iterative procedure like Newton-Raphson, Gauss-Newton, etc. can lead to the global optimum.

1.1.5.2 Maximum-Likelihood

In a probabilistic context the problem is reversed. Instead of trying to find the optimal coordinates that minimise the error, the coordinates that maximise the probability are sought after. By taking the log of (1.7), we get

$$\ln(\Pr(\boldsymbol{R}=\boldsymbol{r})) = \ln(\frac{1}{(2\pi)^{M/2}|\boldsymbol{C}|^{1/2}}) - \frac{1}{2}(\boldsymbol{r}-\boldsymbol{d})^T\boldsymbol{C}^{-1}(\boldsymbol{r}-\boldsymbol{d}). \tag{1.28}$$

The first term is independent of $\hat{\boldsymbol{\theta}}$ and as a result plays no role in the optimisation, so it can be ignored.

Instead of maximising a negative term, we can minimise it is positive dual. Let the cost function J be

$$J(\hat{\theta}) = (\boldsymbol{r}-\boldsymbol{f}(\hat{\boldsymbol{\theta}}))^T\boldsymbol{C}^{-1}(\boldsymbol{r}-\boldsymbol{f}(\hat{\boldsymbol{\theta}})) \tag{1.29}$$

and the optimal θ^* is found same as before

$$\boldsymbol{\theta}^* = \arg\min_{\hat{\boldsymbol{\theta}}} J(\hat{\boldsymbol{\theta}}). \tag{1.30}$$

The cost function of the ML case can be thought as a generalisation of the NLS case. Basically $\boldsymbol{C}$ plays the role of weighting the measurements, where due to the inverse, large noise measurements weigh less in the optimisation, while low noise measurements have a higher weight and influence more the result.

1.1.6 Noise and NLoS in Position Location

A major issue in urban and indoors position-location is the high number of NLoS signals. In both time-based and power-based measurements, NLoS paths will add a bias to the noise, as the signal takes longer and attenuates more over the larger distance it traverses, compared to the equivalent Line-of-Sight (LoS) path. As a

result localisation accuracy can be seriously affected.

Using (1.1) we have

$$\boldsymbol{r} = \underbrace{f(\theta)}_{d} + \boldsymbol{\eta}.$$

Let a specific agent be g and a specific anchor be λ. Then we have

$$r_{g,\lambda} = d_{g,\lambda} + \eta_{g,\lambda}, \tag{1.31}$$

where the noise can be modelled as a zero mean Gaussian

$$\eta_{g,\lambda} \sim \mathcal{N}(0, \sigma^2_{g,\lambda}) \tag{1.32}$$

with noise variance

$$\sigma^2_{g,\lambda} = K_e d^{\beta_\lambda}_{g,\lambda}, \tag{1.33}$$

where β_λ is the path loss exponent and K_e is a proportionality factor, capturing the combined physical layer and receiver effect [22], as well as other effects. More information on the model can be found in [14, 23, 24]. In the case of $\beta_\lambda = 2$ the model applies for both ToA and RSS measurements.

It is assumed that NLoS estimates add a positive bias, cf. [25], to the error of the distance measurements. Hence, (1.31) becomes

$$r_{g,\lambda} = d_{g,\lambda} + \eta_{g,\lambda} + b_g. \tag{1.34}$$

The bias error will be $b_g > 0$, and is assumed to be independent of $n_{g,\lambda}$. Without making any assumptions on the underlying bias generating distribution, b_g can be derived from a uniform distribution

$$b_g \sim \mathcal{U}(0, B_{\max}), \tag{1.35}$$

where $B_{\max}$ is the largest propagation distance the signal can travel given the physical layer parameters like the transmission power and the propagation environment. Additionally it is assumed that $B_{\max}$ is much larger than the variance, i.e., $B_{\max} >> \sigma_{g,\lambda}$.

Directly using NLoS measurements in multilateration can greatly increase the localisation error. Despite that, there are times, where using NLoS measurements will actually improve the position location. This creates two main problems on NLoS that are highly active research topics. NLoS Identification and NLoS Mitigation

- NLoS Identification

  The problem here is about distinguishing between LoS and NLoS signals. Statistical NLoS identification schemes for cellular systems have been proposed in [26, 27, 28]. Of special note is the use of Ultra WideBand (UWB). UWB signals posses a higher temporal resolution and robustness than narrowband and sideband signals, making them ideal for position location measurements. Consequently UWB signal localisation has seen a large interest from the research community. Firstly properties of ultra-wide-band signals had been investigated and their suitability for localisation has been verified both theoretically [29, 30, 31] and empirically [32, 33]. In [31] Donlan et al. use data from the DARPA NETEX project to model the UWB indoor channel. More recent measurement campaign is undertaken in [34]. NLoS message identification for UWB systems has been considered in [35, 36, 37].

  Additionally, the statistics of ToA, RSS and the temporal dispersion of RSS have been analysed extensively in various scenarios and it has been shown in [36] that by using the RMS delay spread, LoS and NLoS signals can be identified with high accuracy.

- NLoS Mitigation

  Given the ability to distinguish between LoS and NLoS signals the question of how to handle the NLoS signals remains. A simple solution would be

to discard them completely. However it has been shown [38] that by simply discarding NLoS measurements it does not necessarily improve performance.

A high number of NLoS mitigation methods have been proposed, e.g., using an SDP approach, cf. [39] or a quadratic programming approach, cf. [40], at the cost of high complexity, cf. [41]. Another technique of using NLoS measurements is to selectively use them as long as they help the convergence of a solution,cf. [23]. NLoS mitigation for the deterministic case has been tackled in [35, 42, 37, 43, 44]. A survey covering NLoS mitigation methods can be found in [45]. Finally, [46] presents a variety of algorithms for both centralized and distributed localisation, involving real-time estimation of the NLoS noise pdf in order to better mitigate it in the localisation calculations.

1.1.7 Performance Analysis of Position Location Algorithms

The evaluation of a Position Location algorithms performance is not as straightforward as it initially seems. There are many aspects that need to be taken into account besides localisation accuracy. For the localisation of a single node the most important metric is the Mean Square Error (MSE) which is defined as

$$\mathrm{MSE}(\hat{\boldsymbol{\theta}}) = \frac{\sum_{i}^{M}(\hat{\boldsymbol{\theta}}_i - \boldsymbol{\theta}_i)^2}{M}, \tag{1.36}$$

where M is the size of $\boldsymbol{\theta}$. The MSE, provides a measurement of the variance of the error. And respectively we can define the RMS error, which measures the standard deviation of the error, i.e. the RMS error is defined as:

$$\mathrm{RMS}(\hat{\boldsymbol{\theta}}) = \sqrt{\mathrm{MSE}(\hat{\boldsymbol{\theta}})}. \tag{1.37}$$

An important limit to compare the MSE is the Cramer-Rao lower bound (CRLB). It provides a lower bound on the variance of an unbiased estimate and can be readily computed for $\hat{\boldsymbol{\theta}}$, and in this case we can use it for a lower boundon

MSE. It should be noted that it is possible for MSE($\hat{\boldsymbol{\theta}}$) to be lower than CRLB, given a biased estimator and specific scenarios. Mathematically, CRLB is defined as

$$\text{CRLB}(\hat{\boldsymbol{\theta}}) = [\boldsymbol{I}]^{-1} = \left[\mathbb{E}\frac{\partial^2 \ln \Pr(\boldsymbol{R}=\boldsymbol{r})}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}^T}\right]^{-1}, \tag{1.38}$$

where $\boldsymbol{I}$ is the fisher information matrix. Assuming zero-mean Gaussian distributed noise, $\boldsymbol{I}$ can be computed as

$$\boldsymbol{I}(\boldsymbol{\theta}) = \left[\frac{\partial \boldsymbol{f}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}\right]^T \boldsymbol{C}^{-1} \left[\frac{\partial \boldsymbol{f}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}\right]. \tag{1.39}$$

Various proposed lower bounds for position location can be found in the literature. For the case of cooperative localisation CRLBs were proposed in [3, 41]. In [47], Winn et al. derived the Squared Position Error Bound (SPEB), a lower bound specifically for localisation in multipath Rayleigh channels. In [48], Spirito defined the accuracy measure, a metric which takes into account both the measurement noise variance with the geometry of the neighbouring noise in order to derive a lower bound for a node's localisation.

It is important to note the importance that the geometry between the agent and the anchors play in localisation. Even if there was no noise in the measurements, a bad geometry can be detrimental to the accuracy. This effect is captured in the Geometric Dilution of Precision (GDoP)[49], which is readily used in the analysis of GPS systems. GDoP can be defined as $\boldsymbol{I}(\boldsymbol{\theta})$ without the covariance matrix. That is,

$$\text{GDOP} = \left[\frac{\partial \boldsymbol{f}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}\right]^T \left[\frac{\partial \boldsymbol{f}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}\right]. \tag{1.40}$$

Generally, we are not interested in the localisation of simply a single node, instead in both cooperative and non-cooperative localisation, we have a large number of nodes that are trying to self-localise. In this context we are more interested in the

RMS localisation error that can be defined as

$$\bar{\Omega} = \sqrt{\frac{1}{N}\sum_{i=1}^{n} \mathbb{E}\left[\|\hat{\boldsymbol{\theta}}_i - \boldsymbol{\theta}_i\|_2^2\right]}, \tag{1.41}$$

where N is the number of nodes trying to self-localise. Another metric in the multiple agents scenario is the outage probability. The outage probability can be defined as the probability that an agent has localised below a certain error threshold e_{th}, and mathematically is defined as

$$\Pr_{\text{out}}(e_{th}) = \mathbb{E}\{\mathbf{1}\{\|\boldsymbol{\Theta} - \hat{\boldsymbol{\Theta}}\| > e_{th}\}\}, \tag{1.42}$$

where $\boldsymbol{\Theta} = [\theta_1, \ldots \theta_N]$ and $\mathbf{1}(\cdot)$, is the indicator function defined as

$$\mathbf{1}(X = x) = \begin{cases} 1, \text{if } X = x \\ 0, \text{ otherwise} \end{cases}. \tag{1.43}$$

1.2 Problems Addressed and Contributions

Another important issue to consider is the tractability of the cooperative localisation in real-life devices which have limited battery and processor capabilities. Consequently, the main focus and the common theme presented in all Chapters of this thesis is the objective to design algorithms that have realistically low computational and communication costs. It was also important to accomplish the above with as little compromise to the accuracy as possible, even in high noise scenarios. Monte Carlo simulations were run for all algorithms to prove the above. The use of Monte Carlo simulations was chosen as it provided the flexibility to easily, affordably and reliably compare the algorithms presented in a large variety of different noise models with all the relevant existing algorithms in the literature in both complexity and accuracy. This allowed us to obtain results from cooperative networks of large numbers of nodes in various 3D scenarios with different noise models and different IMU sensor characteristics. A real life measurement campaign, albeit interesting, would have been by necessity very limited due to manpower and budget constraints and

not able to get adequate results for a large scale network.

In more detail, the following major issues were tackled:

- *3D probabilistic cooperative localisation*. Probabilistic cooperative localisation algorithms have been developed with great success in the 2D case, cf. [9], resulting to a huge number of message passing variants. Unfortunately as mentioned above, the aforementioned are computationally expensive algorithms and become even more so in the 3D case due to the added dimension in the solution space. We propose a novel cooperative localisation algorithm, namely HEVA, that drastically decreases the computational cost in localisation, while at the same time improving localisation accuracy, even in high noise, 3D environments with NLoS affected communications. Monte Carlo simulations were run to showcase the improvements in both accuracy and computational costs relative to published algorithms in the literature, cf. chapter 3.

- *Reference-free 2D localisation.* A hidden issue in cooperative localisation is the definition of the reference point. All literature either assumes that all nodes have a common reference point with precise knowledge of their relative location of it, or in the best of cases simply acknowledge the issue and the notes localized relative to each other and some arbitrary chosen reference point. We propose a novel cooperative localisation algorithm, namely Grid-BP, that implements a unique ID grid reference system, overcoming the reference point issue. At the same time the algorithm offers parametric message passing allowing for very fast and efficient computational time.

- *Combining local sensor based pedestrian tracking with cooperative localisation.* There has been great development in the use of sensor based pedestrian tracking, or pedestrian tracking/GPS hybrid algorithms, but the research on pedestrian tracking/cooperative localisation algorithms has been quite limited. As a result, we have proposed a novel probabilistic hybrid INS/PDR tracking algorithm, named PHIMTA. PHIMTA can be used in a probabilistic

temporal model with Grid-BP to offer high accuracy pedestrian tracking in GPS-denied environments.

- *Message Scheduling in Distributed Message Passing algorithms.* A lot of research has been done in message scheduling of centralized message passing algorithms, as the algorithm can be both optimized in communication overhead, and solution accuracy. Unfortunately there has been almost no research done in the distributed case due to the inherent lack of message scheduler with global knowledge of the network. We propose a truly distributed message scheduling algorithm, namely SR-BP, which extends the advantages of R-BP in distributed scenarios, such as cooperative localisation or distributed spectrum sensing.

1.3 Publications

The current PhD thesis resulted in the following publications:

- P.-A Oikonomou-Filandras and Kai-Kit Wong. Hybrid non-parametric belief propagation for localization in wireless networks. In *Sensor Signal Processing for Defence (SSPD 2011)*, pages 1–5, Sept 2011

- P.-A. Oikonomou-Filandras, Kai-Kit Wong, and Yangyang Zhang. Heva: Cooperative localization using a combined non-parametric belief propagation and variational message passing approach. *Journal Commun. Netw.*, to be published

- P.-A. Oikonomou-Filandras, Kai-Kit Wong, and Yangyang Zhang. Informed scheduling by stochastic residual belief propagation in distributed wireless networks. *"IEEE Wireless Commun. Lett.*, 4(1):90–93, Feb 2015

1.4 Thesis Organization

The rest of the thesis is organized as follows:

- Chapter 2 provides the mathematical background necessary for the development and analysis of the algorithms to follow. A brief review is given on

probabilistic graphical models, message passing, approximate inference, and contraction mappings, as the main tools that will be used.

- Chapter 3 presents HEVA for 3D cooperative localisation in NLoS ToA metric scenarios.
- Chapter 4 presents Grid-BP for 2D localisation using a grid based reference system.
- Chapter 5 presents the Pedestrian Hybrid Inertial Measuring Tracking Algorithm (PHIMTA), as a probabilistic pedestrian tracking algorithm and suggests the use of PHIMTA/Grid-BP as a combined approach to pedestrian tracking in GPS-denied environments.
- Chapter 6 presents SR-BP, as a distributed scheduler for message passing algorithms.
- Chapter 7 concludes the thesis offering final remarks and conclusions as well as possible future research directions.

Chapter 2

Mathematical Background

2.1 Graphs and Probabilistic Graphical Models

In this section, background information on PGMs, relevant to the later parts of the thesis is provided, as well as the corresponding references for further reading on the subject. The Section will start with the basics of PGMs and provide a quick overview. Then some more in-depth analysis on selected topics that are of interest will follow. As the nomenclature on PGMs is still not completely standardised, the terms used in [53] will be preferred. Initially some basics on probabilistic reasoning and some basic graph concepts will be presented. Followed by a quick overview of Markov networks.

2.1.1 Basic Graph Concepts

Graphical models are a way to describe the dependence and independence relations between variables in problems modelled as distributions. Their importance is due to their ability to provide a general framework that is easy to understand visually of all the interactions in a model. In this manner the act of designing the model, with the act of getting information from the model are explicitly separated. This allows for the same inference algorithms to be usable, when the model is changed, or vice versa different inference algorithms can be run on the same model. This has allowed for the design of a large variety of algorithms that can be used on PGMs that can be applied to highly diverse problems. Therefore for a given problem there are two steps to consider:

- **Modelling**. Identify all possible variables that are relevant for a given model. Then find the interaction between these variables and model them as edges (directed or undirected) between them. This can be done both by taking expert advise on the problem and by the use of automated algorithms. These algorithms will generally not be able to give the direction of dependence but are usually able to provide valuable insight on how certain variables interact with each other.

- **Inference**. After the probabilistic model is constructed, various information can be derived by using probabilistic inference on the model. Typical questions can be about the most likely value of certain variables conditioned on a fixed state of others, or the most likely state of all variables in general. Another typical need is to find sufficient statistics that will describe the model with accuracy.

2.1.2 Graphs

A typical aim in a graph model is to infer the interaction between the states of an event we want to investigate and data that we have obtained. This can be modelled in the following way. Define a number of parameters $\boldsymbol{\Theta}$ that are of interest, and then obtain a set of observed data $\boldsymbol{d}$ of the random variable $\boldsymbol{D}$ that affect $\boldsymbol{\Theta}$. Given some prior knowledge on how $\boldsymbol{\Theta}$ is generated then we can use the Bayes rule to formulate the problem as

$$\underbrace{\Pr(\boldsymbol{\Theta} \mid \boldsymbol{D} = \boldsymbol{d})}_{\text{posterior}} = \frac{\overbrace{\Pr(\boldsymbol{D} = \boldsymbol{d} \mid \boldsymbol{\Theta})}^{\text{likelihood}} \overbrace{\Pr(\boldsymbol{\Theta})}^{\text{prior}}}{\underbrace{\Pr(\boldsymbol{D} = \boldsymbol{d})}_{\text{evidence}}}, \tag{2.1}$$

where $\Pr(\boldsymbol{D} = \boldsymbol{d}) = \int \Pr(\boldsymbol{D} = \boldsymbol{d} \mid \boldsymbol{\Theta}) \Pr(\boldsymbol{\Theta}) d\boldsymbol{\Theta}$.

The main idea here is that if we have some prior knowledge $\Pr(\boldsymbol{\Theta})$ and data that we acquired, then given our model of $\boldsymbol{\Theta}$, i.e., $\Pr(\boldsymbol{D} = \boldsymbol{d} \mid \boldsymbol{\Theta})$, we aim to infer the posterior $\Pr(\boldsymbol{\Theta} \mid \boldsymbol{D} = \boldsymbol{d})$, or more simply we want to refine our belief of how $\boldsymbol{\Theta}$ behaves, based on the data we have observed. The term likelihood is used for the

probability of obtaining the data we observed given the model parameters, or more simply, on how likely it is to have observed this data, given the parameters we have envisioned.

A graph is essentially a graphical representations of a set of independence assertions. These by the use of the chain rule, define the joint distribution of a given model and can be used to form the posterior, eq. (2.1), that are of interest.

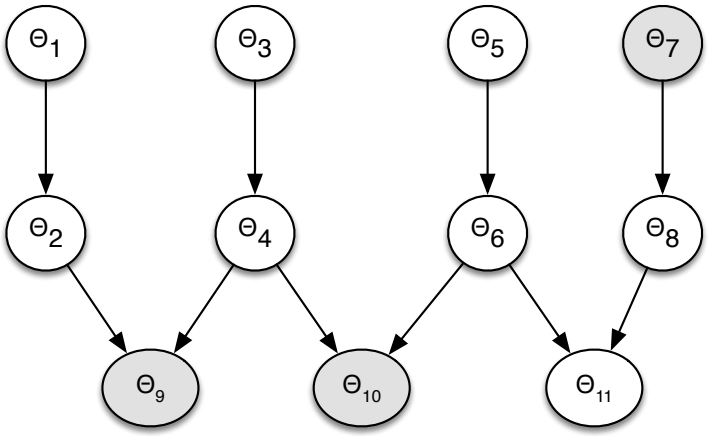

Figure 2.1: An example of a directed graph. Greyed out variables Θ_9, Θ_{10} are observed.

Definition 1. *A graph is a data structure consisting of vertices (nodes) and edges (links) between the vertices, e.g. Fig. 2.1. Let* $\{\Theta_1, \ldots, \Theta_n\}$ *be the set of nodes. Two nodes can be connected with a directed edge* $\Theta_i \to \Theta_j$ *or an undirected edge* $\Theta_i - \Theta_j$*. A graph with all edges directed is called a directed graph, while one with all edges undirected is called undirected graph.*

Directed edges denote dependence, as in the parent node Θ_i conditions the child node Θ_j in $\Theta_i \to \Theta_j$. Undirected edges show the existence of a relationship between two nodes, but the direction of dependence is unknown.

As the aim of PGMs is to provide a graphical representations of a probabilistic model, it is important to note the relationships between a graph and a distribution. The first thing required is a definition of the independence $\boldsymbol{\Theta}$ assertions in a distribution.

Definition 2. *Let P be a distribution over X. We define* $\mathscr{I}(P)$ *to be the set of independence assertions of the form* $(\Theta_x \perp\!\!\!\perp \Theta_y \mid \Theta_z)$ *that hold in P.*

Let $\mathcal{G}$ be a graph and $\mathcal{I}(\mathcal{G})$ are its independence assertions. If all the assertions that are in $\mathcal{I}(\mathcal{G})$ exist in $\mathcal{I}(\mathcal{P})$ or equivalently $\mathcal{I}(\mathcal{G}) \subseteq \mathcal{I}(\mathcal{P})$, then $\mathcal{I}(\mathcal{G})$ is an I-map of $\mathcal{I}(\mathcal{P})$. This implies that $\mathcal{I}(\mathcal{P})$ could have more independence assertions that are not described in $\mathcal{G}$. More formally a graph is defined as follows

Definition 3. *Let $\mathcal{K}$ be any graph object associated with a set of independencies $\mathcal{I}(\mathcal{K})$. We say that $\mathcal{K}$ is an I-map of a set of independencies $\mathcal{I}$ if $\mathcal{I}(K) \subseteq \mathcal{I}$.*

If a distribution P factorizes as described by a graph G, then G is an I-map of P. It is important to understand that a distribution could have many different valid I-maps. So a graph that is an I-map of P is not a sufficient condition for a good graph model.

In more complicated graphs it is important to understand the "flow" of dependence between different nodes. Besides the obvious direct connection between two nodes $\Theta_i \rightarrow \Theta_j$ there are four basic indirect two-edge trails that are used as basic blocks for dependence. The four trails are the following as found in nodes in Fig. 2.1:

- **Causal Trail** A trail between three nodes, where the first node is not observed. $\Theta_1 \rightarrow \Theta_2 \rightarrow \Theta_9$: active if and only if Θ_4 is not observed.
- **Evidential Trail** A trail between three nodes, where the first node is observed. $\Theta_7 \rightarrow \Theta_8 \rightarrow \Theta_{11}$: Θ_7 is observed and is active if and only if Θ_8 is not observed.
- **Common Cause** A trail between three nodes, where the middle node is the parent and other two nodes are the children. $\Theta_{10} \leftarrow \Theta_6 \rightarrow \Theta_{11}$: active if and only if Θ_6 is not observed.
- **Common Effect** A trail between three nodes, where the middle node is the child, and the two other nodes are the parents. $\Theta_2 \rightarrow \Theta_9 \leftarrow \Theta_4$: active if and only if either Θ_9 or one of its descendants is observed.

In the more general case a trail between two nodes Θ_i, Θ_j is active if there is a path of interconnects nodes that connects them and the two following conditions

hold. Let Θ_o be a subset of observed variables, then:

- Whenever in the path between Θ_i and Θ_j there is a v-structure, like $\Theta_{l-1} \rightarrow \Theta_l \leftarrow \Theta_{l+1}$, then either Θ_l or one to its descendants are in Θ_o.
- No other node along the trail is in Θ_o.

With the above we can define two nodes as d-separated if there is not active trail between them or more formally:

Definition 4. *Let $\boldsymbol{\Theta}_x, \boldsymbol{\Theta}_y, \boldsymbol{\Theta}_z$ be three sets of nodes in $\mathscr{G}$. We say that $\boldsymbol{\Theta}_x$ and $\boldsymbol{\Theta}_y$ are d-separated given $\boldsymbol{\Theta}_z$, denoted as d-sep$_{\mathscr{G}}(\boldsymbol{\Theta}_x, \boldsymbol{\Theta}_y \mid \boldsymbol{\Theta}_z)$, if there is no active trail between any node $\Theta_x \in \boldsymbol{\Theta}_x$ and $\Theta_y \in \boldsymbol{\Theta}_y$ given $\boldsymbol{\Theta}_z$. The set of independences that correspond to d-separation is equivalent to $\mathscr{I}(\mathscr{G})$.*

In the current analysis of Position Location, undirected graphs will be used. Hence, a more in depth overview will be provided. For a much more complete discussion on PGMs and their properties [53, 54] are recommended reads.

2.1.2.1 Markov Networks

An undirected graph can be called a Markov network. Nodes represent variables and edges represent the existence of some form of direct probabilistic interaction between the variables. Remodelling our example as a Markov network, can be seen in Fig. 2.2, where all active trails as defined above need to be represented with edges, and all directional information has been lost.

It is not necessary for each node to have only one variable. The quite opposite in fact can be quite useful. So we define a factor ϕ to be a function of the values of a set of random variables $\boldsymbol{D}$ to $\mathbb{R}$. The set of variables $\boldsymbol{D}$ is called the scope of a factor.

As there is no real knowledge of the direction of dependence of two connected variables, it is easier to think of the connected nodes as trying to agree, or disagree depending on the relative values of the joint factor matrix. The values assigned to each factor should not be thought as CPDs but instead as a potential ϕ. As a result, in order to get information from a Markov network, all connected nodes have to be

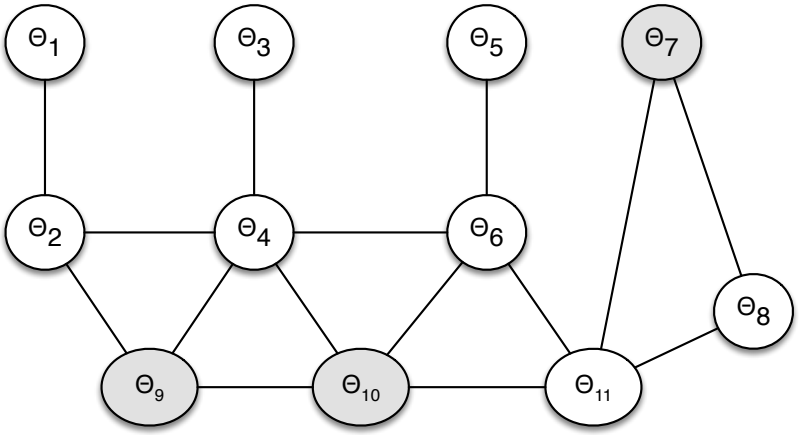

Figure 2.2: An undirected graph, based on Fig. 2.1

taken into account, instead of just the relevant CPDs. Continuing the example from Fig. 2.2 we have:

$$\Pr(\Theta_1,\ldots,\theta_{11}) = \frac{1}{Z}\phi_1(\Theta_1,\Theta_2,\Theta_9)\phi_2(\Theta_3,\Theta_4,\Theta_{10},\Theta_8))\phi_3(\Theta_5,\Theta_6)\phi_4(\Theta_7,\Theta_8,\Theta_{11}),$$

where Z is called the partition function. It is a normalising constant and is defined as

$$Z = \sum_{\Theta_i \in \{\Theta_1,\ldots\Theta_{11}\}} \phi_1(\Theta_1,\Theta_2,\Theta_9)\phi_2(\Theta_3,\Theta_4,\Theta_{10},\Theta_8)\phi_3(\Theta_5,\Theta_6)\phi_4(\Theta_7,\Theta_8,\Theta_{11}). \tag{2.2}$$

A factor contains variables that are fully interconnected between them. The subset of vertices that is maximally connected, for a given undirected graph, is defined as a clique.

Definition 5. *A Markov Random field can be defined as a set of distribution* $\Pr(\Theta_i \mid ne(\Theta_i))$*, where* $i \in 1,\cdots n$*, is the distribution index and* $ne(\Theta_i)$ *are the neighbouring nodes of variable* Θ_i*.*

2.1.3 Temporal Graphs

Another fundamental issue is the modelling of dynamic systems that change in time. In this case we assume a temporal model where the Probability Distribution Func-

tions (pdfs) of the state variables $\mathbf{\Theta}_i$ for node i, change as time passes. Hence, we denote $\mathbf{\Theta}_i^{(t)}$, which is now called a template variable and it is instantiated at different points in time t. Hence our goal is to represent the joint distribution of the random variables over their possible values for each relevant time (t).

Firstly, we discretize the timeline into a set of timeslots. Thus we are now only interested at the value of the random variables at a given timeslot t. Obviously the variables will be also dependent on their state at the previous timeslots. A further simplification can be made. We assume, that each state is dependent only on the previous respective timeslot, i.e. $\Pr(\mathbf{\Theta}_i^{(t)} \mid \mathbf{\Theta}_i^{(0:t-1)}) = \Pr(\mathbf{\Theta}_i^{(t)} \mid \mathbf{\Theta}_i^{(t-1)})$. Such a system is called Markovian, and this is called the Markov assumption. Assuming our example changes in time then we will get a temporal model such as the one in Fig. 2.3. As would be expected temporal models are a great tool for localizing mobility nodes and will be further discussed in Chapter 5.

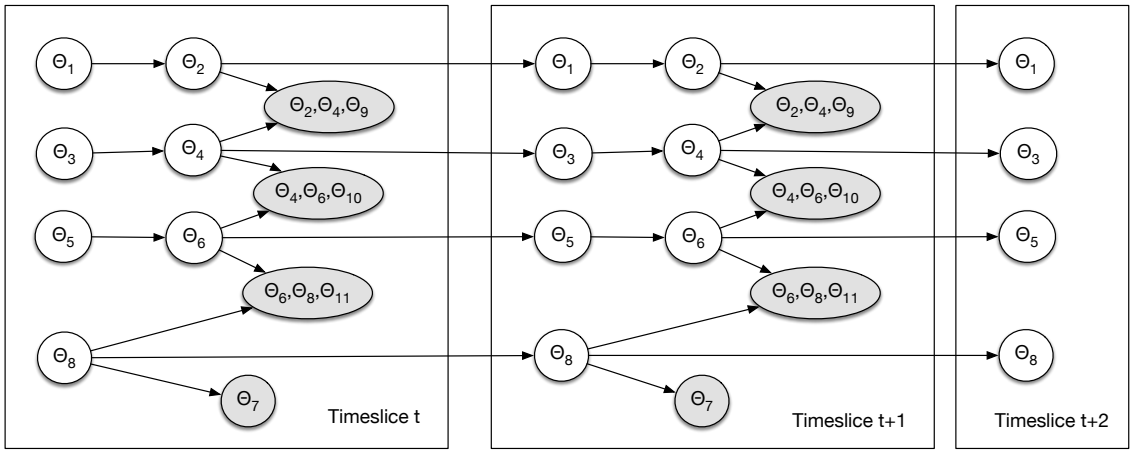

Figure 2.3: Temporal model example of Fig. 2.2

2.2 Inference

Inference is the act of obtaining relevant information about distributions out of a probabilistic model, such as the most likely state of a variable conditioned on the states of other variables, or the computation of a marginal distribution. In PGMs it is important that inference is detached from the design of the model, so even if the model changes, the same inference algorithm will still work. The section will focus

on the most typical type of query done in inference, the conditional query, i.e., find $\Pr(\boldsymbol{\Theta} \mid \boldsymbol{D} = \boldsymbol{d})$, where $\boldsymbol{d}$ represents an observed vector known for random variable vector $\boldsymbol{D}$.

2.2.1 Variable Elimination

During inference one of the most important steps is the marginalisation of variables. As marginalisation involves summing out all possible values of the variables in question, it can get easily computationally expensive, especially in highly complex models with a large number of variables. A typical way to marginalise in a graph is via Variable Elimination (VE).

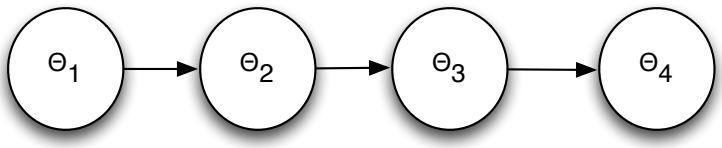

Figure 2.4: A simple directed chain.

Before discussing a more complicated example like Fig. 2.1, we will present VE in a easier example for clarity. Let there be a chain graphical model, i.e. the one in Fig. 2.4, and we want to obtain the marginal $\Pr(\Theta_4)$. By using VE the following

calculations are done

$$\Pr(\Theta_4) = \sum_{\Theta_1,\Theta_2,\Theta_3} \Pr(\Theta_1,\Theta_2,\Theta_3,\Theta_4) \tag{2.3}$$

$$= \sum_{\Theta_1,\Theta_2,\Theta_3} \Pr(\Theta_4 \mid \Theta_3) p(\Theta_3 \mid \Theta_2) \Pr(\Theta_2 \mid \Theta_1) \Pr(\Theta_1) \tag{2.4}$$

$$= \sum_{\Theta_3} \Pr(\Theta_4 \mid \Theta_3) \sum_{\Theta_2} \left\{ \Pr(\Theta_3 \mid \Theta_2) \underbrace{\left(\sum_{\Theta_1} \Pr(\Theta_2 \mid \Theta_1) \Pr(\Theta_1) \right)}_{\tau(\Theta_2)} \right\} \tag{2.5}$$

$$= \sum_{\Theta_3} \Pr(\Theta_4 \mid \Theta_3) \underbrace{\sum_{\Theta_2} \Pr(\Theta_3 \mid \Theta_2) \tau(\Theta_2)}_{\tau(\Theta_3)} \tag{2.6}$$

$$= \sum_{\Theta_3} \Pr(\Theta_4 \mid \Theta_3) \tau(\Theta_3). \tag{2.7}$$

Generalizing in an arbitrary large chain we would have

$$\Pr(\Theta_{n+1}) = \sum_{\Theta_n} \Pr(\Theta_{n+1} \mid \Theta_n) \Pr(\Theta_n). \tag{2.8}$$

So VE sums in steps following the nodes, instead of trying to sum everything at the same time. In each step a message $\tau()$ is calculated which will be used in the following node for the next calculations. In order to calculate $\tau()$, k^2 multiplications and $k \times (k-1)$ additions are required in order to sum up the corresponding entries, consequently the cost of each step is $\mathcal{O}(k^2)$, where k is the number of values variable Θ_n has. If there are N variables to be summed out then the total cost is $\mathcal{O}(Nk^2)$. The above process is called variable elimination. Alternatively if we calculated the joint distribution and then summed out the corresponding values we would require $\mathcal{O}(k^n)$ time, hence VE having a linear time cost is much more efficient.

More formally VE is defined as

Definition 6. *Let $\boldsymbol{\Theta}$ be some set of variables and let ϕ be a set of factors such that for each $\phi \in \Phi$, the scope of $\phi \subseteq \boldsymbol{\Theta}$. Let $\boldsymbol{I} \subset \boldsymbol{\Theta}$ be a set of query variables and let $\boldsymbol{K} = \boldsymbol{\Theta} - \boldsymbol{I}$ then for any ordering over K the variable elimination VE$(\boldsymbol{\Theta}, \boldsymbol{K})$ returns*

a factor $\phi^*(\boldsymbol{I})$ *such that:* $\phi^*(\boldsymbol{I}) = \sum_K \prod_{\phi \in \Phi} \phi$.

