Cooperative, Range-based Localization for Mobile Sensors

Andrew Symington
Green Templeton College
University of Oxford

Dissertation submitted for the degree of
Doctor of Philosophy

Hilary Term 2013

I owe my gratitude to many people who have helped shape this research. However, I am particularly indebted to Dr Niki Trigoni, Dr Simon Julier and Dr Andrew Markham for their valuable support over the last few years. As a supervisor, Niki has always been approachable and encouraging, and her publication-oriented mentorship has been an asset to my work. Simon has a seemingly-endless vault of expertise and contacts, which he has readily shared with me. He is incredibly helpful, and has patiently worked with me numerous times to grasp new concepts. Andrew is an inspiring researcher, who is not afraid to engage with experimentation, and he never seems too busy to talk over a problem or idea. And, I am incredibly appreciative for all of the emails, sample code, and soldering lessons he has given me.

I am also thankful for the excellent feedback from my earlier *Transfer of Status* and *Confirmation of Status* examiners, Dr Dan Olteanu, Dr Stephen Cameron, Dr Ingmar Posner and Dr Phil Blunsom. In addition, I would like to extend my gratitude to Dr Eduardo Lopez, Dr Abhijit Kallapur, Dr Louis Theran, Dr Naomi Fox, and Dr Steven Reece for interesting respective discussions about control models, filtering, graph rigidity and statistics, which has helped give substance and direction to several components of my research. Finally, I'd like to thank Dr Sebastian Madgwick from x-io.co.uk for providing excellent technical support for the x-IMU units we purchased.

I am also very fortunate to have worked in such an engaging office atmosphere, surrounded by enthusiastic researchers. In particular, I'd like to thank Dr Renzo de Nadi, Dr Sonia Waharte, Dr Sarfraz Nawaz, Dr Ricklef Wohlers, Dr Julian de Hoog, Dr Muzammil Hussain, Paul Ward, Hongkai Wen, Zhuoling Xiao, Anna Zawilska and Savvas Papaioannou for lengthy discussions about research problems. In addition, I'd like to thank Jonah Rimer, Sarah-Ann Burger, Susan Portalupi, Ryan Hogarth, Joseph Feyertag and Dr Alex Rhodes for willingly helping out as test subjects in earlier localization experiments. Thanks also to Dr Alison Foster for allowing me to run my tracking experiments at the Oxford Botanic Garden.

Pursuing a post-graduate degree can be an costly endeavour. I am incredibly fortunate to have received an EPSRC *Doctoral Studentship* to support my studies, and an EPSRC *Pathways to Impact* grant to help commercialize the research. I would also like to thank the *Anthony Storr Bursary Fund*, *Green Templeton College* and the *Department of Computer Science* for providing critical financial support, which helped 'tide the gap' in my final year of study.

However, I owe the greatest debt to my family and friends for their support over the last four years. In particular, I would like to thank Nadine for motivating me to complete, showing patience during my write-up, and for always being there for me.

Abstract

This thesis describes the development of an offline, cooperative, range-based localization algorithm for use in settings where there is limited or no access to a positioning infrastructure. Motivating applications include underground animal tracking and indoor pedestrian localization. It is assumed that each sensor performs dead reckoning to estimate its current position, relative to a starting point. Each measurement adds error, causing the position estimate to drift further from the truth with time. The key idea behind the proposed algorithm is to use opportunistic radio contacts to mitigate this drift, and hence localize with greater accuracy.

The proposed algorithm first fuses radio and motion measurements into a compact graph. This graph encodes key positions along sensor trajectories as vertices, and distance measurements as edges. In so doing, localization is cast as the graph realization problem: assigning coordinates to vertices, in such a way that satisfies the observed distance measurements. The graph is first analysed to certify whether it defines a localization problem with a unique solution. Then, several algorithms are used to estimate the vertex coordinates. These vertex coordinates are then used to apply piecewise corrections to each sensor's dead reckoning trajectory to mitigate drift. Finally, if sufficient anchors are available, the corrected trajectories are then projected into a global coordinate frame.

The proposed algorithm is evaluated in simulation for the problem of indoor pedestrian tracking, using realistic error models. The results show firstly that 2D and 3D problems become provably more localizable as more anchors are used, and as the experiment duration increases. Secondly, it is shown that widely-used graph realization algorithms cannot be used for localization, as the complexity of these algorithms scales polynomially or greater with graph vertex count. Thirdly, it is shown a novel piecewise drift correction algorithm typically works well compared to a competing approach from the literature, but rare and identifiable graph configurations may cause the method to underperform.

Notation

n	Number of sensors
m	Number of anchors
T	Number of discrete time epochs
δt	Constant epoch duration
$\mathbf{X}^i$	Estimated position of sensor $i \in [1, n]$ at time $t \in [1, T]$
$\mathbf{Y}^i$	Actual position of sensor $i \in [1, n]$ at time $t \in [1, T]$
d	Dimension (either 2D or 3D)
D	Encounter graph distance matrix
W	Encounter graph weighting matrix
E	Encounter graph edge matrix
V	Encounter vertex set
v	Number of vertices in encounter graph
e	Number of edges in encounter graph
X	Coordinates matrix of size $v \times d$
f_r	Radio model $f_r : (\gamma_{1:k}) \rightarrow \mathcal{N}(\mu, \sigma^2)$
f_m	Motion model $f_m : (x_{1:T}, P_{1:T}, s, t) \rightarrow \mathcal{N}(\mu, \sigma^2)$
x_t	PDR filter state $x_t = \begin{bmatrix} p_t & v_t & r_t & \beta_t^\omega & \beta_t^a \end{bmatrix}^T$ at time t
P_t	PDR filter covariance at time t
C_t	PDR filter rotation matrix at time t
p_t	PDR filter state: position at time t
v_t	PDR filter state: velocity at time t
r_t	PDR filter state: Euler attitude at time t
β_t^ω	PDR filter state: gyroscope bias at time t
β_t^a	PDR filter state: accelerometer bias at time t
h, H	PDR filter non-linear and linearized measurement model
f, F	PDR filter non-linear and linearized process model
g, G	PDR filter non-linear and linearized noise perturbation model
Q	Process noise matrix
R	Measurement noise matrix
z_t	Zero velocity measurement $z_t = \mathbf{0}$ at time t
u_t	Control vector $u_t = \begin{bmatrix} \alpha_t & \omega_t \end{bmatrix}^T$
ω_t	Angular velocity measurement at time t in $^{rads}/_s$
α_t	Linear acceleration measurement at time t in $^m/_{s^2}$
w_t	Process disturbance vector $w_t = \begin{bmatrix} w_\omega & w_\alpha & w_{\beta^\omega} & w_{\beta^\alpha} \end{bmatrix}^T \sim \mathcal{N}(0, R)$
w_ω,w_α	Noise perturbation for gyroscope and accelerometer
w_{β^ω},w_{β^α}	Angular random walk of gyroscope and accelerometer
E_g	SI gravity of earth $E_g = 9.80665^m/_{s^2}$
r	Range or distance in meters
γ	Received signal strength in dBm
η	Path loss exponent for radio propagation
σ_r^2	Constant variance of received signal strength measurements
μ	Mean of distance estimate (from radio or motion model)
σ^2	Variance of distance estimate (from radio or motion model)

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Consider a network of small, resource-constrained mobile devices that sense the environment, perform processing tasks, store and communicate data. Examples of such devices, or *sensors*, include smart phones, wildlife tracking collars or unmanned aerial vehicles. For many applications knowing the location of the sensor is critical. For example, an application designed to monitor environmental humidity is substantially more useful when each sensor measurement has a corresponding position in space. *Localization* is the process by which such a position is found.

Outdoor localization can be achieved to decimeter accuracy with post-processed satellite measurements, provided that accelerations are small. Indoor localization, on the other hand, still remains an open problem. The high frequency radio signals used in satellite navigation reflect off buildings, preventing their reception indoors. Walls, floors and other structures also cause multi-path and attenuation, making the accuracy of terrestrial radio localization dependent on the positions of the transceivers, and the clutter topology. Nonetheless, with a relatively static environment, good configuration of anchor points and sufficient training data, radio fingerprinting can position to an accuracy in the order of several meters.

However, many sensor deployments preclude the use of infrastructure-based localization, as anchors may be sparse or infeasible to deploy. When an infrastructure is unavailable, sensors may instrument their own motion and use *dead reckoning* to estimate their position. Cameras and laser scanners provide high-accuracy dead reckoning, but cannot be used in sensor networking problems because of resource constraints. Recent advances in fabrication have given rise to micro-electromechanical system (MEMS) inertial sensors. When used in a strapdown configuration to measure motion these sensors can be used to maintain an estimate of position. However, MEMS sensors suffer noise, which compounds with time to bias this estimate. This

makes dead reckoning inaccurate over long periods[1].

Owing to the fading property of wireless channels, whenever two sensors exchange a beacon it relates their positions in space and time. However, since these encounters occur opportunistically, one cannot guarantee network connectivity at any given point in time. This makes real-time localization challenging, as each sensor has a different version of the problem, described by its locally-sensed measurements. The "offline" version of the same problem seeks to post-processes all inertial and radio measurements. This is well-suited to delay-tolerant applications, such as wildlife tracking.

The objective of this thesis is to develop and evaluate a novel offline, range-based cooperative localization algorithm for localizing mobile sensors. This algorithm converts inertial and radio measurements – derived from either time of flight or signal strength – into a connected graph. Vertices in this graph represent device positions, while edges represent distances, or *ranges*, between vertices. Since this graph abstracts from the individual measurements, it is compact and efficient to optimize over. Moreover, it provides a convenient representation for certifying whether the problem is uniquely-defined, as well as solving the problem.

This chapter is organised in the following way. Sec. 1.1 formally defines the scope of the research problem, while Sec. 1.2 describes three applications that motivate the need for offline, range-based cooperative localization. Sec. 1.3 then outlines the proposed approach, and Sec. 1.4 lists the primary novel contributions of this research. This chapter concludes with a publication list in Sec. 1.5, and an outline in Sec. 1.6.

1.1 Problem definition and scope

Consider a bounded 2D or 3D environment containing m anchors and n mobile sensors. Each anchor k has a known, fixed location $\mathbf{a}_k$. Consider the problem of tracking the sensors' positions over a T second period. For convenience, we'll denote the truthful position of sensor i at time t as $\mathbf{y}_t^i$.

Each sensor i measures its own motion using an *Inertial Measurement Unit* (IMU) in a strapdown configuration. It is assumed that the sensors are resource-constrained, and so the energy, storage and communication cost of each data bit is high. This makes it infeasible to store or communicate all raw inertial measurements, while simultaneously preserving sensor lifetime. Therefore, each sensor performs *dead reckoning* online by recursively filtering its locally-sensed inertial measurements at a high frequency. The filter estimates the sensor's trajectory relative to a starting position,

[1] This is true in general, but a significant error can be mitigated by exploiting domain knowledge

which will be assumed $\mathbf{x}_0^i = \mathbf{0}$, and the $t$the measurement is comprised of a mean $\mathbf{x}_{0:T}^i$ position and covariance $P_{0:T}^i$. As time progresses, so measurement error accumulates, and the estimated position $\mathbf{x}_t^i$ drifts further from $\mathbf{y}_t^i$. In order to conserve energy, $\mathbf{x}_{0:T}^i$ and $P_{0:T}^i$ are down-sampled to a much lower frequency.

Periodically, sensor i moves within radio communication range of another sensor or anchor j, and they exchange a beacon. This event results in a scalar measurement $r_t^{i,j}$ of the pairwise distance between the two devices at time t, derived from radio ranging, either using time-of-flight (TOF) or received signal strength (RSS) measurements. The key idea is to use these opportunistic constraints to mitigate drift, and localize the sensors with a greater precision.

This localization problem is challenging for the following reasons:

1. **Connectivity** - It is assumed that sensor mobility is unknown. Connectivity is therefore opportunistic, and there is no guarantee that a sensor encounters a sufficient number of peer sensors or anchors over time T. This motivates the need for a localization algorithm that certifies whether the set of measurements uniquely defines a problem, in addition to trying to solve the problem.

2. **Noise** - Even if a given localization problem has a single solution, the solution itself may be difficult to find. In particular, sensing instrumentation suffers noise, which manifests as measurement error. This motivates the need for a localization algorithm that takes into account measurement uncertainty when obtaining a solution.

3. **Scalability** - Many localization algorithms can be reduced to a graphical representation of the problem. The run-time complexity of optimizing over this graph is normally a polynomial function of the graph vertices. This motivates the need for a localization algorithm that operates on an abstract representation of the full localization problem.

The specific focus of this thesis is therefore a offline solution to cooperative, range-based localization. The method is offline in the sense that measurements are collected and post-processed to arrive at a localization solution. The online solution is substantially more difficult, because intermittent network connectivity causes information asymmetry across the network. The *range-based* prefix shows that all inertial and radio measurements are reduced to a pairwise distance measurements between anchor or sensor positions. The *cooperative* prefix shows that all measurements are used

simultaneously to obtain a localization solution. The advantage with cooperative approaches is that they do not require anchors.

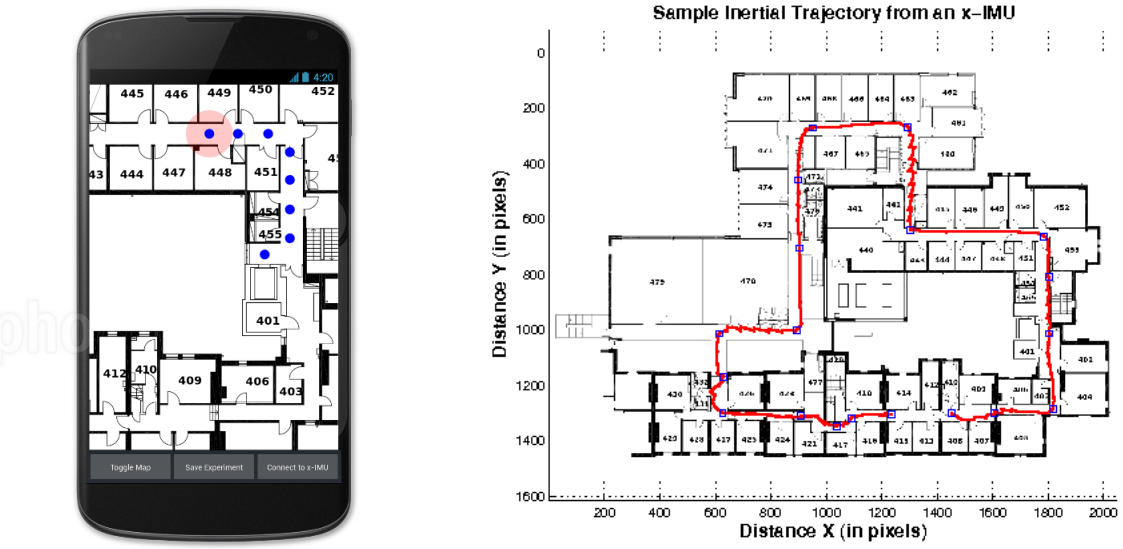
The cooperative localization problem is further complicated by the following two issues, both of which are important but are not addressed in this thesis:

- **Synchronization** - A sensor's perception of time may drift with respect to a reference clock, as a result of noise. Clock crystals typically suffer noise in the order of 20 to 50 parts per million, which equates to approximately a minute per month. The crystal suffers further error in environments with large temperature fluctuations. Time synchronization is a well studied research field, and the reader is referred to Sivrikaya and Yener [85] for a survey of the literature.

- **Security** - Possible exploits in cooperative localization include third parties gaining access to the data, sensors spoofing their identity, or maliciously reporting erroneous measurements. In many sensitive applications it may be undesirable for an untrusted party (either a peer in the network or an outside observer) to access or reconstruct sensors' location information.

1.2 Motivating applications

This section introduces three real-world applications that motivate the need for offline range-based cooperative localization. Sec 1.2.1 discusses the first of these applications, which involves monitoring how customers explore retail spaces. The second application, discussed in Sec. 1.2.2, involves monitoring the total number of people an office floor, in order to comply with Health and Safety requirements in large buildings. The third application involves monitoring badgers underground using collar sensors, and is discussed in Sec. 1.2.3.



(a) A smart phone sensor.

(b) An example customer trajectory.

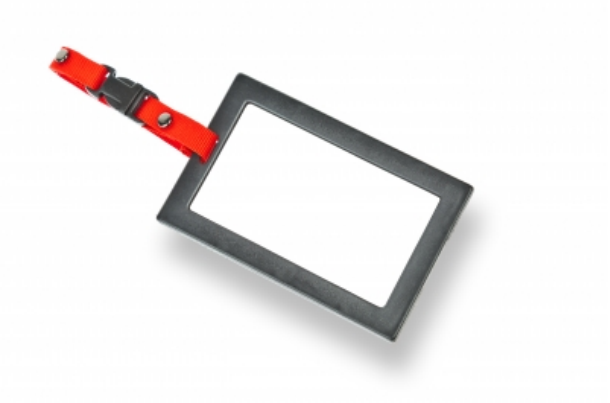
Figure 1.1: Cooperative, range-based localization for customer profiling in retail.

1.2.1 Customer tracking in retail settings

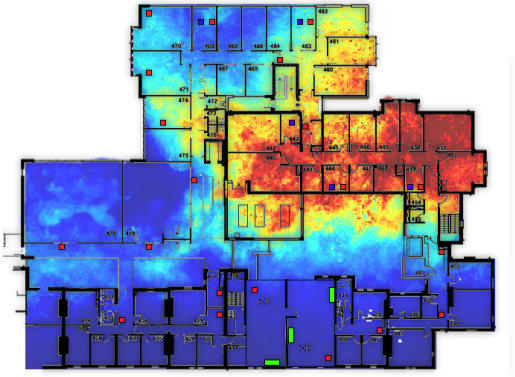
Most smart phones contain at least a low-cost accelerometer and a magnetometer, and are thus able to measure device orientation and acceleration. When combined with step counting, it is possible to obtain a crude trajectory for the user. More recent smart phones contain a variety of additional sensors, including gyroscopes and barometric pressure sensors, allowing more advanced dead reckoning techniques.

Consider a retail application designed for a smart phone, which provides information that improves a customer's experience within a shopping centre (see Fig. 1.1a). Such an application could for example provide real-time pricing, exclusive discounts, navigation assistance and additional product information. In exchange for this information, the application would gather fine-grained location data, which would help management understand how customers navigate the space (see Fig. 1.1b). This may improve marketing strategies or help set rental pricing models for retail space, based on aggregate information about how customers behave.

Cooperative, range-free localization can be used to fuse motion information with opportunistic GPS fixes or wireless encounters – from either Bluetooth, WiFi or a cellular network – in order to localize the customer. However, in addition to the ethical questions raised by such a system, pedestrian dead reckoning with smart phones remains a difficult problem, due to the fact that the device itself is not rigid with respect to the person it seeks to localize.



(a) RFID access control tag.



(b) Heatmap showing how office space is used..

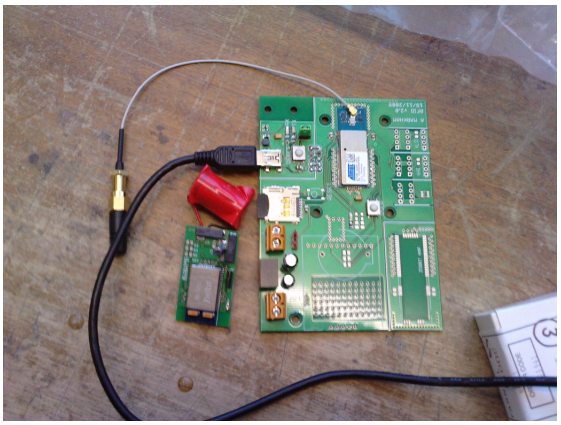
Figure 1.2: Offline, range-based localization for office space monitoring.

1.2.2 Office space usage monitoring

Understanding how office space is used by employees or visitors is important to businesses. An obvious use of this information is security. However, this information may also help with resource allocation or compliance with legislation. For example, switching the location of two divisions may even the density of employees across the building, preventing contention for resources, such as kitchens, washrooms, photocopying rooms or meeting rooms. Alternatively, Health & Safety regulations state the maximum number of people per room, based on the number of emergency exits. It may be impractical to fit all thoroughfares with access control, and so even monitoring office space usage at a coarse level is difficult. Currently, large companies rely on intermittent manual counts to verify compliance.

Consider embedding an inertial sensor into a Radio-Frequency Identification (RFID) access control tag, such as the one depicted in Fig. 1.2a. These tags are carried by each employee throughout the building. Much like the previous motivating application, this tag would be used to track the employee's position through dead reckoning. Periodic card swipes would provide highly-accurate opportunistic encounters with known points. Cooperative, range-based localization can be used to fuse these opportunistic encounters with the motion measurements to build an occupancy map, such as the one depicted in Fig. 1.2b. This map indicates which parts of the building are occupied by the most number of people over a fixed period.

(a) A badger with sensing collar for measuring position using magneto-inductive tracking.



(b) Internals of the collar sensor (left) with red power source, pictured alongside the base station (right).

Figure 1.3: Offline, range-based localization for badger habitat monitoring.

1.2.3 Underground badger tracking

As of writing, badgers are being culled across the United Kingdom, as part of a pilot programme to halt the spread of tuberculosis to cattle. In order for interventions like these to be effective, one must first understand how different badger populations interact. An important precursor to this is the ability to monitor the animal's location. However, tracking badgers is difficult, because they are burrowing animals that spend a significant amount of time per day underground in *setts*. Radio signals do not propagate well through soil, making GPS or radio positioning ineffective for tracking.

Markham et al. [61] have shown how magnetic fields – which are not generally disrupted by soil – may be used to localize badgers underground using collar sensors (see Fig. 1.3a). However, magnetic field strength decays cubically with distance, and so its usable range is effectively limited to several meters from a base station (see Fig. 1.3b). What is needed is a system that will continue to track the location of tagged animals underground, albeit at a lower accuracy, whenever the badgers move outside the reach of the magnetic field.

Consider a collar that records motion information, magnetic and radio encounters. Cooperative, range-based localization could be used to localize tagged animals by fusing inertial measurements with opportunistic encounters, either between tagged animals or with magnetic (underground) or radio (overground) anchors. Periodically, an animal might emerge from the ground, providing a brief moment where radio measurements may be used to correct the position estimate. Such anchor encounters may also be used to project the solution into a global coordinate frame.

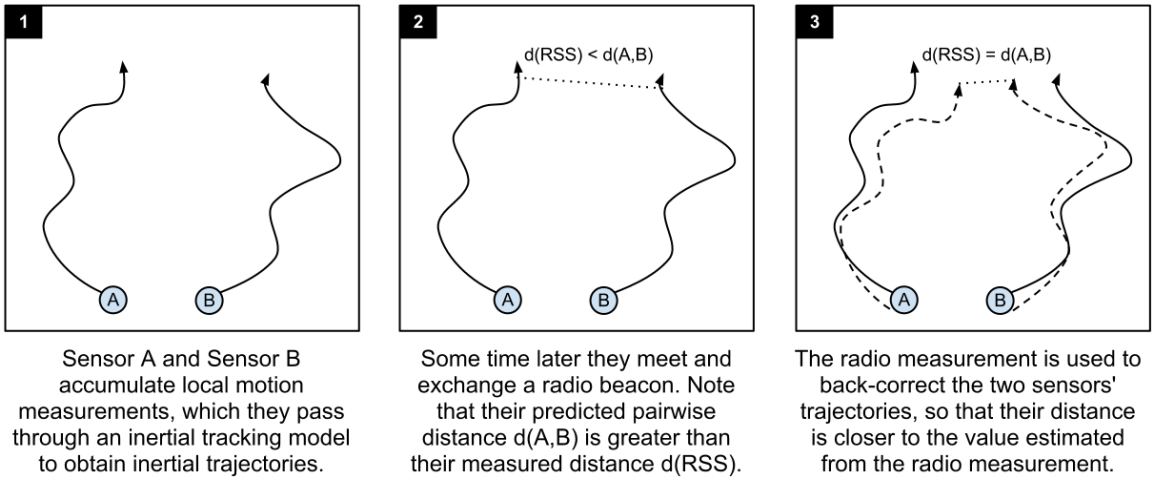


Figure 1.4: Using opportunistic radio encounters to mitigate drift.

1.3 Proposed approach

The primary contribution of this thesis is an offline, cooperative, range-based localization algorithm. The key idea behind this algorithm is that whenever two devices exchange a radio beacon with each other, it provides information that relates their unknown locations in space and time. The algorithm fuses many such encounters together, along with inertial measurements and anchor positions (if available) to solve for all unknown sensor trajectories.

Fig. 1.4 illustrates this key idea. Sensors A and B begin at known locations and use dead reckoning to maintain an estimate of their location. Some time later the two sensors come within communication range and exchange a radio beacon. At this point the sensors have a *predicted* pairwise distance of $d(A, B)$. A radio path loss model is used to convert the radio beacon *Received Signal Strength* (RSS) to a pairwise distance $d(RSS)$ in meters. The residual error between the predicted and measurement distance permits the two trajectories to be corrected.

The proposed algorithm extends this fundamental idea by fusing many encounters between multiple sensors and anchors into a graph, and casting cooperative localization as a graph realization problem. Not only is this representation compact, but it also provides a method of checking whether a given set of measurements provides sufficient information to define problem with a unique solution. The algorithm is split into three phases – *Graph Construction*, *Graph Realization* and *Trajectory Reconstruction*. These phases are outlined briefly in Sec. 1.3.1, 1.3.2 and 1.3.3, and will be discussed in greater detail over the next three chapters.

1.3.1 Graph Construction

Consider the example problem shown in the top-left subfigure of Fig. 1.5, which contains two sensors, with trajectories shown as red and blue dashed arrows. At times $T = 1, 2, 3$ the two sensors encounter each other, or a base station. Each encounter is shown as a black edge. If one considers each event to be a case of static localization, the problem yields a disconnected graph. Since the anchor positions are known, this information can be incorporated as pairwise distances between anchors. This is illustrated in the top-right subfigure. Although the resulting graph is connected, it does not form a rigid structure, which means that there is insufficient information to uniquely localize the sensors. Under the assumption that dead reckoning works reasonably well over short periods, it can be used to measure distances between encounter points along a single sensor's trajectory. This information is shown in the bottom left of Fig. 1.5. Finally, the radio encounters, motion measurements, and anchor positions can be fused together to build the graph shown in the bottom-right subfigure, which captures the relationship between key points along the sensors trajectories.

Note that, for clarity, the vertices are drawn with relative positions in Fig. 1.5. In practice, the relative coordinates of the vertices (anchors excluded) are unknown. The *Encounter Graph* is completely described by sparse and symmetric distance and weighting matrices. If the graph contains an edge between vertices i and j, then it will have a positive element at index (i, j) of both the distance matrix and the weighting matrix. The element of the distance matrix reflects the estimate of the distance between vertices i and j, while the weighting matrix reflects the confidence in this estimate, expressed as the variance of a zero-mean Gaussian distribution.

1.3.2 Graph Realization

The Graph Realization phase of the proposed localization seeks to find a $k \times d$ matrix of d-dimensional graph vertex coordinates, which best satisfies the measured elements of the distances and weighting matrices. The phase is divided into three steps, which are shown in Fig. 1.6. The objective of the first step is to determine whether a specific sparse distance matrix defines a problem with exactly one solution (a unique set of vertex coordinates in $\mathbb{E}^d$). This can be achieved using only the distance matrix, and the process is discussed further in Sec. 1.3.2.1. Given that a unique solution exists, the objective of Sec. 1.3.2.2 is to calculate this solution using only the distance matrix.

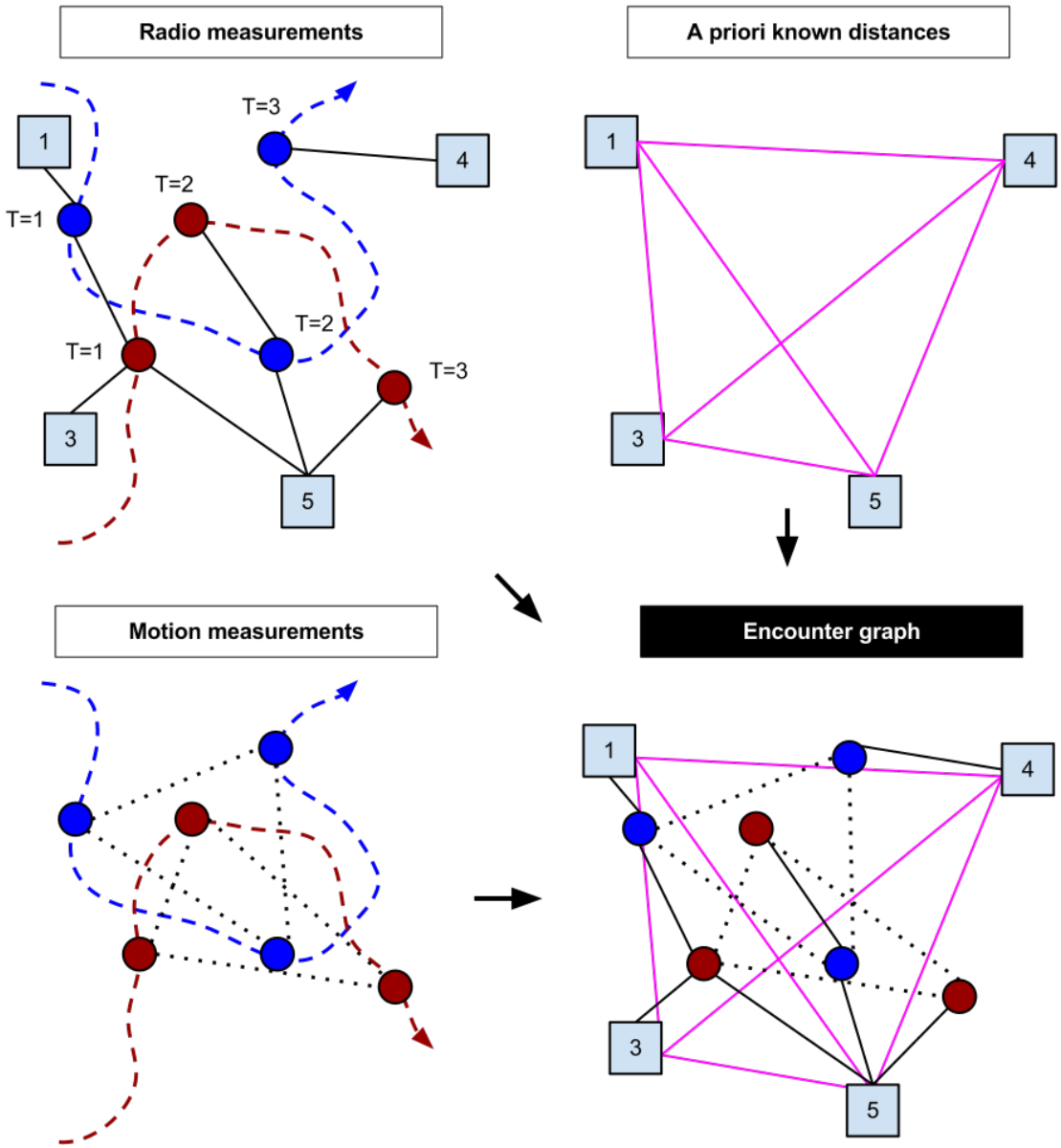


Figure 1.5: Diagram showing how the encounter graph is constructed for a toy example containing four anchors, denoted by squares, and two sensors, denoted by trajectories coloured red and blue. Radio measurements (top-left) are fused with motion measurements (bottom-left) and distances between anchors (top-right) to build the encounter graph (bottom-right).

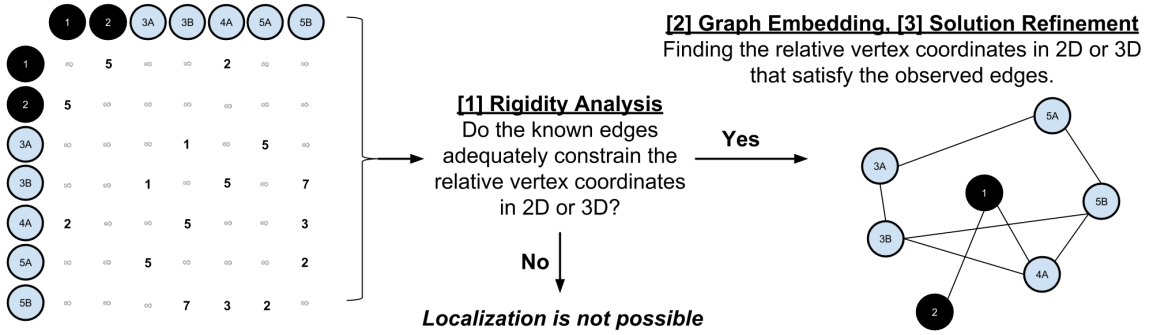


Figure 1.6: The first step of the graph realization phase is to determine whether the graph given by a symmetric distance matrix (left) sufficiently constrains the relative vertex coordinates. If this is not the case, there is an inherent ambiguity in the solution space defined by the problem, and localization is not possible. Under the assumption that there is no solution ambiguity, the next step of graph realization is to find a suitable graph embedding. The third and optional step is to refine this embedding, taking into account measurement error.

Finally, Sec. 1.3.2.3 discusses refinement algorithms for improving this solution, which take into account the weighting matrix.

1.3.2.1 Rigidity Analysis

In order to operate in infrastructure-free scenarios, the proposed localization algorithm first finds the relative positions of vertices in the encounter graph, and then projects this constellation into a global reference frame if anchors are available.

The 2D localization problem depicted in Fig. 1.7 illustrates that a single distance matrix may have multiple valid embeddings. Such problems are impossible to solve, as there is an inherent ambiguity in the solution space defined by the problem. In the *Rigidity Analysis* step of the proposed localization algorithm, graph rigidity theory is used to determine whether the set of available measurements defines a cooperative localization problem with exactly one solution (i.e. a provably localizable graph). The proposed localization algorithm provides a deterministic method of certifying localizability in $\mathbb{E}^2$, and a probabilistic method of certifying localizability in $\mathbb{E}^3$ that never returns a false positive result.