Finally, the same technique can be used in undirected graphs, with factors instead of probabilities, an idea which leads to message passing.

2.2.2 Message Passing

The whole idea of message passing is derived from the computations that need to be done "locally" at each step of VE. The graph can be considered as a railway system with nodes as stations and edges being the tracks. At each iteration the "messages" get transported to the next station, after they have been updated with any incoming information in the current station. Before a more in depth analysis, some important properties relevant to message passing in undirected graphs, need to be discussed.

2.2.2.1 Clique and Cluster Graph

We define a cluster graph as a set of factors Φ over $\boldsymbol{\Theta}$ in an undirected graph, each of whose nodes is associated with a subset $\boldsymbol{C}_i \subseteq \boldsymbol{\Theta}$. A cluster graph must be family preserving i.e. each factor $\phi \in \Phi$ must be associated with a cluster, denoted $a(\phi)$ such that scope$[\phi] \subseteq \boldsymbol{C}_i$. Each edge between a pair of clusters $\boldsymbol{C}_i$ and $\boldsymbol{C}_j$ is associated with a sepset $\boldsymbol{S}_{i,j} \subseteq \boldsymbol{C}_i \cap \boldsymbol{C}_j$, i.e. a subset of variables that is common to both clusters connected by the edge.

If there are no loops, the graph is called a clique tree. In clique trees, exact inference can be calculated. On the contrary, if there are loops, the graph model is called a cluster graph. In cluster graphs it is not possible to run exact inference, but often running message passing algorithms gives quite good approximate results.

An important property of cluster graphs is the running intersection property. This states that for a pair of clusters $\boldsymbol{C}_i, \boldsymbol{C}_j$, and variables $\boldsymbol{\Theta} \in \boldsymbol{C}_i \cap \boldsymbol{C}_j$ there must exist a unique path between $\boldsymbol{C}_i$ and $\boldsymbol{C}_j$ for which all clusters and sepsets contain $\boldsymbol{\Theta}$. Or simply put the set of clusters and sepsets in a graph that contain $\boldsymbol{\Theta}$ must form a tree.

Message passing will firstly be described in the context of clique trees and then generalised for the case of cluster graphs. In a clique tree the process is precisely the same as the variable elimination algorithm described above. Let T be a clique

tree with cliques $\boldsymbol{C}_1, \cdots, \boldsymbol{C}_k$. First all factors belonging to a clique are multiplied together in order to calculate the initial potentials, i.e., for clique $\boldsymbol{C}_i$,

$$\psi_i(\boldsymbol{C}_i) = \prod_{\phi: a(\phi)=j} \phi. \tag{2.9}$$

Due to the family preservation property we have

$$\prod_{\phi} \phi = \prod_{j} \psi_j. \tag{2.10}$$

A clique is selected as the root and VE is performed as described previously starting from the leaves and moving inward, towards the root. The message from clique c_i to clique c_j is calculated as follows

$$\delta_{i\to j}(\boldsymbol{S}_{ij}) = \sum_{\boldsymbol{C}_i - \boldsymbol{S}_{ij}} \psi_i(\boldsymbol{C}_i) \prod_{k\in \mathrm{ne}(i)-\{j\}} \delta_{k\to i}(\boldsymbol{S}_{ki}). \tag{2.11}$$

Each clique $\boldsymbol{C}_i$ is ready to transmit to a neighbouring clique $\boldsymbol{C}_j$, when it has received messages from all other neighbouring nodes, besides $\boldsymbol{C}_j$. Then it multiplies all incoming messages with its own initial potential, and sums out all variables that are not in the sepset $\boldsymbol{S}_{ij}$ between itself $\boldsymbol{C}_i$ and the clique $\boldsymbol{C}_j$, that is is going to transmit. Once the final message arrives to the root, the root clique will multiply it with its own potential. The result will be its belief denoted $\beta_r(\boldsymbol{C}_r)$ and

$$\tilde{P}_\phi(\boldsymbol{C}_r) = \sum_{\boldsymbol{\Theta} - \boldsymbol{C}_r} \prod_{\phi} \phi. \tag{2.12}$$

In order to calculate the belief for all cliques, each clique will need to multiply its own initial potential with all incoming messages from its adjacent cliques. As the root clique is the final clique to receive incoming messages, after one pass of the algorithm it is ready to calculated its belief. For the rest of the cliques, a "downwards" pass has to be done, starting from the root and ending at the leaves, for them to obtain the incoming messages that they were not calculated in the previous

"upwards" pass. Mathematically belief is

$$\beta_i(\boldsymbol{C}_i) = \psi_i(\boldsymbol{C}_i) \prod_{k \in \text{ne}(i)} \delta_{k \to i}(\boldsymbol{S}_{ki}). \tag{2.13}$$

Based on the above an alternative way to calculate an outgoing message given the belief would be to simply divide the message received from said clique, so that

$$\delta_{i \to j}(\boldsymbol{S}_{ij}) = \sum_{\boldsymbol{C}_i - \boldsymbol{S}_{ij}} \beta_i(\boldsymbol{C}_i) / \delta_{j \to i}(\boldsymbol{S}_{ji}). \tag{2.14}$$

This idea is extremely helpful in a distributed scenario. Consider a wireless sensors network, with a number of nodes. Then each clique is calculated at a different node, and it is much more convenient for nodes to broadcast their own belief, instead of having to compute a different message to transmit at each neighbour they have. Then the incoming message $\delta_{i \to j}$ at node j, will be calculated at the receiving node, by simply dividing by any outgoing message $\delta_{j \to i}$ the node j has from before.

2.2.3 Propagation-Based approximation

If instead of a clique-tree, a cluster graph is considered, the constraints that guarantee exact inference no longer hold and message passing algorithms no longer provide exact results. This is due to belief loops that overpower certain beliefs through constant feedback over others. Despite that, BP algorithms are perfectly applicable to cluster graphs, and can provide quite good approximations to the real beliefs. Due to the feedback loops, message passing is called loopy BP. In loopy BP, two passes are not adequate and the algorithm will need to iterate a number of times before it reaches convergence. Even so, there is the possibility that the algorithm will converge at all. The belief and message update equations become respectively

$$\beta_i^{(t)}(\boldsymbol{C}_i) = \psi_i(\boldsymbol{C}_i) \prod_{k \in \mathrm{ne}(i)} \delta_{k \to i}^{(t)}(\boldsymbol{S}_{ki}), \tag{2.15}$$

$$\delta_{i \to j}^{(t+1)}(\boldsymbol{S}_{ij}) = \sum_{\boldsymbol{C}_i - \boldsymbol{S}_{ij}} \psi_i(\boldsymbol{C}_i) \prod_{k \in \mathrm{ne}(i) - \{j\}} \delta_{k \to i}^{(t)}(\boldsymbol{S}_{ki}) = \tag{2.16}$$

$$= \sum_{\boldsymbol{C}_i - \boldsymbol{S}_{ij}} \psi_i(\boldsymbol{C}_i) \frac{\beta_i^{(t)}(\boldsymbol{C}_i)}{\delta_{j \to i}^{(t)}(\boldsymbol{S}_{ji})}, \tag{2.17}$$

where t denotes the iteration, and the parenthesis is used to distinguish the iteration from an exponent.

A cluster graph is said to be calibrated if for all clusters, for each edge connecting the clusters $\boldsymbol{C}_i$ and $\boldsymbol{C}_j$ the following equation holds

$$\sum_{\boldsymbol{C}_i - \boldsymbol{S}_{ij}} \beta_i^{(t)}(\boldsymbol{C}_i) = \sum_{\boldsymbol{C}_j - \boldsymbol{S}_{ij}} \beta_j^{(t)}(\boldsymbol{C}_j) \tag{2.18}$$

or in words the marginal beliefs of the sepsets between all adjacent cliques are equal.

Because there is no guarantee of the quality of the result, or even if there will be a result, the way the cluster graph is constructed becomes extremely important. In the case of clique trees, all valid graphs will lead to the correct results, with the possible increase in computational cost. In loopy BP though, the resulting beliefs could change drastically. There is a constant compromise between the computational cost and the accuracy of the results, the faster the propagation, the poorer the approximation.

2.2.4 Pairwise Markov Network

A pairwise Markov network is the simplest possible way to construct a valid cluster graph that follows both the family preservation and the running intersection properties. In a pairwise Markov network, each variable has its own univariate cluster. Then a pairwise cluster exists for all edges in the Markov network. By transforming the variables any distribution can be described as a pairwise Markov network.

2.2.5 Bethe cluster graph

A more general graph that can be used to model more complex interactions is the Bethe cluster graph, e.g. Fig. 2.5. This is a bipartite graph with one large and one small layer. The large layer consists of k clusters, where each cluster corresponds to a clique $\phi \in \Phi$, with a scope$[\phi]$. The second layer consists of small univariate clusters one for each variable. Each variable cluster is connected to all clusters in the large layer that have a common variable with it. This advantage of this cluster graph is that it is easy to construct and by design both the two properties will hold. In Fig. 2.5 we can see our example network formulated as a Bethe cluster graph.

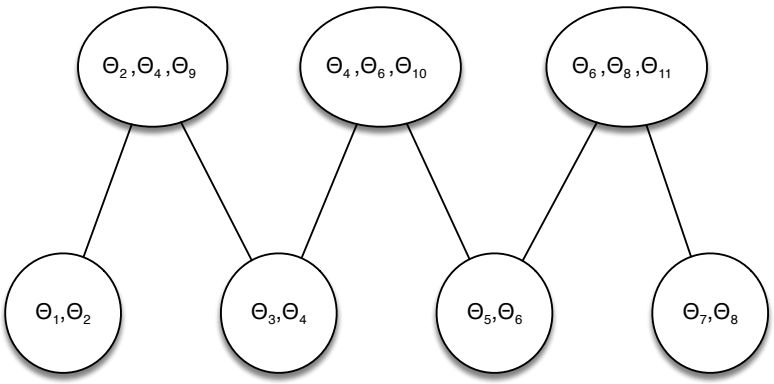

Figure 2.5: Example of a Bethe cluster graph.

2.3 Non-Parametric Density Estimation

Another important issue is how to model each factor distribution/potential in our graphs. If the distribution is discrete a matrix with the probabilities will suffice to describe the distribution, or in the case of a known distribution the parameters that define it can be used. In the more common case, where we have a number of observed events without any knowledge of the parameters of the underlying distribution, the use of non-parametric estimates becomes a valid alternative. Even though a non-parametric form will never be as accurate or as efficient as a valid closed form equation of the distribution it is often the only choice possible, as non-parametric forms require very few underlying assumptions. For an interesting discussion on kernel methods and their applications see [54].

2.3.1 KDE

A way to create a continuous density estimate from a set of observed variables is KDE. The basic idea is to use a function $K(\cdot)$, called the kernel on each data point in order to smooth its density over a region around it.

For L i.i.d. samples $\{\boldsymbol{\theta}_1, \cdots, \boldsymbol{\theta}_L\}$ we could create the KDE

$$\hat{\Pr}(\boldsymbol{\Theta}) = \frac{1}{L}\sum_{i=1}^{L} K_h(\boldsymbol{\Theta} - \boldsymbol{\theta}_i). \tag{2.19}$$

Typically the $K_h(\cdot)$ function is positive, symmetric and integrates to unity in order to create a density estimate. In practice a zero-mean spherically symmetric Gaussian kernel is a preferred choice, such as

$$K_h(\theta) = \mathcal{N}(\theta; 0; hI) \propto \exp(-\|\theta\|^2/2h), \tag{2.20}$$

where h controls the spread of influence that each observation has. There are a number of different methods that can be used to calculate h [55]. A simple heuristic is the *rule of thumb*. In this case the data is assumed to be derived from a Gaussian distribution and we have

$$\boldsymbol{\mu} = \frac{1}{N}\sum(\boldsymbol{\theta}_i) \tag{2.21}$$

$$\Sigma = \frac{1}{N}\sum(\boldsymbol{\theta}_i - \boldsymbol{\mu})(\boldsymbol{\theta}_i - \boldsymbol{\mu})^T \tag{2.22}$$

$$h_{RoT} = C_d \mathrm{Diag}(\Sigma) N^{-2/(4+d)}, \tag{2.23}$$

where a good approximation for C_d is $C_d \simeq 1 \quad \forall d$.

2.4 Non-Parametric Belief Propagation

NBP is a technique proposed by Ihler et al. in [55], and used in position location problems in [56, 15, 57]. The main idea is to use KDE to approximate belief and message distributions in a message passing algorithm. Messages are represented as a sample based density estimate, e.g., each sample can be a spherical kernel, as in eq. (2.20), with a weight, forming a Gaussian mixture approximation of the message

distribution. So given N weighted samples $\{w_n^{(k)}, \boldsymbol{\theta}_n^{(k)}, h^{(k)}\}$, where $n = 1, \ldots N$ and k represents the node index, that the message came from, there are two important operations that need to be considered for NBP.

1. **T**he computations of the updated beliefs as in eq. (2.15).

   The computation of the belief message involves calculating the product of all incoming messages with the nodes own potential. Assuming everything is approximated as a Gaussian mixtures, then we have K mixtures of N components each, allowing each mixture have the same number of components for simplicity, and we aim to calculate the product of the mixtures. However, calculating the N^k-component mixture is for most cases computationally infeasible so it is easier to use a sampling technique instead. Mathematically let the Gaussian mixture pdf for each message $i \in K$ be

   $$p_i(\boldsymbol{\Theta}) = \sum_{n \in N} w_{n,i} \mathcal{N}(\boldsymbol{\Theta}; \boldsymbol{m}_{n,i}, \Sigma_i), \tag{2.24}$$

   where $n \in N$, denotes the component. Our aim is to sample from the N^K component mixture density

   $$p(\boldsymbol{\Theta}) \propto \prod_{i=1}^{K} \mathrm{Pr}_i(\boldsymbol{\Theta}). \tag{2.25}$$

   There are many different techniques that can be used to sample from (2.25). As an example, the Mixture Importance Sampling (MIS) algorithm is shown, as it is commonly used in Position Location, cf. [15, 55, 58]. The main idea of importance sampling is instead of trying to sample from an intractable distribution $p(\boldsymbol{\Theta})$, we sample from a proposal distribution $q(\boldsymbol{\Theta})$, that is easy to handle. We assume that both $p(\boldsymbol{\Theta})$ and $q(\boldsymbol{\Theta})$ can be evaluated up to a normalisation constant. Then $kN \geq N$ $\boldsymbol{\Theta}_j$ samples are drawn from $q(\boldsymbol{\Theta})$ and they are assigned a weight given by $w_j \propto \frac{p(\boldsymbol{\Theta}_j)}{q(\boldsymbol{\Theta}_j)}$. The weights are then normalised by their sum $Z = \sum w_j$ and N samples are drawn with replacement with probability equal to $p(\boldsymbol{\Theta}_j) = \frac{w_j}{Z}$. In the case of MIS a mixture is randomly selected

from the K mixtures, and then a sample is taken from its N components. This is equivalent to having as a proposal distribution the average of the mixtures, i.e.,

$$q(\mathbf{\Theta}) = \frac{1}{K}\sum_i p_i(\mathbf{\Theta}). \tag{2.26}$$

2. **T**he computation of the updated messages as in eq. (2.16).

   Given the incoming messages $\delta_{k\to i}(\mathbf{\Theta}_k) = \{w_n^{(k)}, \boldsymbol{\theta}_n^{(k)}, h^{(k)}\}, \forall k \in \text{ne}(i)$ ~~/~~, we aim to calculate $\delta_{i\to j}(\mathbf{\Theta}_i)$ The trick here is instead of calculating the marginal, doing the summation in (2.16), or integral in the continuous case, to assume that all variables not in the sepset are fixed and draw samples $\mathbf{\Theta}_j$ from the conditional distribution, e.g. in this case $p(\mathbf{\Theta}_j \mid \mathbf{\Theta}_i, \mathbf{\Theta}_x)$, where $\mathbf{\Theta}_x$ includes all other possible variables that need to be marginalised, which in this case would be the parameters of the Gaussian mixtures, which are already known from the samples. The samples are then weighted by the remainder $\frac{w_n^k}{\delta_{j\to i}(\mathbf{\Theta}_j)}$, and sampled again with replacement based on the weights.

2.5 Clustering Algorithms

Clustering Algorithms in machine learning are used to group together, i.e., form clusters of data points with similar properties. This allows us to create categories for the data points, even without any prior information about which groups they belong to and consequently each datapoint is assigned to the most probable category. Examples range from the automatic grouping of published articles, based on their keywords, an online date matching algorithm, or generally every algorithm that matches information into groups, e.g., google news.

In our current discussion though, we take advantage of another use of clustering algorithms, which is the ability to derive an approximate parametric distribution, from a number of non-parametric samples. A simple example would be if there was a number of datapoints all packed close together. Then we could approximate them with a Gaussian distribution, and we would need to find the sufficient statistics, i.e.,

the mean and the variance, in order to get a parametric approximation of all the data points. This allows us to use non-parametric techniques to model some distributions and then use clustering to compact the resulting information in a parametric form which is more compact and easier to transmit. Two of the most common clustering techniques are K-Means and the Expectation-Maximization Algorithm (EM). K-means is a non-probabilistic, "hard-decision" clustering algorithm, while EM is a probabilistic "soft-decision" algorithm. "Hard decision" means that each datapoint is assigned to only one cluster, while in "soft-decision", each datapoint is assigned weights for all clusters.

2.5.1 K-Means

Consider the following problem. There is a dataset of N observed d-dimensional data points, i.e. $\{\boldsymbol{\theta}_1, \dots \boldsymbol{\theta}_N\}$ and our goal is to group them into K categories, or clusters. Each cluster k will get assigned to it all data points that are close to some metric property $\boldsymbol{\mu}_k$, that intuitively can be the centre of the cluster. For each datapoint we define a binary variable $r_{nk} \in \{0,1\}$, which is equal to 1, when datapoint n belongs to cluster k, and 0 otherwise. As this is a hard-decision algorithm each datapoint can only belong to one cluster at any given moment, the one that has a μ_k closest to it. With that in mind we define a cost function that out aim is to minimise

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} ||\boldsymbol{\theta}_n - \boldsymbol{\mu}_k||^2. \tag{2.27}$$

The cost function J represents the sum of the squared distances of all data points to their assigned clusters. In order to minimise it, we need to find the optimal r_{nk} and $\boldsymbol{\mu}_k$. This can be done in a two-step iterative process, where we keep r_{nk} fixed and optimise $\boldsymbol{\mu}_k$, and vice versa until the variables converge or a predefined number of iterations passes. Assuming that we have some initial $\boldsymbol{\mu}_k$, we start by trying to find the optimal r_{nk}, given our initial $\boldsymbol{\mu}_k$. Mathematically this is formulated as

$$r_{nk} = \begin{cases} 1 & \text{if} \quad k = \arg\min_j ||\boldsymbol{\theta}_n - \boldsymbol{\mu}_j||^2 \\ 0 & \text{otherwise} \end{cases} \tag{2.28}$$

or more simply we assign each datapoint $\boldsymbol{\theta}_n$ to the $\boldsymbol{\mu}_k$ closest to it.

Now having optimized r_{nk}, we consider it fixed and optimise $\boldsymbol{\mu}_{nk}$. By taking the derivative of the cost function and assigning to zero, we can solve for $\boldsymbol{\mu}_k$ and we have

$$2\sum_{n=1}^{N} r_{nk}(\boldsymbol{\theta}_n - \boldsymbol{\mu}_k) = 0 \Rightarrow \tag{2.29}$$

$$\mu_k = \frac{\sum_{n=1}^{N} r_{nk}\theta_n}{\sum_{n=1}^{N} r_{nk}}, \tag{2.30}$$

which is simply the mean of all data points assigned to group k.

As J decreases in each iteration, convergence is guaranteed although it might be a local minimum instead of a global one. The end result is highly dependent on the initial values of $\boldsymbol{\mu}_k$, so an option would be to run the algorithm multiple times with different initial values, and keep the result with the minimum cost J.

2.5.2 Expectation-Maximization Algorithm

The EM algorithm generalises means in a probabilistic context. A distribution is proposed with different components, characterised by parameters and the algorithm tries to find the optimal parameter values for each component and respective weights for each datapoint for each component, in order to maximise the log likelihood function of the data points. In the general case we assume a joint probability distribution $\Pr(\boldsymbol{D} = \boldsymbol{d}, \boldsymbol{S} \mid \boldsymbol{\Theta})$, where $\boldsymbol{d}$ is the observation vector of $\boldsymbol{D}$. $\boldsymbol{S}$ represents hidden variables and $\boldsymbol{\Theta}$ are the sufficient statistics parameters, that characterise the distribution. The algorithm aims to maximise $\Pr(\boldsymbol{D} \mid \boldsymbol{\Theta})$ with respect to $\boldsymbol{\Theta}$.

This is accomplished in a similar two-step iterative process as the one described previously in K-means. Firstly initial values for $\boldsymbol{\Theta}$ are chosen, and then the algorithm iterates between optimising either the $\boldsymbol{S}$ hidden variables, or the $\boldsymbol{\Theta}$ parameters, by maximising the complete-data loglikelihood, until either the parameters or the loglikelihood have converged.

Assuming initial $\boldsymbol{\Theta}^0$, the first step is the E step where the posterior of the latent

variables is $\Pr(\boldsymbol{S} \mid \boldsymbol{D}, \boldsymbol{\Theta})$ is calculated. Then in the M step new parameters $\boldsymbol{\Theta}^{\text{new}}$ are calculated by

$$\boldsymbol{\Theta}^{\text{new}} = \arg\max_{\boldsymbol{\Theta}} Q(\boldsymbol{\Theta}, \boldsymbol{\Theta}^{\text{old}}), \tag{2.31}$$

where $Q(\boldsymbol{\Theta}, \boldsymbol{\Theta}^{\text{old}})$ is the expectation of the complete-data loglikelihood and is defined as

$$Q(\boldsymbol{\Theta}, \boldsymbol{\Theta}^{\text{old}}) = \sum_{\boldsymbol{S}} \Pr(\boldsymbol{S} \mid \boldsymbol{D} = \boldsymbol{d}, \Theta^{\text{old}}) \ln \Pr(\boldsymbol{D} = \boldsymbol{d}, \boldsymbol{S} \mid \boldsymbol{\Theta}). \tag{2.32}$$

EM can also be used to find Maximum A Posteriori (MAP) solutions for models, where the parameters $\boldsymbol{\Theta}$ are probabilistic and are defined by a prior distribution $\Pr(\boldsymbol{\Theta})$. In this case in the M step the quantity to be maximised is $Q(\boldsymbol{\Theta}, \boldsymbol{\Theta}^{\text{old}}) + \ln(\boldsymbol{\Theta})$.

As a more concrete example, we consider a dataset that we are trying to fit in a Gaussian mixture distribution. Again let the observed data set of the random variable $\boldsymbol{D}$ be $\boldsymbol{d}$. We assume that the dataset is separated into K clusters and therefore we assume K components in the distribution. A Gaussian mixture can be defined as

$$\Pr(\boldsymbol{D}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{D} \mid \boldsymbol{\mu}_k, \Sigma_k). \tag{2.33}$$

Let us define a latent k-dimensional binary variable S_k, where one element will be 1 and everything else will be equal to 0. Therefore, $S_k \in \{0, 1\}$ and $\sum_k S_k = 1$. We define the joint distribution $\Pr(\boldsymbol{D}, \boldsymbol{S})$ in terms of a marginal distribution $\Pr(\boldsymbol{S})$ and a conditional distribution $\Pr(\boldsymbol{D} \mid \boldsymbol{S})$. The marginal distribution over $\boldsymbol{S}$ is specified by the mixing coefficients of the Gaussian mixture distribution, such that

$$\Pr(S_k = 1) = \pi_k, \tag{2.34}$$

where $0 \leq \pi_k \leq 1$ and $\sum_k \pi_k = 1$ or $\Pr(\boldsymbol{S})$ can be also written as

$$\Pr(\boldsymbol{S}) = \prod_{k=1}^{K} \pi_k^{S_k}. \tag{2.35}$$

Similarly, the conditional distribution can be written as

$$\Pr(\boldsymbol{D} \mid \boldsymbol{S}) = \prod_{k=1}^{K} (\mathcal{N}(\boldsymbol{D}; \mu_k, \Sigma_k))^{s_k}. \tag{2.36}$$

Therefore, by the chain rule, we have

$$\Pr(\boldsymbol{S}, \boldsymbol{D}) = \Pr(\boldsymbol{D} \mid \boldsymbol{S}) \Pr(\boldsymbol{S}) \Rightarrow \tag{2.37}$$

$$\Pr(\boldsymbol{D}) = \sum_{S} \Pr(\boldsymbol{S}, \boldsymbol{D}) \tag{2.38}$$

$$= \sum_{S} \Pr(\boldsymbol{D} \mid \boldsymbol{S}) \Pr(\boldsymbol{S}) \tag{2.39}$$

$$= \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{D}; \boldsymbol{\mu}_k, \Sigma_k). \tag{2.40}$$

Finally, the posterior probability $\Pr(S_k = 1 \mid \boldsymbol{D})$ can be defined as

$$r(S_k) = \Pr(S_k = 1 \mid \boldsymbol{D}) = \frac{\Pr(S_k = 1) \Pr(\boldsymbol{D} \mid S_k = 1)}{\sum_{j=1}^{K} \Pr(S_j) \Pr(\boldsymbol{D} \mid S_j)} \tag{2.41}$$

$$= \frac{\pi_k \mathcal{N}(\boldsymbol{D}; \mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\boldsymbol{D}; \mu_j, \Sigma_j)}. \tag{2.42}$$

The posterior plays the same role of weight as r_{nk} played in the means, but in a "soft manner". Basically it shows the probability of a specific datapoint $\boldsymbol{d}_n$ belonging to each mixture in the distribution, and is called responsibility. The log-likelihood of the mixture can be defined as

$$\ln \Pr(\boldsymbol{D} = \boldsymbol{d}; \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{d}_n; \mu_k, \Sigma_k) \right\}. \tag{2.43}$$

In order to optimise the loglikelihood, we set the derivative with respect to the

parameters to zero and get

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{n=1}^{N} r(s_{nk}) \boldsymbol{d}_n, \tag{2.44}$$

$$\Sigma_k = \frac{1}{N_k} \sum_{n=1}^{N} r(s_{nk}) (\boldsymbol{d}_n - \boldsymbol{\mu}_k)(\boldsymbol{d}_n - \boldsymbol{\mu}_k)^T, \tag{2.45}$$

where

$$N_k = \sum_{n=1}^{N} r(s_{nk}). \tag{2.46}$$

In order to find the optimal π_k, the added constraint of $\sum_k \pi_k = 1$ needs to be taken into account, so a Lagrange multiplier is added to the loglikelihood before the derivative is taken. Equating with zero and solving for π_k gives us

$$\pi_k = \frac{N_k}{N}. \tag{2.47}$$

The EM algorithm will iterate over the two steps, First in the expectation step keeping the parameters fixed, and calculating the responsibilities (2.42). Then the new parameters will be computed by using equations (2.44), (2.45), (2.46) and (2.47). Then the log likelihood (2.43) is calculated. This continues until either the parameters or the loglikelihood converge or a number of predefined iterations passes.

EM generally converges quite quickly as far as optimising the loglikelihood goes. On the other hand the parameters depending on the optimisation might vary wildly as more iterations pass. Again the end result is not guaranteed to be a global optimum, and hence multiple instance of the algorithm need to be run with different starting parameters, and the best one kept. As K-means is much cheaper computationally, one way to get good initial parameters would be to derive them from running K-means first. Also EM suffers from singularity issues, i.e. clusters can , which need to be accounted for explicitly in the code. Finally, another issue is the implicit assumption that the number of clusters K is known beforehand. Therefore

variations have been developed like the variational Bayes (VB) that will be used in HEVA presented in Chapter 3, that overcome these limitations. Despite the above, EM due to its simplicity and generality has been widely used in the last few years in a large variety of different issues, with high success.

2.6 Contraction Mappings

Message passing can be seen as an iterative method trying to solve a non-convex problem. Obviously analysing the convergence properties of such methods is quite important, as they answer if a fixed point, hopefully one that is an optimal solution, can be found, and how long will it take to find it. One of the principal techniques used for convergence analysis is contraction mappings.

In order to define contraction mappings we will assume that the iterative algorithm is one of several that can be written in the form

$$\boldsymbol{x}^{(t+1)} = \mathrm{T}(\boldsymbol{x}^{(t)}) \quad \text{for} \quad t = 0,1,2,\ldots \tag{2.48}$$

where $\mathrm{T}(\cdot)$, is a mapping from a subset $\boldsymbol{X}$ in $\mathbb{R}^N$ into itself and has the property

$$\|\mathrm{T}(\boldsymbol{x}) - \mathrm{T}(\boldsymbol{y})\| \leq a\|\boldsymbol{x} - \boldsymbol{y}\|, \quad \forall \boldsymbol{x}, \boldsymbol{y} \in \boldsymbol{X}, \tag{2.49}$$

where $\|\cdot\|$ is a norm, and a is a constant called modulus of T and we have $a \in [0,1)$. Let there be a mapping $\mathrm{T} : \boldsymbol{X} \longmapsto \boldsymbol{X}$. Then any vector $\boldsymbol{x}^* \in \boldsymbol{X}$ satisfying $\mathrm{T}(\boldsymbol{X}^*) = \boldsymbol{x}^*$ is called a fixed point of T and the iteration $x := \mathrm{T}(\boldsymbol{x})$ can viewed as an algorithm trying to find a fixed point.

Theorem 2.6.1. *(Convergence of Contracting Iterations) Suppose that $T : \boldsymbol{X} \longmapsto \boldsymbol{X}$ is a contraction with modulus $a \in [0,1)$ and that X is a closed subset of $\mathbb{R}^n$. Then:*

- *(Existence and Uniqueness of Fixed Points) The mapping T has a unique fixed point $\boldsymbol{x}^* \in \boldsymbol{X}$.*
- *(Geometric Convergence) For every initial vector $\boldsymbol{x}^{(0)} \in \boldsymbol{X}$, the sequence*

$\{\boldsymbol{x}^{(t)}\}$ generated by $\boldsymbol{x}^{(t+1)} = T(\boldsymbol{x}^{(t)})$ converges to $\boldsymbol{x}^*$ geometrically. In particular,

$$\|\boldsymbol{x}^{(t)} - \boldsymbol{x}^*\| \leq a^t \|\boldsymbol{x}^{(0)} - \boldsymbol{x}^*\|, \quad \forall t \geq 0. \tag{2.50}$$

Convergence analysis on tree PGMs is trivial as the problem is convex and an optimal solution is guaranteed to be found. Unfortunately in loopy graphs, this is not the case and there is no guarantee that a message passing algorithm will find an optimal solution or even converge. Sufficient conditions for message passing algorithms to converge has been investigated in [59, 60, 61]. Convergence will be discussed in more detail in Chapter 6.

2.7 Conclusions

In this chapter, the main mathematical tools that will be used in the following chapter were presented. Firstly the way to design a PGM from probabilistic variables was presented in Section 2.1. Directed and Markov graphs were discussed and their differences as well as temporal graphs that model dynamic networks with variable distributions that change over time. In Section 2.2, the main techniques on conducting inference on PGMs was presented using VE and message passing. Also the Bethe cluster graph was presented as the simplest model that satisfies the necessary properties to use message passing. The next issue discussed, in Section 2.3 was how to model distributions that are continuous and do not fit any of the standard pdf parametric forms. Followed by a general message passing algorithm, namely NBP in Section 2.4 that can handle non-parametric forms of distributions. The way to approximate non-parametric forms with parametric ones using clustering algorithms was presented in Section 2.5. Finally, the theory of Contraction Mapping was briefly discussed in Section 2.6 and how it is used to analyze the convergence properties of message passing algorithms in graphs. In the following chapters the above tools will be used and extended in order to tackle cooperative localization.

Chapter 3

Hybrid Variational Ellipsoid Algorithm

3.1 Introduction

In this chapter we present a novel message passing algorithm for cooperative localisation, named HEVA. HEVA is designed so that it decreases the computational cost of NBP as well as the communication overhead, catering for 3D localisation, where complexity cost drastically increases due to the extra dimension. HEVA overcomes the pitfalls of the aforementioned algorithms and is computationally tractable in the 3D case. Our proposed algorithm can achieve higher localisation accuracy, getting more nodes self-localized within given accuracy and at much less complexity than NBP [15]. Although IPPM in [14] and LS in [9] are less complex, their accuracies are much inferior, especially under high noise environments. HEVA combines the strengths of both VMP and NBP to reach a fine balance between the information used in the probabilistic inference, the message size and convergence speed. HEVA manages to minimize the amount of information lost in the approximations in the inference calculations and as such localizes with high resolution. HEVA also has a simple NLoS mitigation filter that removes unhelpful NLoS messages when using ToA distance measurements.

3.1.1 Our Contributions

Our contributions can be summarized as follows:

- We devise an ellipsoid particle filter that greatly reduces the number of particles required for accurate product calculations in BP,

- We devise a low complexity NLoS mitigation message filter for ToA that removes NLoS messages with no requirement for prior environment information.

- We propose a simple kernel alternative suitable for non parametric forms of localization pdfs used in message passing algorithms.

- We present HEVA a novel algorithm that combines the above with VB. HEVA allows high accuracy in 3D indoors positioning with low complexity and low communications overhead.

3.1.2 Organization

The remainder of this chapter is organized as follows. Section 3.2 formulates the Bayesian localisation problem. Section 3.3 describes the HEVA algorithm. Simulation results are provided in Section 3.4 and finally conclusions are drawn in Section 3.5.

3.2 Problem Formulation

We consider a set of nodes in a 3D environment of size $X \times Y \times Z$ m^3. The nodes consist of N agents and M anchors, where $|\mathscr{V}| = N+M$, $M \geq 4$ and $N \gg M$.[1] Let $\boldsymbol{\theta} = [\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_i, \ldots, \boldsymbol{\theta}_{N+M}]$ denote the locations of all nodes, with $\boldsymbol{\theta}_i = [x_i, y_i, z_i]$ being the coordinates of node i, and $\boldsymbol{\Theta}_i$ be the respective random variable. The nodes communicate wirelessly and it is assumed that the maximum communication range for each node is $r_{\max}$. Time is slotted and time slots are denoted by the time index superscript (t) for $t = 1, \ldots, \infty$.

A network of such can be viewed as a graph. The wireless nodes are represented by the set $\mathscr{V}$ of vertices of the graphical model. If two nodes, say node i and node j, are within range, there will be an edge $e_{ji} \in \mathscr{E}$, connecting the two nodes.

[1] Typically the number of agents is assumed much larger than the number of anchors [9].