1.3.2.2 Graph Embedding

The objective of the *Graph Embedding* step is to assign each graph vertex a d-dimensional coordinate in such a way that the resultant pairwise distances match the elements in the distance matrix. There are a great number of graph embedding

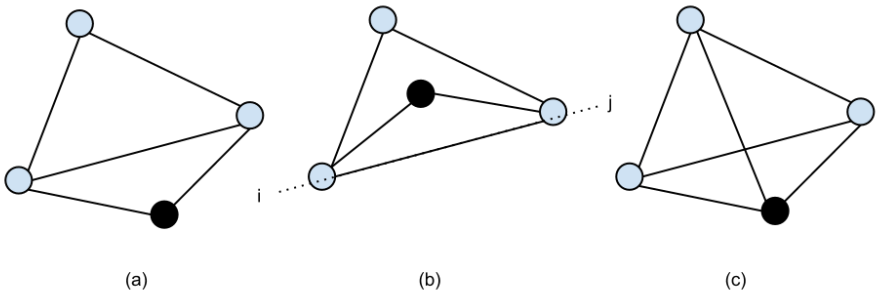


Figure 1.7: Not all measurements uniquely define a localization problem. Consider the three graphs above, representing the same 2D localization problem. Each vertex encodes the 2D position of a sensor, while the edges are observed distance measurements between pairs of sensors. Problem (a) and Problem (b) have the same distance matrices, but different vertex positions. This is caused by reflective symmetry about the (i, j) axis in Problem (b). Figure (c) shows the the same problem, introducing one more measurement. The resulting graph constrains all non-trivial degrees of freedom in 2D and is thus uniquely localizable.

algorithms in the literature, which seek to solve such a problem. The proposed localization algorithm is not prescriptive, but is tested using the following algorithms from the literature:

1. *Multidimensional Scaling* (MDS) - This method by Shang et al. [83] is similar to Principal Component Analysis (PCA), and it reduces the computation of vertex coordinates to finding the eigenvectors of a matrix derived from the *complete* distance matrix. It follows that this algorithm requires an all-pairs shortest path algorithm to complete the distance matrix, which causes sparsely-connected vertices to be pulled toward the centroid.

2. *Distributed Multidimensional Scaling* (MAP) - This method by Shang and Ruml [82] performs separate MDS solutions for each node in the network, using its 2-hop neighbourhood. The resultant solutions are then stitched back together. This method is more accurate for irregularly-shaped graphs.

3. *Spectral Graph Drawing* (SGD) - This method by Broxton et al. [7] attempts to push together vertices with very high weightings, resulting in a realization where highly-connected nodes are located towards the center. Thus, the negative exponent of the distance matrix is typically used as the weighting matrix. Since unknown edges can simply be assigned a weight of zero, unlike MDS, an all-pairs shortest path algorithm is not required.

4. *Degree Normalized Spectral Graph Drawing* (DNS) - This method by Koren [51] is similar to SGD but its objective function is normalized by vertex degree[2], meaning that highly-connected nodes are pushed apart from one another.

1.3.2.3 Solution Refinement

In the *Solution Refinement* step of the proposed algorithm the graph embedding is refined. Small changes are iteratively applied to the embedding, such that the weighted distance error between the predicted and observed distances is minimized. The refinement algorithm terminates after a fixed number of iterations, or when the changes to the state differ by less than a certain threshold value. The actual refinement algorithm is not prescribed by the proposed localization algorithm. However, the proposed algorithm is validated against the following refinement algorithms from the literature.

1. *Non-linear Least-Squares Optimization* (NLO) - This is a widely-used optimization method, based on the Levenberg-Marquardt or Gauss-Newton algorithms. This method assumes that error is calculated as a non-linear function of the current state estimate. The objective is to iteratively make changes to the state to minimize the sum square error. To do this, the method linearizes the function about the current state estimate.

2. *Weighted Multidimensional Scaling* (WDS) - This method by de Leeuw [18] is the weighted analogue of MDS. However, unlike MDS, WDS cannot be solved by a simple eigendecomposition of a matrix. Rather, WDS uses an iterative solver called SMACOF to majorize a carefully-selected proxy function, which causes the WDS objective function to monotonically decrease.

1.3.3 Trajectory Reconstruction

In the *Trajectory Reconstruction* phase of the proposed algorithm, the graph embedding is used to correct segments of the dead reckoning trajectories. The phase is split up into two steps. In the *Drift Correction* step, which is discussed in Sec. 1.3.3.1, each sensor's trajectory is threaded through a subset of vertex coordinates from the embedded *Encounter Graph*. In the *Solution Projection* step, which is discussed in 1.3.3.2 , known correspondences between anchor positions and vertices in the *Encounter Graph* are used to project all sensors' trajectories into a global coordinate frame.

[2]The degree of a vertex is the sum of the weighting of its incident edges.

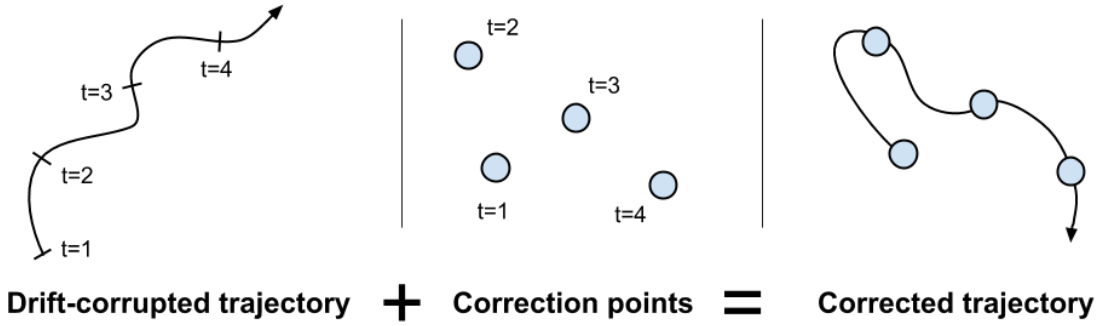


Figure 1.8: In the *Drift Correction* step vertices from the graph embedding are used to correct each sensor's dead reckoning trajectories.

1.3.3.1 Drift Correction

The objective of the correction step is to thread each sensor's *Inertial Trajectory* through a subset of the vertices in the realized graph. More specifically, a time-ordered sequence of vertices can be extracted from the *Encounter Graph*. This is essentially a low-resolution sample of sensor mobility. Temporally-adjacent pairs in this sequence are used to correct segments of the *Inertial Trajectory*. This is a *piecewise method*, because it operates on segments of the *Inertial Trajectory* in isolation, as illustrated in Fig. 1.8. Two piecewise correction methods are proposed:

1. *Linear Drift Correction* (LIN). This method was taken from Constandache et al. [14], and it assumes that inertial drift can be corrected by an offset and correction vector, which is added proportionally over time.

2. *Radial Drift Correction* (RAD) - This algorithm is a novel contribution of this thesis. RAD assumes that inertial drift can be corrected by an offset and scaled rotation, which is added proportionally over time.

1.3.3.2 Solution Projection

The objective of the *Solution Projection* step is to project the *Corrected Trajectories* from a local to a global reference frame, both in $\mathbb{E}^d$. This step is optional, as it requires at least $d+1$ anchors with algebraically independent coordinates.

Procrustes Analysis (see Cox and Cox [15]) is used to calculate a translation, rotation and scaling component that maps the local coordinates of the anchors to their corresponding coordinates in the global reference frame. This transformation

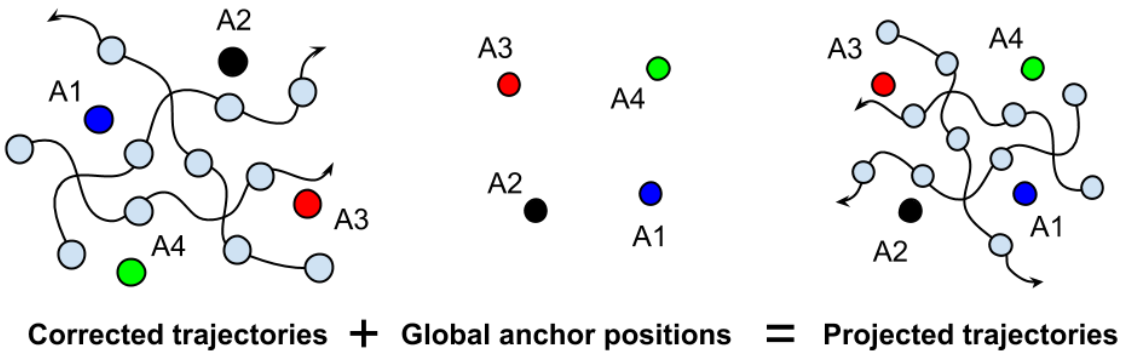


Figure 1.9: In the *Solution Projection* step the sensor trajectories are projected from a relative coordinate frame into a global coordinate frame, both in $\mathbb{E}^d$. The transformation calculation requires at least $(d+1)$ anchors with algebraically independent coordinates.

is applied to all points along all sensor trajectories. Fig. 1.9 illustrates the *Solution Projection* step, using four anchors.

1.4 Contributions of the thesis

The main contributions of this thesis are as follows:

1. The development of a offline, range-based localization algorithm. This algorithm separates localization into six modular steps - *Graph Construction*, *Rigidity Analysis*, *Graph Embedding*, *Solution Refinement*, *Drift Correction* and *Solution Projection*.

   (a) The *Graph Construction* step provides a method of converting inertial and mobility measurements into an encounter graph, which abstracts from low-level measurement detail to capture the broad relationship between sensor and anchor positions over time.

   (b) In the *Rigidity Analysis* step the encounter graph is analysed to determine whether a sufficient number of radio measurements exist to mitigate all sensor position ambiguity. Graph rigidity has been used in the localization literature, but primarily for static 2D localization. This is extended to the mobile case and, using recent graph rigidity results, 3D localizability is certified probabilistically in polynomial time.

   (c) The *Graph Embedding* step uses techniques from static sensor localization to determine graph vertex coordinates, given only pairwise range estimates.

The resulting vertices are then passed to an iterative refinement algorithm that uses measurement weightings to improve the solution.

(d) In the *Drift Correction* step a novel piecewise drift correction algorithm is proposed, called *Radial Drift Correction* (RAD). This is used to remap segments of an *Inertial Trajectory* to lie between known start and end points. This algorithm differs from a competing model in the literature, in that it models piecewise drift correction as a rotation and scale, as opposed to shear.

2. A simulation study to evaluate the proposed localization algorithm. The target application was cooperative pedestrian localization with foot-mounted inertial sensors. Sample traces were synthetically generated using error models calibrated from real measurements. The key findings were the following:

   (a) Through the removal of redundant constraints between sensors, graph rigidity theory provides a mechanism for certifying whether 2D and 3D range-based localization problems contain a sufficient number of measurements to adequately constrain relative sensor positions.

   (b) Existing graph realization algorithms used to perform static sensor localization cannot be used effectively to localize mobile sensors. The reason for this is that the complexity of graph realization algorithms scale at least quadratically with the number of graph vertices, which in turn increases linearly with duration. Simulations show that comparatively small problems yield graphs containing thousands of nodes within the first hour.

   (c) RAD outperforms LIN in cases where the global orientations of the sensors' dead reckoning trajectories are unknown. However, RAD causes piecewise discontinuities at encounter points, and does not constrain one degree of freedom in 3D. In a certain few cases where there no rotational correction component is required, RAD tends to scale segments of the trajectory disproportionately.

1.5 Publications

The list below contains published work that relates to the contents of this thesis, and is sorted in reverse chronological order.

1. Symington, A., & Trigoni, N. (2012, June). *Encounter Based Sensor Tracking.* In Proceedings of the thirteenth ACM international symposium on Mobile Ad Hoc Networking and Computing (pp. 15-24). ACM.

2. Symington, A., Waharte, S., Julier, S., & Trigoni, N. (2010, May). *Probabilistic Target Detection by Camera-equipped UAVs.* In Robotics and Automation (ICRA), 2010 IEEE International Conference on (pp. 4076-4081). IEEE.

The first publication describes how motion and radio measurements may be used to construct a graph, find a 2D embedding, and correct dead-reckoning solutions. Since publication the algorithm has been altered in the following ways:

1. The graph construction method has been optimized for performance.

2. Measurement uncertainty is now used to refine the graph embedding.

3. Sparse non-linear least squares optimization is used in place of graph realization..

4. Rigidity is certified in 3D probabilistically using recent results in graph rigidity theory.

The second publication describes a method of camera-based target tracking from an unmanned aerial vehicle. The proposed method carefully fuses measurements, taking into account how altitude affects precision and recall.

1.6 Structure of the thesis

This thesis is organised in the following way. Chapter 2 provides a background to this research, by contextualising the proposed algorithm with respect to the localization literature. The proposed algorithm is broadly decomposable into three phases – *Graph Construction*, *Graph Realization* and *Trajectory Reconstruction*. These three phases are discussed in detail from Chapter 3 to 5. Chapter 6 describes a simulation study that was carried out to compare the proposed localization algorithm to competing models from the literature. Then, Chapter 7 discusses simulation results, while Chapter 8 concludes this thesis, and summarises directions for future research.

Chapter 2

Background and Related Work

The objective of *localization* is to determine the positions – in two or three dimensional space – of a set of static or mobile devices, given a time-indexed series of measurements. The key distinction between *tracking* and localization is that, in tracking, the cooperation of the device being positioned is not necessarily required. The localization problem presents itself in many forms across a variety of academic disciplines, including sensor networking, robotics, bioinformatics, mathematics and geomatics. As a result, there is an inconsistent use of nomenclature. This thesis will adopt the terminology from sensor networking. The term *"sensor"* will be used to refer to the static or mobile device with a position that is unknown, and the term *"anchor"* will refer to a device with a position that is fixed and known *a priori*. In the robotics literature sensors and anchors are referred to as *robots* and *landmarks* (or *features*) respectively.

Localization is enabled by instrumenting relative motion directly (odometry) using *proprioceptive sensing*, or by relating the motion or position of the sensor to a known reference point (landmarking) using *extereoceptive sensing*. Tab 2.1 provides a list of the most commonly-used proprioceptive and extereoceptive sensor for localization. Although the algorithm proposed in this thesis is general, it will be experimentally tested in a cooperative pedestrian localization application. This application requires a sensor that operates unobtrusively and cost-effectively, and restrictions are therefore placed on cost, size and weight. This effectively limits the scope to small wireless radios or Micro Electro-Mechanical system (MEMS) sensors, such as barometers, accelerometers, gyroscopes and magnetometers.

This chapter is structured as follows. Sec. 2.1 discusses static localization from the perspective of the sensor networking literature. Sec. 2.2 continues with a discussion of mobile localization, from the perspective of the robotics literature. The contribution of this thesis is an offline, range-based mobile localization algorithm that borrows

Table 2.1: Categorisation of localization instrumentation

Sensor	Proprioceptive		Extereoceptive	
	Measurement	**Physical**	**Measurement**	**Physical**
Accelerometers	Acceleration	Acceleration	Geogravitation	Orientation
Gyroscopes	Angular velocity	Angular velocity		
Magnetometers			Geomagnetism	Orientation
			User magnetic field	Distance
Barometers			Barometric pressure	Altitude
Rotary encoders	Wheel odometry	Displacement		
Camera	Visual odometry	Pose change	Feature tracking	Distance
Laser scanner	Visual odometry	Pose change	Feature tracking	Distance
Radio			Angle of arrival	Orientation
			Time of arrival	Distance
			Proximity	Proximity
			Signal strength	Distance
Ultrasound			Time of Arrival	Distance

techniques from both research fields. Sec. 2.3 concludes with a survey of models specific to pedestrian localization.

2.1 Static localization

The static localization problem in sensor networking is concerned with finding the location of static sensors using measurements between sensors and anchors, or between peer sensors. Wireless radios may offer several different types of measurement, which are outlined in Sec. 2.1.1, and these measurements define which localization techniques can be used. Broadly speaking, the range of static localization algorithms can be divided into the following three main categories. *Single-hop* algorithms, discussed in Sec. 2.1.2, require a sufficient number of sensor-to-anchor measurements to localize each sensor. *Multi-hop* algorithms, discussed in Sec. 2.1.3, work on ad-hoc networks, and typically localize sensors iteratively; once a sensor is localized it becomes an anchor to its neighbours. Finally, Sec. 2.1.4 discusses *Cooperative* algorithms, which use all measurements simultaneously to arrive at a localization solution that minimizes some objective function, like measurement error.

2.1.1 Measurements

This section outlines popular radio measurements that are used for static localization. Received Signal Strength (RSS), which is discussed in Sec. 2.1.1.1, is a widely-used

measurement because it is implemented by most radio hardware. Other measurements, such as those discussed in Sec 2.1.1.2-2.1.1.5 Time of Arrival (TOA), Time Difference of Arrival (TDOA), Angle of Arrival (AOA) and Doppler Velocity (DV) have been shown to provide superior results to RSS, but require more specialised hardware, often increasing the size and cost of the sensor.

2.1.1.1 Received signal strength

Received signal strength (RSS) is a measurement of the energy spectral density at the receiver's radio, during reception of a frame. At the most basic level, RSS provides a measurement of *proximity*: whether two devices are sufficiently close so that they are able to communicate. In a *disc model* isotropic propagation is assumed; the receiver is assumed to be located within a fixed-radius sphere centred on the transmitter. Assuming that the transmitting power and any fixed losses are known, this energy is related to the environment between the transmitted and the receiver. In free space, RSS decreases approximately logarithmically with respect to distance. In cluttered environments reflection and absorption have a significant effect on RSS, and therefore the log-distance free space model generally cannot be used.

Under the assumption that sensors transmit beacon frames at a fixed rate, if two sensors don't exchange a beacon at a given point in time, they are unlikely to be close to each other. This *negative information* is derived from the fact that wireless channels fade, and being disconnected is correlated with being distant.