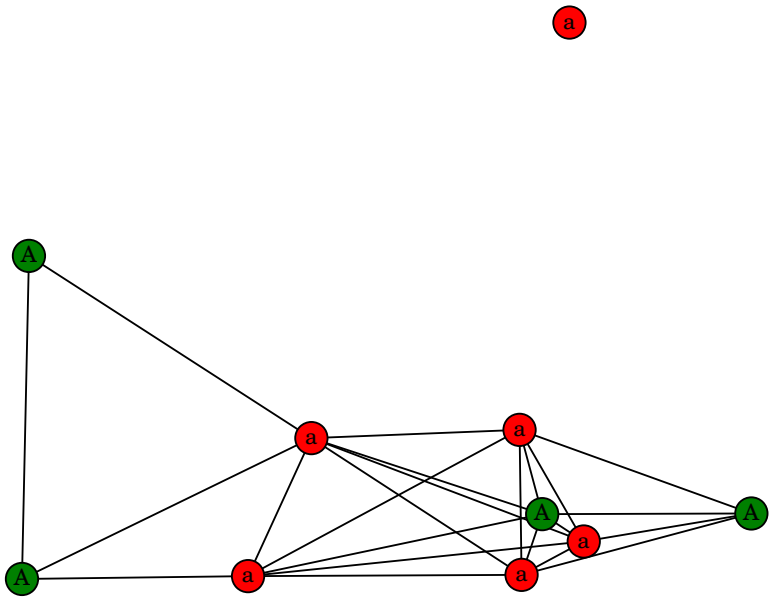

Figure 3.1: Example of a wireless sensors network. Agents are represented by lowercase "a" (red circles), anchors are represented by capital "A" (green circles) and branches (or edges) correspond to communication between them.

The set of all nodes j with edges e_{ji} to node i is denoted as the neighbourhood $\text{ne}(i)$. A simple network graph example with agents, represented as "a" labelled red circles and anchors, represented as "A" labelled green circle, is depicted in Fig. 3.1.

Let $\Pr^{(t)}(\boldsymbol{\Theta}_i)$ be the pdf, i.e., the belief that node i has about its location at time t. Initially, the belief for the agents can be an information-less uniform pdf over the grid, while the anchors' pdfs would be spherical multivariate Gaussians with mean being their exact locations and a covariance matrix of $\sigma_a^2 \boldsymbol{I}$ for some noise power σ_a^2.

Nodes can measure a corresponding distance estimate via ranging. For example, node i receiving a message from node j at time slot t can derive a noisy estimate $r_{j\to i}^{(t)}$ of the distance between them. It is assumed that the measurement taken from node i receiving a message from node j and the one from node j receiving a message from node i are the same and as a result, the direction of the message plays no role, i.e., $r_{j\to i}^{(t)} = r_{i\to j}^{(t)} = r_{ji}^{(t)}$. In practice, distance measurements will differ and the

nodes can share their measurements and use the average. We assume that the nodes use ToA distance measurements, as in [24] and we define the random variable r_{ji} as

$$r_{ji} = d_{ji} + \eta_{ji} \equiv \|\boldsymbol{\theta}_i - \boldsymbol{\theta}_j\| + \eta_{ji}, \tag{3.1}$$

where η_{ji} is a noise factor following a Gaussian distribution with variance $K_e\|\boldsymbol{\theta}_i - \boldsymbol{\theta}_j\|^{\beta_{ji}}$, in which K_e is a proportionality constant capturing the combined physical layer and receiver effect [22], and β_{ji} denotes the path loss exponent. In the case of line-of-sight (LoS) η_{ji} is assumed to have zero mean, and $\beta_{ji} = 2$, i.e., $\eta_{ji} \sim \mathcal{N}(0, K_e\|\boldsymbol{\theta}_i - \boldsymbol{\theta}_j\|^2)$. Alternatively, in the case of NLoS, the Gaussian random variable has a positive mean $b_{ji} \gg s_{ji}^2$, where $b_{ji} \sim \mathcal{U}(\frac{d_{ji}}{3}, \frac{2d_{ji}}{3})$ (i.e., a uniform distribution) and $\beta_{ji} = 3$, i.e., $\eta_{ji} \sim \mathcal{N}(b_{ji}, K_e\|\boldsymbol{\theta}_i - \boldsymbol{\theta}_j\|^3)$.

Let $\Pr^{(t)}(R_{ji} = r_{ji}^{(t)}|\boldsymbol{\Theta}_i, \boldsymbol{\Theta}_j)$ be the Conditional Probability Distribution Function (cpdf) of observing distance $r_{ji}^{(t)}$ at time slot t, given the location beliefs $\Pr^{(t)}(\boldsymbol{\Theta}_i)$ and $\Pr^{(t)}(\boldsymbol{\Theta}_j)$ for node i and node j, respectively. Assuming statistically independent noise and statistically independent priors between nodes, the joint pdf of the whole probabilistic model with $\boldsymbol{\vartheta} \triangleq (\{\forall \boldsymbol{\Theta}_i \in \mathcal{V}\})$ and $\boldsymbol{R} \triangleq (\{R_{ji}\}_{\forall i,j \in \mathcal{E}})$, for a given time slot t, is found as

$$\overset{(t)}{\Pr}(\boldsymbol{\vartheta}, \boldsymbol{R}) = \overset{(t)}{\Pr}(\boldsymbol{R}|\boldsymbol{\vartheta}) \overset{(t)}{\Pr}(\boldsymbol{\vartheta}) = \prod_{i,j \in \mathcal{E}} \overset{(t)}{\Pr}(R_{ji} = r_{ji}|\boldsymbol{\Theta}_i, \boldsymbol{\Theta}_j) \prod_{\boldsymbol{\Theta}_i \in \mathcal{V}} \overset{(t)}{\Pr}(\boldsymbol{\Theta}_i). \tag{3.2}$$

Our aim is to find the most probable position for every node given the observed positions and prior information, i.e., find the maximum a posteriori (MAP)

$$\hat{\boldsymbol{\vartheta}} = \arg\max_{\boldsymbol{\vartheta}} \overset{(t)}{\Pr}(\boldsymbol{\vartheta}|\boldsymbol{R} = \boldsymbol{r}), \tag{3.3}$$

where $\boldsymbol{r}$ is a vector representing all the observed distances. To accomplish this, we employ a message passing algorithm [53], in which information from the graph can be summarized in local edge information, allowing for an efficient distributed algorithm, despite its lack of guaranteed optimal solution or even convergence for the given random graph geometry.

3.2.1 Belief Message Passing

We model the network as a Bethe cluster graph, cf. Section 2.2.5, and we apply a loopy belief message passing algorithm. Let the lower factors, which represent the node beliefs, be composed of univariate potentials $\psi(\mathbf{\Theta}_i)$, whereas the upper region, which represents the interactions between the node variables is composed of "large" factors equal to $\psi(\mathbf{\Theta}_i, \mathbf{\Theta}_j, r_{ji})$. An example of the network in Fig. 3.1 is shown in Fig. 3.2. The lower factors are set to the initial node beliefs for the given time slot (t), and the upper factors to the corresponding cpdfs, i.e.,

$$\psi(\boldsymbol{\vartheta}_i) = \overset{(t)}{\Pr}(\mathbf{\Theta}_i), \tag{3.4}$$

$$\psi(\mathbf{\Theta}_i, \mathbf{\Theta}_j, r_{ji} = r_{ji}^{(t)}) = \overset{(t)}{\Pr}(r_{ji} = r_{ji}^{(t)} | \mathbf{\Theta}_i, \mathbf{\Theta}_j). \tag{3.5}$$

Messages are then passed between nodes for multiple iterations until the node beliefs have converged, or a predetermined number of iterations has passed. We define the message passing equation (2.17), from node j to node i at BP iteration $(s+1)$ by

$$\delta_{j\to i}^{(s+1)}(\mathbf{\Theta}_i) = \int \psi(\mathbf{\Theta}_i, \mathbf{\Theta}_j, R_{ji} = r_{ji}^{(t)}) \frac{b_{j\to i}^{(s)}(\mathbf{\Theta}_j)}{\delta_{i\to j}^{(s)}(\mathbf{\Theta}_j)} \mathrm{d}\mathbf{\Theta}_j, \tag{3.6}$$

where $r_{ji}^{(t)}$ is the observed value of the distance between the nodes, at time slot t. Intuitively, a message $\delta_{j\to i}^{(s+1)}(\mathbf{\Theta}_i)$ is the belief that node j has about the location of node i and is a function of $\mathbf{\Theta}_i$. After it calculates all the incoming messages, each node updates its belief by (2.15), i.e.

$$b_i^{(s+1)}(\mathbf{\Theta}_i) = \lambda \psi(\mathbf{\Theta}_i) \prod_{k \in \mathrm{ne}(i)} \delta_{k\to i}^{(s+1)}(\mathbf{\Theta}_i) + (1-\lambda) b_i^{(s)}(\mathbf{\Theta}_i), \tag{3.7}$$

in which, $\lambda \in [0,1]$ is a dampening factor used to facilitate convergence, c.f. [53].[2] The algorithm continues until convergence or a set number of iterations $s_{\max}$ has elapsed. At that point, the belief that represents an approximation to the true

[2]Dampening is not required for 2D localisation, but simulations show that it helps in 3D localisation where the increased ambiguity results in oscillating behaviours more often.

marginal, for each node is

$$p^{(t+1)}(\mathbf{\Theta}_i) = b_i^s(\mathbf{\Theta}_i). \tag{3.8}$$

This can be thought of as a process of merging every node's belief about a specific node's location to get the best estimate. This message passing analysis naturally leads to a distributed cooperative system because each node is only required to do calculation concerning its local factors and messages.

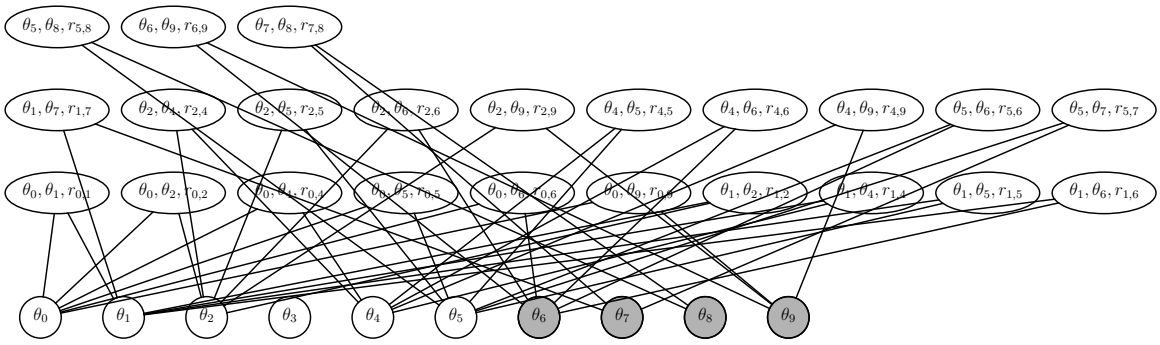

Figure 3.2: A Bethe cluster graph of the network in Fig. 3.1.

A major issue in calculating the messages (3.6) and (3.7) is the computational cost. If $\mathbf{\Theta}_i$ can take one of D discrete values, then the messages and marginals in the 3D case will be represented by dimensional vectors of cardinality D. The integral in (3.6) becomes a matrix-vector product and this in general requires $\mathcal{O}(D^2)$ operations [62], making the problem intractable because the dimensions of the grid and the resolution of the required localisation will increase quadratically. This makes the use of approximation methods necessary.

In the literature, various methods have been proposed, such as particle filtering methods [63, 64], Monte Carlo methods [16], and the more general NBP, e.g., [57, 15, 65]. They try to approximate the cpdf by a Gaussian mixture of particles, achieving a complexity of $\mathcal{O}(L^2)$, where L denotes the number of particles, and $L \ll D$. If the density of the number of particles is to remain approximately the same in the 3D case as in 2D, then the number of particles will increase by a factor

of $2d_{ji}$ (d_{ji} being the exact distance). This phenomenon is known as the curse of dimensionality [66]. This would require finding the product of large Gaussian mixtures, which is already an expensive operation and can quickly become prohibitively expensive. Also it requires a considerable amount of overhead, since the parameters for all the components of each particle message need to be communicated. Another important issue is the convergence of the algorithm, although this thesis does not directly tackle this issue. Due to the loopy characteristics of the graph, beliefs of factors will move around closed loops, creating opportunities for some factors to overpower their beliefs and oscillations to occur in the graph, essentially affecting the convergence. In practice, most nodes would actually converge to a local optimum with only a few nodes oscillating unable to converge [53]. In the next section, we will present HEVA to overcome those critical issues.

3.3 HEVA

3.3.1 Overview

HEVA is a hybrid method that combines elements of both parametric and non-parametric approximations for optimizing the local computational cost at each node, the communication overhead, and the convergence speed. In HEVA, the factors of $\psi(\mathbf{\Theta}_i)$ are first approximated by a Gaussian mixture, while the factors of the cpdfs $\psi(\mathbf{\Theta}_i, \mathbf{\Theta}_j, R_{ji} = r_{ji})$ are approximated by weighted particles with a spherical Gaussian kernel density. Two novel filters are proposed here. The first filter provides NLoS mitigation by removing the messages with positive bias characteristics in their respective ranging measurements. The second one is designed to intelligently decrease the number of particles in each mixture in order to minimize calculations and is defined as an ellipsoid particle filter which utilizes the intuition that the location of a node will be close to the intersections between two belief cpdfs. Finally, we use MIS to calculate the product in eq. (3.7) and we use VB to obtain the parameters of the Gaussian mixture $\psi(\mathbf{\Theta}_i)$. In so doing, only the parameters of the Gaussian mixture distribution require transmission.

The algorithm is summarized in Algorithm 1. It begins by all nodes broadcast-

Algorithm 1 HEVA

Initialize beliefs $p^{(0)}(\boldsymbol{\Theta}_i)\forall i \in \text{Nodes}$
for $t = 0$ to T **do**
  for all $i \in \text{Nodes}$ **do**
    Broadcast current belief $\Pr^{(t)}(\boldsymbol{\Theta}_i)$
    for all $j \in \text{ne}(i)$ **do**
      Collect distance estimates $r_{ji}^{(t)}$
    end for
  end for
  Initialize $\psi(\boldsymbol{\Theta}_i) = \Pr^{(t)}(\boldsymbol{\Theta}_i)$
  Initialize $\psi(\boldsymbol{\Theta}_i, \boldsymbol{\Theta}_j, R_{ji} = r_{ji}) = \Pr^{(t)}(R_{ji} = r_{ji}^{(t)} | \boldsymbol{\Theta}_i, \boldsymbol{\Theta}_j)$
  repeat
    for all $i \in \text{Nodes}$ **do**
      for all $j \in \text{ne}(i)$ **do**
        Receive $b_j^{(s)}(\boldsymbol{\Theta}_j)$
        if $b_j^{(s)}(\boldsymbol{\Theta}_j)$ passes NLoS message filter (Algorithm 2) **then**
          Calculate $\delta_{j\to i}^{(s+1)}(\boldsymbol{\Theta}_i)$ using Spherical Gibbs sampling (Algorithm 3)
        end if
      end for
      Calculate $b_i^{(s+1)}(\boldsymbol{\Theta}_i)$ using the proposed ellipsoid particle filter (Algorithm 4) and VB clustering
      Check for convergence
      Broadcast $b_i^{(s+1)}(\boldsymbol{\Theta}_i)$
    end for
    s = s+1
  until all messages have converged or the maximum number of iterations is reached
  Update belief $p^{(t+1)}(\boldsymbol{\Theta}_i)$, using (3.8)
end for

ing their initial beliefs. In the first iteration, the agents have a uniform distribution and hence skip transmitting their beliefs, as it would not add any useful information. As more iterations pass, agents begin to have non-uniform beliefs about their locations and start to transmit. After each agent receives the messages (3.7) from their neighbours, the next step is to pass them from the NLoS mitigation filter.

3.3.2 NLoS Mitigation Filter

In order to understand the effect of the NLoS mitigation filter, it is important to understand the impact of NLoS propagation in ToA measurements. ToA measurements are affected by positive bias and removing biased messages does not affect the CRLB of the localisation error, c.f. [26]. Assuming a simple 2D example with three anchors communication with one agent, it is easy to visualize that in the case of no noise, the incoming message "ring"-shaped pdfs will combine to a single Gaussian centred on the intersection of the messages. Alternatively, if one anchor is NLoS, then due to the positive bias, the measured distance estimate will account to a larger radius in the "ring"-shaped pdf message skewing the centre of the Gaussian further away from the intersection, and possibly creating new components in the Gaussian mixture.

To mitigate that, the proposed NLoS message filter checks all incoming messages in the following manner. Each node compares the distance between itself and the corresponding neighbour computed using the current beliefs of both nodes with the range estimate. If it is larger than the estimate, the corresponding marginal (3.6) will be calculated and used in estimating the product (3.7); otherwise it will be dropped. The intuition behind is simple. As stated in Section 3.2, NLoS measurements are affected by a positive bias, which means that they will be greater than the real estimates. Let the estimated position of node i be $\hat{\boldsymbol{\theta}}_i^{(t)}$, that of node j be $\hat{\boldsymbol{\theta}}_j^{(t)}$, and the respective distance measurement be $r_{ji}^{(t)}$. As a result, if $\|\hat{\boldsymbol{\theta}}_i^{(t)} - \hat{\boldsymbol{\theta}}_j^{(t)}\| \geq r_{ji}^{(t)}$, then the corresponding messages will be calculated as normal; otherwise, the message from node j will be dropped.

To take full advantage of the information in the distribution, the following idea is proposed. Firstly, we draw L weighted samples $\{w_j^{(l)}, \boldsymbol{\theta}_j^{(l)}\}_{l=1}^{L}$ from

$b_j^{(s)}(\boldsymbol{\Theta}_j)\ \forall j \in \text{ne}(i)$ and another L weighted samples $\{w_i^{(l)}, \boldsymbol{\theta}_i^{(l)}\}_{l=1}^{L}$ from $\psi(\boldsymbol{\Theta}_i)$. Given the samples from the belief pdfs in $\{w_j^{(l)}, \boldsymbol{\theta}_j^{(l)}\}_{j=1,l=1}^{|\mathcal{N}_i|,L}$ and $\{w_i^{(l)}, \boldsymbol{\theta}_i^{(l)}\}_{l=1}^{L}$, the convex hull of each node set is calculated and the maximum distance between the two convex hulls of node i and any of its neighbours, node j, is compared with the distance measurement $r_{ji}^{(t)}$. This is done by using the maxdist algorithm in [67]. For $j \in \text{ne}(i)$, if

$$\begin{aligned} \mathrm{m}axdist\Big(&convex\_hull(\{w_i^{(l)}, \boldsymbol{\theta}_i^{(l)}\}_{l=1}^{L}), \\ &convex\_hull(\{w_j^{(l)}, \boldsymbol{\theta}_j^{(l)}\}_{l=1}^{L})\Big) \geq r_{ji}^{(t)}, \end{aligned} \tag{3.9}$$

then the message is kept and added to the set $\mathcal{Q}_i$; otherwise, it is dropped. Note that $\mathcal{Q}_i$ will hold only the messages that will be used in subsequent calculations to reduce the complexity. By using this condition, NLoS messages will be used in the first iterations of the algorithm where there is not enough information about the belief of the nodes, but in the later iterations, NLoS messages will tend to be dropped. It should be noted that no effort is made to identify if the message is NLoS or not by using any NLoS identification technique. This means that there is a probability that both LoS message could be ignored and NLoS message might pass the filter. The NLoS message filter algorithm is summarized in Algorithm 2.

Algorithm 2 NLoS message filter

Require: non-uniform $b_{i\to\forall j\in\text{ne}(i)}^{(s)}(\boldsymbol{\Theta}_i)$

Sample $\{w_i^{(l)}, \boldsymbol{\theta}_i^{(l)}\}_{l=1}^{L} \sim b_{i\to\forall j\in\text{ne}(i)}(\boldsymbol{\Theta}_i)$
Initialize $\mathcal{Q}_i$ to empty
for all $j \in \text{ne}(i)$ **do**
  Sample $\{w_j^{(l)}, \boldsymbol{\theta}_j^{(l)}\}_{l=1}^{L} \sim b_{j\to i}^{(s)}(\boldsymbol{\Theta}_j)$
  $\mathsf{C}H_i$=convex_hull($\{w_i^{(l)}, \boldsymbol{\theta}_i^{(l)}\}_{l=1}^{L}$)
  $\mathsf{C}H_j$=convex_hull($\{w_j^{(l)}, \boldsymbol{\theta}_j^{(l)}\}_{l=1}^{L}$)
  if $\mathsf{m}axdist(\mathsf{C}H_i, \mathsf{C}H_j) > r_{ji}^{(t)}$ **then**
    Add $\{w_j^{(l)}, \boldsymbol{\theta}_j^{(l)}\}_{l=1}^{L}$ to $\mathcal{Q}_i$
  end if
end for
return $\mathcal{Q}_i$

3.3.3 Filtering Operation

The next step of Algorithm 1 HEVA is for each node, say node i, to compute the received messages $\delta_{j\to i}^{(s+1)}(\boldsymbol{\Theta}_i)$ from its neighbours, where $\delta_{j\to i}^{(s+1)}(\boldsymbol{\Theta}_i))$ is the product of the distance cpdf with the neighbour location belief integrated over the neighbourhood variable $\boldsymbol{\Theta}_j$, for BP iteration $(s+1)$, i.e.,

$$\delta_{j\to i}^{(s+1)}(\boldsymbol{\Theta}_i) = \int \frac{\psi(\boldsymbol{\Theta}_i, \boldsymbol{\Theta}_j, R_{ji} = r_{ji}^{(t)}) b_{j\to i}^{(s)}(\boldsymbol{\Theta}_j)}{\delta_{i\to j}^{(s)}(\boldsymbol{\Theta}_j)} d\boldsymbol{\Theta}_j \tag{3.10}$$

with $\psi(\boldsymbol{\Theta}_i, \boldsymbol{\Theta}_j, r_{ji} = r_{ji}^{(t)}) = \Pr^{(t)}(r_{ji} = r_{ji}^{(t)} | \boldsymbol{\Theta}_i, \boldsymbol{\Theta}_j)$. Although node i has received the beliefs $b_j^{(s)}(\boldsymbol{\Theta}_j)$ and obtained the measurements $r_{ji}^{(t)}$ for deriving the cpdf, for complexity reasons, this will be done by using particle filtering. Specifically, given the set of particles $\mathscr{Q}_i$, for each subset j of particles $\{w_j^{(l)}, \boldsymbol{\theta}_j^{(l)}\}_{l=1}^{L}$, we calculate a set of parameters

$$\mathscr{G}_{j\to i} \triangleq \left\{ w_{j\to i}^{(l)}, \boldsymbol{\mu}_{j\to i}^{(l)}, \boldsymbol{\Sigma}_{j\to i} \right\}_{l=1}^{L} \tag{3.11}$$

that approximate

$$\delta_{j\to i}^{(s+1)}(\boldsymbol{\Theta}_i) \simeq \sum_l w_{j\to i}^{(l)} \mathscr{N}(\boldsymbol{\Theta}_i; \boldsymbol{\mu}_{j\to i}^{(l)}, \boldsymbol{\Sigma}_{j\to i}), \tag{3.12}$$

where $w_{j\to i}^{(l)}$ is a weighting factor and $\mathscr{N}(\boldsymbol{\Theta}_i; \boldsymbol{\mu}_{j\to i}^{(l)}, \boldsymbol{\Sigma}_{j\to i})$ refers to a Gaussian distribution in $\boldsymbol{\Theta}_i$ with mean vector $\boldsymbol{\mu}_{j\to i}^{(l)}$ and covariance matrix $\boldsymbol{\Sigma}_{j\to i}$. As such, the estimated lth sample $\boldsymbol{\theta}_i^{(l)}$ will be close to the surface of a sphere with radius $r_{ji}^{(t)}$ around the sample $\boldsymbol{\theta}_j^{(l)}$ and the mean vector for the mixture $\delta_{j\to i}^{(s+1)}(\boldsymbol{\Theta}_i)$ is given by

$$\boldsymbol{\mu}_{j\to i}^{(l)} = \boldsymbol{\theta}_j^{(l)} + r_{ji}^{(t)} + \nu^l \begin{bmatrix} \sin(\rho^{(l)})\cos(\phi^{(l)}) \\ \sin(\rho^{(l)})\sin(\phi^{(l)}) \\ \cos(\rho^{(l)}) \end{bmatrix}, \tag{3.13}$$

where $\phi^{(l)} \sim \mathscr{U}[0, 2\pi), \rho^{(l)} \sim \mathscr{U}[-\frac{\pi}{2}, \frac{\pi}{2}]$ and $\nu^{(l)} \sim p_\nu$, in which $\mathscr{U}(\cdots)$ denotes a uniform distribution and p_ν is the noise pdf, with a standard deviation of σ_ν. In addition, a covariance matrix $\boldsymbol{\Sigma}_{j\to i}$ is assigned to all Gaussians (independent of l)

and is calculated as

$$\mathbf{\Sigma}_{j\to i} = \frac{1}{5.991}\left[\frac{4\sigma_v\left(3(r_{ji}^{(t)})^2 + 4\sigma_v^2\right)}{L}\right]^{\frac{2}{3}}\mathbf{I}. \tag{3.14}$$

The analysis leading to the above expression is given in the appendix. On the other hand, the weight of the lth sample is given by

$$w_{j\to i}^{(l)} \propto \frac{w_j^{(l)}}{\delta_{i\to j}^{(s)}(\boldsymbol{\theta}_j^{(l)})}. \tag{3.15}$$

This can be considered as the belief pdf of node j with the influence of node i from the previous iteration being removed (i.e., $\delta_{i\to j}^{(s)}(\mathbf{\Theta}_j)$), in order to avoid overpowering of a node's belief due to loops [57]. Alternatively, we can concentrate the samples taking advantage of the angle in the same manner as Parsimonious NBP, c.f. [57]. We call this this variation of HEVA as Parsimonious HEVA, or PHEVA.

After calculating the approximations of all cpdfs for all the incoming messages, we add all sets $\mathscr{G}_{j\to i}$ to the superset $\mathscr{G}_i$. This set can be viewed as $|\text{ne}(i)|$ Gaussian mixtures with L components each, and will be used to calculate the product of all incoming beliefs with the belief of node i (3.7). The steps are summarized in Algorithm 3. Note that the terms "component" and "particle" will be used interchangeably thereafter.

Algorithm 3 Spherical Gibbs Sampling

1: Initialize $\mathscr{G}_i$ as an empty set
2: **for all** $j \in \mathscr{Q}_i$ **do**
3: Sample $\{\phi^{(l)}\}_{l=1}^L \sim \mathscr{U}[0, 2\pi]$
4: Sample $\{\rho^{(l)}\}_{l=1}^L \sim \mathscr{U}\left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$
5: Sample $\{v^{(l)}\}_{l=1}^L \sim \mathscr{N}(0, \sigma_v^2)$
6: Calculate the mean vector, using (3.13)
7: Calculate the covariance matrix $\mathbf{\Sigma}_{j\to i}$, using (3.14)
8: Calculate the weights $w_{j\to i}^{(l)}$, using (3.15)
9: Add $\{w_{j\to i}^{(l)}, \boldsymbol{\mu}_{j\to i}^{(l)}, \mathbf{\Sigma}_{j\to i}\}_{l=1}^L$ to $\mathscr{G}_i$
10: **end for**
11: **return** $\mathscr{G}_i$

3.3.4 Product Operation: Gaussian Mixture Product Calculation

A commonly used approach to prevent the huge computational cost for evaluating the product of Gaussian mixtures is the use of the MIS technique [55]. As each Gaussian mixture in the product (3.7) has L components, the computational cost will be proportional to L^2 [55]. Unfortunately, even in the case of MIS, the cost can become prohibitive in the 3D case, because of the increased number of components per mixture, as discussed in Section 3.1. To avoid this, we propose the use of a novel filter on the particles of the incoming messages, minimizing the total number of particles in MIS and hence the number of calculations, thereby making MIS feasible.

The filter takes advantage of the intuition that the relevant particles should be close to the intersection of the "sphere"-shaped particle sets. This inter-sectional area can be enclosed inside an ellipsoid, and therefore the filter is referred to as an ellipsoid particle filter. Fig. 3.3 shows a 2D example of the filter. As the majority of the particles are not near the ellipsoid, and in no way near the intersections, they can be safely removed to avoid a huge amount of unnecessary computations.

The proposed ellipsoidal filter is a "soft" decision probabilistic filter. Instead of simply removing all particles that are outside the filter area, it weights them based on their Euclidean distance from the ellipsoid and then resamples the $V = \alpha L$ particles from each message, where $0 < \alpha < 1$. This gives the filter a greater flexibility allowing to consider for situations, where the real location is further away from the intersection due to high noise. We define a box-shaped volume as shown in Fig. 3.3, with the edges defined by the coordinates as

$$\begin{cases} x_{\min} = \max\limits_{j}(\min\limits_{l}(x_j^{(l)})), \\ y_{\min} = \max\limits_{j}(\min\limits_{l}(y_j^{(l)})), \\ z_{\min} = \max\limits_{j}(\min\limits_{l}(z_j^{(l)})), \end{cases} \text{and} \begin{cases} x_{\max} = \min\limits_{j}(\max\limits_{l}(x_j^{(l)})), \\ y_{\max} = \min\limits_{j}(\max\limits_{l}(y_j^{(l)})), \\ z_{\max} = \min\limits_{j}(\max\limits_{l}(z_j^{(l)})). \end{cases} \tag{3.16}$$

From the superset $\mathscr{G}_i$, we select all the particles that reside inside the box. From

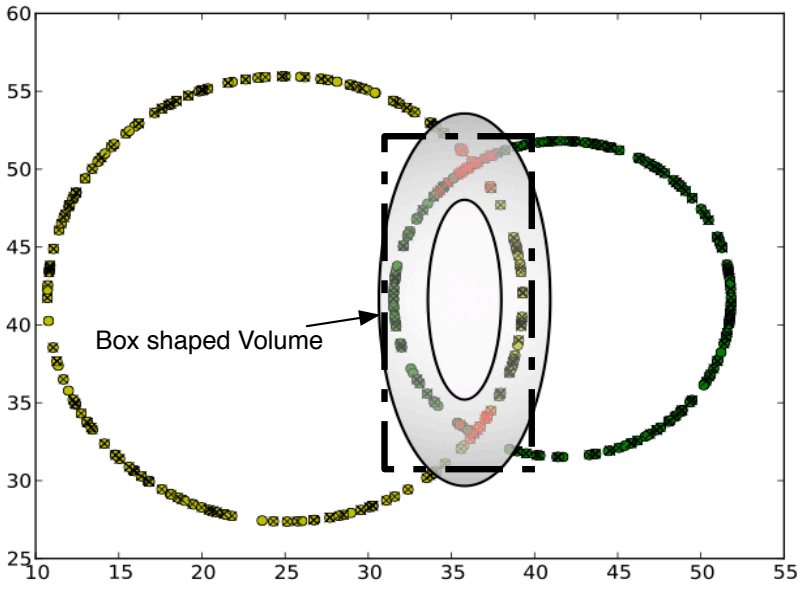

Figure 3.3: A 2D example showing the ellipsoidal filter in operation. The mean and covariance for the ellipsoid are calculated based on the particles inside the box, and then resampling with replacement is done for each message proportional to their distance from the ellipsoidal area.

these particles, we calculate the mean vector $\boldsymbol{\mu}_{box}$ and the covariance matrix $\boldsymbol{\Sigma}_{box}$. Then we calculate the ellipsoid weight for every particle $\{w^{(l)}_{j\to i}, \boldsymbol{\mu}^{(l)}_{j\to i}, \boldsymbol{\Sigma}_{j\to i}\}^{L}_{l=1}$ in every message set $\mathscr{G}_{j\to i}$, by using the mean of the particle in the following function and normalize the weights for each message set. Mathematically, that is,

$$w^{(l)}_{\mathsf{e},\boldsymbol{\mu}_{j\to i}} = w_{ellipsoid}(\boldsymbol{\mu}^{(l)}_{j\to i}) \propto \exp\left(-\frac{\gamma}{2}\left(\sqrt{(\boldsymbol{\mu}^{(l)}_{j\to i}-\boldsymbol{\mu}_{box})^{T}\boldsymbol{\Sigma}^{-1}_{box}(\boldsymbol{\mu}^{(l)}_{j\to i}-\boldsymbol{\mu}_{box})}-1\right)^{2}\right), \tag{3.17}$$

where the variable γ is defined as the filter precision, and it controls the steepness of the filter. We add the relevant weight to every element in the superset $\mathscr{G}_i$:

$$\left\{\left\{w^{(l)}_{\mathsf{e},\boldsymbol{\mu}_{j\to i}}, w^{(l)}_{j\to i}, \boldsymbol{\mu}^{(l)}_{j\to i}, \boldsymbol{\Sigma}_{j\to i}\right\}^{L}_{l=1}\right\}_{j\in\mathscr{N}_i}. \tag{3.18}$$

Afterwards, V particles are randomly sampled with replacement, from each

message set $\mathscr{G}_{j\to i}$, according to their weights (3.17), giving a new particle set $\{w_{j\to i}^{(v)}, \boldsymbol{\mu}_{j\to i}^{(v)}, \boldsymbol{\Sigma}_{j\to i}^{(v)}\}_{v=1}^{V}$ that is added to the set $\mathscr{J}_i$. The ellipsoid particle filter is summarized and formally described in Algorithm 4. Additionally, we draw V

Algorithm 4 ellipsoid particle filter

1: Given all particles $\{w_{j\to i}^{(l)}, \boldsymbol{\mu}_{j\to i}^{(l)}, \boldsymbol{\Sigma}_{j\to i}\}_{l=1}^{L} \in \mathscr{G}_i$, find (3.16)
2: Calculate box borders based on $\begin{bmatrix} x_{\min} & x_{\max} \\ y_{\min} & y_{\max} \\ z_{\min} & z_{\max} \end{bmatrix}$
3: Calculate $\boldsymbol{\mu}_{\mathsf{box}} = \mathsf{mean}(\boldsymbol{\mu}_{j\to i}^{l}) \;\forall\; \boldsymbol{\mu}_{j\to i}^{l}$ inside box
4: Calculate $\boldsymbol{\Sigma}_{\mathsf{box}} = \mathsf{covariance}(\boldsymbol{\mu}_{j\to i}^{l}) \;\forall\; \boldsymbol{\mu}_{j\to i}^{l}$ inside box
5: Weight all samples using (3.17)
6: Initialize $\mathscr{J}_i$ to be an empty set
7: **for all** $j \in \mathscr{Q}_i$ **do**
8: Re-sample from $\{w_{j\to i}^{(l)}, \boldsymbol{\mu}_{j\to i}^{(l)}, \boldsymbol{\Sigma}_{j\to i}\}_{l=1}^{L}$ proportionally to their weights $\{w_{\mathsf{e},\boldsymbol{\mu}_{j\to i}}^{(l)}\}_{l=1}^{L}$, aL times with replacement.
9: Add $\{w_{j\to i}^{(l)}, \boldsymbol{\mu}_{j\to i}^{(l)}, \boldsymbol{\Sigma}_{j\to i}\}_{l=1}^{aL}$ to $\mathscr{J}_i$
10: **end for**
11: **return** $\mathscr{J}_i$

samples from $\psi(\boldsymbol{\Theta}_i)$ to complete the calculation (3.7). Essentially, we have a new product of Gaussian mixtures, but with V components, instead of L. In order to compute the product, we employ MIS [55]. From each message $j \in \mathrm{ne}(i)$, we draw V samples $\boldsymbol{\theta}_j^{(v)}$ and weight them by

$$w_j^{(v)} = \frac{\psi(\boldsymbol{\theta}_j^{(v)}) \prod_{j\in\mathrm{ne}(i)} \delta_{j\to i}^{(s+1)}(\boldsymbol{\theta}_j^{(v)})}{\psi(\boldsymbol{\theta}_j^{(v)}) + \sum_{j\in\mathrm{ne}(i)} \delta_{j\to i}^{(s+1)}(\boldsymbol{\theta}_j^{(v)})} \frac{1}{w_{ellipsoid}(\boldsymbol{\theta}_j^{(v)})}. \tag{3.19}$$

It should be noted that we divide by the weight $w_{ellipsoid}(\boldsymbol{\theta})$ to cancel out the asymptotic effect of drawing samples from the ellipsoid particle filter. After that, L particles are sampled with replacement from the $(|\mathrm{ne}(i)|+1)V$ samples $\{w_j^{(v)}, \boldsymbol{\theta}_j^{(v)}\}_{j=1,v=1}^{|\mathrm{ne}(i)|+1,V}$ proportional to the weights $w_j^{(v)}$, to give $\{\boldsymbol{\theta}^{(l)}\}_{l=1}^{L}$. As we keep only V particles for each incoming message, the computational cost can be drastically reduced, while the computational cost of weighting all the particles is linear with the total number of particles, having complexity of $\mathscr{O}(V^2(|\mathrm{ne}(i)|+1))$ compared to the $\mathscr{O}(L^2(|\mathrm{ne}(i)|+1))$ if using all components. The MIS algorithm is

summarized in Algorithm 5.