2.1.1.2 Time of arrival

Time of arrival (TOA) systems estimate the pairwise distance between two sensors by multiplying the *propagation time* of a signal by its fixed *propagation velocity*. For radio signals the propagation velocity in a vacuum is $c = 299,792,458$ kilometers per second. For radio systems a 10 nanosecond timing error thus equates to approximately 3 meters in positioning error. A one-way TOA system therefore requires the sender and receiver clocks to be synchronized to the nanosecond level, which is difficult to achieve in real-world deployments. The *Cricket Mote* by Priyantha et al. [72] is a one-way TOA system that uses the same principle, but with ultrasonic signals. Ultrasound travels at the 343.2 meters per second, which means that a 10 millisecond error equates to around a 3 meter ranging error. The device emits radio and ultrasonic signals simultaneously. The radio signal arrives almost instantaneously at the receiver. The time it takes for the ultrasound signal to arrive at the receiver is directly related to the distance separating the transmitter and the receiver.

In practice, the range of one-way TOA with ultrasound is extremely limited. The ultrasonic signals themselves do not propagate more than several tens of meters, and are significantly effected by clutter. As a workaround to the time synchronization problem in one-way ranging, two different systems are used. In *two-way* TOA the propagation time is averaged over a round-trip, preventing the two device clocks from requiring synchronization. However, in order for this system to function, high resolution clocks and an accurate knowledge of the delays imposed by both the transmitter and receiver radios are still required. In addition, two-way TOA ranging is sensitive to multi-path, and normally coupled with spread-spectrum modulation schemes. Ubisense[1] and NanoLOC[2] are two TOF ranging systems that use ultra-wideband (UWB) and chirp spread spectrum (CSS) modulation respectively.

2.1.1.3 Time difference of arrival

In *Time difference of arrival* (TDOA) systems there is a single transmitter and multiple, time-synchronized receivers. The emitted signal arrives at the receivers at slightly different times, because of the varying distances between the transmitter and each receiver. Each measurement falls on a hyperbola or hyperboloid in 2D and 3D respectively, and the emitter position is solved for using multilateration . In problems where time-synchronized anchors are available, it is possible to carry out TDoA.

2.1.1.4 Angle of arrival

Angle of arrival (AOA) systems use measurements of the angle, or bearing, between devices. This is achieved by measuring the phase difference between the same signal arriving at multiple antennas. Multiple antenna arrays not only increase sensor cost, also its size; antennas must be spaced at half or quarter wavelength which, at 2.4GHz, means that the footprint of the antenna array is over 3×3 centimeters. A suitable application for angle of arrival is mobile phone positioning, where the bulk of the system's complexity lies in large, expensive and highly-sensitive base stations with multiple receiving elements. Each base station determines the angle of signal arrival and, by intersection, is able to calculate user position.

2.1.1.5 Doppler-velocity

Doppler velocity (DV) enables a source station to measure the velocity of a moving target. A control signal is transmitted at a known frequency from a source device.

[1] http://www.ubisense.net
[2] http://www.nanotron.com

This signal then reflects of the surface of a target device, where it is finally received and processed by the source device. The Doppler effect causes the received frequency to shift either up or down, depending on the relative velocities of the two devices.

2.1.2 Single-hop positioning

In *Single-hop localization* each sensor is localized with respect to anchors in one-hop communication range. The *Global Positioning System* (GPS) is an example of a single-hop localization from TOA measurements. GPS is the most widely-adopted type of Global Navigation Satellite System (GNSS), although other services such as Galileo and GLONASS do exist. In the GPS system all satellites (anchors) continually transmit a pseudo-random sequence. By comparing the phase to a identical local copy of the signal, a receiver (sensor) can calculate the propagation delay of the signal, which is in turn related to the distance between the receiver and each satellite [33]. As a result of their frequency, GNSS signals cannot penetrate solid structures, and so it is difficult to obtain a fix indoors. Moreover, GPS modules require a significant quantity of energy to continually search for satellite signals.

2.1.2.1 Direct methods

Fig. 2.1 illustrates the three geometric methods of combining radio measurements to localize sensors. Sensors, anchors and measurements are shown as light circles, black circles and lines respectively. Although shown in two dimensions, all concepts extend to higher dimensions with additional anchor information. *Triangulation* uses AOA measurements. The position of the sensor is given by the intersection of the lines formed by the anchor positions and measured angles. In *Trilateration* distances to anchors are estimated, either using TOA or RSS. The position of the sensor is given by the intersection of the three circles, centered on the anchors. In *Multilateration* the TDOA of a single sensor's signal at three or more non-collinear anchor positions is needed. The position solution is found by the intersection of the hyperbolic curves.

In some cases the position of the sensor may be *over-constrained*. For example, in 2D lateration a distance to three anchors is required to localize the sensor, but four or more distances may in reality be known. In such a case a maximum likelihood estimator is used to determine the position of the sensor, given some knowledge of measurement error. In other cases an insufficient number of measurements is obtained, and the sensors position is said to be *under-constrained*.

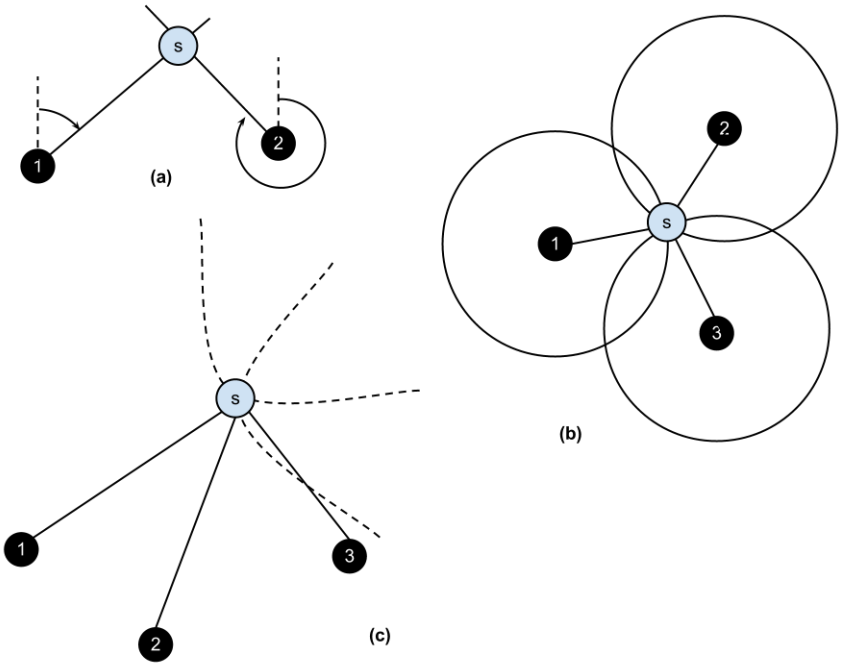


Figure 2.1: Examples of (a) triangulation, (b) trilateration, and (c) multilateration

2.1.2.2 Interferometric methods

Interferometric methods require a minimum of four devices. Two of these devices simultaneously transmit on nearby frequencies. The two signals' interfere, and the interference pattern is then measured by two independent receivers. The difference between the measurements is then related to some underlying physical quantity, such as position or velocity, given knowledge of the signal frequencies but without the need for high precision timing.

Maróti et al. [62] showed how to calculate phase offset using off-the-shelf Zigbee radios, by transmitting low-frequency sine waves and using RSS as a proxy measurement. The author showed that centimeter accuracy was attainable, but that larger distances yield larger errors. Subsequently Kusy et al. [54] showed that, by measuring Doppler shift rather than phase offset, one can simultaneously calculate both position and velocity.

2.1.2.3 Fingerprinting methods

Fingerprinting methods, such as RADAR by Bahl and Padmanabhan [4], exploit multi-path error to improve localization. A *fingerprint* is a vector of RSS values recorded at a particular point in space. Localization is carried out in the following two

phases. In the *training phase* many RSS measurements are recorded at observation points, and they are used to build a *radio map* of the environment. In the *localization phase* inference is performed over the radio map in order to determine the most likely position of the sensor, given a vector of RSS measurements. Although it takes a significant amount of time to generate and maintain the radio map, this is currently the state of the art in terrestrial anchor-based localization.

Notable fingerprinting methods include HORUS by Youssef and Agrawala [96] and the EZ Localization System by Chintalapudi et al. [11], which Rai and Chintalapudi [73] have shown to yield median errors in the 3m range. The EZ algorithm was designed as a computationally efficient algorithm. Ferris et al. [26] presented a fingerprinting method that used a Gaussian Process [74] to model the received signal strength as a function of multidimensional continuous space. Ferris et al. [27] extended their Gaussian Process RSS fingerprinting approach [26] to operate in the absence of a training set. This was achieved using Gaussian Process Latent Variable Models [56], a technique in which the sensor position is considered unobservable. Similar RSS fingerprints imply similar locations, and this fact is used to close the trajectory loop when a sensor returns to a previous position.

2.1.3 Multi-hop positioning

The overarching objective of *Multi-hop localization* (also referred to as *ad-hoc localization*) is to provide a localization mechanism for scenarios where sensors may not be in single-hop communication range of a sufficient number of anchors for *Single-hop localization* to take place. *Multi-hop localization* techniques may be broken down into three approaches – *Iterative*, *Distance Vector* and *Cooperative*. Fig. 2.2 shows an example problem, which will be used to illustrate these three approaches. In the diagram, sensors and anchors are depicted by squares and circles, and a black edge indicates that a measurement exists between two devices. A node that is currently being localized is coloured red, along with the measurement edges that it uses.

The top row of the Fig. 2.2 shows two rounds of the *Iterative localization* algorithm by Savvides et al. [78]. In the first round, sensors that are neighbours with at least three anchors localize themselves using trilateration. These sensors are then treated as anchors in subsequent rounds. Any localization error therefore compounds, causing positioning accuracy to decrease as subsequent iterations are carried out.

The bottom-left image of Fig. 2.2 shows the same problem as before, using a *Distance Vector* approach. Niculescu and Nath [67] show that a sensor's position may be calculated using lateration in conjunction with *network distance* – either the

summed hop count or shortest path cost. In other words, the shortest path to the nearest $(d+1)$ anchors is used to calculate a position in $\mathbb{E}^d$. In the case of error-free measurements, network distance is always equal to or larger than the true distance. The accuracy of this approach therefore depends on the relative device positions and, in particular, how well the network distance approximates true distance.

2.1.4 Cooperative positioning

Cooperative localization algorithms differ from *Iterative* and *Distance Vector* approaches in the sense that all measurements are used simultaneously to determine the *relative* positions of sensors and anchors, in some *local coordinate frame*. This approach therefore has the potential to solve a much richer variety of localization problems. For example, some applications preclude the use of anchors, because they are too dangerous to install, like in hostile environments, or too expensive in terms of cost or energy to maintain, like in wildlife tracking. Once the relative positions are found, if anchors are available then the solution is then projected into a global reference frame using a transformation that is derived from the correspondences between the positions of the anchors in the local and global coordinate frames. Patwari and Hero [69] provide a survey of cooperative localization algorithms.

2.1.4.1 Stress methods

Each algorithm in this variety of cooperative localization has some notion of the *stress* that results from a given estimate of vertex coordinates. Shang et al. [83] define stress to be the sum squared error between the observed and predicted distances. They then show that *Multidimensional Scaling* (MDS) can be used to determine the vertex coordinates that minimize stress, assuming a complete distance matrix is given. The complete distance matrix can be estimated using a shortest-path algorithm, such as *Floyd-Warshall*. Shang and Ruml [82] later extended this method to distribute its computation and perform better on irregularly shaped graphs.

2.1.4.2 Spectral methods

Broxton et al. [7] defined stress to be the predicted distances between vertices, weighted by the negative exponent of the observed distances. The advantage with this approach is that unknown edges may be weighted zero, and thus have no effect on the solution. The author showed that a method by Koren [51] could be used to determine the assignment of coordinates to vertices that minimizes the stress function.

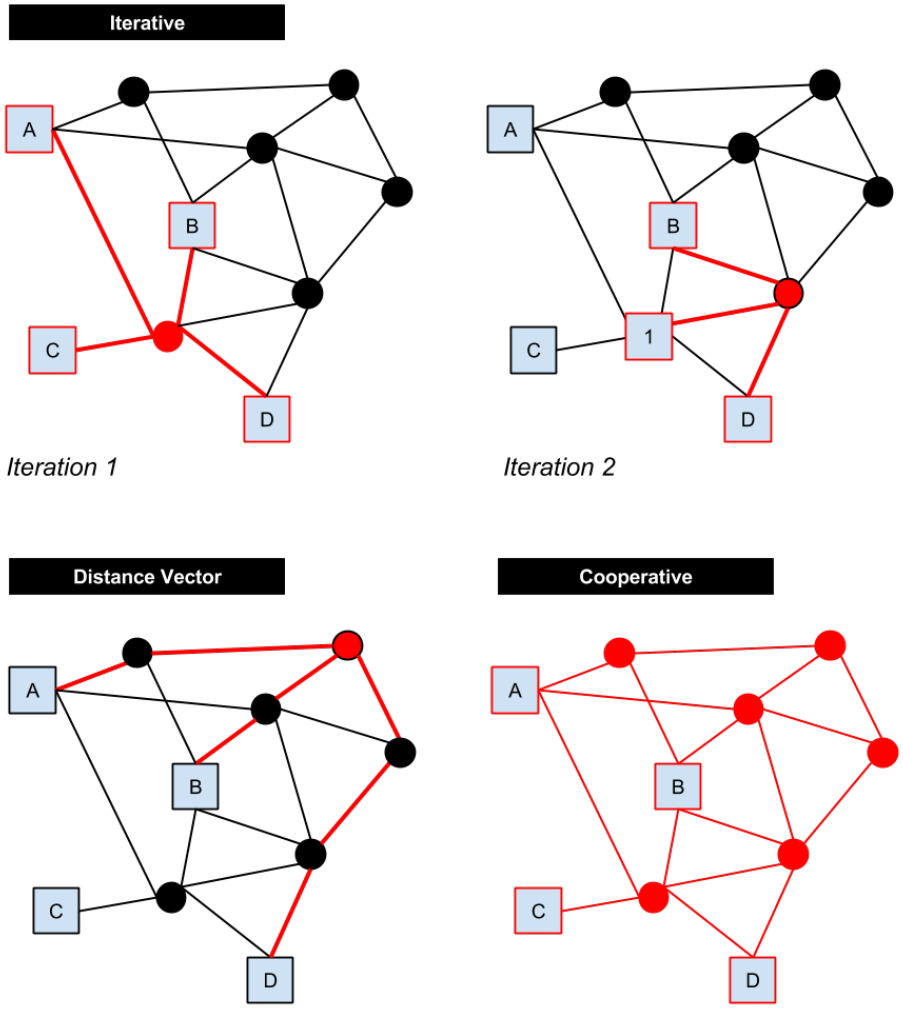


Figure 2.2: The diagrams above illustrate the differences between the *Iterative*, *Distance Vector* and *Cooperative* static localization approaches. Circles and squares denote sensors and anchors respectively, while edges denote distance measurements. Red has been used to indicate which of these devices and measurements are being used in the current iteration to carry out localization. In the first step of *Iterative* localization (top left) sensors connected to a sufficient number of anchors localize themselves. In the next iteration (top-right) the localized sensors are then treated as anchors, and used to localize peers. This process continues until all sensors are localized, or no further sensors can be localized. In *Distance-Vector* approaches (bottom-left) the distance to each anchor is measured as the shortest multi-hop distance, or *network distance*. These distances are combined to then localize the sensor. In *Cooperative* approaches (bottom-right) all measurements are used simultaneously to arrive at a localization solution.

2.1.4.3 Semidefinite methods

The problem of assigning coordinates to vertices in such a way that satisfies known edge lengths is non-convex (there may be local minima). However, it is possible to solve a relaxation of the problem by an *Interior Point Algorithm* [94]. Biswas and Ye [5] applied this technique to the problem of sensor localization with noisy range estimates. Subsequently, Kim et al. [50] expanded on this work, exploiting matrix sparsity to reduce the amount of time required to obtain a solution. Recent results by Javanmard and Montanari [43] provide analytical performance bounds for SDP localization where nodes are distributed randomly according to a uniform distribution within a bounded d-dimensional hypercube.

2.2 Mobile localization

Motivated by autonomous robot navigation, the vast majority of research regarding mobile localization originates from the robotics community. Thrun et al. [89] proposed that mobile localization can be categorised by whether (i) the problem is solved offline or online, (ii) the environment is static or dynamic, (iii) sensors can control their own mobility, and (iv) the problem contains one or many sensors. This thesis is concerned with localization problems in a static environment with known anchor positions and multiple sensors that cannot control their own mobility. In the literature, these problems are typically referred to as *localization* and *cooperative (multi-robot) localization* for the single or multiple sensor case respectively. Localization is a special case of a more general problem called *Simultaneous Localization and Mapping* (SLAM) in which the anchor positions are also unknown.

The remainder of this section is structured as follows. Sec. 2.2.1.1 discusses *recursive localization methods*, which is an online solution that processing measurements as they arrive. Then, Sec. 2.2.2 discusses *full localization methods*. These methods differ from recursive methods in the sense that the solution is solved for by minimizing some cost function, taking into account all measurements over time. Finally, in Sec. 2.2.3 we narrow the scope to range-only methods, which assume that points on a map and trajectory are associated only by range estimates.

2.2.1 Recursive localization

Recursive localization is typically used for online localization, because of its memory and computational efficiency. A recursive localization algorithm maintains a current

state estimate, which propagates forward in time (dynamically or kinematically) and then corrects using measurements. Parametric filters, which are discussed in Sec. 2.2.1.1, encode a single hypothesis as a state estimate, with an associated error distribution. Non-parametric filters, which are discussed in 2.2.1.2 take a Monte-Carlo approach. This allows multiple state hypotheses and arbitrary (including multi-modal) error distributions to be maintained within the filter, at a computational cost.

2.2.1.1 Parametric filters

The *Kalman Filter* provides a method of recursively filtering measurements to estimate the state of a system. The state and its uncertainty are expressed as a mean vector and covariance matrix. In the prediction step of a KF, the state is updated using a *process model*. This model assumes that the relationship between the previous and next state is linear. Similarly, in the correction step, the KF models the relationship between the current state and measurements with a *measurement model*, which it again assumes is linear. Under the assumption that measurement error is Gaussian distributed, the KF is an optimal state estimator.

In reality the system and measurement models used by the KF are rarely linear. In the *Extended Kalman Filter* (EKF) the prediction and correction models are linearized about the current state estimate. This approximation means that it is not an optimal filter, and its stability is highly dependent on the non-linearity of the process and correction functions. Another drawback of the KF and EKF is that all elements of the covariance matrix must be updated whenever a single anchor is observed..

Notable extensions to the EKF include the *Compressed EKF (CEKF)* by Guivant and Nebot [34], which seeks to reduce the computational complexity of EKF in situations where a significantly large number of anchors are present in the state. The CEKF uses the fact that large sequences of filter updates are independent of a significant number of elements in the state. The CEKF therefore separates and operates on a smaller, local state, which is eventually merged into the global state. The *Constrained EKF* by Ungarala et al. [91] introduces a third *constraint step* to the EKF, in which the state is corrected based on a set of linear constraints. This is particular useful when holonomic constraints preclude certain state combinations.

The *Information Filter* (IF) is similar to the KF, with the notable exception that it propagates the inverse of the covariance matrix, which is referred to as the *information matrix*. Aulinas et al. [3] argue that this representation is advantageous for three reasons. Firstly, the filter is more accurate that the KF, because the update step involves simple summations. Secondly, the filter is more stable than the KF. Thirdly,

it is faster than the KF when the dimensionality of the state is large. Like the KF, the IF has a non-linear counterpart, the Extended Information Filter (EIF). However, in practice, the KF is chosen over the IF, because matrix inversion is required in the correction step of the filter to recover the state.

In an EKF the state error is approximated by a multivariate Gaussian, and propagated analytically through the linearized system and measurement models. Julier and Uhlmann [45] showed that this introduces significant error, and proposed an alternate filtering strategy called the *Unscented Kalman Filter* (UKF). The UKF propagates a carefully-chosen set of points, called *sigma points*, through the process and measurement models in order to capture the true posterior mean and covariance.

2.2.1.2 Non-parametric filters

A *Particle Filter* (PF) provides a method of recursively estimating the state of a system, by iteratively refining a large set of hypotheses. Each hypothesis (particle) is initially assigned values that are drawn from a prior distribution. At each time step the PF propagates the particles forward in time, and a random perturbation is added to their values to account for process uncertainty. The weight of each particle is then calculated using measurements. The particles are resampled using these weights and used to initialize the next time step. In Hol et al. [39] several different resampling strategies are discussed.

The primary advantage with the PF is that it does not prescribe how the state estimate or measurements are distributed. PFs are therefore able to effectively represent multi-modal error distributions. The primary drawback with PFs is that they are computationally expensive to implement, and that the number of particles and iterations must to be selected to balance time and error performance. As the dimensionality of the particles increases, so more particles are required to effectively sample the volume. In the localization problem the particle dimensionality increases with additional sensors, while in the SLAM problem, the dimensionality increases also as more anchors are added. The *particle deprivation problem* is a drawback with PFs where no particles exist in the vicinity of the true state estimate. This is typically caused by a few high likelihood particles biasing the resampling method. To mitigate this problem, Thrun et al. [89] suggest adding a fixed number of random particles to each iteration.

The FastSLAM algorithm by Montemerlo et al. [65] uses *Rao-Blackwellization* to reduce the complexity of the PF for SLAM problems. Even for small examples the dimensionality of each particle is too high, making implementing a PF impossible.

FastSLAM works by factoring out the anchor positions from the sensor trajectory. This factorization implies that if one knew the sensor trajectory, each anchor position is independent the other anchor positions (they are conditioned solely on the sensor trajectory). FastSLAM estimates the sensor trajectory using a particle filter. The conditional independence between anchors allows a separate parametric filter (EKF or UKF) to be used to estimate each anchor position. The FastSLAM 2.0 algorithm by Roller et al. [77] improves upon this method by taking into account measurements when resampling particles.

2.2.2 Full localization

The full localization problem differs from the recursive localization problem in the sense that measurements over all time are used at once to obtain a localization solution. The full localization problem is a special case of *Full SLAM* problem, where the anchor positions are known *a priori*. This increases the computational complexity of the problem, but allows for greater estimation accuracy. As such, these approaches are typically used offline. The *de facto* approach is non-linear least squares optimization, which is discussed in Sec. 2.2.2.1. There are a great number of extensions to this baseline approach, which are discussed in Sec. 2.2.2.2.

2.2.2.1 Non-linear least-squares optimization

Durrant-Whyte and Bailey [23] assert that, from a theoretical standpoint, *Full SLAM,* and hence *Full Localization* are solved problems. The classic solution to the problem is referred to as *Bundle Adjustment (BA)* or *Structure From Motion (SFM)* in the fields of photogrammetry and computer vision respectively. In BA the problem is formulated as a graph, where vertices represent unknown states and edges represent measurements. Each measurement relates one or more states by some non-linear function. Iterative *stochastic gradient descent* algorithms, such as Guass-Newton or Levenberg-Marquardt [63], are then used to solve for the unknown state values. This non-linear least-squares optimization technique, however, is sensitive to initialization and convergence to local minima. In the case where measurement error is assumed Gaussian, BA is an optimal least squares solver. Lu and Milios [59] were the first authors to apply this technique to SLAM problem.

2.2.2.2 Extensions to the baseline method

Although solved theoretically, BA has many practical implementation issues. Notably, the time required by baseline BA scales cubically with the state space, and therefore a significant amount of research focuses lowering this computational complexity, while retaining the estimation accuracy. The ultimate goal being to incrementally solve the full localization problem in constant time. The remainder of this section summarizes variants of BA that work towards achieving this goal. Triggs and McLauchlan [90] survey such improvements, many of which have been incorporated into *GraphSLAM* by Thrun and Montemerlo [88], its multi-robot extension by Kim et al. [49], and the *General Framework for Graph Optimization* by Kummerle et al. [53].

As a result of finite sensing ranges, the graphical model formed by the relationship between states and measurements is inherently sparse. Lourakis and Argyros [57] proposed a generic BA implementation, called *Sparse Bundle Adjustment* (SBA), which exploits this natural structure to lower the computational complexity of solving the problem. SBA partitions the sensor trajectories from the anchor positions, and an optimized version of Levenberg-Maquardt with a sparse implementation of Cholesky decomposition is used to solve the normal equations, and iteratively refine an initial estimate. **?**] also replaced Levenberg-Marquardt with a conjugate gradient solver, designed to perform well with large, sparse matrices.

The initial state estimate has a significant effect on the performance of BA. If this estimate lies far from the ground truth, it may take a significant number of iterations for the solver to converge, depending on the nonlinearity of the problem. Olson et al. [68] proposed a novel non-linear optimization algorithm to rapidly find the "macroscopic solution" to the BA problem, which approximates the final solution – the overall solution is accurate, but the detail at the microscopic level requires further refinement. This model is typically used to bootstrapping the algorithm with an initial estimate. The algorithm is a modification to stochastic gradient descent, in which an alternative state space is used. Each iteration of this approximation algorithm has a run-time complexity of $O(\log k)$ for a problem containing k states.

Many authors have increased performance on large problems using sub-maps. Notable work includes *Atlas* by Bosse and Newman [6], *Tectonic SAM* by **?**] and *Treemap* by Frese [30]. In *Atlas* the map is divided into sub-maps, each of which maintains its own coordinate system. Features between maps are related by spatial transforms. Tectonic SAM provides an exact solution to the SLAM problem by dividing the map with a k-way cut. Treemap provides a good approximation to the full solution by representing the high-dimensional optimization problem as a binary

tree of smaller optimization problems, which effectively partitions independent parts of the global map.

Frese et al. [31] showed that the *information form* of a SLAM problem can yield several optimizations that improve performance. In *Square Root SAM by* Dellaert and Kaess [19] the information matrix is factorized by Cholesky decomposition and and updated incrementally. The *Incremental Smoothing and Mapping* (iSAM) algorithm by Kaess et al. [46] performs incremental QR factorization using Givens rotations. One drawback with this approach is that it periodically requires a batch relinearization, which is computationally expensive. Subsequently, Kaess et al. [47] improved iSAM to perform continual "fluid" relinearization, and further exploit the sparse structure of the information matrix.

Finally, in *Adaptive Relative Bundle Adjustment* by Sibley et al. [84] the performance of the solver is increased by relaxing the requirement that states be optimized in a single Euclidean reference frame. Rather, optimization takes places in a metric-space defined by a connected Riemannian manifold. This technique affords a constant-time incremental solution to the full SLAM problem. However, converting the relative solution to one that is embedded in Euclidean space is computationally expensive, and cannot be performed incrementally.

2.2.3 Range-only methods

Robotic platforms frequently use cameras or laser range-finders for autonomous navigation. These sensors typically provide a significant number of measurements that when coupled with a robust data association technique, such as DDF-SAM by Cunningham et al. [17], provide a strong relationship between positions in the map and positions along the sensor trajectories. However, cameras or laser range-finders cannot be used in certain applications, either because of size restrictions or because they cannot work in the operating environment.

Newman and Leonard [66] identified autonomous underwater navigation as one such problem. More specifically, they proposed using TOF measurements from anchors with unknown positions to localize an underwater vehicle. The authors propose a non-linear optimization method that solves for the vehicle trajectory and beacon positions, under the assumption that the vehicle travels at a constant velocity. Olson et al. [68] extended this approach to multiple vehicles, incorporated measurement outlier rejection (through spectral graph partitioning), and showed how to use a consensus approach to initializing the anchor positions.

Range-only SLAM has also been applied to localizing robots within a sensor network. Menegatti et al. [64] proposed an EKF for tracking the 2D position of a single robot. RSS measurements were passed through a log-distance path loss model to relate robot positions to landmarks. As was the case with underwater navigation, this approach is sensitive to the initial anchor position estimate. Thus, rather than randomly initializing the anchor position estimates, the authors use trilateration. Djugash et al. [22] proposed a multi-robot extension to this approach, which acknowledges that the anchors connectivity may not be sufficiently high to support trilateration. Their proposed algorithm waits until the graph created by the anchors and vehicle odometry has no vertex with degree less than four, and then uses multidimensional scaling is used to determine the relative anchor positions.

2.3 Indoor Pedestrian Localization

The overarching goal of indoor pedestrian localization is to estimate the path taken by a walking human, by combining locally-sensed measurements. While millimeter-precision localization is possible with motion capture systems, such as Vicon's T-Series[3], this method requires that the human subject wears calibration markers, and maintains line-of-sight with a costly array of infrared cameras. The focus of the research in this thesis is on localizing with measurements from less invasive devices, such as watches or mobile phones, to record opportunistic radio and motion measurements, which can then be processed online or offline to localize the subjects.

The recent surge in indoor pedestrian localization research has been driven by the reduction in size and cost of MEMS inertial sensors. Such sensors are integrated into smart phones, fitness devices, watches or other personal electronic devices, and can be used to measure physical quantities that relate to motion. The challenges with using such devices are the following:

1. **Resource constraints** - A target platform may have sensing, communication, computation or memory. Sensing constraints are usually because the cost or energy requirements of a particular sensor is too high. Communication is normally constrained by the wireless medium or by energy budgets. Computation and memory constraints are imposed by the on-board microcontroller.

2. **Noise** - Automotive grade inertial sensors are typically used for PDR, because they are small and inexpensive. However, this grade of inertial sensor tends to be

[3] http://www.vicon.com

Table 2.2: Grade Classification of Inertial Measurement Units

Grade	Cost	Drift / hour	Applications
Marine	$1,000,000	75 m	Submarines, ships, military aircraft
Navigation	$100,000	1.5 km	Commercial aircraft
Tactical	$10,000	19 km	Missiles, military UAVs
Industry	$1000	190 km	Industrial robotics, commercial UAVs
Automotive	$10	7900 km	Mobile phones, pedestrian navigation

perturbed by a significant quantity of error, which biases position quadratically with time.

3. **Activity** - In many cases, and notably with smart phones, the devices may not be rigidly fixed to the person being tracked. This decouples the measured motion from the subject. Kern et al. [48] show that accurate classification of activity from inertial measurements remains an open problem.

Sec. 2.3.1 discusses the grading of Inertial Measurement Units (IMUs), and the sources of measurement error. Then, Sec. 2.3.2 shows how IMUs are used to achieve pedestrian dead reckoning. The measurement error compounds with time to cause position drift. Finally, Sec. 2.3.3 discusses offline, range-based approaches for integrating dead reckoning with radio measurements in order to mitigate this drift.

2.3.1 Inertial measurement units

Although IMU accuracy-cost grading is not standardized, Groves [33] proposes the approximate grading shown in Tab. 2.2. The availability, cost and size of any IMU above *automotive grade* (those used in mobile phones and video games) precludes widespread adoption. Automotive grade IMUs use MEMS sensors that are rigidly fixed to the subject being tracked. This is often referred to as a *strapdown* configuration.

MEMS sensors are made up of components up to 100 micrometers in size. The five most commonly-used MEMS devices for dead reckoning are:

1. *Accelerometers* measure net acceleration, which is inclusive of both gravity and specific force. With additional domain knowledge it is sometimes possible to separate the two components from each other.

2. *Gyroscopes* measure angular velocity, which is typically integrated with respect to time to update orientation.

3. *Magnetometers* measure the magnetic flux density. This sensor is typically used to measure Earth's magnetic field, and hence the devices bearing from magnetic north. Such measurements are however perturbed by local iron deposits.

4. *Barometric pressure sensors* detect small atmospheric pressure fluctuations. Over short intervals the atmospheric pressure changes as a function of altitude. Over longer periods pressure changes response to weather.

5. *Temperature sensors* are used to measure the ambient temperature. The small electromechanical components in MEMS devices expand and contract in response to environmental temperature, which causes measurement drift. Temperature sensors are therefore used to compensate for this drift.

Each MEMS device outputs a voltage that corresponds to some physical quantity. The devices above are normally bundled together into an *Inertial Measurement Unit* (IMU). An *Inertial Tracking Model,* referred to more commonly as an *Inertial Navigation System* (INS), fuses measurements from the above devices to estimate position, orientation, acceleration, etc. By convention, *attitude* usually refers to pitch and roll, while heading (or bearing) refers to yaw. The term *orientation* is used to describe all degrees of rotational freedom in either 2D or 3D space.

An *Attitude and Heading Reference System* (AHRS) is a class of INS that produces a drift-free estimate of orientation from motion measurements. This is made possible by the fact that Earth's gravity and magnetic field, detectable by an accelerometer and magnetometer respectively, form an absolute reference frame against which orientation can be corrected. Unlike orientation, position cannot be estimated without external reference points. Thus, position is tracked by a method called *dead reckoning*, in which the INS accumulates displacement estimates. In such a system error also accumulates, causing the position estimate to *drift* further from the truth with time.

2.3.1.1 Error perturbation

Error is introduced to a navigation solution by both the navigation processor and the sensing instrumentation. The navigation processor is limited by finite numeric precision, and the assumptions made by the estimation filter.

Although MEMS sensors perform internal error compensation, this is normally the responsibility of the navigation processor. Each measurement suffers error that is in fact composed of many smaller errors, each having a type and source. Most navigation processors compensate for the following four measurements types.

- *Bias* - This error is independent of the value being measured, and is the dominant form of error in all inertial instrumentation. Typical automotive grade MEMS accelerometers and gyroscopes may be biased in the order of $0.1^{m}/_{s^2}$ and $5 \times 10^{-4}{}^{\circ}/_{s}$ respectively.

- *Scaling* - This error represents a divergence between the measured value (the kinematic unit obtained by multiplying an electrical measurement) and the truthful value. Automotive grade sensors have scaling errors as high as 0.01.

- *Cross-coupling* - This error is introduced by a misalignment of the instrumentation's axes with respect to the body frame, caused by fabrication limits. Like scale factor errors, cross coupling errors may be as high as 0.01 for automotive grade sensors, and may be different for positive and negative measurements.

- *Noise* - This error originates from a variety of sources. Firstly, automotive grade gyroscopes and accelerometers have a low signal-to-noise ratio. As the period over which sampling takes place – or *averaging time* – increases, so the noise deviation decreases; there is thus a trade-off between sampling resolution and noise. Secondly, certain gyroscopes use vibrating masses and are therefore resonant frequencies, such as those from a motor, may cause correlated noise. Thirdly, an analogue to digital (ADC) converter is used to quantize voltage readings, thereby adding noise.

Each of the four error types above can manifest in the four different ways below.

- *Fixed contribution* - This is a constant amount of error that is unique to each device. Most manufacturers run a calibration routine for each device to internally correct for this error.

- *Run-to-run* - This is a quantity of error that changes over long periods, but remains fixed for short periods. i.e. for the duration of a single experiment. In many cases it is possible to mitigate this error with an initialization routine.

- *In-run* - This is error that changes during a single experiment, and it is typically the most difficult to correct. In some cases the in-run variation can be offset by integrating with other sensing instrumentation.

- *Temperature-dependent* - This is error which is correlated with environmental temperature. This occurs because the small mechanical devices within the MEMS sensors expand and contract with changes in temperature.