Algorithm 5 MIS

1: **for all** $j \in \mathscr{J}_i$ **do**
2: Initialize $p_{i\to j}(\boldsymbol{\theta}_i) = \sum_{k=1}^{aL} w_{j\to i}^{(k)} \mathcal{N}\left(\boldsymbol{\theta}_i, \boldsymbol{\mu}_{j\to i}^{(k)}, \boldsymbol{\Sigma}_{j\to i}\right)$
3: **end for**
4: Initialize $q_i(\theta_i) = \frac{1}{|\mathscr{J}_i|} \sum_{j'\in\mathscr{J}_i} p_{i\to j'}(\boldsymbol{\theta}_i)$
5: Draw KaL samples $\{\boldsymbol{\theta}_i^{(n)}\}_{n=1}^{KaL} \sim q_i(\boldsymbol{\theta}_i)$, where $K > 1$
6: Weight them by $w_i^{(n)} = \frac{\prod_{j'\in\mathscr{J}_i} p_{i\to j'}(\boldsymbol{\theta}_i^{(n)})}{q_i(\boldsymbol{\theta}_i^{(n)})}$
7: Re-sample from $\{w_i^{(n)}, \boldsymbol{\theta}_i^{(n)}\}_{n=1}^{KaL}$ proportionally to their weights, aL times with replacement.

As such, we have a particle approximation of $b_i^{(s+1)}(\boldsymbol{\theta}_i)$. From this current belief, we randomly choose λL samples and $(1-\lambda)L$ samples from the calculated belief in the previous iteration, in order to estimate the dampened belief as in (3.7).

Given the set of the L particles that approximate $b_i^{(s+1)}(\boldsymbol{\theta}_i)$, the final step is to convert the non-parametric kernel representation of the belief in a parametric form using a Gaussian mixture with parameters $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_K)$, mean vectors $\{\boldsymbol{\mu}_k\}_{k=1}^K$ and covariance matrices $\{\boldsymbol{\Sigma}_k\}_{k=1}^K$ so that

$$\delta_{i\to\forall j}^{\text{lower}\to\text{upper}}(\boldsymbol{\theta}_i) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{\theta}_i; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k), \tag{3.20}$$

where K is regarded as the number of clusters. The optimization of the parameters is addressed in the next subsection.

3.3.5 Variational Bayes

VB is a Bayesian learning algorithm that finds the parameters that best fit an approximate distribution to a dataset [18]. The idea is similar to the well-known EM algorithm discussed in Chapter 2, but with the major difference that the parameters of the mixtures are considered themselves stochastic variables derived from prior distributions.

The advantage of this is threefold. First, singularity issues encountered in EM

can be completely avoided. Also, there is no issue of fitting too many clusters, as even if a large number of initial clusters are initially considered, the superfluous ones will degenerate to zero through self-optimization. This allows the algorithm to start with a large K that will decrease as the algorithm progresses and more information received resolves ambiguities. Essentially, this gives a compromise among the message size, computational cost and information loss. Finally, the optimal number of clusters can be obtained without using techniques such as cross-validation. For more information, interested readers are referred to [18, Chapter 10]. These prior pdfs are chosen to be conjugate priors of the corresponding distributions in order to facilitate the derivations.

In VB, the parameters in (3.20) are considered stochastic and their priors are a Dirichlet distribution for $\boldsymbol{\pi}$ and a Gaussian-Wishart distribution for $\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k$, where $\boldsymbol{\Lambda}_k = \boldsymbol{\Sigma}_k^{-1}$. That is,

$$\Pr(\boldsymbol{\pi}) = \mathsf{D}ir(\boldsymbol{\pi}|\boldsymbol{a}) = C(\boldsymbol{a}) \prod_{k=1}^{K} \pi_k^{a_k - 1} \tag{3.21}$$

$$\Pr(\boldsymbol{\mu}, \boldsymbol{\Lambda}) = \Pr(\boldsymbol{\mu}|\boldsymbol{\Lambda}) \Pr(\boldsymbol{\Lambda}) \tag{3.22}$$

$$= \prod_{k=1}^{K} \mathcal{N}(\boldsymbol{\mu}_k|\boldsymbol{m}_k, (\beta_k \boldsymbol{\Lambda}_k)^{-1}) \mathcal{W}(\boldsymbol{\Lambda}_k|\boldsymbol{W}_k, \nu_k), \tag{3.23}$$

where $\boldsymbol{a} = \{a_1, \ldots, a_K\}$ can be interpreted as the effective prior number of observations associated with each component, $\boldsymbol{W}_k$ is a positive definite matrix, and ν_k denotes the number of degrees of freedom of the distribution for the kth cluster. The parameters of the priors are called hyper-parameters. In the rest of the section the steps of the algorithm are presented, but for a complete derivation of the algorithm, see [18].

The method to optimize the parameters in (3.20) is iterative and cycles between

two steps, similar to the EM steps, with some initialization of the hyper-parameters

$$a_k = a_0 \ \forall k, \tag{3.24}$$

$$\beta_k = \beta_0 \ \forall k, \tag{3.25}$$

$$\boldsymbol{m}_k = \boldsymbol{m}_0 \ \forall k, \tag{3.26}$$

$$\boldsymbol{W}_k = \boldsymbol{W}_0 \ \forall k, \tag{3.27}$$

$$\nu_k = \nu_0 \ \forall k. \tag{3.28}$$

In the first step, referred to as the E-step, the parameters of the distributions are considered constant and the responsibilities of each component (cluster k) for each sample (particle $\boldsymbol{\theta}_l$ obtained in Section III-D), denoted as r_{lk}, are calculated. Very briefly, r_{lk} indicates how probable it is for particle l to belong to cluster k. In more detail, the E-step calculates

$$\ln \tilde{\boldsymbol{\Lambda}}_k = \sum_{i=1}^{F} \psi\left(\frac{\nu_k + 1 - i}{2}\right) + F \ln 2 + \ln \mathsf{det}(\boldsymbol{W}_k), \tag{3.29}$$

$$\ln \tilde{\pi}_k = \psi(a_k) - \psi\left(\sum_{k=1}^{K} a_k\right), \tag{3.30}$$

$$r_{lk} \propto \tilde{\pi}_k \tilde{\boldsymbol{\Lambda}}_k^{\frac{1}{2}} \exp\left\{-\frac{F}{2} - \frac{\nu_k}{2}(\boldsymbol{\theta}_l - \boldsymbol{m}_k)^T \boldsymbol{W}_k (\boldsymbol{\theta}_l - \boldsymbol{m}_k)\right\}, \tag{3.31}$$

where $\psi(\cdot)$ is the digamma function, and F is the number of features. In 3D cases, $F = 3$ but in 2D, $F = 2$.

In the M-step, given the responsibilities r_{lk}, we recalculate the parameters for

maximizing the log-likelihood by

$$L_k = \sum_{l=1}^{L} r_{lk}, \tag{3.32}$$

$$\bar{\boldsymbol{\theta}}_k = \frac{1}{L_k} \sum_{l=1}^{L} r_{lk} \boldsymbol{\theta}_l, \tag{3.33}$$

$$\boldsymbol{S}_k = \frac{1}{L_k} \sum_{l=1}^{L} r_{lk} (\boldsymbol{\theta}_l - \bar{\boldsymbol{\theta}}_k)(\boldsymbol{\theta}_l - \bar{\boldsymbol{\theta}}_k)^T, \tag{3.34}$$

$$a_k = a_0 + L_k, \tag{3.35}$$

$$\beta_k = \beta_0 + L_k, \tag{3.36}$$

$$\boldsymbol{m}_k = \frac{1}{\beta_k} (\beta_0 \boldsymbol{m}_0 + L_k \bar{\boldsymbol{\theta}}_k), \tag{3.37}$$

$$\boldsymbol{W}_k^{-1} = \boldsymbol{W}_0^{-1} + L_k \boldsymbol{S}_k + \frac{\beta_0 L_k}{\beta_0 + L_k} (\boldsymbol{\theta}_l - \boldsymbol{m}_0)(\boldsymbol{\theta}_l - \boldsymbol{m}_0)^T, \tag{3.38}$$

$$\nu_k = 1 + \nu_0 + L_k. \tag{3.39}$$

The algorithm continues to iterate between the E- and M-steps until either the parameters or the log-likelihood converge to meet a certain precision. Finally, the values of the optimized parameters in (3.20) are then chosen as the expectations of the corresponding distributions, i.e.,

$$\pi_k = \frac{a_k}{\sum_{k=1}^{K} a_k}, \tag{3.40}$$

$$\boldsymbol{\mu}_k = \boldsymbol{m}_k, \tag{3.41}$$

$$\boldsymbol{\Sigma}_k = (\nu_k \boldsymbol{W}_k)^{-1}. \tag{3.42}$$

It is well understood that VB has the advantage of achieving a high log-likelihood quickly, and gives a good parametric approximation of $\Pr(\boldsymbol{\theta}_i)$. In HEVA, the distribution consists of a relatively large number of components evenly spread out in the area already calculated in for the ellipsoid particle filter. There are two advantages. In case there is a large space of possible locations, the large number of components will provide flexibility for the approximation to fit well with the data.

In addition, if there is little or no ambiguity, most of the components degenerate to zero allowing for a small Gaussian mixture (i.e., small K).

3.3.6 Computational Convergence

A cluster graph is not guaranteed to achieve the optimal solution, but if the running intersection property and the family preservation property are held [53], typically most clusters will converge to a local optimum with just a few clusters oscillating unable to achieve convergence. HEVA uses a dampened BP message passing, which empirically helps even the oscillating nodes converge to a local optimum. An added complexity is inserted in the algorithm because of the use of VB. For this reason, it is important to explicitly define a convergence criterion. As location beliefs are approximated by Gaussian mixtures, Kullback-Leibler Divergence (KLD) between the belief of the previous and present iterations is calculated. If the difference is below a prescribed threshold, the node is said to have converged to a solution. As there is no easy way to calculate the KLD between two Gaussian mixtures, the approximation method proposed in [68] is used.

3.3.7 Complexity

The complexity of HEVA is dominated by three processes: (1) the filtering operation, (2) the product operation and (3) the VB algorithm, i.e., the clustering operation. In order to analyse the complexity, we let:

- $|\mathrm{ne}(i)|$ the number of incoming messages to a specific node,
- L the number of drawn particles,
- α the ratio of particles kept in the product calculation,
- S the number of components in the Gaussian mixture,
- F the number of features,
- $I_{\mathrm{V}B}$ the maximum number of iterations VB will run,
- $I_{\mathrm{H}EVA}$ the number of iterations HEVA will run.

At every iteration of the algorithm, each node first draws particles from each incoming message, namely, the filtering operation, an operation that scales with the number of neighbours, and the number of drawn particles (therefore scales with complexity $L|\text{ne}(i)|F$). Then the samples are passed through the ellipse function and every particle is weighted. Thus, this operation scales with $L|\text{ne}(i)|F$. The next step is to draw αL samples for each message with replacement, with a complexity of $\alpha L|\text{ne}(i)|$. Finally, the MIS algorithm is used to calculate the product. This operation's complexity scales with $(\alpha L)^2|\text{ne}(i)|F$ [55]. Finally, a parametric form is found with VB, an operation that scales with $\alpha LSI_{\mathsf{V}B}F^3$ [18]. The computational complexity of the above operations is summarized in Table 3.1.

Next we compare HEVA with various alternative algorithms used for distributed cooperative localisation. In NBP and its non-parametric variants, the squared L factor in calculating MIS makes the product operation (3.7) computationally dominate the algorithm. As such, the complexity cost is bounded by $\mathscr{O}(L^2|\text{ne}(i)|F)$. NBP variations like Parsinomius Nonparametric BP (P-NB) [57] and Box Nonparametric BP (Box-NBP) [65] aim to decrease the number of particles L required, by focusing the energy mass of the pdfs, but the computational cost remains dominated by (3.7). Similarly they are also computationally bounded by $\mathscr{O}(L^2|\text{ne}(i)|F)$.

In HEVA, we make α small to decrease the computational cost of the product operation. By choosing a suitably low α, the decrease in computational cost will be $\frac{(\alpha L)^2}{L^2} \Rightarrow \alpha^2$. For example, if $\alpha = 0.2$, there will be more than 96% decrease in complexity for the message product operation. Essentially this ameliorates the bottleneck of the product operation. This lowers the complexity bound of HEVA by almost an order of L approaching approximately $\mathscr{O}(L|\text{ne}(i)|F)$, if one chooses $\alpha^2 \approx \frac{1}{L}$. Finally, for Parsinomius HEVA (PHEVA), as will be shown in the sequel, the focused sampling provides betters results for a given number of particles L, albeit at a computational cost.

In Fig. 3.4, we present the average simulation time for a 3D network with 25 nodes, and average connectivity 4.9 for different numbers of L. Computation was

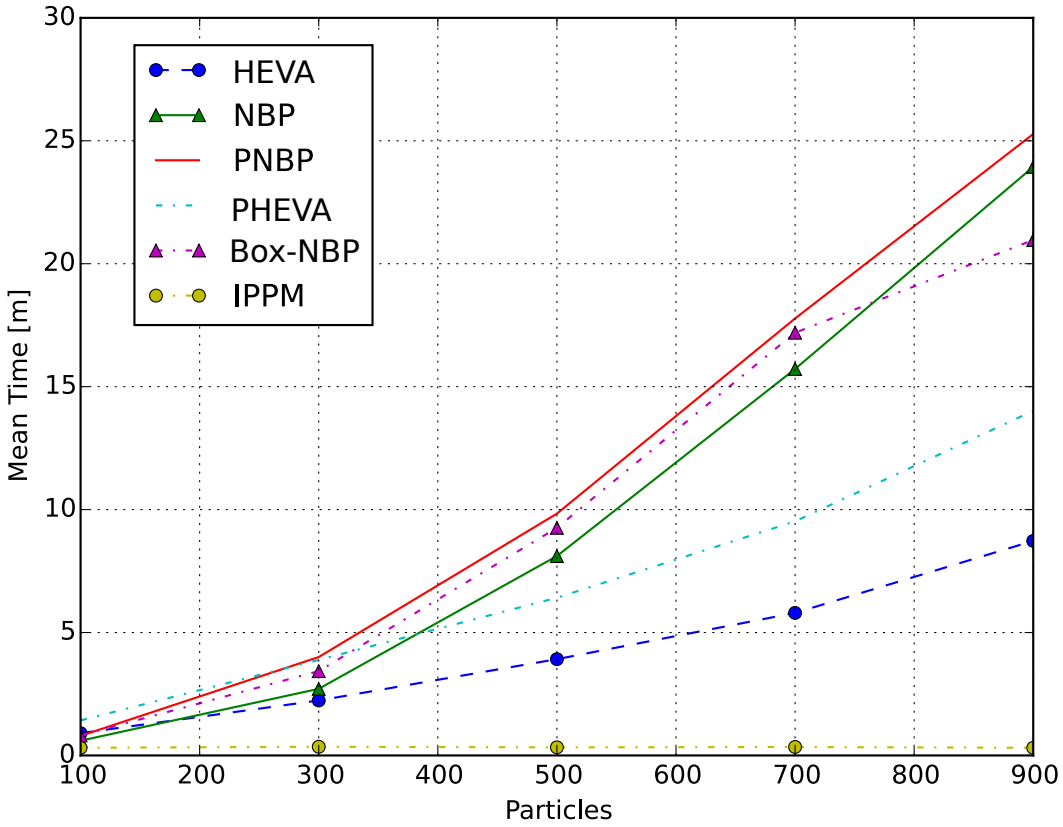

Figure 3.4: The average time versus the number of particles when the average node connectivity is 4.9.

done on an Intel core 2 duo at 2.33GHz. All algorithms were implemented in python using the numpy and scipy libraries [69]. For each scenario, 100 iterations were run and the average time was calculated. As can be seen, for NBP, P-NB and Box-NBP, the processing time all increases quadratically with L, while HEVA is much smoother, and almost approaches a linear increase. In contrast, P-NB has a much steeper increase in processing time than HEVA, providing a compromise between the speed of HEVA, for slightly better accuracy. Finally, it can be seen that IPPM is much faster than the aforementioned family of belief message passing algorithms, but at a higher error rate. IPPM does not use particles, thus the mean time does not vary with the number of particles. Unfortunately, this provides lower accuracy and requires a larger communication overhead, in order to decide on the initial values before the optimization starts, as will be discussed in Section 3.3.8. The complexity

Table 3.1: Computational complexity of HEVA steps for a each agent (say the i-th)on one iteration

Computation	Cost
The following operations are repeated I_{HEVA} times	
Draw samples from messages	$L\|\text{ne}(i)\|F$
Weight using Ellipse function	$L\|\text{ne}(i)\|F$
Draw weighted samples with replacement	$\alpha L\|\text{ne}(i)\|$
Compute particle product	$(\alpha L)^2\|\text{ne}(i)\|F$
Run VB	$\alpha LSI_{VB}F^3$

costs of the various algorithms are given in Table 3.2.

Table 3.2: Computational complexity for each agent (say the i-th) on one iteration of different algorithms

Algorithm	Complexity	Typical Values
NBP	$\mathcal{O}(L^2(\|\text{ne}(i)\|)$	$L = 10^4 - 10^6$
PNBP	$\mathcal{O}(L^2(\|\text{ne}(i)\|)$	$L = 10^2 - 10^3$
Box-NBP	$\mathcal{O}(L^2(\|\text{ne}(i)\|)$	$L = 10^2 - 10^3$
HEVA	$\mathcal{O}((aL)^2\|\text{ne}(i)\|)$	$L = 10^2 - 10^3$
PHEVA	$\mathcal{O}((aL)^2\|\text{ne}(i)\|)$	$L = 10^2 - 10^3$
IPPM	$\mathcal{O}(\|\text{ne}(i)\|)$	–

The average computational time for HEVA, and other algorithms is provided in Fig. 3.5 for different sizes of neighbouring nodes. The same experiment was conducted but with varying communication range R, while keeping the number of particles constant $L = 300$, hence increasing the average number of neighbours. All techniques increase linearly with time, but the gradient of HEVA is much smaller $0.27s/$neighbour, compared to $0.77s/$neighbour for NBP, scaling much better when the average number of neighbours increases. The steepness that can be seen in the graph for less than three neighbours on average can be explained by the lack of computations as there are not enough neighbours to localize. The variations of NBP take similar time to NBP but require slightly more computations as they also need to calculate the angles of the samples.

In Fig. 3.6, results for the average computational time of NBP and HEVA are compared with the respective algorithms of adding to NBP one component of HEVA at a time, for the same 3D scenario as before. First, the new kernel is added, namely

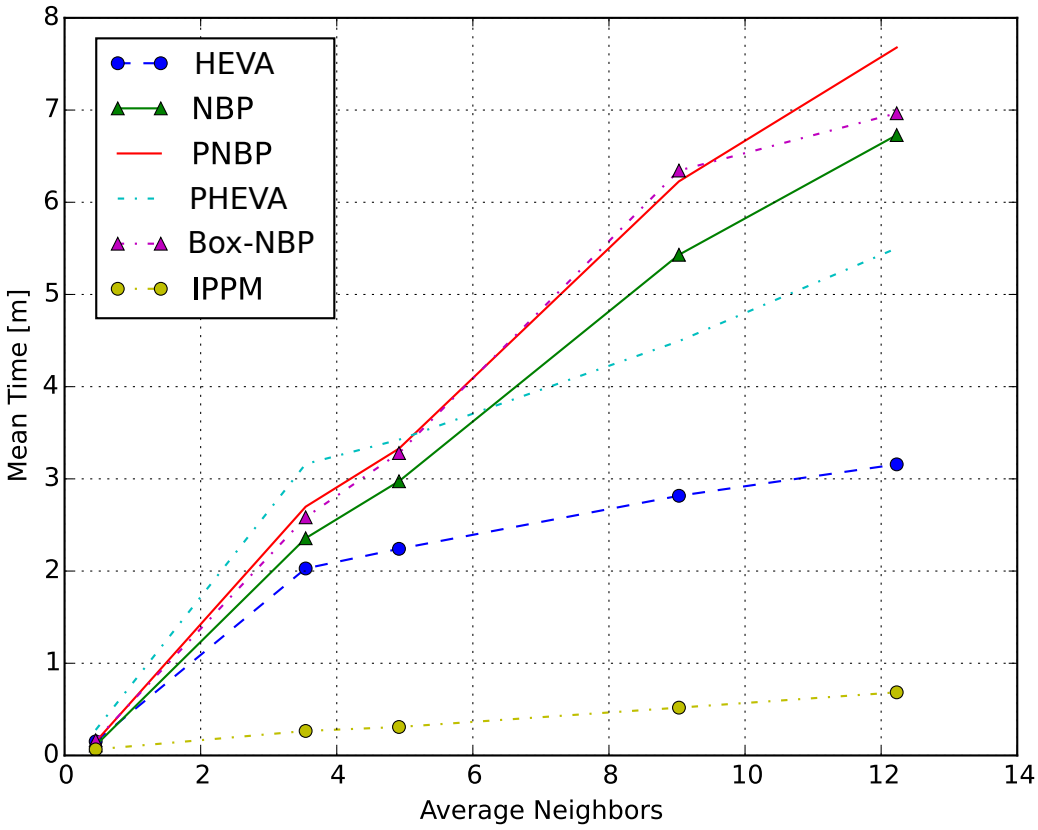

Figure 3.5: The average simulation time versus mean node connectivity. Number of particles is $L = 300$.

"NBP Sphere Kernel", decreasing the computational cost slightly. Then the ellipsoid particle filter is added, namely "ENBP" or "NBP+EF", which manages to decrease the cost considerably. Then if VB is applied, this will actually increase the computation cost, and finally the NLoS message filter, which relatively keeps the cost equal to "EVNBP" or "NBP+EF+VB". It should be noted that as the number of average neighbours increases the computational time of HEVA decreases relative to "EVNBP", as messages will be dropped, from the NLoS message filter.

3.3.8 Communication Overhead

In a distributed algorithm, most of the energy is spent on the local computations and broadcasting messages to one-hop neighbouring nodes. In the literature focus is mostly given in the broadcasting part as the energy required for transmission is

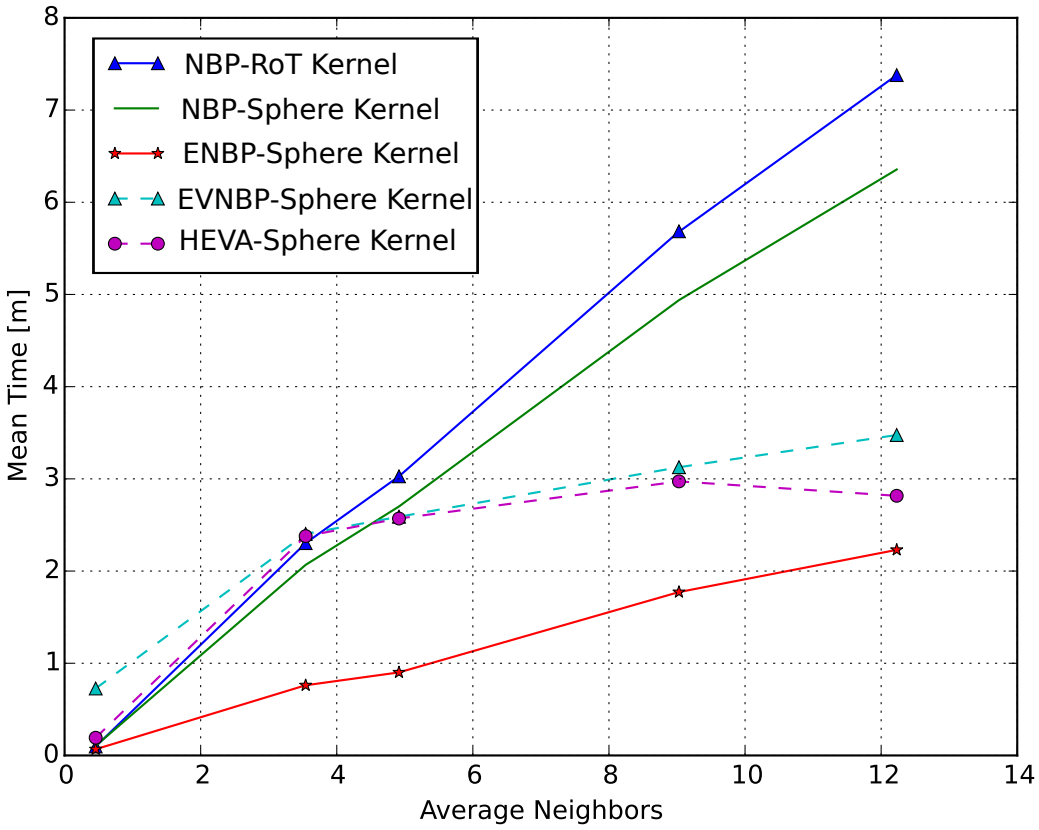

Figure 3.6: The average simulation time versus mean node connectivity. Number of particles is $L = 300$.

much higher than the energy required for a single floating point operation, c.f. [70]. Following the analysis in [71], we assume that all nodes are uniformly distributed over a 3-D unit square grid, and all real values are represented in double precision floating point format (64-bit precision). Hence, the total energy consumed for communication by any cooperative localisation algorithm can be written as

$$\mathbb{E}(|\mathcal{V}|) = b(|\mathcal{V}|)h(|\mathcal{V}|)e(|\mathcal{V}|), \tag{3.43}$$

where $\mathbb{E}(\cdot)$ returns the expectation, $b(|\mathcal{V}|)$ denotes the total number of transmitted bits for $|\mathcal{V}|$ nodes, $h(|\mathcal{V}|)$ is the average number of hops required for transmitting one bit to the destination and $e(|\mathcal{V}|)$ is the average amount of energy required for transmitting one bit over one hop. As all communication is assumed to

be done only by broadcasting to one-hop neighbours we set $h(|\mathcal{V}|) = 1$. The total number of real values transmitted in one iteration for one agent is approximately $L(1+F)$ for non-parametric algorithms and $L_{\text{VB}}(1+F+F^2)$ for HEVA, where L_{VB} is the average number of non zero components in VB. Therefore, we have that $b_{\text{NBP}}(|\mathcal{V}|) = \mathcal{O}(L(|\mathcal{V}|)(1+F))$ and $b_{\text{HEVA}}(|\mathcal{V}|) = \mathcal{O}(L_{\text{VB}}(|\mathcal{V}|)(1+F+F^2))$.

IPPM nodes on the contrary only transmit their point estimate, broadcasting F real values, which gives $b_{\text{IPPM}}(|\mathcal{V}|) = \mathcal{O}((|\mathcal{V}|)F)$. Given a fixed total number of nodes and ignoring $e(|\mathcal{V}|)$, which is assumed the same for all algorithms, we get the energy cost bounds presented in Table 3.3.

Table 3.3: Energy consumed by all nodes on one iteration of different algorithms

Algorithm	Energy Cost	Typical Values
NBP	$\mathcal{O}(L((\|\mathcal{V}\|))$	$L = 10^4 - 10^6$
PNBP	$\mathcal{O}(L((\|\mathcal{V}\|))$	$L = 10^2 - 10^3$
BNBP	$\mathcal{O}(L((\|\mathcal{V}\|))$	$L = 10^2 - 10^3$
HEVA	$\mathcal{O}(L_{\text{VB}}(\|\mathcal{V}\|))$	$L_{\text{VB}} = 10^0 - 10^1$
IPPM	$\mathcal{O}(\|\mathcal{V}\|)$	–

Note that as was shown in Section 3.3.7, even though simply using ENBP is computationally cheaper than EVNBP and HEVA, by transmitting the belief in parametric form using VB. Based on the above analysis, there is a significant decrease in communication overhead. Simulations illustrate that for 3D positioning on average, VB uses $L_{VB} \simeq 4$ and assuming $L = 800$, as in Section 3.4, we have approximately a 98% decrease in communication overhead throughout the network. Even if NBP, used only $L = 100$ particles, the communication cost of HEVA would still be approximately 87% less, showcasing the importance of transmitting a parametric form when minimizing communication throughput is the priority.

The ability to actually achieve consistently better accuracy than NBP and its variants, with a large decrease in computational cost, ameliorating the bottleneck of MIS, as well as a large decrease in communication overhead is the main contribution of HEVA, and important, as it paves the path of practically using probabilistic techniques in 3D localisation.

3.4 Simulation Results

In this section, we present simulation results for HEVA for cooperative localisation. The RMS error is compared with those of NBP and Parsimonious NBP [15] as well as boxed NBP [65]. In experiments with NLoS edges we compare HEVA with IPPM [25] and Expectation-Conditional Maximization (ECM) [71].

P-NB is a variant of NBP that uses the angle of particles between iterations to focus the sampling instead of using a uniform sampling. The equations from [15] have been extended for the 3D case in order for the simulation to work. BNBP is another variant that uses information of anchors in order to box the sampling in a specific area, in the same spirit as HEVA, even though in this case the limit is "hard", as all samples inside the box are taken, and all samples outside are dropped. ECM uses a distributed expectation conditional maximization algorithm while IPPM is a deterministic optimization method, based on the parallel projection method.

The RMS error can be defined as

$$\varepsilon_{\mathsf{R}MS} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\mathbb{E}(\|\hat{\boldsymbol{\theta}}_i - \boldsymbol{\theta}_i\|^2)}, \tag{3.44}$$

where $\mathbb{E}(\cdot)$ denotes the expectation over noise realizations. As a summary, we will show that HEVA provides consistently better results than NBP and its variants with a large decrease in computational cost. In the 3D case, the advantages become much more pronounced, while NBP and its variants become computationally prohibitive. The biggest advantage of HEVA can be seen in the case of NLoS. We will define the edge NLoS probability as the probability that any network communication edge $e_{ji} \in \mathscr{E}$ is in NLoS for a given experiment, and present results for varying the edge NLoS probabilities. Finally, the variants of HEVA using EM and K-means will be shown, compared to our proposal of VB. Note that the higher errors here are due to the spherical Gaussian distributions for the anchor beliefs that increase the inherent noise. The important thing to note is the comparison between algorithms.

We consider a three dimensional scenario of a $95 \times 95 \times 20m^2$ grid with $N = 125$ agents uniformly distributed over the grid and $M = 8$ anchors placed near the 8

corners of the grid. The maximum communication range is $15m$ for every node and the average node connectivity is $|\bar{\mathcal{N}_i}| \approx 6.1$. For each noise level, 300 independent simulations were run and the RMS error was calculated for all algorithms. Also, $L = 800$ and HEVA/PHEVA used a factor $\alpha = 0.2$.

Initially, a comparison will be made between NBP and the various components of HEVA. Thus, NBP is compared to NBP with the novel kernel, NBP with the ellipsoid particle filter, i.e., ENBP, ENBP with VB and finally the complete HEVA.

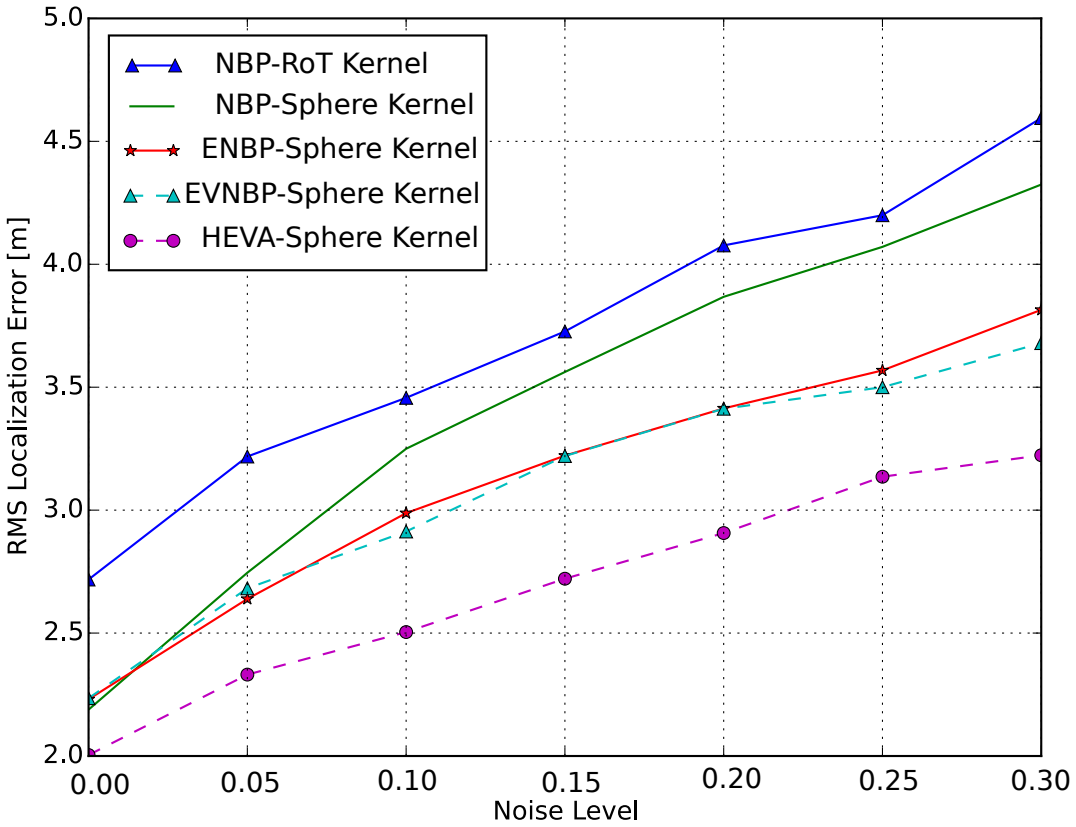

Figure 3.7: The rms error versus the amount of range estimation noise K_e when the average node connectivity is 6.1 and no NLoS.

In Fig. 3.7, results are provided to show that an improvement is achieved as we add the various components of HEVA on NBP. Firstly we see an improvement by adding the kernel to NBP. It should be noted here, that in the 2D case, there is no noticeable improvement in the accuracy by using the kernel compared to the rule of thumb, but a reduction in the computational cost as the kernel covariance

matrix is simpler to compute than the rule of thumb kernel covariance matrix. Also, we see that VB does not improve or deteriorate the results so is used purely in order to convert the pdf to a parametric form. Finally, it is also interesting to note an improvement by using the NLoS message filter even when there are no NLoS edges in the network. This is explained as the filter does not only remove messages from NLoS edges but omits all messages that do not facilitate converge in the later iterations, hence helping convergence.

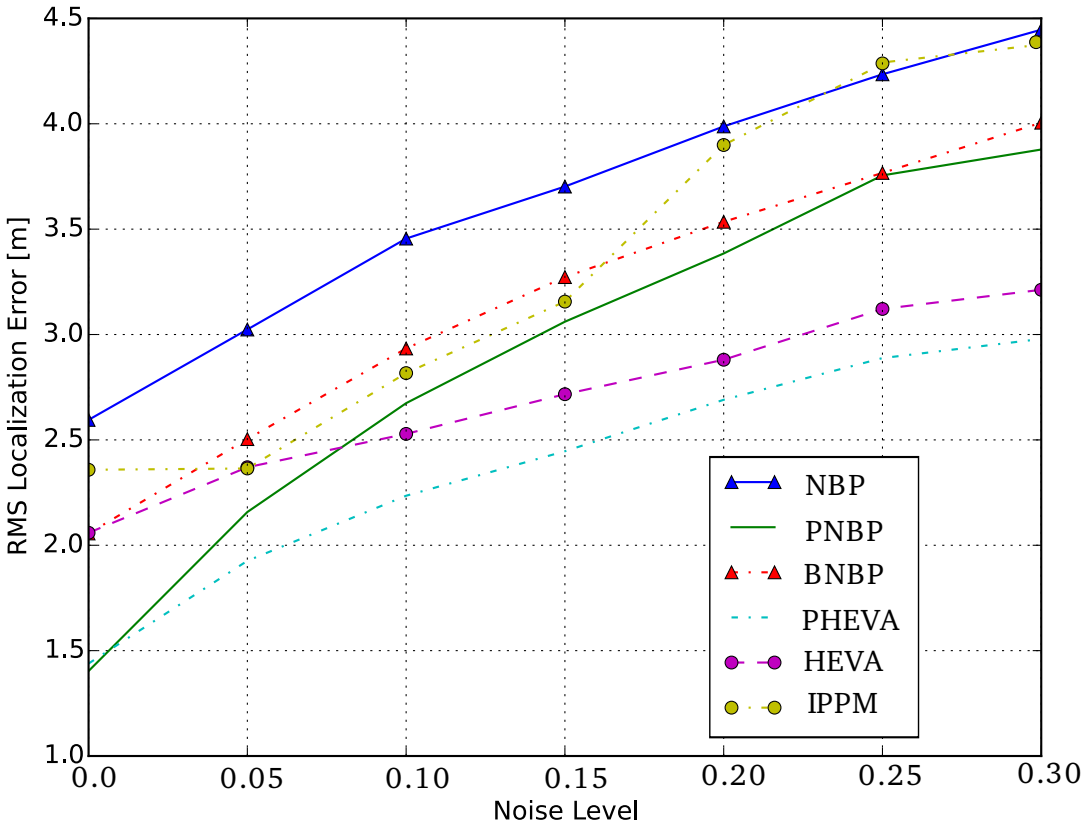

Figure 3.8: The rms error versus the amount of range estimation noise K_e when the average node connectivity is 6.1 and no NLoS.

Next we compare HEVA with the aforementioned competing algorithms in Fig. 3.8. As expected, all algorithms that focus their particles outperform NBP. Further, HEVA and PHEVA outperform all other methods especially in higher noise levels, with HEVA achieving almost equal accuracy to PHEVA for much less computational cost as was shown in Fig. 3.5.

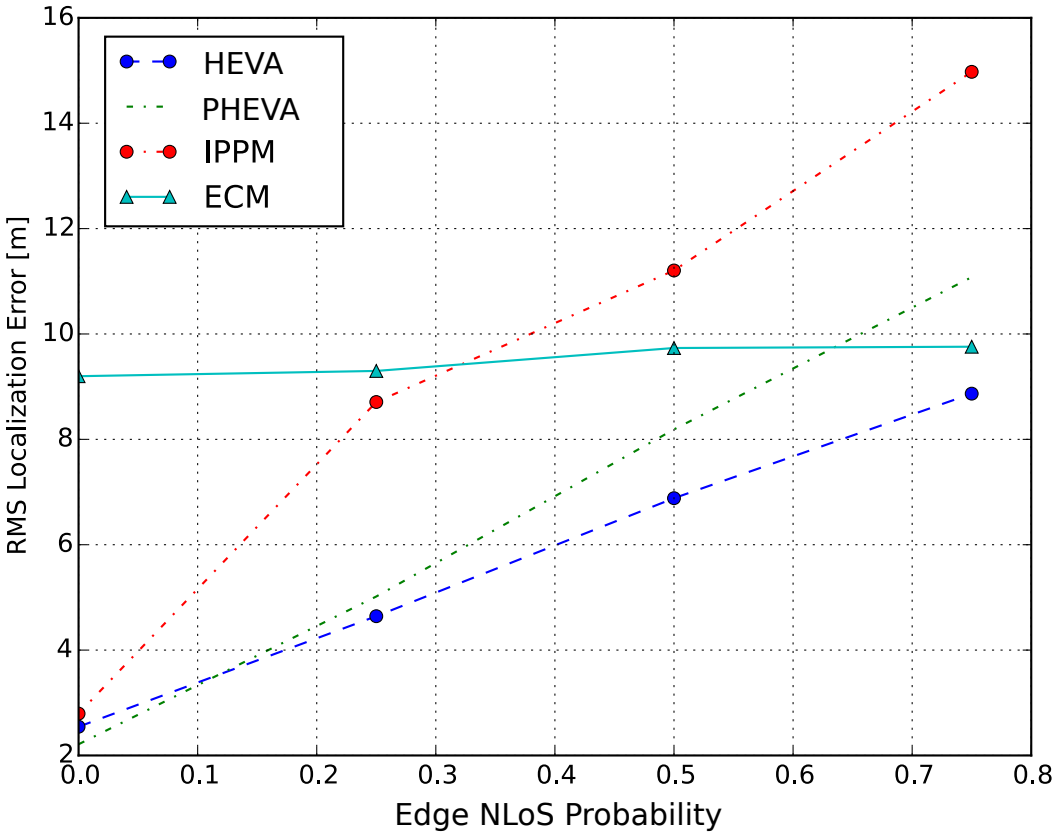

Figure 3.9: The average error versus the edge NLoS Probability, when the average node connectivity is 6.1 .

In Fig. 3.9, we compare the algorithms for various edge NLoS probabilities. HEVA and PHEVA are compared to IPPM and ECM [71], both of which have NLoS mitigation capabilities. As can be seen, both HEVA and PHEVA outperform the other algorithms even in very high NLoS scenarios. As ECM tries to approximate the noise pdf fitting a non-parametric pdf to the measured distances, it manages to keep the RMS error constant at all edge NLoS probabilities. Finally, it is interesting to note that in the case of high edge NLoS probability, HEVA outperforms PHEVA. We believe that as the high noise creates more ambiguity, it is easier for PHEVA to initially focus the particles in the wrong regions, increasing the RMS error.

Finally, we analyse the issue of messages dropped from the NLoS message filter. Depending on the NLoS edge probability, the number of messages dropped varies. The NLoS message filter is quite conservative, so for the NLoS scenario

presented in Section IV, the average number of messages dropped goes from approximately 8% to approximately 13%, as the NLoS edge probability increases. It should be noticed that there is a sweet spot for the NLoS edge probability. Around 0.5 and afterwards, the percentage actually decreases as the NLoS edge probability gets higher. This is because it gets harder to distinguish biased messages and drop them, and the majority will be biased in the first place. Still even in these cases the improvement can be seen in Section V, Fig. 9. Also, as the average number of neighbours, i.e., the average node connectivity, drops so does the average number of messages. Nevertheless, only messages after the first iteration are dropped and the message percentage dropped is at less than 8%, even in really low connectivity, hence not affecting negatively the accuracy.

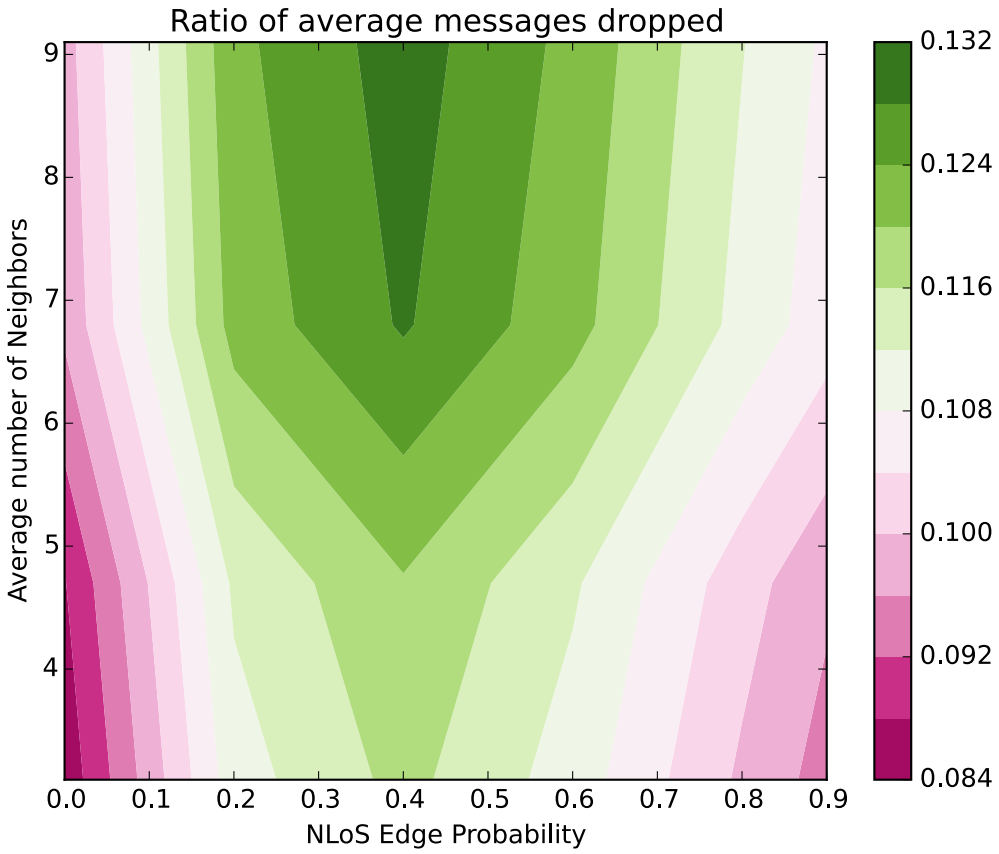

Figure 3.10: The average message drop ratio versus the average neighbours and NLoS edge probability.

3.4.1 Comparison with Other Clustering Techniques

As there is a number of possible clustering techniques that could be used instead of VB, in this subsection, we provide simulation results for two other clustering methods, namely, K-means and EM [18] for comparison. Fig. 3.11 provides the average RMS error results against the number of iterations. As is expected, VB and EM have almost the same results, for both the error cpdfs and the average RMS errors and they both outperform K-means. A close observation for the results further shows that VB performs slightly better than EM. As explained earlier in Section 3.3, the advantage of VB over EM is not only performance but VB can avoid singularity issues,cf. [18] , and can automatically optimize its clustering even if it starts with an arbitrary large number of clusters.

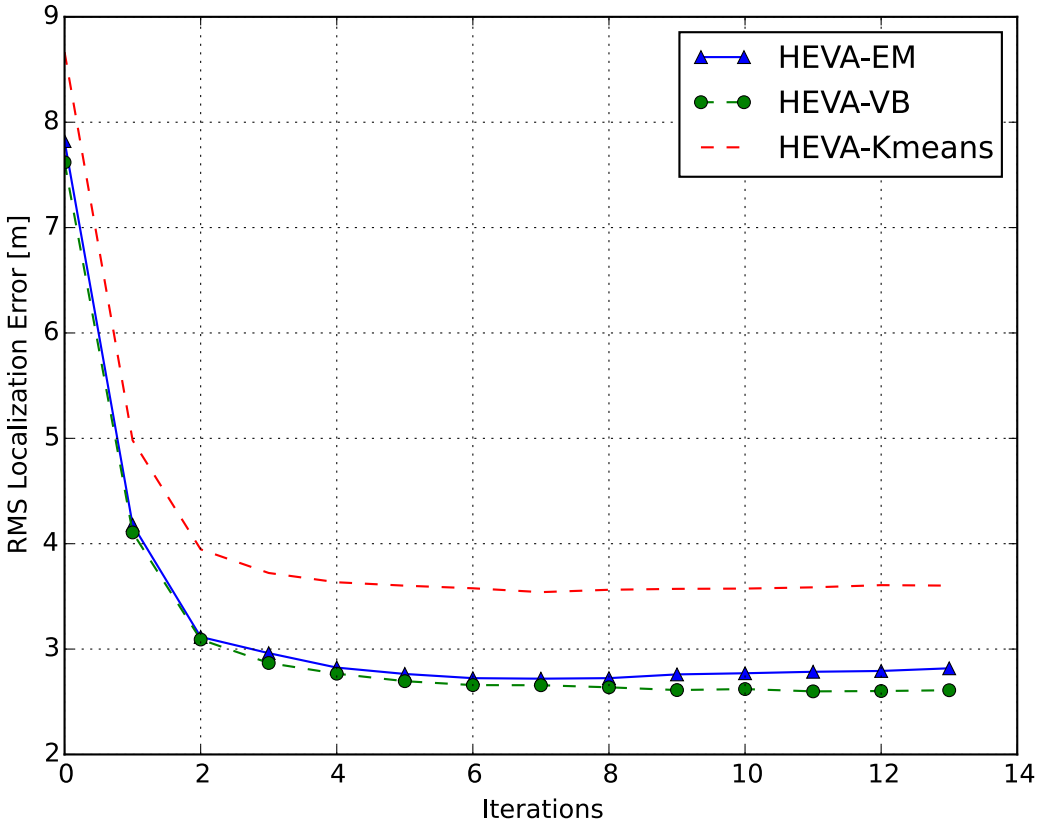

Figure 3.11: The average RMS error versus the number of iterations with $K_e = 0.1$

3.5 Conclusions

In this Chapter, we have presented a novel algorithm for cooperative localisation, named HEVA, that extendsnbp. HEVA combines many novel contributions into a robust, accurate and low complexity algorithm. Firstly we designed a novel low computational cost filter, that intelligently removes NLoS messages. We have created a novel kernel, which is simpler to compute than the "rull-of-thumb" kernel and that outperforms it in accuracy. We have also designed a novel ellipsoid particle filter, that drastically reduces the number of particles required for product calculations in the Sum-Product algorithm. This allows for a decrease in computational cost and also helps improve the solution accuracy. Also we have used a novel conversion of the messages to parametric form, by using VB as a clustering technique. This minimizes the communication overhead, without sacrificing convergence speed, or representational flexibility of the pdf distribution. HEVA has been shown to outperform all other methods, while offering considerable computational advantages and staying robust and computationally cheap in high noise NLoS scenarios and 3D localisation. In Fig. 1.2, HEVA is connected to both the Parametric and Non-Parametric branches as it combines elements of both and manages to overcome the weakness of each technique by taking advantage of the strengths of the other. Still many issues remain. One of which, is the implicit assumption that a known global reference point, i.e. the origin of the axis. In Chapter 4 we resolve this issue.

Chapter 4

Grid Belief Propagation

4.1 Introduction

An often unexplored issue in localisation is how the coordinate system itself works. localisation occurs relative to some commonly accepted reference points. These are called geodetic datums. Two of the more commonly known ones are the WGS84 and the NAD83, cf. [72]. In Global Positioning System (GPS), a polar system is used and the origin of the coordinate system is the center of the planet. In cooperative localisation systems though, typically a Cartesian system is assumed and the nodes localize relative to themselves. Hence, all nodes localise with respect to a *Local Coordinate System (LCS)*. As a result an implicit assumption is hidden in every model: *All nodes know the precise location of the origin point and their relative position to it*. This means that even though the nodes use their own LCS to localize a GCS is assumed to be known by everyone so that every node can convert their LCS to the GCS coordinates. Otherwise the whole coordinate system could be arbitrarily rotated and/or the origin moved arbitrarily around the solutions of the nodes locations would still be valid.

The motivation behind this work is to work around this hidden assumption, of using LCS to localize and then converting the LCS to GCS, thereby allowing a more realistic and flexible system. To the authors' knowledge, the only work that considers the issue is [15], without actually solving it, where Ihler *et al.* proposed a self-calibration algorithm, where a few nodes are arbitrarily chosen as the common

reference point and all other nodes localize relative to them. More typically, the issue just gets completely neglected. Our solution to the problem is to only use a GCS system. This avoids all issues that were created from the LCS. Unfortunately the obvious solution, i.e., using directly the GPS coordinate system, can lead to underflow issues during calculations due to the small distances inherent in indoors localisation. Therefore, the use of constant normalization and scaling is required. Alternatively, we propose the use of a grid-based GCS. This has the advantages of not requiring any normalization of scaling to work and also allow for the easy construction of parametric form pdf leading to efficient and computationally cheap belief message passing.

4.1.1 Our Contributions

We propose a novel technique, called grid-BP that use a grid-based GCS. As a real life example, we use the military grid reference system (MGRS) [73] to showcase our method. This approach does not require a common globally known axis centre, but also has inspired the use of parametric representations using multinomial pdfs. This allows for a fast robust and accurate cooperative localisation algorithm. Even though we use the NATO MGRS coordinate system, any grid-based coordinate system can be used with trivial changes. As a summary, in the algorithm we have made the following contributions:

- We propose a solution to the common reference issue inherent to all distributed cooperative localisation techniques, using the grid-based GCS.
- In addition, we devise parametric approximations to the pdfs, overcoming the computation bottleneck of NBP methods, i.e., the product calculation. Also, the approximations requires no use of an optimization algorithm when calculating the parametric form, thus avoid all the issues of bad local optima that currently plague parametric algorithms.
- Based on the above, we design a novel BP algorithm, namely Grid-BP.
- Simulation results illustrate that the proposed grid-based BP method, or Grid-BP, provides similar accuracy to common techniques.

4.1.2 Organization

The rest of the chapter is organized as follows. In Section 4.2, we first describe how the MGRS by NATO can be employed to provide unique identifiers for multiple resolutions for every point on the planet. We reformulate the equations of the cooperative localisation problem in Section 4.3 in a manner more suitable for the development of the parametric algorithm that will use the MGRS reference system. This is done in Section 4.4, where the parametric BP algorithm which we refer to it as the Grid-BP, is presented. In Section 4.5, simulation results are provided and we have the concluding remarks in Section 4.6.

4.2 Review of MGRS

MGRS is the geo-coordinate standard used by NATO military for locating points on the planet [73]. MGRS is a combination of the the universal transverse Mercator (UTM) grid system and the universal polar stereographic (UPS) grid system, with a different labelling convention. MGRS is a global mesh grid that assigns a unique ID to each grid square. An example id is

$$10QCG12345678, \tag{4.1}$$

where the first part "10Q" is called the Grid Zone Designator (GZD), the second part "CG" is the $100,000$-meter-square identifier, and finally the last numerical part gives the easting (first half digits) and northing (second half digits) inside the square identifier. Every two digits used (for a minimum of 2 and a maximum of 10) increase the resolution by a factor of 10m, down to a resolution of 1m^2 grid squares. Map coordinates are read from west to east first (easting), then from south to north (northing), i.e., left-right, down-up. In cases where the part of the ID is common to all neighbouring nodes, the common part can be dropped and only the rest need to be transmitted or used. For details on the specifics of MGRS the reader is referred to [73]. For the purpose of this chapter and for convenience, we assume that localisation does not occur in these areas and all squares involved are "normal", as the extra controls required are beyond the focus of this thesis and are also quite

straightforward to implement.

4.3 Problem Formulation

We consider a network of nodes in a 2D environment which consists of N agents and M anchors, where $M \geq 4$ and $N \gg M$. Let $\boldsymbol{X} = [X_1, \ldots, X_i, \ldots, X_{N+M}]$ be the random variable vector for the locations of all nodes, with X_i representing the random variable of the MGRS identifier of node i and $X_i \in \{x_1, x_2, \ldots x_k\}$, where k iterates over all possible MGRS IDs. Also let $\boldsymbol{\theta}$ denote the coordinates of all nodes, with $\boldsymbol{\theta}_i$ representing the coordinates of node i, and the domain of $\boldsymbol{\theta}_i$ is $\Re^2$. As before the use of upper case $\boldsymbol{\Theta}_i$ and $\boldsymbol{\Theta}$ represent the respective random variable and vector random variable of the coordinates of every and all nodes. The nodes communicate wirelessly and it is assumed that the maximum communication range for each node is $R_{\max}$. Time is slotted and time slots are denoted by the time index superscript (t) for $t = 1, 2, \ldots, \infty$.

We represent the problem as a joint probability distribution. Let $\Pr^{(t)}(X_i)$ be the pdf, i.e., the belief that node i has about its location at time t. We model $\Pr^{(t)}(X_i)$ as a multinomial distribution with parameters $z_1, \ldots, z_k$ where $\Pr(X_i = x_k) = z_k$ is the probability of node i being in MGRS ID x_k and $\sum_k z_k = 1$. Also let the set of all nodes j within range of node i be denoted as the neighbourhood $\text{ne}(i)$.

Initially, the belief for the agents can be a non-informative uniform pdf over the grid, while the anchors' pdfs are focused in the IDs close to the real position, i.e., within 10m.

Nodes obtain distance estimate via ranging. As in Chapter 3, we assume that the nodes use the average of their corresponding measurements. Consequently, for two nodes i and j we define the random variable R_{ji} with values r_{ji} as

$$r_{ji} = \|\boldsymbol{\theta}_i - \boldsymbol{\theta}_j\| + \eta_{ji}, \tag{4.2}$$

where η_{ji} is a noise factor following a Gaussian distribution with variance $s_{ji}^2 = K_e \|\boldsymbol{\theta}_i - \boldsymbol{\theta}_j\|^{\beta_{ji}}$ in which K_e is a proportionality constant capturing the combined physical layer and receiver effect, and β_{ji} denotes the path loss exponent, as dis-

cussed in the previous chapters.

We define the likelihood of node i and node j measuring distance $R_{ji} = r_{ji}$ between them at time t, given X_i, X_j as

$$\overset{(t)}{\Pr}(R_{ji} = r_{ji} \mid X_i, X_j) \propto \exp\left(-\left(\frac{r_{ji} - ||C_i - C_j||_2}{h}\right)^2\right), \tag{4.3}$$

where h controls steepness, C_i and C_j are the coordinates of the centers of the grids' squares X_i and X_j, respectively. Therefore, our objective is to find the maximum a posteriori (MAP), i.e., the values that maximize $\Pr(\boldsymbol{X}|\boldsymbol{R})$ given distance measurements $\boldsymbol{R} = [R_{ji}]$. For a specific node i, we have

$$\hat{X}_i = \arg\max_{X_i} \overset{(t)}{\Pr}(X_i|\boldsymbol{R}_i). \tag{4.4}$$

Thus, $\Pr^{(t)}(X_i|\boldsymbol{R}_i)$ can be evaluated using the Bayes' rule as

$$\begin{aligned}\overset{(t)}{\Pr}(X_i|\boldsymbol{R}_i) &\propto \overset{(t)}{\Pr}(X_i) \prod_{j\in \mathrm{ne}(i)} \overset{(t)}{\Pr}(R_{ji}|X_i) \\ &\propto \overset{(t)}{\Pr}(X_i) \prod_{j\in \mathrm{ne}(i)} \int \overset{(t)}{\Pr}(R_{ji}|X_i, X_j) \overset{(t)}{\Pr}(X_j) \mathrm{d}X_j,\end{aligned} \tag{4.5}$$

in which the sign "$\propto$" means "is proportional to", and normalization should be done to obtain the pdf.

4.4 The Proposed Grid-BP Algorithm

In this section, we present the proposed grid-based localisation algorithm. Firstly we will formulate a valid cluster graph and the respective message passing equations, as in Chapter 2 for the model discussed in the previous Section 4.3. Then efficient approximation for the marginalization and the product operation will be given.

4.4.1 Belief Message Passing

We model the network as a Bethe cluster graph. The lower factors are composed of univariate potentials $\psi(X_i)$, while the upper region is composed of "large" clusters with one cluster for each factor $\psi(X_i, X_j, R_{ji})$. An example can be seen in Fig. 4.1.

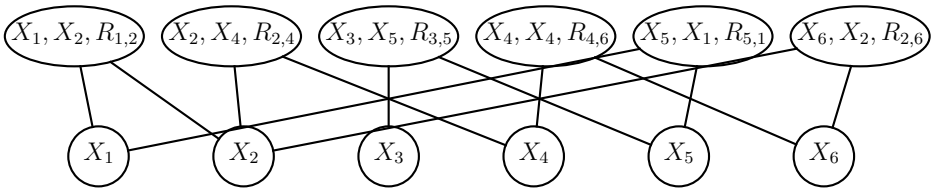

Figure 4.1: The cluster graph for the Grid-BP PGM.

The lower factors are set to the initial beliefs for the given time slot (t), and the upper factors to the corresponding cpdfs

$$\psi(X_i) = \overset{(t)}{\Pr}(X_i), \tag{4.6}$$

$$\psi(X_i, X_j, R_{ji} = r_{ji}^{(t)}) = \overset{(t)}{\Pr}(R_{ji} = r_{ji}^{(t)} | X_i, X_j). \tag{4.7}$$

Messages are then passed between nodes for multiple iterations until the node beliefs have converged. The message from node j to node i, at BP iteration $(s+1)$ is calculated by

$$\mu_{j\to i}^{(s+1)}(X_i) = \int \psi(X_i, X_j, R_{ji} = r_{ji}^{(t)}) \frac{b_{j\to i}^{(s)}(X_j)}{\mu_{i\to j}^{(s)}(X_j)} \mathrm{d}X_j, \tag{4.8}$$

where intuitively, a message (4.8) is the belief that node j has about the location of node i and $r_{ji}^{(t)}$ is the observed value of the distance between the nodes, at time slot t.

Then the belief of node i is updated as

$$b_i^{(s+1)}(X_i) = \lambda \psi(X_i) \prod_{k \in \mathrm{ne}(i)} \mu_{k\to i}^{(s+1)}(X_i) + (1-\lambda) b_i^{(s)}(X_i), \tag{4.9}$$

where λ is a dampening factor used to facilitate convergence.

BP continues until convergence, or if convergence is not guaranteed, s reaches a maximum number of iterations $I_{\max}$. Then the beliefs, representing approximations to the true marginals, are found by (4.9), i.e. $\Pr^{(t+1)}(X_i) = b_i^{(s+1)}(X_i)$, for each node. The proposed Grid-BP is given as Algorithm 6. Each node needs to perform a marginalization operation (4.8) and a product operation (4.9). Approximations are required for both complex operations. In Grid-BP, we take advantage of the multinomial parametric form which we discuss next.

Algorithm 6 Grid-BP

Initialize beliefs $p^{(0)}(X_i)\ \forall i \in$ Nodes
for $t = 0$ to T **do**
  for all $i \in$ Nodes **do**
    Broadcast current belief $p^{(0)}(X_i)$
    for all $j \in \text{ne}(i)$ **do**
      Collect distance estimates $r_{ji}^{(t)}$
    end for
  end for
  Initialize $\psi(X_i) = \Pr^{(t)}(X_i)$
  Initialize $\psi(X_i, X_j, R_{ij}) = \Pr^{(t)}(R_{ij} = r_{ji}^{(t)} \mid X_i, X_j)$
  repeat
    for all $i \in$ Nodes **do**
      for all $j \in \text{ne}(i)$ **do**
        Receive $b_j^{(s)}(X_j)$
        Calculate $\mu_{j\to i}^{(s+1)}(X_i)$, using (4.8) using HSM Gibbs sampling (i.e., Algorithm 7)
      end for
      Calculate $b_i^{(s+1)}(X_i)$, using (4.9).
      Check for convergence
      Send $b_i^{(s+1)}(X_i)$
    end for
  until convergence or s reaches $I_{\max}$
  Update belief $\Pr^{(t+1)}(X_i)$, using (4.9).
end for

4.4.2 Marginalization Operation

The calculation of (4.8) essentially gives the belief node j has about node i. Since the marginalization is too costly, we will use a Gibbs sampler, similar to Chapter 3, to approximate the message. To understand the effect of R_{ji}, we consider the

case that in a specific time slot t', $b_j^{(s)}(X_j)$ has only one ID, i.e., x_j. Then the high probability grid squares of (4.8) will be approximated by IDs of grid buckets that approximately form a circle of radius $r_{ji}^{(t')}$ and centre the x_j. With that in mind, firstly we sample L particles $x_j^{(l)} \sim \mu_{j\to i}(X_j)$. Then we sample L samples $\phi^{(l)} \sim \mathscr{U}(0,2\pi)$ and finally L samples from $\hat{r}_{ji}^{(l)} \sim \mathscr{N}(r_{ji}^{(t)},h)$. The Gibbs sampling algorithm is provided as Algorithm 7.

Algorithm 7 Gibbs Sampling

1: Set $\mathscr{D}_{X_i}$ to empty
2: **for all** $j \in \text{ne}(i)$ **do**
3: Sample $x_j^{(l)} \sim \mu_{j\to i}(X_j)$ which is a multinomial pdf
4: Sample $\phi^{(l)} \sim \mathscr{U}[0,2\pi]$
5: Sample $\hat{r}_{ji}^{(l)} \sim \mathscr{N}(r_{ji}^{(t)},h)$
6: $x_i^{(l)} = \mathsf{MAP\text{-}DMtoID}(x_j^{(l)}, r_{ji}^{(t)}, \phi^{(l)})$ which maps the distance metric to IDs
7: Add $\{x_i^{(l)}\}_{l=1}^{L}$ to $\mathscr{D}_{X_i}$
8: **end for**
9: **return** $\mathscr{D}_{X_i}$

Sampling is repeated, for all incoming messages, and then for all $\{x_j^{(l)}, \hat{r}_{ji}^{(t)}, \phi^{(l)}\}_{l=1,j=1}^{L,|\text{ne}(i)|}$, we use

$$x_i^{(l)} = \text{MAP} - DMtoID(x_j^{(l)}, \hat{r}_{ji}^{(t)}, \phi^{(l)}) \tag{4.10}$$

to get the set $\mathscr{D}_i = \{x_i^{(l)}\}$. Intuitively we do this by counting for each sampled ID $x_j^{(l)}$ the number of IDs to the east and to the north, node i will be, given the measured distance $r_{ji}^{(t)}$ normalized by D as follows

$$\begin{bmatrix} d_H \\ d_V \end{bmatrix} = \text{int}\left(\frac{\hat{r}_{ji}^{(t)}}{D}\begin{bmatrix} \cos(\phi^{(l)}) \\ \sin(\phi^{(l)}) \end{bmatrix}\right). \tag{4.11}$$

Then, the displacement d_H, d_V is translated to a new ID and return it as $x_i^{(l)}$. For MGRS IDs the translation is done by adding d_H, d_V to the easting and northing components of the ID of $x_j^{(l)}$, as discussed in Section 4.2. The distance to ID mapping function is given as Algorithm 8. It should be noted that no reverse mapping is required in the case of MGRS.

Algorithm 8 MAP-DMtoID

1: Calculate horizontal and vertical steps using (4.11)
2: Map horizontal ID $x_j^{(l)} \rightarrow b_j^{(l)}$
3: $b_h^{(l)} = b_j^{(l)} + d_H$
4: Inverse horizontal mapping $b_h^{(l)} \rightarrow x_h^{(l)}$
5: Map vertical ID $x_h^{(l)} \rightarrow b_v^{(l)}$
6: $b_i^{(l)} = b_v^{(l)} + d_V$
7: Inverse vertical mapping $b_i^{(l)} \rightarrow x_i^{(l)}$
8: **return** $x_i^{(l)}$

Finally, as will be shown in the sequel, the set $\mathscr{D}_{X_i}$ of all samples obtained from all incoming messages is used to find (4.9).

4.4.3 Product Operation

To obtain (4.9), firstly we create parametric forms of the incoming messages (4.8), by using the particles in $\mathscr{D}_{X_i}$. We assume that the parameters of the multinomial are random variables $\boldsymbol{Z}_i$ with a uniform Dirichlet prior with parameters α_k, where $k \in \{1,\ldots,K\}$ and K is the number of unique IDs in the multinomial. We use the samples from each incoming message as observations and get the MAP estimate $\hat{\boldsymbol{z}}_i$ of the parameters $\boldsymbol{Z}_i$ of each (4.8). We also consider that all incoming messages have the same prior distribution. Based on the above we calculate the parameters for each multinomial as follows

$$\hat{\boldsymbol{z}}_i = \mathbb{E}\left[\Pr(\boldsymbol{Z}_i|\mathscr{D}_{X_i})\right] = \mathbb{E}\left[\Pr(\mathscr{D}_{X_i}|\boldsymbol{Z}_i)\Pr(\boldsymbol{Z}_i)\right], \tag{4.12}$$

which gives

$$\hat{z}_{i,k} = \frac{M_k + \mid \mathrm{ne}(i) \mid \alpha_k}{\mid \mathrm{ne}(i) \mid \sum_k (M_k + \alpha_k)}, \tag{4.13}$$

where M_k is number of particles x_k in $\mathscr{D}_{X_i}$. The algorithm is presented in Algorithm 9.

For clarity, the quantities of each ID in the samples are shown as being found by a count function but in practice it can be done during the Gibbs sampling step allowing for a more efficient algorithm.