An INS will correct measurements for known fixed contribution bias, scaling and cross-coupling errors, before passing them into the filter. An advanced INS may also perform temperature compensation. In-run error must be modelled within the filter state itself, increasing the computational cost of obtaining a navigation solution. As such, normally only gyroscope and accelerometer in-run bias is compensated, as it forms the largest component of the remaining error. However, military-grade INSs may model many other error sources in their filter's state.

2.3.2 Pedestrian dead-reckoning

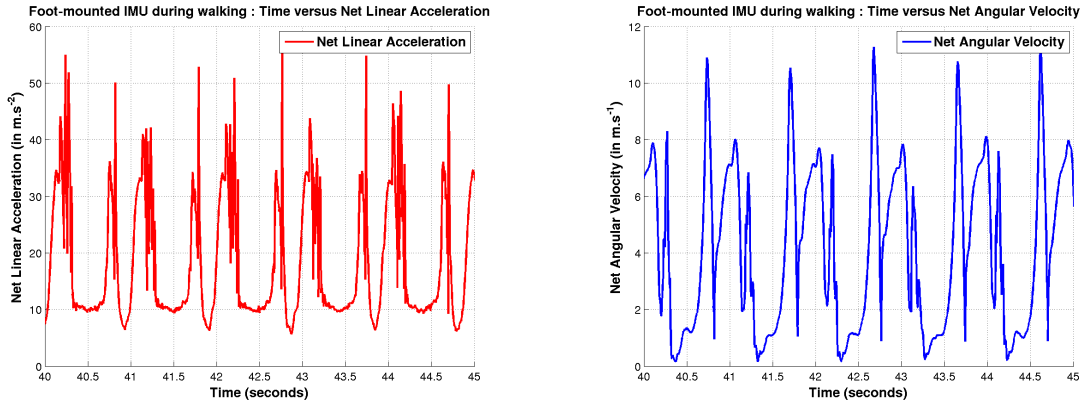
Pedestrian Dead Reckoning (PDR) is the process by which the position of a pedestrian, with respect to some starting point, is estimated using only locally-measured motion. Step detection is a fundamental requirement of all PDR implementations, and so it is discussed in Sec. 2.3.2.1. Harle [37] surveyed the indoor pedestrian localization literature, and proposed that PDR algorithms be divided into *Inertial Navigation Systems* (INS) and *Step and Heading Systems* (SHS). The former, which is discussed in Sec. 2.3.2.2 maintains a full 3D trajectory solution at a resolution defined by the sampling rate of the inertial sensors. The latter, which discussed in 2.3.2.3, integrate estimates of distance with bearing or attitude to perform dead reckoning at a step-by-step resolution.

2.3.2.1 Step detection

Pedometers are devices worn on the body, which detect and count the number of steps taken. Harle [37] showed that steps can be detected using a variety of different sensors, including pressure, mechanical switches, ultrasound, piezoelectric, or electromyography sensors. However, MEMS sensors are typically used, because they are less invasive, reliable, smaller and do not fatigue.

The step count is multiplied by an estimate of step length to determine the aggregate distance traveled. Groves [33] showed that an individuals stride length changes as a function of gender, height, frequency, body weight, terrain inclination, fatigue and load. However, simpler models that use only the first two or three of these parameters tend to work well in practice. Foot-mounted sensors are favourable, because they are most sensitive to step-triggered acceleration. However, in practice one cannot feasibly instrument the foot. Nevertheless, the *Nike+*[4] sensor is an example of a

[4]http://www.nikeplus.com



(a) Net linear acceleration

(b) Net angular velocity

Figure 2.3: Measured motion of the foot during regular walking

foot-mounted pedometer that is calibrated by the user over time to personalise step length.

Intuitively, when the foot hits the ground, its net acceleration remains constant and equal to gravity. Figure 2.3 shows plots of a foot's net acceleration and net angular velocity during walking, as a function of time. At times 40.5, 41.5, etc. the net acceleration plot in Fig. 2.3a is lower-bounded by the earth's gravity. This represents the point at which the foot was planted on the ground (no specific force) meaning that total acceleration was equal to gravity. Note also that these points correspond directly to the near-zero net angular velocity events in Fig. 2.3b. Groves [33] observed that the specific force measured at the torso exhibits a double-peaked oscillatory pattern, and that step events can be detected by evaluating zero crossings.

2.3.2.2 Inertial navigation systems

An *Inertial Navigation System* (INS) fuses motion measurements from a strapdown IMU to estimate a 3D pose, comprising of a position and orientation. A full INS requires at least a triaxial gyroscope and triaxial accelerometer. In an INS orientation is first updated by integrating angular velocity measurements with respect to time. This orientation is then used to rotate the linear acceleration measurements into an inertial reference frame. Gravity is subtracted from the resulting inertial-frame acceleration, and the resulting *specific force* is then integrated twice with respect to time to update position.

The accuracy of an open-loop INS is a function of the orientation and acceleration error. Small orientation errors are amplified by gravity times the tilt degree, and

subsequently integrated twice, causing the displacement error to grow quadratically with time. As a result, an IMU with an orientation error of 0.02 degrees will produce a position that drifts about 3.4 meters in 10 seconds. Typical automotive and tactical grade IMU errors are an order of magnitude larger than this, making this approach infeasible for tracking over long periods. Orientation error, which is predominantly caused by in-run gyroscope bias, can be mitigated through fusion with different sensors. Notably, if the IMU also contains a triaxial magnetic sensor, then the Earth's magnetic and gravitational fields provide measurable axes against which orientation can be corrected [80, 60]. Collin et al. [12] showed that a tactical-grade ring laser gyroscopes can be used as an alternative to MEMS gyroscopes, but this technology remains too expensive to be feasible.

On striking the ground a human foot is approximately stationary with respect to the inertial reference frame for a brief moment, and hence its velocity must be approximately zero. If steps can be detected, this fact can be used to periodically correct the linear velocity estimate. This *Zero Velocity Update* (ZUPT) mechanism [29] has the result of limiting the period over which an open-loop double-integration of acceleration can occur to the arc shown in Fig. 2.4. During the stride phase the position error typically grows to between 1 and 3 centimeters. However, if the step event is missed, this error continues to grow cubically until the next step is detected. Skog et al. [86] critically compare three different ZUPT detection algorithms. Their results suggest that angular velocity is a key measurement to maximizing step detection accuracy.

2.3.2.3 Step and heading systems

A *Step and Heading System* (SHS) estimates the position of a pedestrian at a step-by-step granularity. In a SHS an attitude and heading reference system is used to estimate the bearing, or heading, of the pedestrian. A step detector is then used to identify step events. Step length is then calculated, using domain knowledge and measurements. A pose estimate (x_t, y_t, θ_t) for the pedestrian at a step event t is calculated using the previous estimate $(x_{t-1}, y_{t-1}, \theta_{t-1})$ in conjunction with step length z_l and orientation change z_θ measurements using Eqn 2.1 - Eqn 2.3.

$$\theta_t = \theta_{t-1} + z_\theta \tag{2.1}$$

$$x_t = x_{t-1} + z_l cos\theta_t \tag{2.2}$$

$$y_t = y_{t-1} + z_l sin\theta_t \tag{2.3}$$

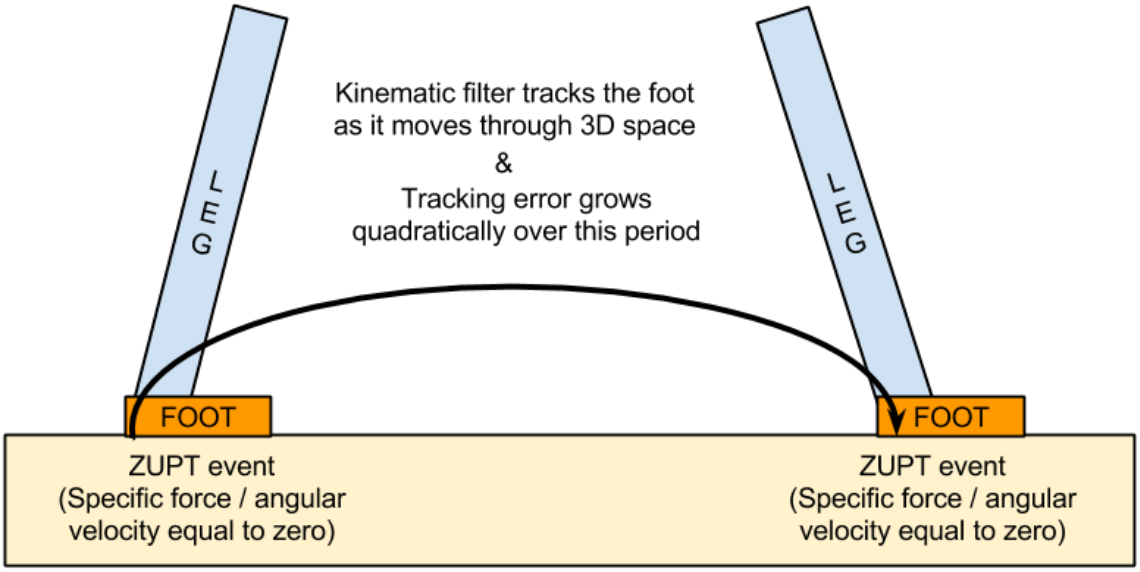


Figure 2.4: When the foot strikes the ground, its net angular velocity is approximately zero, and detectable by a gyroscope. This triggers the filter to reset velocity, meaning that an open loop double-integration of linear acceleration occurs only when the foot is swinging between step events.

Heading is usually calculated in a similar way to an INS, i.e. by integrating angular velocity measurements with respect to time. When gyroscopes are not available, orientation can be crudely estimated from accelerometer and magnetometer measurements. In this case it is impossible to properly separate specific force from gravity, and so assumptions must be made about the dynamics of the pedestrian.

Pratama and Hidayat [71] show that adult males and females have a typical stride length of 0.413 and 0.415 respectively, times their height. Therefore, step length is often assumed constant for each individual. However, Weinberg [92] showed that there is in fact variance in the walking speed of each individual, and proposed a method that estimates stride length to within 8% of the true value. This method relates the stride length to the absolute displacement of the hip, which can be measured from a waist-mounted device. Yang and Li [95] proposed another model that relates step frequency to step length, resulting in length estimates with an error of 5.6%.

2.3.3 Fusion methods

As discussed in the previous section, pure PDR methods result in cumulative error that causes position to drift. The focus of this section will be on pedestrian localization methods that fuse PDR with other measurements, in an effort to improve

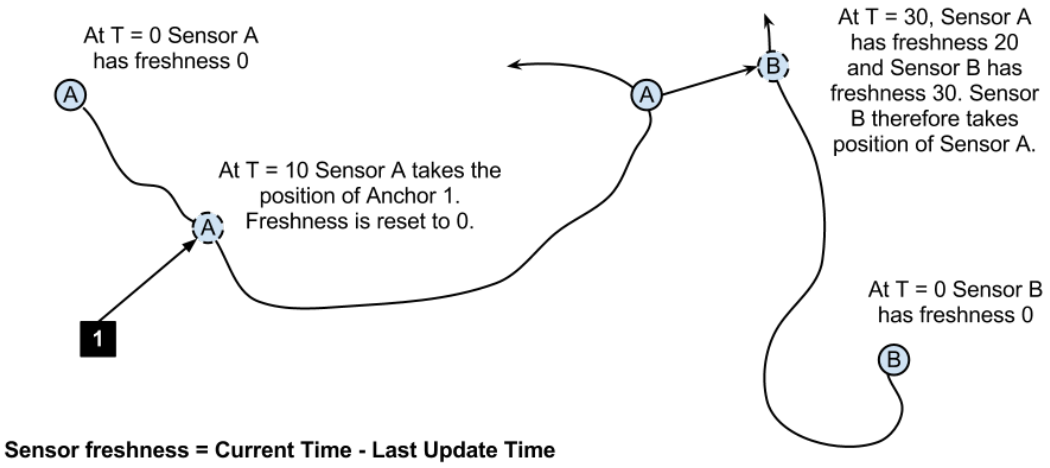


Figure 2.5: Directed diffusion tracking using proximity only

long-term localization accuracy.

Directed diffusion is a method of controlling the flow of information across an ad-hoc network. Constandache et al. [14] proposed an online, cooperative pedestrian localization algorithm that uses directed diffusion to control position updates between mobile sensors. Each sensor uses SHS to perform PDR. When two sensors encounter each other, they calculate which of the two has a *fresher* position estimate; the freshness of each sensor's position is simply the last time that the position was updated by an anchor. When an encounter takes place, the less-fresh sensor adopts the fresher position, and uses linear drift correction to correct its trajectory from the last encounter position to the current point in time.

Directed diffusion is illustrated in Fig. 2.5. At time zero, Sensor A and Sensor B (both denoted by circles) begin with a location freshness factor of zero. Then, at time $T = 10$ Sensor A encounters Anchor 1 (denoted by a square). As a result of rule (2) above, Sensor A assumes the position of Anchor 1 and sets its own freshness to zero. Then, at time $T = 30$, Sensor A and B encounter one another. Sensor A is the dominating sensor and, therefore, by rule (3), Sensor B assumes Sensor A's position and a freshness of 20. Since range is not taken into account, this algorithm can only perform well when the radio sensing range is very short.

Krach and Robertson [52] proposed a cascaded filter for PDR, which is divided into an upper and lower half. The lower half of the filter runs a full EKF for PDR. The INS uses ZUPTs to clamp drift by identifying footsteps The upper half of the filter runs at a frequency equal to the step frequency. On each step event, the displacement

from the previous step is reduced to a bearing and distance measurement. This measurement is then passed into the upper half of the filter to update the particle values. Map information is integrated to the filter by lowering weights of particles that violate constraints.

Woodman and Harle [93] proposed a similar cascaded filter, which also integrates WiFi measurements in addition to map constraints. Their approach uses an occupancy grid to represent the state space of all possible positions. A training phase identifies the visible access points at each grid location. Particles that violate map constraints are weighted zero. The likelihood or weight of each particle is also derived from the similarity of the measured number of access points to the training set, given the proposed particle location. The number of particles is also adapted in each iteration in order to balance the error and time performance of the filter.

The particle filter by Faragher [25] estimates the pedestrian's latitude, longitude, step length and compass bias. The method uses a foot-mounted sensor and accelerometer thresholding to detect step events. When a step event occurs, particles are propagated by the step length through the unbiased compass bearing, and a small amount of noise is added. The key to this model is that when two similar RSS fingerprints are measured, it implies similar location, and this event is used to "close the loop" in the sensor trajectory. Corrections may also occur if an opportunistic GNSS measurement is received. The authors propose an initialization period, where measured displacements from a GNSS system are used to calibrate step length (an element of the particle filter state) to the individual being tracked.

FootSLAM by Angermann and Robertson [1] represents 2D sensor position as a probabilistic transition model over a hexagonal grid. The authors claim to exploit human perception in a manner that is unproven from a theoretical standpoint. A Rao-Blackwellized particle filter is used to identify the most likely map (state transition model over the hexagonal representation) and sensor trajectory. FeetSLAM by Robertson et al. [76] is an extension to FootSLAM, in which the learned maps from peers are used as priors to the. PlaceSLAM by Robertson et al. [75], and WiSLAM byBruno and Robertson [8] extends FootSLAM by integrating extereoceptive measurements in the form of proximity (RFID) and range (RSS).

2.4 Discussion

This chapter began with a discussion of static localization using radio measurements, from the perspective of the sensor networking literature. The focus then turned to the

robotics literature, and mobile localization from odometry and range measurements was surveyed. Following this, the scope was narrowed to pedestrian localization, a special case of the localization problem in which inertial and radio measurements are used to track human subjects.

The range-only localization algorithm proposed in this thesis blends ideas from both robotics and sensor networking. It casts the mobile sensor localization problem as the graph localization problem: assigning coordinates to vertices in such a way that agrees with observed distance estimates, which are derived from radio and motion sensing. Firstly, this representation is compact, and therefore it is computationally more efficient to work with, compared to a pose graph containing *all* measurements. Secondly, theory from sensor networking can be used to evaluate whether this smaller graph is uniquely localizable, without requiring the vertex coordinates. The advantage being that one is able to discard problems with inherent ambiguity, which no algorithm can solve. Finally, those problems with provably unique solutions may be solved using graph embedding algorithms. The resulting coordinates are passed to piecewise correction algorithms to obtain a fast, but low-resolution, trajectory for each sensor.

The closest work from the literature is by Djugash et al. [22]. This model fuses odometry with range measurements to obtain a pose graph, which is used to ultimately initialize the anchor positions in an EKF. This method is overly-conservative in the sense that it approximates rigidity by waiting until each vertex in the graph has a degree greater than four, and then resolving flip ambiguities using a handedness from the original trajectory. In contrast, the approach proposed in this thesis certifies rigidity directly on any encounter graph that it is given.

The next chapter discusses the *Graph Construction* step of the proposed localization algorithm. More specifically, it shows how inertial and radio measurements from a foot-mounted sensor are converted into a graph that describes sensor mobility.

Chapter 3

Graph Construction

This chapter discusses the first phase of the proposed cooperative localization algorithm. This phase is called *Graph Construction*, and it is illustrated by Fig. 3.1. Graph Construction depends on the existence of domain-specific *Motion* and *Radio* models. Both models are domain-specific, but essentially reduce sensor measurements to a distance, or range, estimate between two points in space. Sec. 3.1 discusses the Graph Construction method, showing how measurements are used to build a graphical model, called the *Encounter Graph*. This graph captures the connectivity between sensors and anchors over space and time, but not the fine-grained measurements. *Motion* and *Radio* models specific to pedestrian localization – the application used to validate the proposed model – are discussed in Sec. 3.2 and Sec. 3.3 respectively.

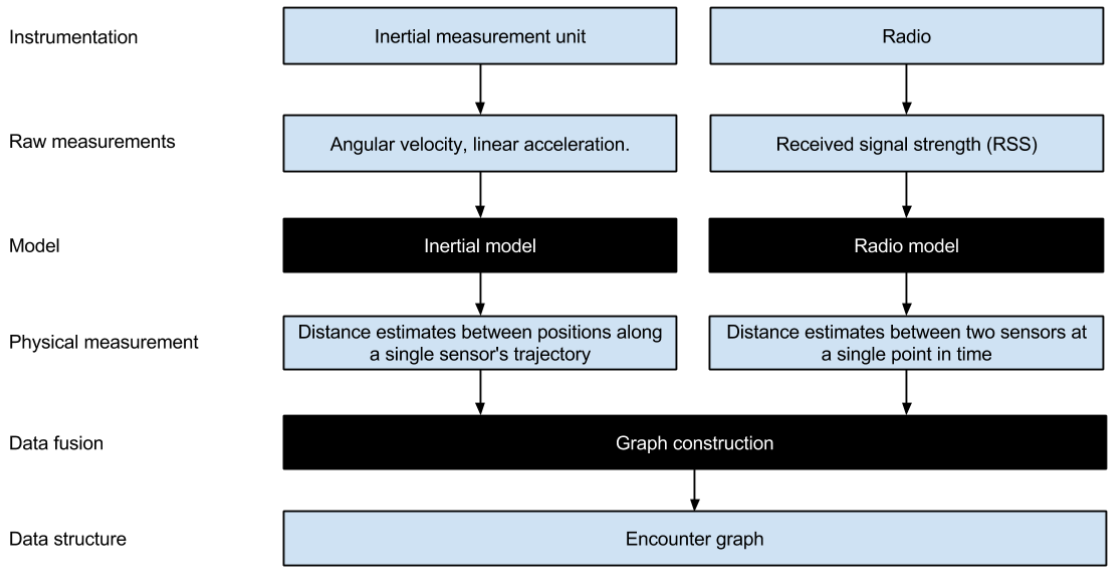


Figure 3.1: How the encounter graph is constructed

3.1 Graph Construction

Consider a localization problem containing n sensors and m anchors. Assume that an experiment is run from 0 to T seconds, and that each sensor records its local motion and radio encounters. At the end of the experiment the data is collected and post-processed by the *Graph Construction* step to build an *Encounter Graph*. In order to do so, the existence of the following two models is assumed:

1. A Motion *Model* $f_m : (x_{1:T}, P_{1:T}, s, t) \rightarrow \mathcal{N}(\mu, \sigma^2)$ that samples a sensor trajectory estimate $x_{1:T}$ and uncertainty $P_{1:T}$ at timestamps $s < t$ as inputs, estimates the distance between the two positions as a Gaussian-distributed variable with mean μ and variance σ^2.

2. A *Radio Model* $f_r : (\gamma_{1:k}) \rightarrow \mathcal{N}(\mu, \sigma^2)$ that, converts a variable-length set of received signal strengths $\gamma_{1:k} = \{\gamma_1, ..., \gamma_k\}$ (in dBm) into a Gaussian-distributed distance with mean μ and variance σ^2.

The *Encounter Graph* is an abstract representation of a cooperative localization problem. Each vertex in the graph represents either the fixed location of an anchor, or the location of a sensor at a specific point in time. Each edge associates two vertices by a Gaussian-distributed Euclidean distance estimate. Edges that relate two vertices belonging to a single sensor are referred to as *Inertial Edges*, while edges that relate vertices belonging to two different devices are referred to as *Radio Edges*. Finally, edges that relate the static position of two anchors are referred to as *Prior Edges*.

The remaining subsections discuss how the *Encounter Graph* is constructed and optimized. Sec. 3.1.1 begins by describing the fundamental idea, which is illustrated with an example toy problem. Sec. 3.1.2 then outlines an extension to this method, which quantizes time and bounds the size of the *Encounter Graph*. Finally, Sec. 3.1.3 argues that vertices not incident to any *Radio Edges* are redundant, and then shows how to remove them from the graph.

3.1.1 Fundamental Construction Method

An earlier paper [87] described the basic construction method, which uses only distances and not variances. This method is explained with the example problem shown in Fig. 3.2, which contains two sensors (A and B) and two anchors (1 and 2). On

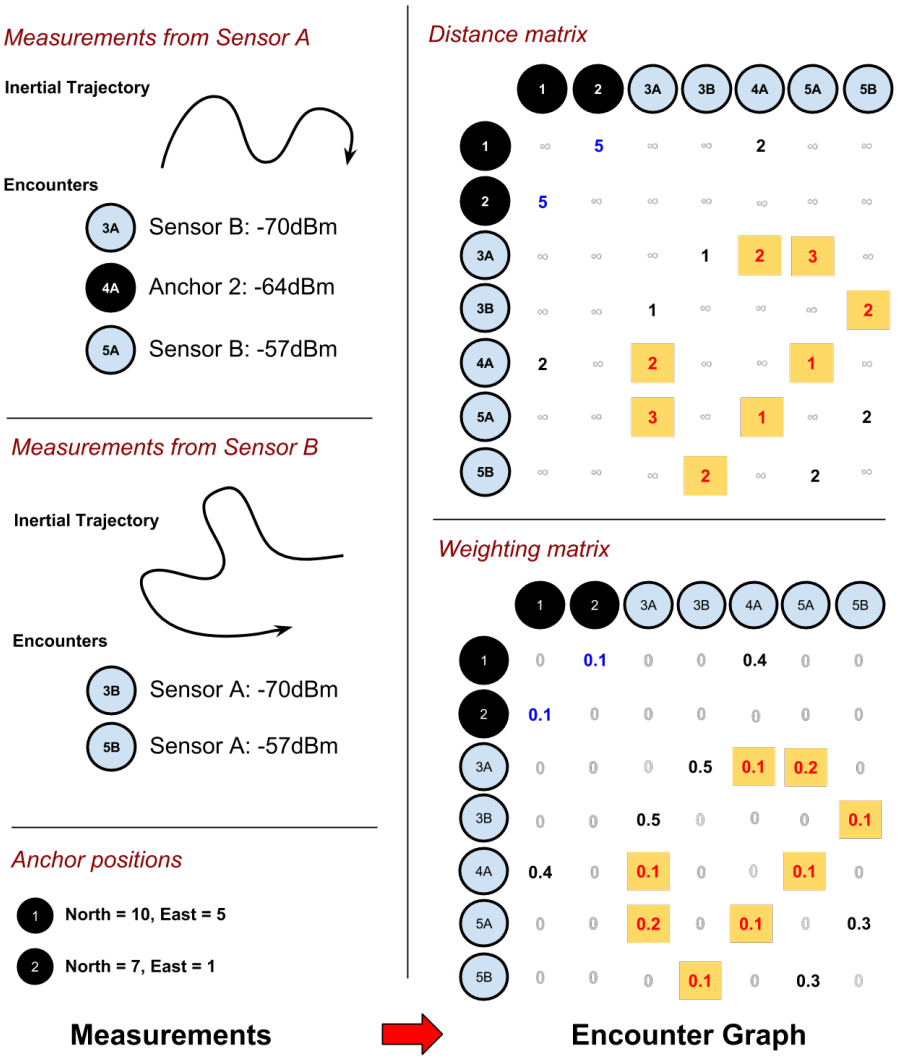


Figure 3.2: The fundamental graph construction method

the left of the figure are the measurements that act as input data to the construction algorithm, which comprise of inertial trajectories and radio encounters for each sensor, as well as the known positions of all anchors. On the right of the image are the distance and weighting matrices, which are the result of the construction algorithm. In the example graph there are seven vertices and eight distance measurements (off-diagonal elements associating an edge with a distance and variance). The edges are undirected and so both matrices are symmetric.

The connected graph has itself not been drawn, because that would imply that the 2D coordinates of each vertex is known; only the anchor positions are known. The step-by-step construction process for the example problem is provided below:

- At time $t = 0$ a vertex is added for each anchor, representing its static position. In our toy example, these vertices are denoted by black circles marked 1 and 2. Since the anchor positions are known *a priori*, the graph is fully-connected with known edge lengths. These vertices are then connected with an edge having a distance of 5 and a variance of 0.1. This event adds blue off-diagonal elements to the distance and variance matrices, associating vertices (1) and (2).

- At time $t = 3$ *Sensor A* and *Sensor B* encounter each other with an estimated distance of 1 and variance of 0.5. These values are obtained from passing a radio measurement through the *Radio Model*. Two vertices, indicated by circles 3A and 3B, are then added to the *Encounter Graph*, each representing the location of a sensor at the time when the encounter took place. This event adds black off-diagonal elements to the distance and variance matrices, associating vertices (3A) and (3B).

- At time $t = 4$ *Sensor B* encounters *Anchor 1* with a distance estimate of 2 and variance 0.4 (again, the *Radio Model* is used). Since the anchor's position is static, only one vertex is added, which represents the position of *Sensor B* at the time of encounter. Once again, this adds black off-diagonal elements to the distance and variance matrices.

- At time $t = 5$ *Sensor A* and *Sensor B* encounter one another again, this time with an estimated distance of 2 and variance 0.3. Two vertices, indicated by circles 5A and 5B, are added to the *Encounter Graph*. As before, this adds black off-diagonal elements to the distance and variance matrices.

- Finally, the inertial information is used to fully-connect the vertices along a single sensor's trajectory. For example, vertices 3A, 4A and 5A represent the Sensor A's position at times $t = 3$, $t = 4$ and $t = 5$ respectively. The *Motion Model* is used to estimate the distances between all points along the sensor's trajectory. The process is repeated for *Sensor B*. The resulting information is shown by red off-diagonal elements in the matrices.

One subtle point to note is that the construction method adds two extra vertices for each sensor, representing its start and end position. These are omitted from the graph for clarity. These vertices will ensure that *Drift Correction* (discussed in Sec. 5.1) is able to correct the entire trajectory. Omitting these vertices causes corrections ambiguities about the first and last encounter points.

Consider a localization problem with n sensors and m anchors. Let k_s and k_a be the number of sensor-to-sensor and sensor-to-anchor encounters respectively, and c_i be the total number of encounters recorded by sensor i. The number of *Encounter Graph* vertices v and edges e are given by Eqn 3.1 and 3.2 respectively.

$$v = \underbrace{m}_{\text{anchors}} + \underbrace{\overbrace{k_a + 2k_s}^{\text{encounters}} + \overbrace{2n}^{\text{endpoints}}}_{\text{sensors}} \tag{3.1}$$

$$e = \underbrace{\left(\frac{m(m-1)}{2}\right)}_{\text{anchor}} + \underbrace{\sum_{i=1}^{n} \frac{c_i(c_i-1)+2}{2}}_{\text{inertial}} + \underbrace{\left(k_a + k_s\right)}_{\text{radio}} \tag{3.2}$$

3.1.2 Time Quantization Method

In real sensor network deployments, radio beacons are typically transmitted at a fixed rate. If sensors are collocated for reasonably long periods of time, this can result in a great number of redundant encounters being recorded. Eqn 3.1 showed that, for the basic construction method, both the number of graph vertices and edges scale as a function of the number of encounters. The time complexity of localization is intimately related to the size of the Encounter Graph. Hence, the marginal amount of information provided by a fast beaconing rate is outweighed by the additional computation that is required to perform localization. This motivates the need to bound the size of the graph.

The vertex merging algorithm seeks to bound the size of the graph, by fusing temporally-adjacent encounters between the same two devices. This approach operates in the following way. Time is divided into T discrete time units. Then, $m + nT$

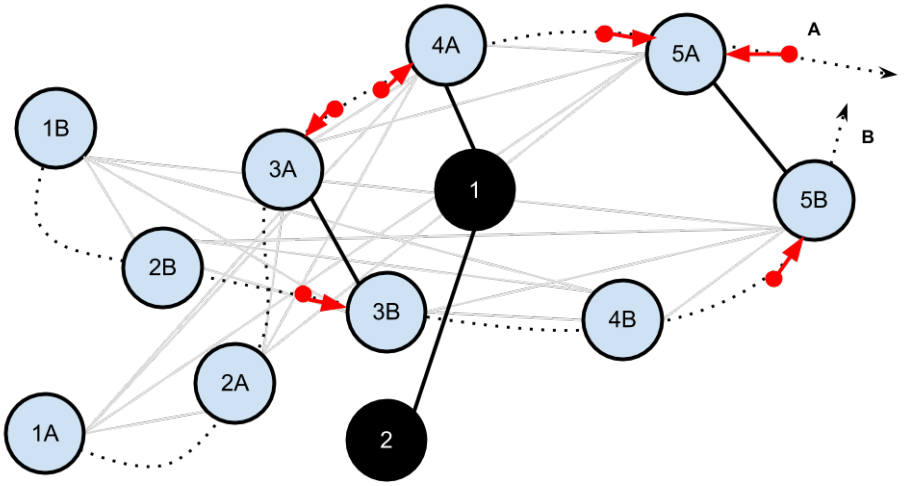


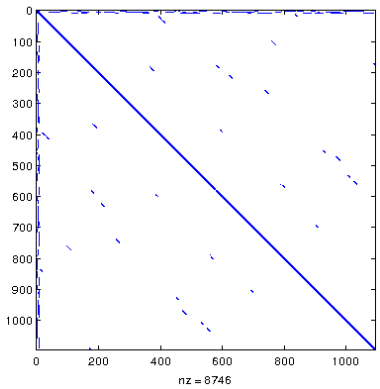
Figure 3.3: In the Time Quantization Method, encounters are mapped to their nearest discrete-time vertex. In the figure above, the exact encounter position is shown by a red dot. In all cases, this dot is shifted to its nearest vertex. In some cases, multiple encounters map to the same vertex, which is why the *Radio Model* requires a method of fusing multiple measurements together.

vertices are inserted into the *Encounter Graph* – one for each of the m anchors, and one for each of the n sensors at every point in time. Each radio encounter is then mapped to its nearest discrete time unit. Encounters between the same two devices that map to the same discrete time unit are merged into a single distance estimate using the *Radio Model* (see Fig. 3.3).

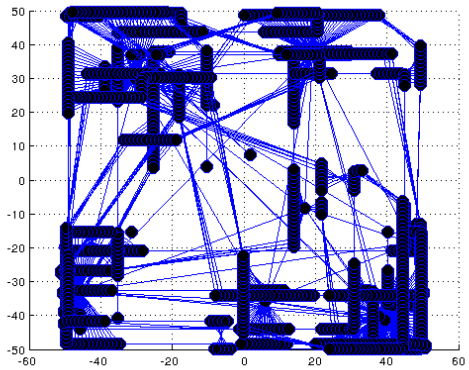
Quantizing time and merging encounters leads to a fixed number of vertices in the *Encounter Graph*. Retaining the notation from the previous subsection, and assuming that time is divided into T discrete time units, the new number of vertices v and edges e in the *Encounter Graph* is given by Eqn 3.3 and 3.4 respectively. As before, c_i is the number of encounters made by sensor i.

$$v = \underbrace{m}_{\text{anchors}} + \underbrace{nT}_{\text{sensors}} \tag{3.3}$$

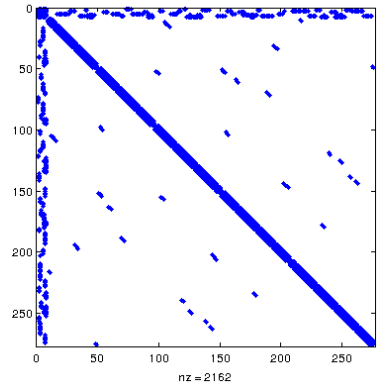
$$e = \underbrace{\left(\frac{m(m-1)}{2}\right)}_{\text{anchor}} + \sum_{i=1}^{n} \left(\underbrace{\frac{c_i(c_i-1)}{2}}_{\text{inertial}} + \underbrace{c_i}_{\text{radio}} \right) \tag{3.4}$$



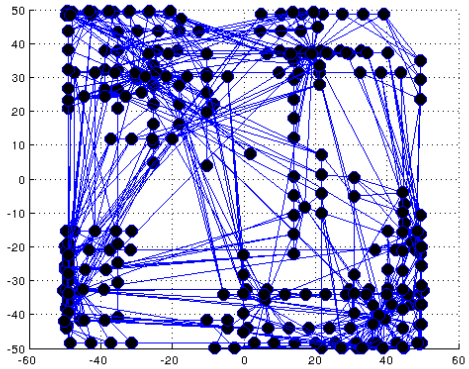
(a) Distance matrix before optimizing the graph



(b) Graph vertices before optimizing the graph



(c) Distance matrix after optimizing the graph



(d) Graph vertices after optimizing the graph

Figure 3.4: Reducing the size of the encounter graph by vertex pruning

3.1.3 Vertex Pruning Method

In the Time Quantization Method, selecting the value for T is not straightforward: If T is selected too low, then radio encounters are merged unnecessarily. If T is selected too high, then very few encounters are merged and the resulting graph is very large. Let k_s and k_a be the number of sensor-to-sensor and sensor-to-anchor encounters. If $k_a + 2k_s < nT$ then the Time Quantized Graph is at least as big as the basic graph, because the majority of the nt vertices are *redundant* (not incident to any radio edge).

This section introduces the Vertex Pruning Method, which removes redundant vertices, and hence reduces the size of the *Encounter Graph*. To achieve this, any rows and columns from the edge matrix which contain only inertial measurements are removed. The only exception to this rule is that any row or column that corresponds to the first or last vertex of a sensor's trajectory is not removed. This is so that *Drift Correction* can be performed over the entire trajectory. Vertex pruning is particularly useful in cases where the trajectory sampling rate is high, but the number of encounters is low. Fig. 3.4 illustrates that by using a vertex pruning with a 10 second resolution reduces the number of vertices in the graph by three-quarters.