Algorithm 9 MAP Parameter Estimation

1: Let $|X_i|$ be the number of unique IDs in $\Pr(X_i|\boldsymbol{Z}_i)$
2: Calculate $M_k = count(x_k, \mathscr{D}_{X_i})\ \forall k \in |X_i|$
3: **for all** $k \in |X_i|$ **do**
4: Calculate $\hat{z}_{i,k}$, with (4.13)
5: **end for**
6: **return** $\hat{\boldsymbol{z}}_i$

After obtaining the pdfs involved in the calculation of (4.9), the resulting pdf will simply be the dot product of the $\boldsymbol{z}_i$ parameters of each incoming message. The parameters of (4.9) can be directly calculated by adding the logs of the corresponding $\hat{\boldsymbol{z}}_i$'s.

4.4.4 Message Filtering

As it makes no sense to keep all the MGRS ids on the planet, we can assume that each node constructs pdfs with the IDS within 100m of the IDs it received in the first iteration. To reduce the ids further we propose a simple filter that only keeps the most probable IDs summing up to an energy threshold of the respective cdf. The rest are assumed to share uniformly the remainder of the pdf energy. After Monte Carlo simulations, it was evident that by keeping $\sim 80\%$ of the total energy of the pdf, the size of the messages transmitted is decreased by $\sim 90\%$ with barely any increase in localisation error. Thus, assuming that each message covers a $100 \times 100\text{m}^2$ grid, the total number of IDs used without the filter would be 10^4. After the filter, however, only $\sim 10^2$ IDs will be transmitted.

4.4.5 Convergence

A cluster graph is not guaranteed to determine the optimal solution. However, if the running intersection property and the family preservation property are held, typically most clusters will converge to a local optimum with only a few clusters oscillating unable to converge [53]. To facilitate convergence, as in HEVA, see Chapter 3, a dampened BP message passing is used, which empirically helps and even the

oscillating nodes converge to a local optimum. This is accomplished by setting

$$\delta_{i\to\text{all } j\in\text{ne}(i)}^{\text{lower}\to\text{upper}}(X_i) = \lambda\,\psi(X_i)\prod_{k\in\text{ne}(i)}\delta_{k\to i}^{\text{upper}\to\text{lower}}(X_i) + (1-\lambda)\delta_{i\to\text{all } j\in\text{ne}(i)}^{(\text{old})\text{lower}\to\text{upper}}(X_i), \tag{4.14}$$

where λ is the dampening factor. The Kullback-Leibler (KL) divergence between the current and previous iterations message is calculated and if it falls within a certain threshold the node will consider that it has converged to a solution.

4.4.6 Complexity

The complexity for the marginalization filtering is $\mathcal{O}(L|\text{ne}(i)|)$ and the computational cost for the product operation is also $\mathcal{O}(L|\text{ne}(i)|)$, which corresponds to the complexities suggested in [62] for parametric techniques. For completeness, the different complexities for discretized, non-parametric and parametric techniques are shown in Table 4.1 as shown in [62].

Table 4.1: Comparison of complexity costs for Discretised, non-parametric and parametric techniques where L is the number of particles and M is the number of messages involved in the operations.

Approach	Operation	complexity	Value of L
Descretised	Marginalization	$\mathcal{O}(L^2)$	Large
Descretised	Product	$\mathcal{O}(LM)$	Large
Non-Parametric	Marginalization	$\mathcal{O}(L)$	Small
Non-Parametric	Product	$\mathcal{O}(L^2M)$	Small
Parametric	Marginalization	$\mathcal{O}(L)$	Small
Parametric	Product	$\mathcal{O}(LM)$	Small

4.5 Simulations

To evaluate the performance, 100 Monte Carlo simulations were conducted and the localisation error cdf was calculated. In each simulation 100 nodes with 20 anchors are placed randomly in a 100m $\times$ 100m area and the communication range is limited to 12m and $|\text{ne}(i)|_{\text{avrg}} = 4.03$. Anchor locations are modelled as multivariate Gaussian pdfs with an identity variance matrix. The grid resolution for Grid-BP is

$D = 1$m. We compare Grid-BP with the NBP [57], and Hybrid-BP [74].[1] We also compare Grid-BP with HEVA-BP, a computationally cheaper variation of NBP in [50]. In addition, $I_{\max} = 15$ and 800 particles were sampled. Finally, the noise factor used was $K_e = 0.3$. Furthermore, a variant of HEVA that uses GPS coordinates as a reference system was also provided. In HEVA-GPS messages contain GPS coordinates that every nodes converts an LCS before calculating (4.8) and (4.9).

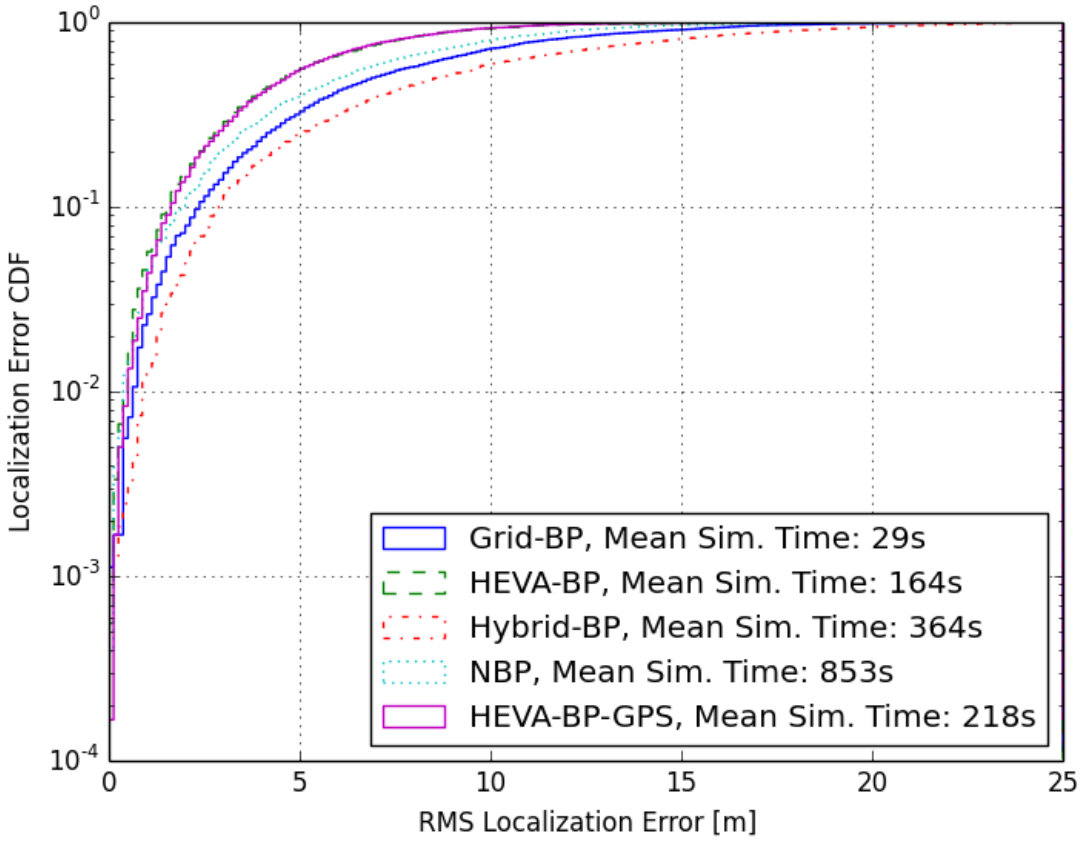

Figure 4.2: Comparison of localisation error cdf with Grid-BP.

Fig. 4.2 shows the localisation error cdf of all the algorithms. Results illustrate that HEVA-BP, NBP and Grid-BP achieve similar results with Grid-BP having slightly better results than Hybrid-BP, which is the only other parametric algorithm. Note that all three algorithms, HEVA, NBP and Hybrid-BP have the strong assumption of sharing knowledge of the GCS origin, while Grid-BP and HEVA-GPS do

[1] For Hybrid-BP, we did not use information given from satellites as in [74].

not (the realistic scenario). Even though HEVA-BP-GPS achieves as good results as HEVA-BP, the 25% increase in computational cost with HEVA-BP due to the mapping of the GPS coordinates to a local Cartesian reference frame, is evident in the mean simulation time. Also, the relative computational efficiency of Grid-BP can also be seen. This is due to the very efficient calculations of both message passing operations compared to the other algorithms. Note that all simulations were run on an Intel i7 2.6GHz, using Python for scientific computing [75].

4.6 Conclusions

Concluding, in this chapter we have presented a novel parametric algorithm for BP that uses a grid based system, Grid-BP. We have solved the issue of coordinate system ambiguity in distributed cooperative localization and have also designed a novel parametric algorithm with extremely low complexity cost. The resulting algorithm combines a global reference system that alleviates the hidden issue of reference, and a parametric representation that allows quick BP. In the tree in Fig. 1.2, Grid-BP fits smoothly in the parametric family of algorithms where it manages to be the only one currently that avoids the requirement of an optimization algorithm to calculate the parameters. Grid-BP's performance is very good even in high noise simulations. This allows for a very low complexity algorithm which can achieve high accuracy in the Position Location problem. It should be noted that the algorithm can be implemented for any arbitrary GCS, depending on application and required accuracy avoiding potential overflowing issues or coordination between nodes. In the next chapter we will extend Grid-BP in a temporal model, adding information from internal sensors to allow for mobility.

Chapter 5

Probabilistic Hybrid INS/PDR Mobility Tracking Algorithm

5.1 Introduction

Coping with mobility has been key in localisation research. The typical scenario is to complement GPS with information from IMUs and odometers to provide uninterrupted navigation solutions during GPS outages. The integration is typically achieved by a Kalman Filter [76], or some variants, e.g., the extended Kalman filter[77]. Unfortunately, the significant errors of MEMS inertial sensors as well as the time-varying models cannot be modelled accurately by the Kalman Filter linearised models. As an alternative to better capture the non-linearities, the use of a particle filter has been proposed [78].

The information from the sensors provides all the necessary information required for tracking of human movement. There has been much research on PDR and its applications. In PDR, the frequency of the pedestrians steps is extracted from the sensor information and assuming some statistical model for the length and course of the steps, the pedestrians direction and distance travelled are calculated by summing up all the steps, cf. [79]. There has also been extensive research in the classification and modelling of sensor outputs with different human movement, cf. [80, 4], but the reality is that erratic movement, e.g., walking in slopes, or abrupt movements, that can typically occur in a battlefield, will make the error

grow quickly out of hand.

Alternatively, pedestrian tracking can be treated as an application of a strap-down INS. In this case, the orientation of a sensor module is tracked by integrating the angular velocities, which are subsequently used to determine the acceleration components in the GCS. Then the gravity acceleration is subtracted and the remaining acceleration is integrated over time to find the sensors displacement. Unfortunately, low cost MEMS-IMUs are susceptible to errors, such as misalignment errors, scale factor, bias turn-on error, bias drift error, etc. Though deterministic errors can typically be removed via calibration, stochastic errors cannot be removed and can increase quickly. Analysis and modelling of the MEMS-IMU errors can be found in [81, 78, 82].

A solution proposed in [83] has been to provide a synergism between PDR and INS. Essentially the movement of the pedestrian will be calculated by finding the orientation and number of steps as in PDR, but the characteristics will be derived from the IMU measurements instead of using a statistical model.

The combination of PDR localisation and cooperative localisation for GPS denied environments however has not been well investigated. The SPAWN framework, cf. [9] considered mobility, but it was demonstrated in [84] that it is too computationally expensive for real-world hand-helds and a heuristic cooperative localisation algorithm, called SnG, was proposed as an alternative. SnG keeps the computational cost low for mobile devices while synergising with PDR. Still due to the heuristics in SnG, the network is highly susceptible to node placement and if the placement is not uniform enough, then the whole network localisation will collapse.

5.1.1 Our Contributions

In this chapter, we present PHIMTA a novel algorithm that was designed as a robust low-cost algorithm for pedestrian mobility tracking in GPS-denied environments. Firstly, we devise dynamic Bayesian network in which we generalize the INS/PDR pedestrian tracking algorithm in [83] by using probabilistic particle representations. Then we combine it with Grid-BP, cf. Chapter 4, to have a fully distributed and robust probabilistic model for mobile pedestrian tracking which combines low com-

putational cost, as well as robustness in different node geometries, and high localisation accuracy. In the context of cooperative localisation mobility, typically a local distribution is assumed that will be used as initial potential for each node, INStead of the non-informative uniform distribution of agents. In this work we designed a stochastic implementation, based on the algorithm above, combined with a particle filter that will be used to integrate the local information derived form the MEMS-IMU and the incoming messages received from the nodes neighbours.

Our contributions can be summarized as follows:

- We design a probabilistic pedestrian tracking technique which is referred to as the particle hybrid inertial measurement tracking algorithm (PHIMTA).
- Also, we combine PHIMTA with Grid-BP, creating a novel dynamic Bayesian network model most suitable for mobile localisation in GPS-denied environments. Grid-BP was chosen due to the low computational cost, but any message passing variant can be used.
- We conduct Monte Carlo simulations using real data from [85, 86] that show that PHIMTA/Grid-BP provides consistently equivalent accuracy with drastically decreased computational cost, compared to the literature.

5.1.2 Organization

The rest of the chapter is organized as follows. In Section 5.2 the problem is formulated and an extended temporal PGM is presented. In Section 5.3 the BP timeslots are briefly analyzed. Then, a more in depth discussion of the IMU Timeslots is presented in Section 5.4. In Section 5.5 the complexity of the algorithm is discussed. Afterwards, in Section 5.6 simulation results are shown from Monte Carlo simulations using data from real life measurement campaigns. Finally the chapter is concluded in Section 5.7.

5.2 Problem Formulation

We consider a similar model as in the previous chapters and extend it by allowing mobility for the nodes. We assume as before a network of nodes in a 2D environ-

ment which consists of N agents and M anchors, where $M \geq 4$ and $N \gg M$. Let $\boldsymbol{X} = [X_1, \ldots, X_i, \ldots, X_{N+M}]$ be the locations of all nodes, with X_i representing the unique identifier of node i and $X_i \in \{x_1, x_2, \ldots x_k\}$, where k iterates over all possible IDs. Also, let $\boldsymbol{Z}$ denote the coordinates of all nodes, with $\boldsymbol{Z}_i$ representing the coordinates of node i, and the domain of $\boldsymbol{Z}_i$ is $\Re^2$. The nodes communicate wirelessly and it is assumed that the maximum communication range for each node is $R_{\max}$. Time is slotted and time slots are denoted by the time index superscript (t) for $t = 1, 2, \ldots, \infty$.

Based on the behaviour of the nodes, there are two types of time slots. Firstly, the nodes might move and use IMU information to update their information, namely IMU time slots, or they might stay idle and cooperate with their neighbours to update their location estimate, namely BP time slots. The nodes use cooperative localisation every n time slots, while in between they have a probability $\Pr(W^{(t)})$ at each time slot to wait or to move. If a node is moving during a BP time slot, then it will not participate in the message passing algorithm. It uses the SHOE filter, cf. Section 5.4, to discriminate between being idle or not.

Let $\Pr(X_i^{(t)})$ be the pdf, i.e., the belief that node i has about its location at time t. We model $\Pr(X_i^{(t)})$ as a multinomial distribution with parameters θ_k, where θ_k is the probability of node i being in ID x_k and $\sum_k \theta_k = 1$. Let $\Pr(\boldsymbol{X}^{(t)})$ denote the state of the system at time slot (t). We assume that the system is Markovian and represent it as a pdf. Then we have

$$\Pr(\boldsymbol{X}^{(0)}, \boldsymbol{X}^{(1)}, \ldots, \boldsymbol{X}^{(t)}) = \Pr(\boldsymbol{X}^{(0)}) \prod_{\tau=0}^{t-1} \Pr(\boldsymbol{X}^{(\tau+1)} | \boldsymbol{X}^{(\tau)}), \tag{5.1}$$

where $\Pr(\boldsymbol{X}^{(0)})$ is the initial system state and $\Pr(\boldsymbol{X}^{(\tau+1)} | \boldsymbol{X}^{(\tau)})$ is the transition probability. Depending on the type of time slot the transition probability will change. Thus, we have

$$\Pr(\boldsymbol{X}^{(\tau+1)} | \boldsymbol{X}^{(\tau)}) = \begin{cases} \Pr(\boldsymbol{X}^{(\tau+1)} | \boldsymbol{X}^{(\tau)}, \boldsymbol{R} = \boldsymbol{r}), \text{for BP time slots,} \\ \Pr(\boldsymbol{X}^{(\tau+1)} | \boldsymbol{X}^{(\tau)}, \boldsymbol{O} = \boldsymbol{o}), \text{for IMU time slots,} \end{cases} \tag{5.2}$$

in which $\boldsymbol{r}$ denotes a vector with all the distance measurements between nodes, and $\boldsymbol{o}$ is a vector with the IMU observations. The above can be described graphically in Fig. 5.1.

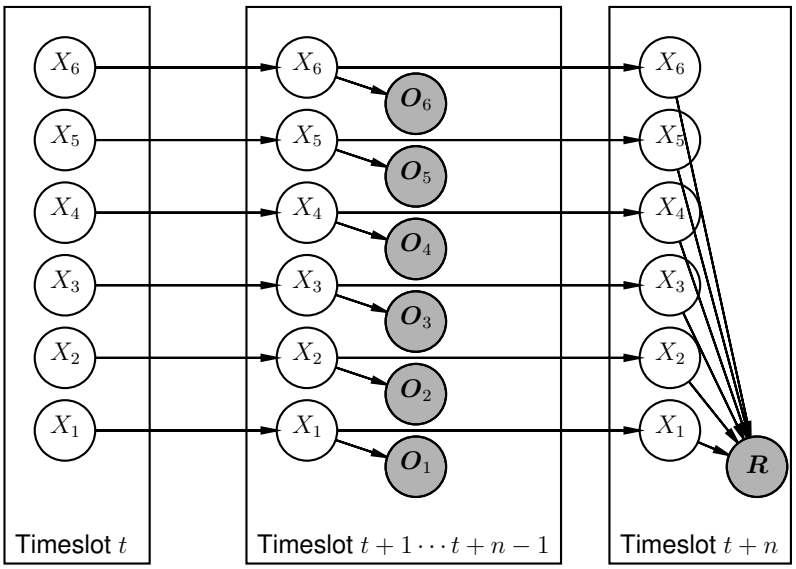

Figure 5.1: A dynamic Bayesian network model.

During the BP time slots, let the set of all nodes within the range of node i be the neighbourhood $\mathcal{N}_i$. Initially, the belief for the agents can be a non-informative uniform pdf over the grid, while the anchors' pdfs are focused in the IDs close to the real position, e.g., within 10m. Node i receiving a message from node j at time slot t can derive, using ToA measurements,[1] a noisy estimate $r_{j\to i}^{(t)}$ of the distance between them. For convenience, we assume $r_{j\to i}^{(t)} = r_{i\to j}^{(t)} = r_{ji}^{(t)}$.

Thus, as before, for ToA distance measurements, we define the random variable $R_{ji}^{(t)}$ with its value $r_{ji}^{(t)}$ modelled as

$$r_{ji}^{(t)} = \|\boldsymbol{z}_i^{(t)} - \boldsymbol{z}_j^{(t)}\| + \eta_{ji}, \tag{5.3}$$

where η_{ji} is a Gaussian noise with variance $\sigma_{ji}^2 = K_e\|\boldsymbol{z}_i^{(t)} - \boldsymbol{z}_j^{(t)}\|^{\beta_{ji}}$ in which K_e is a proportionality constant capturing the combined physical layer and receiver effect,

[1] The assumption of using ToA is not restrictive on the proposed algorithm because it can easily be used with other measurement models.

and β_{ji} denotes the path loss exponent. In the case of line-of-sight (LoS), η_{ji} is assumed zero mean, and $\beta_{ji} = 2$, i.e., $\eta_{ji} \sim \mathcal{N}(0, \sigma_{ji}^2)$.

We define the likelihood of node i and node j measuring distance $R_{ji}^{(t)} = r_{ji}^{(t)}$ between them at time t, given X_i, X_j exactly as in chapter 4, eq. (4.3):

$$\Pr(R_{ji}^{(t)} = r_{ji}^{(t)} | X_i^{(t)}, X_j^{(t)}) \propto \exp\left(- \left(\frac{r_{ji}^{(t)} - \|C_i^{(t)} - C_j^{(t)}\|_2}{h} \right)^2 \right), \tag{5.4}$$

where h controls steepness, $C_i^{(t)}$ and $C_j^{(t)}$ are the coordinates of the centres of the grids' squares $X_i^{(t)}$ and $X_j^{(t)}$, respectively. Thus, we aim to find the maximum a posteriori (MAP), i.e., the values that maximize $\Pr(\boldsymbol{X}^{(t+1)} | \boldsymbol{R}^{(t+1)}, \boldsymbol{X}^{(t)})$ given distance measurements $\boldsymbol{R}^{(t+1)} = [R_{ji}^{(t+1)}]$. For node i, we have

$$\hat{X}_i = \arg\max_{X_i} \Pr(X_i^{(t+1)} | \boldsymbol{R}_i^{(t+1)}, X^{(t)}). \tag{5.5}$$

Consequently, $\Pr(X_i^{(t+1)} | \boldsymbol{R}_i^{(t+1)}, X_i^{(t)})$ can be evaluated using the Bayes' rule as

$$\begin{aligned}
&\Pr(X_i^{(t+1)} | \boldsymbol{R}_i^{(t+1)}, X_i^{(t)}) \\
&\quad \propto \Pr(X_i^{(t)}) \prod_{j \in \mathcal{N}_i} \Pr(R_{ji}^{(t+1)} | X_i^{(t+1)}) \\
&\quad \propto \Pr(X_i^{(t)}) \prod_{j \in \mathcal{N}_i} \int \Pr(R_{ji}^{(t+1)} | X_i^{(t+1)}, X_j^{(t+1)}) \Pr(X_j^{(t+1)}) \mathrm{d}X_j,
\end{aligned} \tag{5.6}$$

in which the sign "$\propto$" means "is proportional to", and normalization should be done to obtain the pdf.

Similarly, during the IMU time slots, we have

$$\Pr(X_i^{(t+1)} | \boldsymbol{O}_i^{(t+1)}, X_i^{(t)}) \propto \Pr(X_i^{(t)}) \Pr(\boldsymbol{O}_i^{(t+1)} | X_i^{(t+1)}). \tag{5.7}$$

The model is summarized in Algorithm 10.

In the next sections, we will describe how Grid-BP is used to solve (5.6) and then how PHIMTA solves (5.7).

Algorithm 10 Dynamic Bayesian Network

```
for all i ∈ N do
    Initialize Pr(X_i^(0)), v_i^(0), ω_i^(0), a_i^(0), μ_i^(0)
    for all t ∈ Time slots do
        if time slot is BP then
            calculate (5.2) using Grid-BP, Algorithm 6
        else
            calculate (5.2) using PHIMTA, Algorithm 11
        end if
    end for
end for
```

5.3 BP Time Slots

During the BP time slots, the network can be modelled as a cluster graph and we adopt a Bethe cluster graph, cf. Section 2.2.5. The lower factors are composed of univariate potentials $\psi(X_i)$, while the upper region is composed of factors $\psi(X_i, X_j, R_{ji})$, see Fig. 5.2.

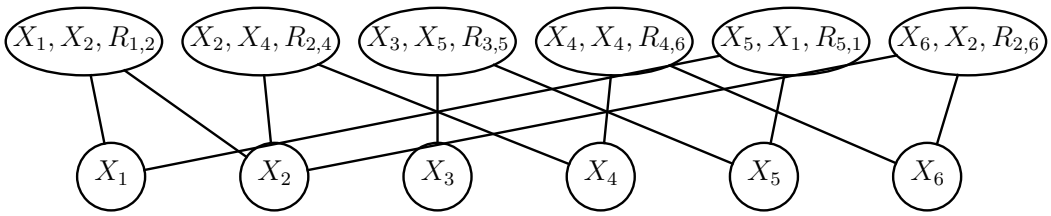

Figure 5.2: The cluster graph. Lower row factors denote the node position beliefs. Upper row factors denote the ranging interactions between the nodes.

The lower factors are set to the initial beliefs for the given time slot (t), and the upper factors are set to the corresponding conditional pdfs (cpdfs):

$$\psi(X_i) = \Pr(X_i^{(t)}), \tag{5.8}$$

$$\psi(X_i, X_j, R_{ji} = r_{ji}^{(t)}) = \overset{(t)}{\Pr}(R_{ji} = r_{ji}^{(t)} | X_i, X_j). \tag{5.9}$$

Messages are then passed between nodes for multiple iterations until the node beliefs have converged. The message from node j to node i, at BP iteration $(s+1)$ is

calculated by

$$\mu_{j\to i}^{(s+1)}(X_i) = \int \psi(X_i, X_j, R_{ji} = r_{ji}^{(t)}) \frac{b_{j\to i}^{(s)}(X_j)}{\mu_{i\to j}^{(s)}(X_j)} \mathrm{d}X_j, \tag{5.10}$$

where intuitively, a message (5.10) is the belief that node j has about the location of node i and $r_{ji}^{(t)}$ is the observed value of the distance between the nodes, at time slot t.

Then the belief of node i is updated as

$$b_i^{(s+1)}(X_i) = \lambda\, \psi(X_i) \prod_{k\in\mathcal{N}_i} \mu_{k\to i}^{(s+1)}(X_i) + (1-\lambda) b_i^{(s)}(X_i), \tag{5.11}$$

where λ is a dampening factor used to facilitate convergence.

BP continues until convergence, or if s reaches a maximum number of iterations $I_{\max}$. Then the beliefs, representing approximations to the true marginals, for each node are found by (5.11), i.e., $\Pr(X_i^{(t+1)}) = b_i^{(s+1)}(X_i)$. Each node will need to perform a marginalization operation (5.10), and a product operation (5.11). The algorithm follows precisely the procedure and approximations described in chapter 4.

5.4 IMU Time Slots

During the IMU time slots, our is aim is to approximate the transition probability in (5.2). This is accomplished by using a non-parametric particle representation [53]. Assuming at time slot (t) the location pdf for node i is $\Pr(X_i^{(t)})$, we represent it by a set of L random samples, or particles $\mathscr{S}_i^{(t)} = \{s_i^{(t,l)}\}_{l=1}^{L}$, sampled from $\Pr(X_i^{(t)})$. The i-th sample is denoted as $s_i^{(t,l)} = (x_i^{(t,l)}, \pi_i^{(t,l)})$, where $x_i^{(t,l)}$ is the value of the node state and $\pi_i^{(t,l)} = \frac{1}{L}$ is the respective weight. Then the motion model is applied to each sample $s_i^{(t,l)}$ and we obtain a new sample

$$s_i^{(t+1,l)} = \left(x_i^{(t+1,l)}, \frac{1}{L} \right), \tag{5.12}$$

where

$$x_i^{(t+1,l)} \sim \Pr(\boldsymbol{X}^{(t+1)} | X_i^{(t)} = x_i^{(t,l)}, \boldsymbol{O}_i^{(t)} = \boldsymbol{o}_i^{(t)}), \tag{5.13}$$

and $\boldsymbol{o}_i^{(t)}$ are the IMU measurements at time (t). Finally, the parameter estimation presented in Algorithm 9, is used to obtain a multinomial parametric form of (5.2) from the samples.

5.4.1 PHIMTA

Our aim now is to derive an updated location particle given a location particle $x_i^{(t+1,l)}$ and the IMU observations $o_i^{(t)}$. We will derive the displacement, speed and attitude vectors, from the IMU sensors. We assume a typical MEMS sensor, consisting of an accelerometer, a magnetometer and a gyroscope. The measurements provided by the IMU and the magnetometer comprise the control input $\boldsymbol{o}^{(t)} = [\boldsymbol{a}^{(t)}, \boldsymbol{\omega}^{(t)}, \boldsymbol{\mu}_x^{(t)}]$, and we denote their respective noise vector as $\boldsymbol{w}^{(t)} = [\boldsymbol{n}_a^{(t)}, \boldsymbol{n}_\omega^{(t)}, \boldsymbol{n}_\mu^{(t)}]$. We assume that the input signal vectors from each sensor have length Y. Also, the noise is assumed to be white Gaussian noise and independent of previous states. A rotation matrix that maps the LCS of the sensors to the GCS of the node is required, as the sensor axes may not match the nodes. Then the accelerometer observations will be mapped to the nodes coordinate system and used to calculate the displacement and speed of the node.

To alleviate the noise a number of schemes will be used. First is the fact that pedestrian walking is cyclical and significantly consistent. Each stride can be split into two phases. The stance phase, i.e., when the foot or part of the foot is placed on the ground, and the swing phase, i.e., when the foot is mid-air. Both the velocity and the angular velocity can be reset to zero at each stance phase, thus reducing the drift error accumulation. As the gyroscopes cannot be used in the static phase, signals from the accelerometer and magnetometer have to be used to calculate the orientations of the sensor module. To overcome tilt errors, the algorithm presented in [83] is used. The stance phase can be easily detected using peak detection, taking into consideration of the existence of zero crossings, e.g., [80]. In this work, we use the SHOE algorithm [87].

Hence, the system iterates through the following steps:

- Stance phase
  - Reset angular velocity to zero
  - Reset velocity to zero
  - Use magnetometer and accelerometer data to calculate rotation matrix
- Swing phase
  - Use gyroscope data to calculate the rotation matrix
  - Calculate the velocity and displacement using accelerometer data

Our derivation follows closely the work in [83]. In subsequent sections, as everything involves internal calculations at each agent, the node subscript is dropped for simplicity.

5.4.2 Coordinate Systems and Transformation Matrix

The global cartesian coordinate system used is the north-east-down (NED) frame (x^n, y^e, z^d). Consequently, the rotation matrix derived by using direction cosine representations is given by (5.14)

$$\boldsymbol{R}^{(t)} = \begin{pmatrix} \cos(p)\cos(a) & -\cos(r)\sin(a)+\sin(r)\sin(p)\cos(a) & \sin(r)\sin(a)+\cos(r)\sin(p)\cos(a) \\ \cos(r)\sin(a) & \cos(r)\cos(a)+\sin(r)\sin(p)\sin(a) & -\sin(r)\cos(a)+\cos(r)\sin(p)\sin(a) \\ -\sin(p) & \sin(r)\cos(p) & \cos(r)\cos(p) \end{pmatrix}, \tag{5.14}$$

where p, r, a, correspond to the pitch, roll, and attitude, respectively, and the time slot superscript has been dropped for simplicity.

5.4.3 Swing Phase

During the swing phase, the orientation of a moving object is tracked by integrating the angular velocity vector $\boldsymbol{\omega}^{(t)} = [\omega_x^{(t)} - n_{\omega x}^{(t)}, \omega_y^{(t)} - n_{\omega y}^{(t)}, \omega_z^{(t)} - n_{\omega z}^{(t)}]$, obtained from the gyroscope after we correct for noise. Let the sampling period δt be short and

$\delta\boldsymbol{\Psi} = [\delta a, \delta p, \delta r]$ be the rotated angle vector of the sensors. Then $\delta\boldsymbol{\Psi} = \boldsymbol{\omega}\delta t$. Assuming a small δt the rotation matrix for a period can be approximated by

$$\boldsymbol{C}^{(t)} = \begin{pmatrix} 1 & -\delta a & \delta p \\ \delta a & 1 & -\delta r \\ -\delta p & \delta r & 1 \end{pmatrix} = \boldsymbol{I} + \boldsymbol{\Omega}^{(t)}\delta t, \tag{5.15}$$

where

$$\boldsymbol{\Omega}^{(t)} = \begin{pmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{pmatrix}. \tag{5.16}$$

This allows us to relate the rotation matrix $\boldsymbol{R}^{(t)}$ with the rotation matrix of the next sampling period $\boldsymbol{R}^{(t+\delta t)}$. Then

$$\boldsymbol{R}^{(t+\delta t)} = \boldsymbol{R}^{(t)} \times \boldsymbol{C}^{(t)}, \tag{5.17}$$

where we have overloaded the superscript to mean the current sampling period besides the time slot. This gives

$$\frac{d\boldsymbol{R}^{(t)}}{dt} = \boldsymbol{R}^{(t)} \times \boldsymbol{\Omega}, \tag{5.18}$$

$$\boldsymbol{R}^{(t+\delta t)} = \boldsymbol{R}^{(t)} \times \exp\left(\int_t^{t+\delta t} \boldsymbol{\Omega} dt\right). \tag{5.19}$$

The rotation matrix update equation is obtained as each new angular velocity samples comes by

$$\boldsymbol{R}^{(t+\delta t)} = \boldsymbol{R}^{(t)}\left(\boldsymbol{I} + \frac{\sin(\|\boldsymbol{\omega}\|\delta t)}{\|\boldsymbol{\omega}\|}\boldsymbol{\Omega} + \frac{1-\cos(\|\boldsymbol{\omega}\|\delta t)}{\|\boldsymbol{\omega}\|^2}\boldsymbol{\Omega}^2\right). \tag{5.20}$$

With the rotation matrix updated, at each sample, the accelerometer data can easily

be mapped from LCS to GCS by

$$\boldsymbol{a}^{(G,t)} = \boldsymbol{R}^{(t)} \cdot \boldsymbol{a}^{(t)}, \tag{5.21}$$

where G specifies that the vector is the GCS. Finally, the updated velocity vector is given by

$$\begin{bmatrix} v_n^{(t+1)} \\ v_e^{(t+1)} \\ v_d^{(t+1)} \end{bmatrix} = \begin{bmatrix} v_n^{(t)} \\ v_e^{(t)} \\ v_d^{(t)} \end{bmatrix} + \begin{bmatrix} a_x^{(G,t)} - n_{ax}^{(t)} \\ a_y^{(G,t)} - n_{ay}^{(t)} \\ a_z^{(G,t)} - n_{az}^{(t)} - g \end{bmatrix} \delta t \tag{5.22}$$

and the corresponding displacement vector

$$\begin{bmatrix} d_x^{(t+1)} \\ d_y^{(t+1)} \\ d_z^{(t+1)} \end{bmatrix} = \begin{bmatrix} v_n^{(t)} \delta t \\ v_e^{(t)} \delta t \\ -v_d^{(t)} \delta t \end{bmatrix} \tag{5.23}$$

is used in Algorithm 7, to obtain the particle $x_i^{(t+1,l)}$.

5.4.4 Static Phase

During the static phase data from the accelerometer and the magnetometer are used to derive the pitch, roll, and attitude required for the rotation matrix, using (5.14). To compensate the tilt errors the following algorithm is used as presented in [83].

First, a linear-phase Finite Impulse Response (FIR) Low Pass Filter (LPF) is used to filter the accelerometer signal. The LPF is designed with a cut-off frequency of less than 1Hz, as a typical human stride takes $\approx$ 1s. The filtered acceleration $g^{(L)}$is then normalized and redefined as a gravity vector in LCS. The normalized GCS gravity vector is then given by

$$\boldsymbol{g}^{(G)} = \boldsymbol{R} \cdot \boldsymbol{g}^{(L)}, \tag{5.24}$$

where $g^{(G)} = [0,0,1]^T$.

Solving the above equation for roll and pitch gives

$$p^{(t)} = \text{atan2}\left(g_x^{(t)}, \sqrt{(g_y^{(t)})^2 + (g_z^{(t)})^2}\right), \tag{5.25a}$$

$$r^{(t)} = \text{atan2}\left(g_y^{(t)}\text{sign}(\cos(p^{(t)})), g_z^{(t)}\text{sign}(\cos(p^{(t)}))\right). \tag{5.25b}$$

After both pitch and roll have been found from the acceleration data, the attitude can be calculated from the magnetic field data. Let $\boldsymbol{\mu}^{(L,t)} = [\mu_x^{(t)}, \mu_y^{(t)}, \mu_z^{(t)}]$ be the LCS magnetometer readings. Then the compensated magnetic field can be calculated as

$$\begin{aligned} h_x^{(t)} &= \mu_x^{(t)}\cos(p^{(t)}) + \mu_y^{(t)}\sin(p^{(t)})\sin(r^{(t)}) \\ &\quad + \mu_z^{(t)}\sin(p^{(t)})\cos(r^{(t)}) \end{aligned} \tag{5.25c}$$

$$h_y^{(t)} = \mu_y^{(t)}\cos(r^{(t)}) - \mu_z^{(t)}\sin(r^{(t)}) \tag{5.25d}$$

$$a^{(t)} = \text{atan2}(-h_y^{(t)}, h_x^{(t)}) - D, \tag{5.25e}$$

where D is the magnetic declination, or the difference between the magnetic north and the true north, caused by the tilt of the earth magnetic field generator relative to the earth spin axis.

The algorithm is summarized as Algorithm 11.

5.5 Complexity

For the BP time slots, the complexity is due to the message passing algorithm used. In our case, as Grid-BP is a parametric form message passing algorithm, the computational cost is $\mathcal{O}(\mathcal{N}_i L)$, cf. chapter 4. This makes the algorithm an order of magnitude faster than non-parametric BP algorithms, e.g., SPAWN [9]. We also compare Grid-BP with the SnG algorithm [84], which has a complexity of $\mathcal{O}(\bar{\mathcal{N}_i} L)$, where $\bar{\mathcal{N}_i}$ symbolizes the average pseudo-anchors of node i. The number of particles used in both algorithms is approximately $L = 100$, while the number of average pseudo-anchors will be less or equal to the average number of neighbours. As such, the algorithms tend to have similar complexity with SnG being slightly faster. Even

Algorithm 11 PHIMTA

Sample $\{\boldsymbol{\pi}_i^{(t,l)}, x_i^{(t,l)}\}_{l=1}^{L} \sim \Pr(X_i^{(t)})$
Detect Stride Phase using SHOE Algorithm
if Stride Phase is *Stance* **then**
    Set $\boldsymbol{\omega}^{(t)} = 0$
    Set $\boldsymbol{v}^{(t)} = 0$
    Extract g from a using LPF
    Calculate p, r, a using equations (5.25)
    Calculate Rotation Matrix $\boldsymbol{R}^{(t)}$ using (5.14)
else
    sample $\{\boldsymbol{n}_{\omega}^{(l,t)}\}_{l=1}^{L} \sim \mathcal{N}(\boldsymbol{n}_{\omega})$
    For each sample calculate $\boldsymbol{R}^{(t)}$ using (5.20)
end if
Sample $\{\boldsymbol{n}_{a}^{(l,t)}\}_{l=1}^{L} \sim \mathcal{N}(\boldsymbol{n}_{a})$
Sample $\{\boldsymbol{n}_{\mu}^{(l,t)}\}_{l=1}^{L} \sim \mathcal{N}(\boldsymbol{n}_{\mu})$
for each sample calculate $\{a^{(l,G,t)}\}_{l=1}^{L}$ using (5.21)
Calculate $\{\boldsymbol{x}^{(l,t+1)}\}_{l=1}^{L}$ using (5.23) and Algorithm 7
Convert to parametric form using Algorithm 9
Update belief $\Pr(X_i^{(t+1)})$

though the two algorithms seem to have the same computational cost, a step by step comparison in Table 5.1 clearly shows that Grid-BP is faster, as it has fewer steps and there is no need to optimize the objective function at every iteration. For the comparison, we assume that both algorithms approximately use the same number of particles, and $|\mathrm{ID}|$ is the cardinality of relevant IDs in Grid-BP, while U and $U_{\max}$ is the number of candidate points and number of highest likelihood candidate points respectively. Finally I_{IWLS} is the number of iterations IWLS can run.

Table 5.1: Complexity of Grid-BP vs SnG

Grid-BP		SnG [84]	
Step	Complexity	Step	Complexity
sample incoming message	L	sample incoming message	L
count sample IDS	$\|\mathrm{ID}\|$	count-sort likelihoods	$\bar{\mathcal{N}_i}L$
multiply multinomial pdfs	$N_i\|\mathrm{ID}\|$	sort candidate points	$U\log(U)$
filter IDs	$\|\mathrm{ID}\|\log(\|\mathrm{ID}\|)$	get centroid	$U_{\max}\log(U_{\max})$
—	—-	IWLS	$\bar{\mathcal{N}_i}I_{\mathrm{IWLS}}$

For the IMU time slots, the complexity is due to PHIMTA. All the steps are proportional to either the number of signals Y obtained from MEMS, or the number of particles used in the calculations L. Each step is given with the corresponding cost in Table 5.2. Hence the complexity is $\mathcal{O}(YL)$. We compare PHIMTA with the PDR algorithm in [88]. Even if the complexity scales in the same way, PDR has fewer computations per iteration than the hybrid algorithm. Essentially it is a compromise between computational cost, and accuracy as will be seen in the sequel. By using Grid-BP though, the computational increase from PHIMTA can be easily compensated.

Table 5.2: Complexity of PHIMTA

Step	Complexity
sample $\Pr(X_i^{(t)})$	L
SHOE Algorithm	Y
Stride phase	YL
Quasi-static phase	YL
update position	L
convert to parametric form	L

Table 5.3: Complexity costs of Grid-BP/PHIMTA vs SnG/PDR

Algorithm	Complexity
Grid-BP	$\mathscr{O}(\mathscr{N}_i L_{\text{Grid-BP}})$
PHIMTA	$\mathscr{O}(Y L_{\text{PHIMTA}})$
SnG	$\mathscr{O}(\mathscr{N}_i L_{\text{SNG}})$
PDR	$\mathscr{O}(Y L_{\text{PDR}})$

Note that $L_{\text{Grid-BP}} = L_{\text{PHIMTA}} = L_{\text{SNG}} = L_{\text{PDR}} \simeq 100$.

5.6 Simulation Results

Our proposed algorithm was evaluated using Monte Carlo simulations. We considered a 2D grid 20m $\times$ 20m with 4 anchors at the corners of the grid. Ten nodes are randomly placed INSide the grid and are trying to localize. We first consider a static scenario in which the RMS error localisation error is compared between Grid-BP and SnG and NBP [57], as an implementation of the SPAWN framework.

In Fig. 5.3, we illustrate the average RMS error error for various communication ranges. We assume that the number of particles used is 300 and $K_e = 0.001$. As expected NBP outperforms both SnG and Grid-BP but at a greater computational cost. While Grid-BP provides similar RMS error to SnG for lower communication range, it is interesting to note that both NBP and Grid-BP take advantage of the availability of more neighbours while SnG seems to keep a constant RMS error error. We should also mention that SnG would collapse if the average number of neighbours is too low. Finally, for the SnG simulations, we initially ran cooperative least-square (LS), cf. [9], as it requires initial estimates to run, which is not required for NBP and Grid-BP.

Now, we consider a mobility scenario where the 10 nodes move randomly for a period of 180s. Simulation parameters are the same as before with a communication range of 12m. We used the test data in [85, 86], as a pool of possible movements that a node can follow with the respective MEMS measurements. Each node decides by a stationary probability $\text{Pr}_i(s)$ if it will wait or not and for how many seconds. If it will wait, then it will be used in the cooperative localisation algorithm. Alternatively, it will pick randomly a movement from the ones provided in the test data and

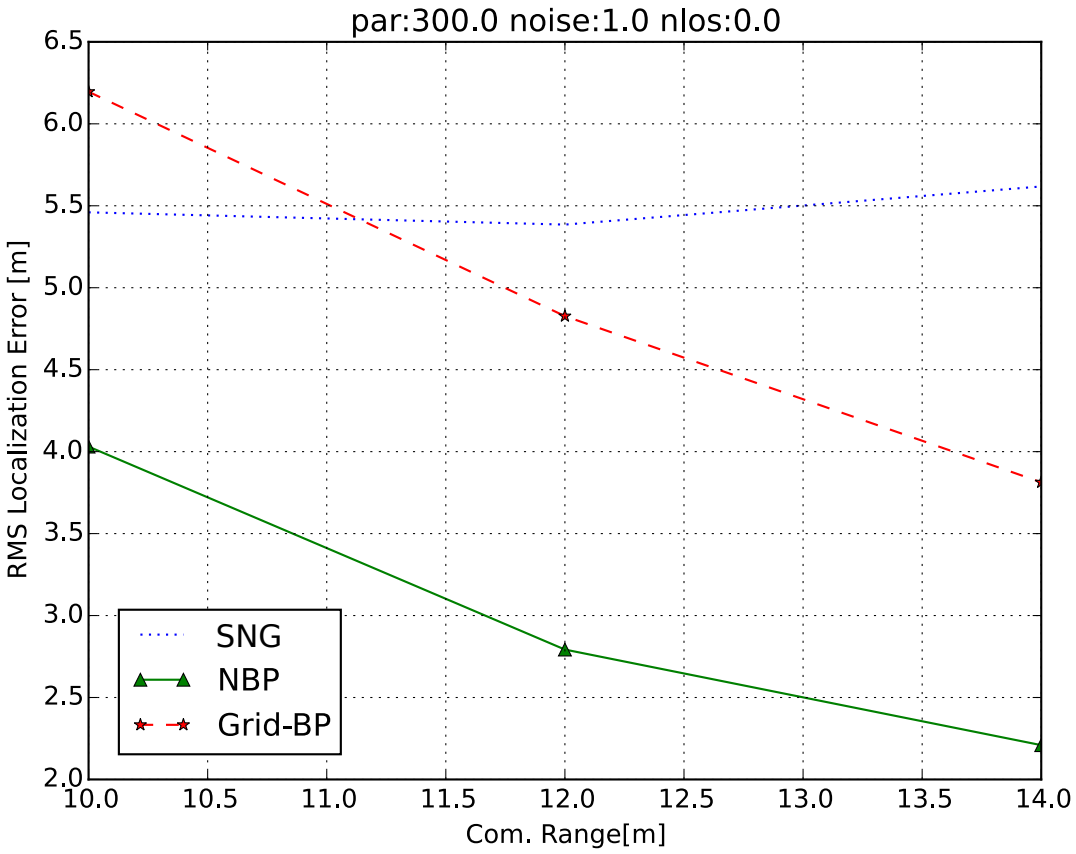

Figure 5.3: The average RMS error error versus the communication range.

move accordingly until the movement time elapses and the procedure repeats until the simulations finish.

The cooperative localisation steps occur, to obtain starting locations and afterwards every 10s. In Fig. 5.4, we present the average RMS error tracking error results, which is the average RMS error localisation error per second for the network. Obviously, as the stationary probability increases, the nodes move less and consequently are more readily available in the cooperative localisation steps, improving their RMS error tracking error. At the boundary scenario, we have $\Pr_i(s) = 0$ which means that all nodes are constantly moving and hence besides the initial cooperative localisation step, nodes will only use MEMS information algorithms. As we can see, the superiority of PHIMTA over PDR is evident in all mobility scenarios. Secondly, SnG slightly outperforms Grid-BP. This is due to the ability of SnG to

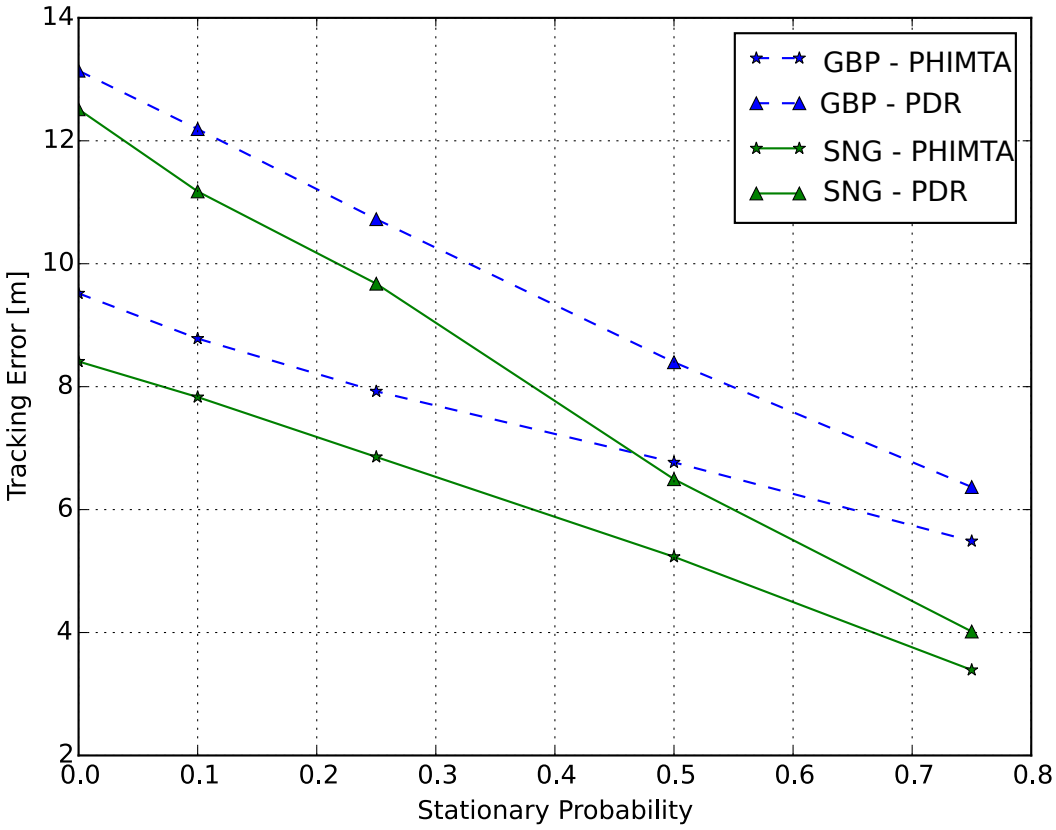

Figure 5.4: The average RMS error tracking error with $K_e = 0.01$.

filter out nodes that mistakenly believe they are stationary while in reality their are moving. Despite that, the difference is small, given the drastic decrease in computational cost. Finally it is clear that using cooperative localisation is a great addition to MEMS localisation.

5.7 Conclusions

In this chapter, we have extended our cooperative localisation algorithm, Grid-BP, with a hybrid MEMS tracking system. We have designed a novel hybrid INS/PDR mobility tracking algorithm and also designed a novel particle filter that combines probabilistically the former with our cooperative localization algorithm. The end result is a powerful and promising mobility tracking algorithm with very low omputational requirements compared to the literature. PHIMTA allows for a node to

be mobile for long periods of time and use cooperative localization to verify its position whenever possible. This can extend a cooperative localization network over a larger area and make it much more robust as the two systems compliment each other. As such, PHIMTA, combines both the self-localized branch and the cooperative localization branch in the position location tree, cf. Fig. 1.2. As INS sensors become increasingly cheaper and more accurate, we believe that hybrid systems that use cooperative localization with INS sensors like PHIMTA will become the norm in GPS-denied environments mobile tracking.

Chapter 6

Stochastic Residual Belief Propagation

6.1 Introduction

In the previous chapters, the focus had been on improving the forms of the messages in distributed BP in order to achieve better accuracy and lower cost. In this chapter we take a step back and investigate a quite overlooked part of the use of message passing by wireless distributed sensors. Specifically we investigate the *message scheduling*.

In order to talk about message scheduling, as discussed in Section 2.6 a different perspective to message passing needs to be approached. In particular, Yedidia *et al.*, cf. [89], demonstrated that BP can be interpreted as performing a constrained minimization of the so-called Bethe free energy. Convergence conditions were proposed in [61, 59, 90]. Moreover, algorithms that ameliorate the effects of cycles by weighting messages have been proposed in [91, 60]. Remarkably, the importance of message scheduling in BP has also been recognized, and R-BP has hence been proposed as an algorithm implementing a greedy informed schedule for message passing, cf. [92]. This gave rise to a number of variants of BP in Low-Density Parity Check (LDPC) decoding that provide more elaborate informed schedules for R-BP, e.g., [93].

Distributed wireless networks, such as in the application of *cooperative spec-*

trum sensing, however, pose new challenges for BP as there is practically no central entity to manage global information, hence making algorithms like Tree Reweighing BP (TR-BP) [91] and R-BP *unusable* in the distributed case. In [94, 95], Wymeersch *et al.* devised a distributed variant of TR-BP, called Uniformly Re-Weighted BP (URW-BP), but did not study the use of informed scheduling for distributed networks.

6.1.1 Our Contributions

In this chapter, we propose a novel R-BP algorithm for distributed wireless networks that employs a stochastic message scheduler based on the residuals at each node. This will result in faster convergence, less overhead, and improved results, but also can be integrated with enhancing algorithms such as URW-BP [94, 95] for further performance increase.

Our contributions can be summarized as follows:

- We devise a SR-BP as a practical alternative to perform R-BP for distributed inference.
- We prove that when BP converges, given similar conditions to R-BP, SR-BP will also converge.
- We propose a probability density function parametrized by the residuals to use as local "soft" decision for message propagation.
- Simulation results for the cooperative spectrum sensing application show that SR-BP greatly outperforms BP in distributed inference and compliments UTR-BP.

6.1.2 Organization

The rest of the chapter is organized as follows. In Section 6.2 the inference using BP is quickly revisited and a convergence analysis of BP is presented. In Section 6.3 SR-BP is presented, and it's convergence properties are discussed and proven. In Section 6.4 we present results from Monte Carlo simulations for both the Ising

model and a more realistic cooperative spectrum sharing scenario. Finally, in Section 6.5 the chapter is concluded and some ideas for future work are discussed.

6.2 Inference using BP

Let $\boldsymbol{X} = [X_1 \cdots X_n]$ be a finite set of N random variables and let $\boldsymbol{x}$ denote an assignment to $\boldsymbol{X}$. Also let $\boldsymbol{x}_c$ denote an assignment to a subset of variables $\boldsymbol{X}_c \subseteq \boldsymbol{X}$. We can represent a joint distribution over $\boldsymbol{X}$ as a set of factors Φ_c for $c \in \mathscr{C}$, where each c is associated with the corresponding variables $\boldsymbol{X}_c$ and Φ_c is a function from the set of possible assignments $\boldsymbol{X}_c$ to $\mathbb{R}^+$. The joint distribution over $\boldsymbol{X}$ is then defined as

$$p(\boldsymbol{X}) = \frac{1}{Z} \prod_{c \in \mathscr{C}} \Phi_c(\boldsymbol{X}_c), \tag{6.1}$$

where $Z \triangleq \sum_{\boldsymbol{X}} \prod_{c \in \mathscr{C}} \Phi_c(\boldsymbol{X}_c)$ is a normalization constant and is commonly referred to as the partition function.

Let $\mathscr{G} = (\mathscr{V}, \mathscr{E})$ represent a cluster graph with vertices $\mathscr{V}$ and edges $\mathscr{E}$. We can place at each node a cluster Φ_s associated with a subset of variables $\boldsymbol{X}_s$ of the undirected graph. Nodes are connected with edges $\mathscr{E}$. A message between two clusters s and t is a factor over their common variables $\boldsymbol{X}_{s,t} = \boldsymbol{X}_s \bigcap \boldsymbol{X}_t$. In sum-product BP, the message from one cluster to another is defined as eq. 2.16

$$\mu_{s \to t}(\boldsymbol{X}_{s,t}) \triangleq \sum_{\boldsymbol{X}_t - \boldsymbol{X}_{s,t}} \Phi_s(\boldsymbol{X}_s) \prod_{r \in \mathscr{N}_s \backslash t} \mu_{r \to s}(\boldsymbol{X}_{r,s}), \tag{6.2}$$

where $\mathscr{N}_s \backslash t$ denotes the set of neighbouring clusters for cluster s, excluding cluster t. In principle, messages will be initiated and passed through the cluster graph until convergence. This occurs when both sides of the update equations for each cluster in the graph are equal, or in other words a fixed point of the Bethe energy function has been found.

A different viewpoint was presented in [92]. Each message can be viewed as residing in some message space $(\mathscr{R} \subset \mathbb{R}^+)^d$, where d is the dimension of the

messages.[1] Hence, the set of messages $\mathscr{M}$, in a cluster graph, is a subset of $\mathscr{R}^{|\mathscr{M}|}$, where $|\mathscr{M}| = 2|\mathscr{E}|$. Let m denote the index of individual messages, $\boldsymbol{v}_m$ denote the mth message and $v \in \mathscr{R}^{|M|}$ denote a joint assignment of messages. The update equation (6.2) can therefore be seen as a mapping function $f_m : \mathscr{R}^{|\mathscr{M}|} \to \mathscr{R}$ that defines the mth message as a function of a subset of the messages in $\mathscr{R}^{|\mathscr{M}|}$. Then we can define the iterative method

$$\boldsymbol{v}_m^{(t+1)} = f_m(\boldsymbol{v}^{(t)}). \tag{6.3}$$

Assuming convergence, we have the fixed point

$$f_m(\boldsymbol{v}^*) = \boldsymbol{v}_m^*. \tag{6.4}$$

Finally, the *global* update functions can be defined for both the synchronous and asynchronous cases, respectively, as

$$F^s(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{|\mathscr{M}|}) = \left(f_1(\boldsymbol{v}), \ldots, f_{|\mathscr{M}|}(\boldsymbol{v})\right), \tag{6.5}$$

$$F_m^a(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{|\mathscr{M}|}) = \left(\boldsymbol{v}_1, \ldots, f_m(\boldsymbol{v}), \ldots, \boldsymbol{v}_{|\mathscr{M}|}\right). \tag{6.6}$$

In the asynchronous case, we assume that there is a set of times $T = \{0, 1, 2, \ldots\}$ at which one or more components $\boldsymbol{v}_m$ are updated. Also let T^m be the set of times $\boldsymbol{v}_m$ is updated. Then for the asynchronous case we adopt Assumption 3.1 of [92].

Assumption 6.2.1. *For every message m, there is a finite time T_m so that for any time $t \geq 0$, the update $\boldsymbol{v} := f_m^a(_v)$ is executed at least once in the time interval $[t, t + T_m]$.*

Essentially this means that all messages will be updated infinitely until convergence.

[1] For convenience all messages are assumed to have the same dimension d.

6.2.1 Convergence Analysis

One of the main tools in convergence analysis is the contraction. Assuming a finite dimensional vector space $\boldsymbol{V}$ that has a vector norm $\|\cdot\|$, we define a mapping $F : \boldsymbol{V} \to \boldsymbol{V}$ to be a $\|\cdot\|$-contraction if

$$\|F(\boldsymbol{v} - F(\boldsymbol{w})\| \leq a\|\boldsymbol{v} - \boldsymbol{w}\|, \tag{6.7}$$

for some $0 \leq a < 1$, for all $\boldsymbol{v}, w \in V$. Respectively if $F(\cdot)$ is a $\|\cdot\|$-contraction then a unique fixed point $\boldsymbol{v}^*$ is guaranteed to exist, and applying $F(\cdot)$ iteratively we have that

$$\boldsymbol{v}^{(t+1)} = F(\boldsymbol{v}^{(t)}) \tag{6.8}$$

is guaranteed to converge to $\boldsymbol{v}^*$, for all possible initial vectors $\boldsymbol{v}^{(0)} \in \boldsymbol{V}$. In the message space $\mathscr{R}$ we define a message norm $\|\boldsymbol{v}_m - \boldsymbol{w}_m\|_m$ that measures distances between individual messages and a global norm that measures distances between points in $\mathscr{R}^{|M|}$. Following the analysis in [92], we also use the max norm $\|\cdot\|_\infty$ for the global norm defined as

$$\|\boldsymbol{v} - \boldsymbol{w}\|_\infty = \max_{m \in |\mathscr{M}|} \|\boldsymbol{v}_m - \boldsymbol{w}_m\|_m. \tag{6.9}$$

Assuming convergence is guaranteed for the synchronous BP, i.e. F^s is a $\|\cdot\|_\infty$-contraction, Elidan et al. showed that the asynchronous BP, will also converge if there is a propagation schedule that guarantees that every message will be updated until convergence, i.e. Assumption 6.2.1. Also they suggested the intuition that in an asynchronous message passing scheme messages that "carry" more information should be propagated first as they will help the algorithm converge faster, and defined the residual of a message as

$$r_m(\boldsymbol{v}) = \|f_m(\boldsymbol{v}) - \boldsymbol{v}_m\|_m. \tag{6.10}$$

This led to the proposal of the the R-BP message passing scheme, where at

each iteration (t) of BP, all residuals are calculated and the message with the largest residual is chosen to be propagated, namely

$$m^{(t+1)} = \arg\max_{m} r_m(\boldsymbol{v}^{(t)}). \tag{6.11}$$

Empirically, Elidan *et al.* also showed that even in complicated real life application PGMs, R-BP will converge more often and with less messages than both synchronous and asynchronous BP propagation schemes, which led to a variety of R-BP variants especially in the LDPC decoding. Unfortunately R-BP and its variants, require a centralized entity to compare all the residuals, find the largest, and propagate the corresponding message, making it unsuitable for BP in distributed networks. In the sequel we present stochastic R-BP (SR-BP), which overcomes the requirement of a centralized entity by using a stochastic propagation scheme, suitable for distributed networks.

6.3 SR-BP

Firstly we reformulate BP so that each node propagates its belief at each iteration instead of a distinct message for each neighbour. The belief, that is an approximation to the true marginal [91], at each timeslot is calculated as

$$b_s^{(t)}(\boldsymbol{X}_s) = \Phi_s(\boldsymbol{X}_s) \prod_{r \in \mathcal{N}_s} \mu_{r\to s}^{(t)}(\boldsymbol{X}_s) \tag{6.12}$$

and the messages calculated from the incoming beliefs $\mu_{r\to s}^{(t)}(\boldsymbol{X}_{r,s})$ are calculated as

$$\begin{aligned} \mu_{r\to s}^{(t)}(\boldsymbol{X}_s) &\propto \sum_{\boldsymbol{X}_r} \Phi_{r,s}(\boldsymbol{X}_r, \boldsymbol{X}_s) \prod_{t \in \mathcal{N}_r \neg s} \mu_{t\to r}^{(t-1)}(\boldsymbol{X}_r) \\ &\propto \sum_{\boldsymbol{X}_r} \Phi_{r,s}(\boldsymbol{X}_r, \boldsymbol{X}_s) \frac{b_r^{(t-1)}(\boldsymbol{X}_r)}{\mu_{s\to r}^{(t-1)}(\boldsymbol{X}_r)}. \end{aligned} \tag{6.13}$$

Consequently, $|\mathscr{M}|$ is equal to the number of nodes. In the distributed case there is no centralized entity to compare all residuals and decide on a message schedule. As a result, each node will have to decide on itself if its message is "important"

enough to transmit. Following the intuition behind residuals, we propose a stochastic message passing schedule, where each node transmits its belief at timeslot (t) with probability $\Pr(r_m^{(t)})$. The probability is calculated as

$$\Pr(r_m^{(t)}) = S(r_m^{(t)}) = \frac{1}{1+\exp(-r_m^{(t)})}, \tag{6.14}$$

where $S(\cdot)$ is the sigmoid function. We now analyze the convergence of SR-BP. To do so, we use Theorem 3.2 of [92] propagation that states

Theorem 6.3.1. *If F^s is a max-norm contraction, then any asynchronous propagation schedule that satisfies Assumption 6.2.1 will converge to a unique fixed point.*

Assuming that a PGM already satisfies the max-norm contraction condition for the synchronous BP case, we only need to prove that SR-BP, satisfies Assumption 6.2.1.

Proof. By definition the residual $r_m^{(t)} \geq 0$, $\forall m \in \mathscr{M}$. Then by construction we have

$$r_m^{(t)} \geq 0 \Rightarrow \frac{1}{1+\exp(-r_m^{(t)})} \geq 0.5. \tag{6.15}$$

Consequently, there will always be a positive probability that message m will be transmitted, hence there will be a T_m, so that for any time $t \geq 0$, message m will be updated and Assumption 6.2.1 is satisfied. Therefore, SR-BP will converge to a fixed point. □

It is hard to analyze the convergence rate of SR-BP but intuitively it should be somewhere between the synchronous case, and the centralized residual case. In the sequel we will provide experimental results that showcase this intuition.

6.4 Experimental Results

Monte carlo simulations were carried out to analyze the convergence rate, message overhead and quality of the marginals of SR-BP. Comparisons were made with synchronous BP, i.e. BP, asynchronous BP (ABP). Also we compare it with centralized R-BP that of course couldn't be used in practice in a distributed scenario. Finally

we also use the aforementioned message scheduling schemes with URW-BP [94]. It should be noted that all algorithms use respectively the same codebase and only the message scheduling differs.

6.4.1 Ising model

Firstly, random grids parameterized by the Ising model [54], are considered. These allow for a systematic way to analyze an algorithm, as both the difficulty and the scale of the inference task can be readily controlled. Random grids with 11 nodes were created, with univariate potentials $\Phi_i(X_i)$ sampled from $\mathscr{U}[0,1]$ for each variable, and pairwise potential given by

$$\Phi_{i,j}(X_i,X_j) = \exp\left(\lambda C * (2\mathbf{1}(X_i = X_j) - 1)\right), \tag{6.16}$$

where λ is sampled uniformly from $[-0.5, 0.5]$ allowing randomly some nodes to agree and some to disagree with each other. In addition, $\mathbf{1}(X_i = X_j)$ is the indicator function. Finally C is an agreement factor, where higher values impose stronger constraints on the "negotiations" between nodes, making convergence harder. 200 simulations were run for a network with 11 nodes, where $C = 10$ and the algorithms were allowed to run until convergence or 300 iterations had passed. The results are summarised in Table 6.1.

There are a few interesting points to be made here. Firstly, as expected R-BP achieves convergence every time in the given scenario, requiring 73% less iterations on average and almost 96% less messages. In addition, SR-BP is pretty close to R-BP achieving only slightly worse convergence rate, and requiring 64% less iterations than BP and around 95% less messages, with the corresponding decrease in computational cost and overhead of course. The other thing of interest is the large increase in convergence rate when UTR-BP is used to compute the messages, as well as the decrease in required iterations and messages to reach it, for BP and ABP. UTR-R-BP seems to work slightly better than normal R-BP, and UTR-SR-BP seems to converge a bit slower and worse than before. Nevertheless, it actually achieves a better approximation to the real marginals, and the number of messages

Table 6.1: Results for the Ising model.

	Conv. %	Avrg. Conv. Iterations	Avrg. Messages	KLD
BP	83%	78.5	8634	0.255
ABP	88%	63.2	6905	0.257
R-BP	100%	21.5	315	0.369
SR-BP	94.5%	28.4	393	0.365

required continues to be quite small.

Figures 6.1 and 6.2 show the cumulative percentage of convergence of the different algorithms as a function of iterations passed (essentially computational time given to the algorithm). Again it is worthy to note that in both cases, even though SR-BP converges less than centralized R-BP, when it will converge it converges much faster.

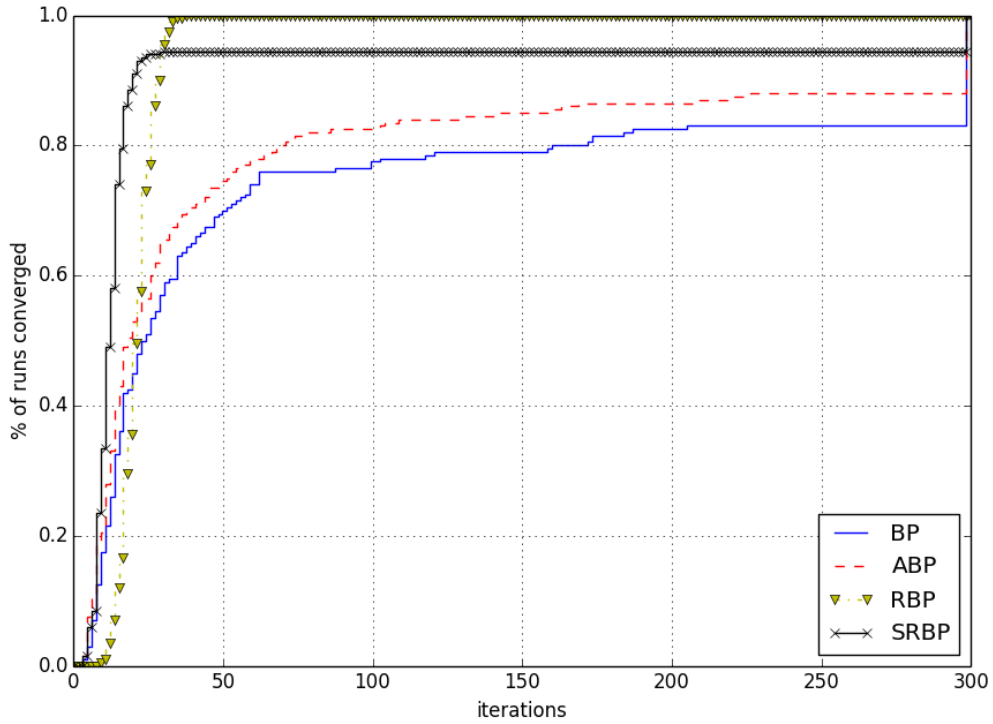

Figure 6.1: Cumulative Convergence percentage vs iterations for 11x11 node ising grid

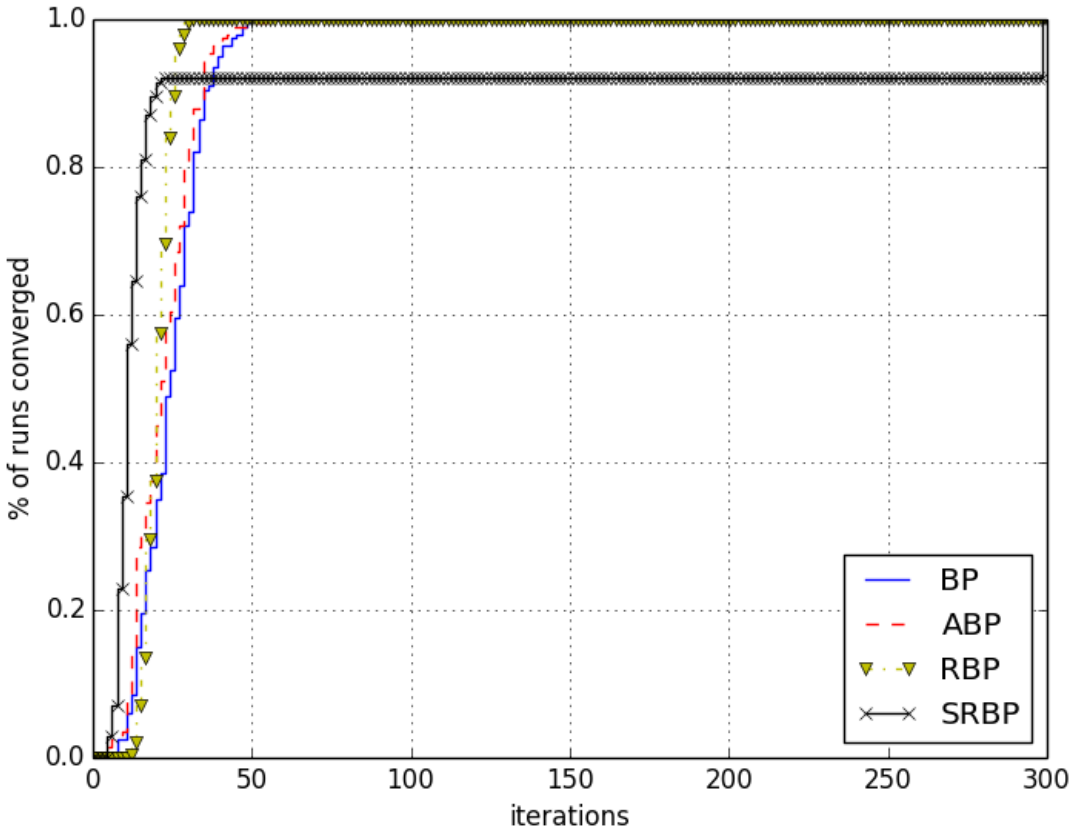

Figure 6.2: Cumulative Convergence percentage vs iterations for 11x11 node ising grid using UTR message passing

6.4.2 Cooperative Spectrum Sensing model

Next we study cooperative spectrum sensing, using the same model presented in [95], as a real life wireless distributed application. We assume that a primary user (PU) is transmitting and secondary users (SU) are collaboratively sensing the spectrum trying to decide if the channel is busy or not. SU that are close by, are more likely to sense the same channel state. Let $X_i \in \{0,1\}$ be binary variable representing the state of the channel close to SU i, and let $Y_i = y_i$ represent the observation made by SU i. Then we have univariate variables $\Phi_i(X_i) = \Pr(y_i \mid x_i)$ and pairwise potentials $\Phi_{i,j}(X_i,X_j) = \exp\left(\lambda_{i,j}\mathbf{1}(X_i = X_j)\right)$, where $\lambda_{i,j}$ is the correlation factor between nodes i and j and in this model is sampled uniformly between $[0.2,4]$. The observations Y_i are sampled from complex Gaussian distributions with noise variance $\sigma_n^2\mathbf{I}_s$ and signal variance $\sigma_s^2\mathbf{I}_s$ for $X_i = 0$ and $X_i = 1$ respectively. $\mathbf{I}_s$ is the

Table 6.2: Results for the cooperative spectrum sensing model.

	Conv. %	Avrg. Conv. Iterations	Avrg. Messages
BP	9.8%	256.1	2817.26
ABP	13.4%	259.1	2849.6
R-BP	19.5%	269.0	279.0
SR-BP	18.5%	267.9	804.7

$s \times s$ identity matrix, and s is the number of i.i.d signal received samples, used in the channel detection. Nodes are deployed randomly in a circular area with unit diameter, and $R = 0.7$ is set as the maximum communication range. 100 Monte Carlo simulations were run, for a 100 transmission timeslots each, with a maximum iteration number of 300. The results are summarized in Table 6.2. As it quite evident, convergence is much more difficult to achieve. This is due to the large number of interconnections cause by the high communication range, and the relatively big correlation factors $\lambda_{i,j}$ Still SR-BP manages to double the convergence percentage, using approximately 29% of the message propagations required by BP. In summary in a real life application SR-BP achieves better convergence, with a 71% decrease in required transmitted messages, while at the same time improving slightly the RoC curve of the spectrum sensing nodes.

It is hard to compute exact marginals for the spectrum sensing case. Instead we compute the roc curves for the various algorithms, as can be seen in Figures 6.3 and 6.4. As can be seen in both cases SR-BP achieves a better ROC curve than BP and ABP, achieving a curve that almost matches the one by centralized R-BP.

6.5 Conclusions

We have presented a novel distributed message scheduling algorithm for running inference algorithms in wireless networks, based on R-BP. We have proven that SR-BP message schedule will converge to a fixed point if a synchronous schedule would converge, and have showcased the superiority of the algorithm even in more general non-convergent cases, where it consistently manages to achieve higher convergence rates, better accuracy, and lower overhead and computational cost. It should be noted that this work is far more general and can be used in a huge number of ap-

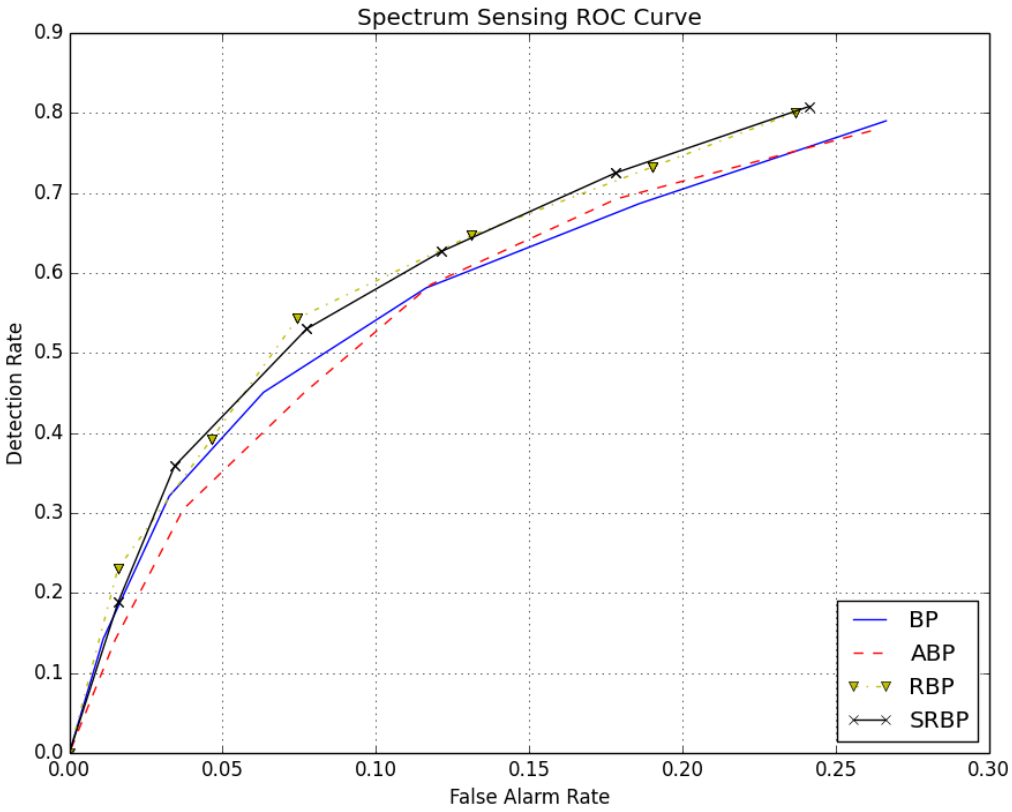

Figure 6.3: ROC Curve

plication where distributed iterative algorithms are used. Future work will involve experimentation with more complicated discrete and continuous pdf. Analysis of the contraction rate of SR-BP and possible alternative distributions that could be used to instigate message propagation. Concluding, the analysis of message schedules for distributed algorithms has been quite overlooked by the research community despite all the advantages a good message schedule clearly provides. We hope that this work will trigger an increase in interest for this interesting field, with so many applications.

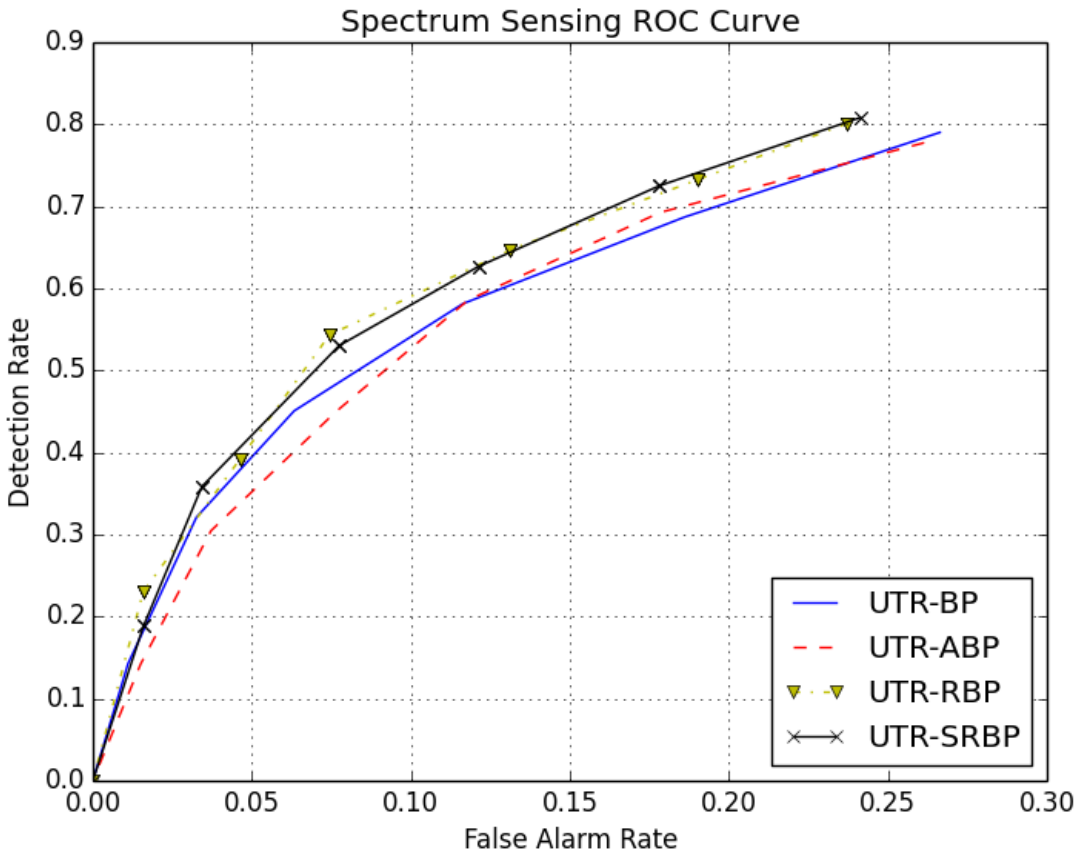

Figure 6.4: UTR ROC Curve.

Chapter 7

Conclusions & Future Work

7.1 Conclusions

Indoors localisation has been a very hot research topic in the years during which this Ph.D. was conducted. This can be explained as it presents a very interesting and challenging problem as in order to achieve good localisation it requires a combination of mathematics, machine learning, signal processing, engineering and code writing. In addition, it has immediate and diverse applications ranging from military to marketing.

The aim of this thesis was to solve a number of specific challenging issues in indoors localisation and specifically when using cooperative localisation and PGMs. Our aim was to develop novel algorithms for cooperative localisation that would take into account issues of capacity and complexity in cooperative localisation. We tackled the above limitations in a number of different ways.

In Chapter 3, we presented a novel algorithm called HEVA. HEVA is based on the SPAWN framework, and uses message passing of probabilistic beliefs. HEVA provided NLoS mitigation for ToA measurements. It also achieved a large decrease in complexity, as well as capacity and communication overhead by using an ellipsoid filter, and clustering to parametrize the messages. This allowed it to provide excellent results with low capacity and computational costs even in 3D scenarios with high noise and NLoS measurements.

In Chapter 4, we suggested the issue of coordinate system inherent in most

cooperative localisation algorithms. We showed that most algorithms localize using an LCS. Then a GCS is assumed to be known by all the nodes in order to get the real coordinates. We explained why this is not trivial and presented an elegant alternative. We developed a novel algorithm, namely Grid-BP, which uses a grid based reference system. Grid-BP, can be readily applied to existing grid-based GCS, like NATO's MGRS. This eliminates the use of any LCS and all the need to map the LCS to a GCS afterwards. In addition, Grid-BP allows for a parametric message passing which is extremely fast and efficient computationally.

In Chapter 5, we extended the use of Grid-BP to mobile users. We created a novel probabilistic algorithm, namely PHIMTA, which combines INS and PDR localisation based on MEMS sensors. We showed that the combination of Grid-BP and PHIMTA, allows for accurate and efficient localisation. We conducted experiments using measurement campaign data provided by [85, 86], and demonstrated the aforementioned.

Finally, in Chapter 6, we presented a distributed stochastic message scheduler for message passing algorithms, namely SR-BP. We showed that we can achieve faster convergence with less transmitted messages, by allowing the benefits provided by R-BP, to be used in a purely distributed environment. We proved that SR-BP will converge whenever the centralized R-BP would converge and used a cooperative spectrum sensing scenario, to showcase the improvement in overhead and convergence rate of SR-BP.

The above algorithms besides solving the described challenges in localisation, also compliment each other and can be used together to provide a low complexity and overhead cooperative localisation with high accuracy, in many "difficult" scenarios.

7.2 Future Work

Indoors localisation has been extensively researched in the last few years. It would be safe to say that in the very near future it will be used by end users in everyday life products and services. Nevertheless not all issues have been resolved yet as it

provides a very interesting problem that encompasses digital signal processing, machine learning and wireless communications. We believe that further investigation in the following areas is of great importance, as further work on the subject needs to take into account stricter limitations on CPU and power consumption of the mobile devices.

1. Further investigation in NLoS mitigation for distributed cooperative localisation while under power constraints in mobile devices is necessary, in different schemes, e.g. TDoA, AoA, etc.

2. Further investigation in the integration of localisation systems, i.e. cooperative localisation, with fingerprinting and INS systems, again while under power constraints.

3. Further reduction of unnecessary communications between devices, and development of pdfs for SR-BP with more information, e.g. taking into account mobility INS information, etc.

Bibliography

[1] Reza Zekavat and R Michael Buehrer. *Handbook of Position Location: Theory, Practice and Advances (IEEE Series on Digital & Mobile Communication)*. Wiley-IEEE Press, 2011.

[2] Elliott D Kaplan and Christopher J Hegarty. *Understanding GPS*. Principles and Applications. Artech House, 2005.

[3] N. Patwari, J.N. Ash, S. Kyperountas, A.O. Hero, R.L. Moses, and N.S. Correal. Locating the nodes: cooperative localization in wireless sensor networks. *IEEE Signal Process. Mag.*, 22(4):54–69, July 2005.

[4] Kerem Altun and Billur Barshan. Pedestrian dead reckoning employing simultaneous activity recognition cues. *Measurement Science and Technology*, 23(2):025103, 2012.

[5] S Y Cho, Seong Yun Cho, Chan Gook Park, and C G Park. MEMS Based Pedestrian Navigation System. *J. Navigation*, 2006.

[6] Lei Fang, P.J. Antsaklis, L.A. Montestruque, M.B. McMickell, M. Lemmon, Yashan Sun, Hui Fang, I. Koutroulis, M. Haenggi, Min Xie, and Xiaojuan Xie. Design of a wireless assisted pedestrian dead reckoning system - the navmote experience. *IEEE Trans. Instrum. Meas.*, 54(6):2342–2358, Dec 2005.

[7] M. Malleswaran, V. Vaidehi, and M. Mohankumar. A hybrid approach for gps/ins integration using kalman filter and idnn. In *Advanced Computing (ICoAC), 2011 Third International Conference on*, pages 378–383, Dec 2011.

[8] A.R. Jimenez, F. Seco, C. Prieto, and J. Guevara. A comparison of pedestrian dead-reckoning algorithms using a low-cost mems imu. In *Intelligent Signal Processing, 2009. WISP 2009. IEEE International Symposium on*, pages 37–42, Aug 2009.

[9] H. Wymeersch, J. Lien, and M.Z. Win. Cooperative localization in wireless networks. *Proc. IEEE*, 97(2):427–450, Feb 2009.

[10] Xingkai Bao and Jing Li. Cotar: An accurate, cost-effective cooperative wireless localization strategy for mobile nodes. Technical report, 2010.

[11] R.M. Vaghefi and R.Michael Buehrer. Cooperative sensor localization with nlos mitigation using semidefinite programming. In *Positioning Navigation and Communication (WPNC), 2012 9th Workshop on*, pages 13–18, March 2012.

[12] Ning Wang and Liuqing Yang. A semidefinite programming approach for cooperative localization. In *Signals, Systems and Computers (ASILOMAR), 2010 Conference Record of the Forty Fourth Asilomar Conference on*, pages 487–491, Nov 2010.

[13] Yi Shang, W. Rumi, Ying Zhang, and M. Fromherz. Localization from connectivity in sensor networks. *IEEE Trans. Parallel Distrib. Syst.*, 15(11):961–974, Nov 2004.

[14] Tao Jia and R.M. Buehrer. Collaborative position location with nlos mitigation. In *2010 IEEE 21st Int. Symp. Personal, Indoor and Mobile Radio Commun. Workshops (PIMRC Workshops)*, pages 267–271, Sept 2010.

[15] A.T. Ihler, J.W. Fisher, R.L. Moses, and A.S. Willsky. Nonparametric belief propagation for self-calibration in sensor networks. In *Information Processing in Sensor Networks, 2004. IPSN 2004. Third International Symposium on*, pages 225–233, April 2004.

[16] Sebastian Thrun, Dieter Fox, Wolfram Burgard, and Frank Dellaert. Robust monte carlo localization for mobile robots. *Artificial intelligence*, 128(1):99–141, 2001.

[17] C. Pedersen, T. Pedersen, and B.H. Fleury. A variational message passing algorithm for sensor self-localization in wireless networks. In *Information Theory Proceedings (ISIT), 2011 IEEE International Symposium on*, pages 2158–2162, July 2011.

[18] Christopher M Bishop and Christopher M Bishop. *Pattern recognition and machine learning*. Springer, New York, NY, 2006.

[19] Hakan Koyuncu and Shuang Hua Yang. A survey of indoor positioning and object locating systems. *IJCSNS International Journal of Computer Science and Network Security*, 10(5):121–128, 2010.

[20] Fernando Seco, A Jimenez, Carlos Prieto, Javier Roa, and Katerina Koutsou. A survey of mathematical methods for indoor localization. *Intelligent Signal Processing*, pages 9–14, 2009.

[21] Xinzheng Zhang, Ahmad B Rad, and Yiu-Kwong Wong. A robust regression model for simultaneous localization and mapping in autonomous mobile robot. *Journal of Intelligent and Robotic Systems*, 53(2):183–202, 2008.

[22] Yihong Qi and Hisashi Kobayashi. On relation among time delay and signal strength based geolocation methods. In *Global Telecommunications Conference, 2003. GLOBECOM '03. IEEE*, volume 7, pages 4079–4083 vol.7, Dec 2003.

[23] Tao Jia and R.M. Buehrer. Collaborative position location with nlos mitigation. In *Personal, Indoor and Mobile Radio Communications Workshops (PIMRC Workshops), 2010 IEEE 21st International Symposium on*, pages 267–271, Sept 2010.

[24] R.M. Buehrer, Tao Jia, and B. Thompson. Cooperative indoor position location using the parallel projection method. In *Indoor Positioning and Indoor Navigation (IPIN), 2010 International Conference on*, pages 1–10, Sept 2010.

[25] Tao Jia and R.M. Buehrer. A set-theoretic approach to collaborative position location for wireless networks. *IEEE Trans. Mobile Comput.*, 10(9):1264–1275, Sept 2011.

[26] Yihong Qi, Hisashi Kobayashi, and H. Suda. Analysis of wireless geolocation in a non-line-of-sight environment. *IEEE Trans. Wireless Commun.*, 5(3):672–681, March 2006.

[27] Saipradeep Venkatraman and James Caffery Jr. Statistical approach to non-line-of-sight bs identification. In *Wireless Personal Multimedia Communications, 2002. The 5th International Symposium on*, volume 1, pages 296–300. IEEE, 2002.

[28] Kegen Yu and Y Jay Guo. Statistical nlos identification based on aoa, toa, and signal strength. *IEEE Trans. Veh. Commun.*, 58(1):274–286, 2009.

[29] Dajana Cassioli, Moe Z Win, and Andreas F Molisch. The ultra-wide bandwidth indoor channel: from statistical model to simulations. *IEEE J. Sel. Areas Commun.*, 20(6):1247–1257, 2002.

[30] S. Gezici et al. Localization via ultra-wideband radios: a look at positioning aspects for future sensor networks. *IEEE Signal Process. Mag.*, 22(4):70–84, July 2005.

[31] B.M. Donlan, D.R. McKinstry, and R.M. Buehrer. The uwb indoor channel: large and small scale modeling. *IEEE Trans. Wireless Commun.*, 5(10):2863–2873, Oct 2006.

[32] D. Jourdan, D. Dardari, and M.Z. Win. Position error bound for uwb localization in dense cluttered environments. *IEEE Trans. Aerosp. Electron. Syst.*, 44(2):613–628, April 2008.

[33] Wang Yang, Zhang Qinyu, Zhang Naitong, and Wang Yasong. A statistical model for indoor uwb los channels. In *Wireless Communications, Networking and Mobile Computing, 2006. WiCOM 2006. International Conference on*, pages 1–4. IEEE, 2006.

[34] L.A. Annoni, R. Minutolo, M. Montanari, and T. Sabatini. The ultra-wide bandwidth indoor channel: Measurement campaign and modeling. In *Software, Telecommunications and Computer Networks (SoftCOM), 2010 International Conference on*, pages 195–199, Sept 2010.

[35] I. Guvenc, Chia-Chin Chong, and F. Watanabe. Nlos identification and mitigation for uwb localization systems. In *Wireless Communications and Networking Conference, 2007.WCNC 2007. IEEE*, pages 1571–1576, March 2007.

[36] S. Venkatesh and R.M. Buehrer. Non-line-of-sight identification in ultra-wideband systems based on received signal statistics. *Microwaves, Antennas Propagation, IET*, 1(6):1120–1130, Dec 2007.

[37] Ismail Güvenç, Chia-Chin Chong, Fujio Watanabe, and Hiroshi Inamura. Nlos identification and weighted least-squares localization for uwb systems using multipath channel statistics. *EURASIP Journal on Advances in Signal Processing*, 2008:36, 2008.

[38] James J Caffery Jr. A new approach to the geometry of toa location. In *Vehicular Technology Conference, 2000. IEEE-VTS Fall VTC 2000. 52nd*, volume 4, pages 1943–1949. IEEE, 2000.

[39] Lance Doherty, Kristofer SJ Pister, and Laurent El Ghaoui. Convex position estimation in wireless sensor networks. In *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 3, pages 1655–1663. IEEE, 2001.

[40] Zhengdao Wang and Georgios B Giannakis. A simple and general parameterization quantifying performance in fading channels. *IEEE Trans. Commun.*, 51(8):1389–1398, 2003.

[41] Erik G Larsson. Cramer-rao bound analysis of distributed positioning in sensor networks. *IEEE Signal Process. Lett.*, 11(3):334–337, 2004.

[42] S. Venkatesh and R.Michael Buehrer. Nlos mitigation using linear programming in ultrawideband location-aware networks. *IEEE Trans. Veh. Commun.*, 56(5):3182–3198, Sept 2007.

[43] Stefano Marano, Wesley M Gifford, Henk Wymeersch, and Moe Z Win. Nlos identification and mitigation for localization based on uwb experimental data. *IEEE J. Sel. Areas Commun.*, 28(7):1026–1035, 2010.

[44] Chee Kiat Seow and Soon Yim Tan. Non-line-of-sight localization in multipath environments. *IEEE Trans. Mobile Comput.*, 7(5):647–660, 2008.

[45] Jasurbek Khodjaev, Yongwan Park, and Aamir Saeed Malik. Survey of nlos identification and error mitigation problems in uwb-based positioning algorithms for dense environments. *annals of telecommunications-annales des télécommunications*, 65(5-6):301–311, 2010.

[46] Feng Yin. *Robust Wireless Localization in Harsh Mixed Line-of-Sight/Non-Line-of-Sight Environments*. PhD thesis, TU Darmstadt, 2014.

[47] Yuan Shen and M.Z. Win. Fundamental limits of wideband localization; part i: A general framework. *IEEE Trans. Inf. Theory*, 56(10):4956–4980, Oct 2010.

[48] M.A. Spirito. On the accuracy of cellular mobile station location estimation. *IEEE Trans. Veh. Commun.*, 50(3):674–685, May 2001.

[49] Jijie Zhu. Calculation of geometric dilution of precision. *IEEE Trans. Aerosp. Electron. Syst.*, 28:893–895, 1992.

[50] P.-A Oikonomou-Filandras and Kai-Kit Wong. Hybrid non-parametric belief propagation for localization in wireless networks. In *Sensor Signal Processing for Defence (SSPD 2011)*, pages 1–5, Sept 2011.

[51] P.-A. Oikonomou-Filandras, Kai-Kit Wong, and Yangyang Zhang. Heva: Cooperative localization using a combined non-parametric belief propagation and variational message passing approach. *Journal Commun. Netw.*, to be published.

[52] P.-A. Oikonomou-Filandras, Kai-Kit Wong, and Yangyang Zhang. Informed scheduling by stochastic residual belief propagation in distributed wireless networks. *"IEEE Wireless Commun. Lett.*, 4(1):90–93, Feb 2015.

[53] D Koller and N Friedman. *Probabilistic graphical models: principles and techniques*. MIT Press, Cambridge, MA, 2009.

[54] David Barber. *Bayesian reasoning and machine learning*. Cambridge University Press, Cambridge, NY, 2011.

[55] A T Ihler. *Inference in sensor networks: Graphical models and particle methods*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, 2005.

[56] V. Savic and S. Zazo. Nonparametric belief propagation based on spanning trees for cooperative localization in wireless sensor networks. In *2010 IEEE 72nd Vehic. Tech. Conf. Fall (VTC 2010-Fall)*, pages 1–5, Sept 2010.

[57] A.T. Ihler, J.W. Fisher, R.L. Moses, and A.S. Willsky. Nonparametric belief propagation for self-localization of sensor networks. *IEEE J. Sel. Areas Commun.*, 23(4):809–819, April 2005.

[58] E. Fox, E.B. Sudderth, M.I. Jordan, and A. Willsky. Bayesian nonparametric inference of switching dynamic linear models. *IEEE Trans. Signal Process.*, 59(4):1569–1585, April 2011.

[59] J.M. Mooij and H.J. Kappen. Sufficient conditions for convergence of the sum-product algorithm. *IEEE Trans. Inf. Theory*, 53(12):4422–4437, Dec 2007.

[60] Dmitry Malioutov, Alan S Willsky, and Jason K Johnson. Walk-sum interpretation and analysis of gaussian belief propagation. In *Advances in Neural Inform. Process. Syst.*, pages 579–586, 2005.

[61] Alexander T Ihler, John Iii, and Alan S Willsky. Loopy belief propagation: Convergence and effects of message errors. In *Journal of Machine Learning Research*, pages 905–936, 2005.

[62] Jaime Lien, Ulric J Ferner, Warakorn Srichavengsup, Henk Wymeersch, and Moe Z Win. A comparison of parametric and sample-based message representation in cooperative localization. *International Journal of Navigation and Observation*, 2012, 2012.

[63] N. Kantas, S.S. Singh, and Arnaud Doucet. Distributed self localisation of sensor networks using particle methods. In *Nonlinear Statistical Signal Processing Workshop, 2006 IEEE*, pages 164–167, Sept 2006.

[64] S. Movaghati and M. Ardakani. Particle-based message passing algorithm for inference problems in wireless sensor networks. *IEEE Sensors J.*, 11(3):745–754, March 2011.

[65] V. Savic and S. Zazo. Nonparametric boxed belief propagation for localization in wireless sensor networks. In *Sensor Technologies and Applications, 2009. SENSORCOMM '09. Third International Conference on*, pages 520–525, June 2009.

[66] Trevor Hastie, Robert Tibshirani, Jerome Friedman, T Hastie, J Friedman, and R Tibshirani. *The elements of statistical learning*, volume 2. Springer, 2009.

[67] Godfried T Toussaint and Jim A McAlear. A simple o (n log n) algorithm for finding the maximum distance between two finite planar sets. *Pattern Recognition Letters*, 1(1):21–24, 1982.

[68] J.R. Hershey and P.A. Olsen. Approximating the kullback leibler divergence between gaussian mixture models. In *Acoustics, Speech and Signal Process-*

ing, 2007. ICASSP 2007. IEEE International Conference on, volume 4, pages IV–317–IV–320, April 2007.

[69] Hans Petter Langtangen. *Python scripting for computational science*, volume 3. Springer, Berlin, Heidelberg, 2006.

[70] G. J. Pottie and W. J. Kaiser. Wireless integrated network sensors. *Commun. ACM*, 43(5):51–58, May 2000.

[71] F. Yin, C. Fritsche, D. Jin, F. Gustafsson, and A.M. Zoubir. Cooperative localization in wsns using gaussian mixture modeling: Distributed ecm algorithms. *IEEE Trans. Signal Process.*, 63(6):1448–1463, March 2015.

[72] Jan Kouba, Josef Popelar, M Elizabeth Cannon, and Gérard Lachapelle. *Modern geodetic reference frames for precise satellite positioning and navigation*. University of Calgary, Department of Geomatics Engineering, 1994.

[73] R Lampinen. Universal transverse mercator (utm) and military grid reference system (mgrs), 2001.

[74] M.A. Caceres, F. Penna, H. Wymeersch, and R. Garello. Hybrid cooperative positioning based on distributed belief propagation. *IEEE J. Sel. Areas Commun.*, 29(10):1948–1958, December 2011.

[75] Travis E. Oliphant. Python for scientific computing. *Computing in Science & Engineering*, 9(3):10–20, 2007.

[76] S. Rezaei and Raja Sengupta. Kalman filter-based integration of dgps and vehicle sensors for localization. *IEEE Trans. Control Syst. Technol.*, 15(6):1080–1088, Nov 2007.

[77] Reza Zekavat and R Michael Buehrer. *Handbook of position location: theory, practice and advances*, volume 27. John Wiley & Sons, 2011.

[78] J. Georgy, A. Noureldin, and M. Bayoumi. Mixture particle filter for low cost ins/odometer/gps integration in land vehicles. In *Vehicular Technology Conference, 2009. VTC Spring 2009. IEEE 69th*, pages 1–5, April 2009.

[79] Wonho Kang, Wonho Kang, Seongho Nam, Seongho Nam, Youngnam Han, Youngnam Han, Sookjin Personal Indoor Lee, Mobile Radio Communications PIMRC 2012 IEEE 23rd International Symposium on, Sookjin Lee, Sookjin Personal Indoor, and Mobile Radio Lee. Improved heading estimation for smartphone-based indoor positioning systems. *Personal Indoor and Mobile Radio Communications (PIMRC), 2012 IEEE 23rd International Symposium on*, 2012.

[80] Z Sun, Zuolei Sun, X Mao, Xuchu Mao, W Tian, Weifeng Tian, Xiangfen Zhang, and X Zhang. Activity classification and dead reckoning for pedestrian navigation with wearable sensors. *Meas. Sci. Technol.*, 2009.

[81] A G Quinchia, A G Quinchia, C Ferrer, C Ferrer, G Falco, G Falco, E Falletti, E Falletti, F Dovis, and F Dovis. Analysis and modelling of MEMS inertial measurement unit. In *Localization and GNSS (ICL-GNSS), 2012 International Conference on*, 2012.

[82] Naser El-Sheimy, N El-Sheimy, Haiying Hou, Haiying Hou, Xiaoji Niu, and Xiaoji Niu. Analysis and Modeling of Inertial Sensors Using Allan Variance. *IEEE Trans. Instrum. Meas.*, 2008.

[83] Chengliang Huang, Zaiyi Liao, and Lian Zhao. Synergism of ins and pdr in self-contained pedestrian tracking with a miniature sensor module. *IEEE Sensors J.*, 10(8):1349–1359, Aug 2010.

[84] T. Higuchi, S. Fujii, H. Yamaguchi, and T. Higashino. Mobile node localization focusing on stop-and-go behavior of indoor pedestrians. *IEEE Trans. Mobile Comput.*, 13(7):1564–1578, July 2014.

[85] M Angermann, A Friese, and M Khider. A reference measurement data set for multisensor pedestrian navigation with accurate ground truth. *European Navigation*, 2009.

[86] M Angermann, P Robertson, T Kemptner, and M Khider. A high precision reference data set for pedestrian navigation using foot-mounted inertial sen-

sors. In *Indoor Positioning and Indoor Navigation (IPIN), 2010 International Conference on*, 2010.

[87] I Skog, P Handel, J O Nilsson, and J Rantakokko. Zero-Velocity Detection, An Algorithm Evaluation. *IEEE Trans. Biomed. Eng.*, 2010.

[88] Yunye Jin, Hong-Song Toh, Wee-Seng Soh, and Wai-Choong Wong. A robust dead-reckoning pedestrian tracking system with low cost sensors. In *Pervasive Computing and Communications (PerCom), 2011 IEEE International Conference on*, pages 222–230, March 2011.

[89] Jonathan S. Yedidia, W.T. Freeman, and Y. Weiss. Constructing free-energy approximations and generalized belief propagation algorithms. *IEEE Trans. Inf. Theory*, 51(7):2282–2312, July 2005.

[90] T.G. Roosta, M.J. Wainwright, and S.S. Sastry. Convergence analysis of reweighted sum-product algorithms. *IEEE Trans. Signal Process.*, 56(9):4293–4305, Sept 2008.

[91] M.J. Wainwright, T.S. Jaakkola, and A.S. Willsky. Tree-based reparameterization framework for analysis of sum-product and related algorithms. *IEEE Trans. Inf. Theory*, 49(5):1120–1146, May 2003.

[92] Gal Elidan, Ian McGraw, and Daphne Koller. Residual belief propagation: Informed scheduling for asynchronous message passing. *arXiv preprint arXiv:1206.6837*, 2012.

[93] Huang-Chang Lee, Yeong-Luh Ueng, Shan-Ming Yeh, and Wen-Yen Weng. Two informed dynamic scheduling strategies for iterative ldpc decoders. *IEEE Trans. Commun.*, 61(3):886–896, March 2013.

[94] H. Wymeersch, F. Penna, and V. Savic. Uniformly reweighted belief propagation: A factor graph approach. In *2011 IEEE Int. Symp. Inform. Theory*, pages 2000–2004, July 2011.

[95] H. Wymeersch, F. Penna, and V. Savic. Uniformly reweighted belief propagation for estimation and detection in wireless networks. *IEEE Trans. Wireless Commun.*, 11(4):1587–1595, April 2012.