3.2 Motion model

The motion model used in the experimental component of this thesis requires a pedestrian dead reckoning (PDR) algorithm to filter inertial measurements and obtain a position solution. This PDR algorithm assumes that inertial measurements are recorded from a 6DoF strapdown IMU that is rigidly fixed to a pedestrian's foot. It is necessary to fix the IMU to the foot, as any position above the heel becomes less stationary during a step event. Angular velocity is integrated once with respect to time to estimate orientation. Acceleration is similarly integrated twice through the estimated orientation to update position. This open-loop double-integration introduces error that compounds quadratically with time. Domain knowledge is used to periodically reset the velocity, so that error only propagates open-loop between steps.

The remainder of this section is structured as follows. Sec. 3.2.1 discusses the general design of the EKF used in this thesis, which comprises of a non-linear *prediction* and *correction* steps. These two steps are discussed in greater detail in Sec. 3.2.1.1 and Sec. 3.2.1.2 respectively. Sec. 3.2.1.3 concludes by showing how both steps linearized and used recursively by the filter.

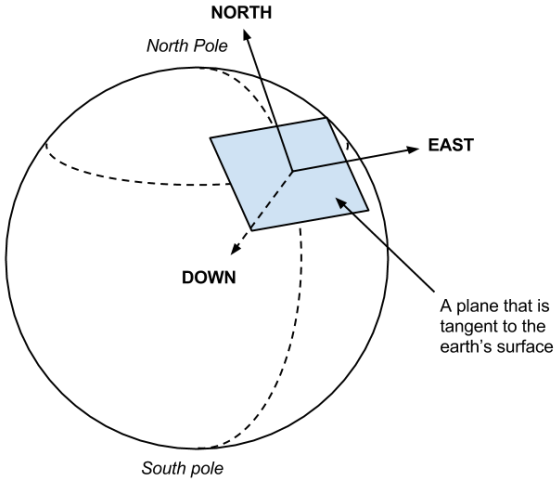


Figure 3.5: The noeth-east-down coordinate frame

3.2.1 Filter design

The PDR algorithm uses two distinct coordinate frames - the *body frame* and the *reference frame*. The body frame is fixed with respect to the foot being tracked. The reference frame is the North East Down (NED) local tangent plane that is shown in Fig. 3.5. A requirement of the NED coordinate frame that the pedestrian being tracked must be sufficiently far from the Earth's north and south poles, where the geogravitational and geomagnetic vectors are no longer strongly orthogonal. For clarity, Greek letters are used to denote variables in the body frame, while roman letters are used to denote variables in the reference frame.

The 15-state EKF encodes kinematic quantities and sensor biases as state variables. The state vector $x_t = \begin{bmatrix} p_t & v_t & r_t & \beta_t^\omega & \beta_t^\alpha \end{bmatrix}^T$ includes the reference-frame position p_t, velocity v_t, Euler orientation r_t, body-frame gyroscope bias β_t^ω and body-frame accelerometer bias β_t^α at some discrete time step t. Estimate uncertainty is encoded in by a 15×15 covariance matrix P_t. The filter assumes an origin-centered starting position and zero biases, e.g. $x_0 = \mathbf{0}$. The external degrees of freedom will ultimately be adjusted by the localization algorithm. Convergence occurs with initial covariance matrix P_0 with $1e-3$ values for kinematic quantities, and$1e-6$ values for biases.

One important aspect of any kinematic filter is the choice of angle representation. While convenient, Euler angles have well-known singularities that prevent them from being an analytically convenient mechanism for modelling rotations. Other popular representations include, amongst others, quaternions and rotation matrices. The EKF filter in this thesis adopts the latter approach. More specifically, orientation is

modelled in the state as an Euler angle. However, in the prediction and correction step of the EKF this representation is first converted to a corresponding rotation matrix. Rotations are performed using this representation, and then mapped back to Euler angles. Importantly, the rotation matrix C_t representing the orientation of the IMU at time C_t is preserved at the end of each time step, and used in the beginning of the next time step.

3.2.1.1 Prediction step

Let δt be the time period in seconds betweens the last state x_{t-1} and the current state x_t. A non-linear process model f is used to update the state estimate on the arrival of new inertial measurements from the IMU.

The IMU measurements are comprised of three-axis linear accelerations α_t and three-axis angular velocities ω_t, which are concatenated into a control vector $u_t = \begin{bmatrix} \alpha_t & \omega_t \end{bmatrix}^T$. The skew symmetric operator $S(w)$ shown in Eqn 3.5 is used to linearize the angular velocity ω

$$S(\omega) = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \tag{3.5}$$

The well-known Cayley transform shown in Eqn 3.6 uses the skew symmetric matrix to find a rotation $R(\omega, \delta t)$. Critically, this transform makes the assumption that the angle change is small, which is typically the case for PDR filters running at rates over 64Hz.

$$R(\omega, \delta t) \quad = \quad (I + S(\omega)\frac{\delta t}{2}).(I - S(\omega)\frac{\delta t}{2})^{-1} \tag{3.6}$$

Eqn 3.8 shows the process model equations that update x_{t-1} to x_t given some time interval δt. It is assumed that the state is perturbed by a Gaussian-distributed noise vector $w_t = \begin{bmatrix} w_\omega & w_\alpha & w_{\beta\omega} & w_{\beta\alpha} \end{bmatrix}^T$. The first equation uses unbiased angular velocity is to update the orientation C_t. The Euler angles r_t can be recovered from the rotation matrix C_t using Eqn 3.7.

$$r_t = \begin{bmatrix} \arctan\left(\frac{C_t^{y,z}}{C_t^{z,z}}\right) \\ \arcsin\left(-C_t^{x,z}\right) \\ \arctan\left(\frac{C_t^{x,y}}{C_t^{x,x}}\right) \end{bmatrix}, \qquad \text{where} \quad C_t = \begin{bmatrix} C_t^{x,x} & C_t^{x,y} & C_t^{x,z} \\ C_t^{y,z} & C_t^{y,y} & C_t^{y,z} \\ C_t^{z,x} & C_t^{z,y} & C_t^{z,z} \end{bmatrix} \tag{3.7}$$

Unbiased linear acceleration is then rotated from the body to the reference frame using the new orientation. The constant gravitational field strength of Earth E_g

is subtracted from the rotated acceleration to obtain specific force in the reference frame. The specific force is then integrated with respect to time to update velocity v_t. Velocity is then integrated with respect to time to update the position p_t. The accelerometer and gyroscope biases are presumed perturbed copies from the previous time step.

$$\begin{aligned} C_t &= C_{t-1}R(\omega_t - \beta_t^\omega + w_\omega, \delta t) \\ v_t &= v_{t-1} + (C_t(\alpha_t - \beta_t^\alpha + w_\alpha) - E_g)\delta t \\ p_t &= P_{t-1} + v_t \delta t \\ \beta_t^\omega &= \beta_{t-1}^\omega + w_{\beta^\omega} \\ \beta_t^\alpha &= \beta_{t-1}^\alpha + w_{\beta^\alpha} \end{aligned} \tag{3.8}$$

3.2.1.2 Correction step

In the correction step of the filter measurements are used to correct the state estimate. For PDR it is possible to integrate a number of different sensors to improve the state estimate [37]. Each sensor increases the accuracy, at the expense of filter complexity, power consumption and device size. Examples include magnetometers, barometric pressure sensors and satellite navigation modules, which can be used to correct the heading, altitude and position respectively. It is common for an off-the-shelf IMUs to include a magnetometer, which can be integrated to achieve attitude estimation. However, the presence of metal in buildings causes spatially-correlated hard and soft iron biases. The EKF proposed in the thesis does not presume the existence of a magnetometer.

The filter makes the conservative assumption that zero-velocity updates are the only corrections that are available. The basic principle is to detect when the foot strikes the ground, and resets the velocity to zero. This is typically done by thresholding net acceleration or angular velocity, but more complex strategies do exist (refer to Skog et al. [86] for a comparison of such approaches). The step detector in this thesis simply determines whether unbiased angular velocity $\|\omega_t - \beta_t^\omega\|_2 < 0.6$ and triggers a step event when this is the case.

It is important to note at this point that a ZUPT can be considered as either a hard or a soft constraint. The topic of integrating hard constraints into the EKF has been addressed by Julier and Laviola [44]. A ZUPT event can be implemented as a hard linear constraint by treating it as a measurement with zero variance. Through experimentation it was found that this approach does not produce good results in the

PDR filter. Intuitively, this because the sensor is vibrating as it strikes the ground, invalidating the assumption that the velocity is exactly zero on step events. When used as a soft constraint, the ZUPT is treated as a measurement. The measurement has an associated variance, which describes how much uncertainty exists – in terms of velocity along each axis – when a step event is triggered. Eqn 3.9 shows the complete update step, comprising of an observation $z_t = \mathbf{0}_{3\times1}$, prediction and measurement model h, which projects the velocity from the state estimate x_t. It was found empirically that the ZUPT measurement noise w_z is drawn from an independent multivariate Gaussian distribution with variance 0.02 along each axis.

$$z_t = h(x_t) + w_z \tag{3.9}$$

3.2.1.3 Linearization

In order to use the EKF for state estimation, the process and correction models must first be linearized. Eqn 3.10 shows the prediction step. The matrix Φ is the linearized process model. The process disturbance vector w_t is drawn from a zero-mean multivariate normal distribution, having process noise covariance Q, with accelerometer and gyroscope noise values along its diagonal. The values can either be obtained from the sensor data sheet, but a better approach is to measure the noise experimentally. The matrix Γ is the linearized noise model, which maps the process disturbance vector from the body frame to the reference frame.

$$x_t = \Phi x_{t-1} + \Gamma w_t \tag{3.10}$$

Eqn 3.11 shows the correction step. The matrix Λ is the linearized measurement model. This model relates sensor measurements z_t to state values x_t. The additive measurement noise vector v_t is drawn from a zero-mean multivariate normal distribution, having measurement noise covariance R, with magnetometer and ZUPT noise values along its diagonal.

$$z_t = \Lambda x_t + v_t \tag{3.11}$$

The process and measurement models shown in Eqn 3.2.1.1 and Eqn 3.9 are nonlinear functions. In each EKF iteration the process and correction models are first linearized about the state estimate, using the approximations in Eqn 3.12.

$$\begin{aligned} \Phi &\simeq I + F\Delta t \\ \Gamma &\simeq G\Delta t \\ \Lambda &\simeq H \end{aligned} \tag{3.12}$$

These approximations require the calculation of the Jacobians in Eqn 3.13. Note that $\mathbf{0}$ and $\mathbf{I}$ are 3×3 zero and identity matrices respectively, while C and A are the rotation matrices defined in Eqn X and Y respectively.

$$\begin{aligned} F &= \begin{bmatrix} \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & A & \mathbf{0} & -C \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \\ G &= \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -C & \mathbf{0} & \mathbf{0} \\ C & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix} \\ H &= \begin{bmatrix} \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \end{aligned} \tag{3.13}$$

When inertial information μ_t arrives from the IMU, the EKF *prediction step* is carried out. Eqn 3.14 is used to update the estimate x_{t-1} to x_t. Likewise, Eqn 3.15 is used to update the covariance in the state estimate from P_{t-1} to P_t. The scalar value Δt represents the time (in seconds) that has elapsed since the previous prediction step. For navigation systems inertial information typically arrives at a rate between 64Hz and 1024Hz, implying that Δt is normally in the order of a few milliseconds

$$x_t = x_{t-1} + f(x_{t-1}, \mu_t)\Delta t \tag{3.14}$$

$$P_t = (I + F\Delta t)P_{t-1}(I + F\Delta t)^T + (G\Delta t)Q(G\Delta t)^T \tag{3.15}$$

When a ZUPT or magnetometer measurement arrives for processing, the *correction step* of the EKF is carried out in order to correct the state. The *a priori* covariance P_t^- is used in conjunction with the linearized measurement model and measurement noise covariance to determine the Kalman gain K (see Eqn 3.16). This gain represents the optimal minimum mean square error estimate for the state, given the current belief and the new information. This Kalman gain is then used to determine the *a posteriori* state estimate x_t^+ and covariance P_t^+.

$$K = P_t^- H^T (H P_t^- H^T + R)^{-1} \tag{3.16}$$

$$x_t^+ = x_t^- + K(z_t - y_t) \tag{3.17}$$

$$P_t^+ = (I - KH)P_t^-(I - KH)^T \tag{3.18}$$

3.2.2 Sampling the inertial trajectory

The PDR algorithm described in the previous section showed how to filter inertial measurements to maintain a single sensor's position estimate. An important requirement of the localization algorithm proposed in this thesis is the ability to estimate the distance $r_{s,t}$ between the position of a single sensor at two points in time. Assume that the EKF filter output for the given sensor is given by a state estimate x_t and covariance in that estimate P_t. For the sake of simplicity, assume further that x_t and P_t contain only position, and not the remaining 12 states of the EKF.

A numerical method is used to estimate the Gaussan-distributed distance between two points along this trajectory. 100 point pairs are sampled from $\mathcal{N}(x_s, P_s^2)$ and $\mathcal{N}(x_t, P_t^2)$ distributions. The distances between the corresponding points in each pair are recorded. The 100 distances samples are assumed Gaussian, and the mean μ and variance σ^2 for the sample set is calculated and returned by the motion model.

3.3 Radio model

Received Signal Strength (RSS) is the most practical measurement for relating the position of two sensors, as it is supported by most off-the-shelf consumer radios. RSS is corrupted by two major sources of error - *multi-path* and *shadowing.* Patwari et al. [70] show that multi-path error depends on both the frequency and the environment, and is caused by signals simultaneously taking different paths to a receiver. The multiple signals then contribute constructively or destructively at the receiver, affecting the received signal energy. Robust modulation techniques effectively spread the signal across a wide frequency range, and therefore mitigate a large quantity of this error. Shadowing error, however, is a product of the environment's reflective and absorptive properties. This error is typically modelled as additive Gaussian noise with mean zero and variance σ_r^2.

This thesis makes the assumption that encounters take place over a short range – within a room – with line-of sight between the transmitter and receiver. In such conditions a log-distance path loss model can be used to relate distance to RSS. Moreover, RSS measurements can be averaged over short intervals using the arithmetic mean. The log-distance path loss model is defined as 3.19. The signal strength γ_r of a radio packet sent from a transmitter that is r meters away can be modelled by some reference power γ_{r_0} at distance r_0 and a path-loss exponent η. The path-loss exponent defines the rate at which the signal drops with distance. Each measurement is assumed to be perturbed by some scalar error e, which is drawn from a zero-centred

Gaussian with variance σ_r^2. Many authors set the reference distance $r_0 = 1$ to simplify the expression.

$$\gamma_r \quad = \quad \gamma_{r_0} - 10\eta \log_{10}\left(\frac{r}{r_0}\right) + e, \qquad e \sim \mathcal{N}\left(0, \sigma_r^2\right) \tag{3.19}$$

A set of RSS measurements $\gamma_{1:k} = \{\gamma_1, ..., \gamma_k\}$ is mapped to a distance estimate through the radio model $f_r : (\gamma_{1:k}) \rightarrow \mathcal{N}(\mu, \sigma^2)$ to approximate the distance between the sender and receiver by a Gaussian-distributed measurement with mean μ and variance σ^2. The average RSS value $\bar{\gamma}$ is first calculated as the arithmetic mean of the RSS measurements. Since it has been assumed that the signal strength is affected only by free space losses, and that the variance in RSS is constant, then the arithmetic mean represents the minimum mean square estimate of the actual received signal strength. 100 samples are then drawn from the distribution $\mathcal{N}(\bar{\gamma}, \sigma_r^2)$ and passed through the path loss model to obtain a range estimate. The mean μ and variance σ^2 for the distance set is then calculated and returned by the radio model.

3.4 Discussion

This chapter showed how radio and inertial measurements are used to construct an encounter graph. Vertices in this graph represent sensors' positions at key points along their trajectory. A motion model and a radio model are required to convert raw measurements into pairwise distance estimates for edges of the encounter graph. This chapter concluded by describing two suitable models for pedestrian dead reckoning. The first of these an inertial navigation system that processes accelerometer and gyroscope measurements to maintain the 3D position of a human foot. The second is a log-distance path loss model, which relates received signal strength to distance.

The following chapter firstly describes how to analyse this encounter graph to determine whether there is a unique set of vertex coordinates that satisfy the constraints imposed by the motion and radio measurements. It then discusses several approaches for estimating such an embedding using distance measurements, as well as refining this estimate using the quantity of information brought by each measurement.

Chapter 4

Graph Realization

The previous chapter showed how inertial and radio measurements are fused to build an *Encounter Graph*. This graph is described entirely by an adjacency matrix, with each edge having a corresponding Euclidean distance. The *Rigidity Analysis* step begins by certifying whether a unique set of coordinates can be assigned to vertices, using only the adjacency matrix. If such a unique realization exists, then the graph is called generically globally rigid, and a unique assignment of relative vertex coordinates exists in either 2D or 3D. Assuming this is true, then the graph may embedded and then optionally refined.

This chapter is organised as follows. Sec. 4.1 discusses the deterministic and probabilistic methods used to certify whether a graph may be uniquely embedded in 2D or 3D respectively. Then, Sec. 4.2 discusses four algorithms for embedding the graph. Sec. 4.3 concludes this chapter with a discussion of two iterative refinement methods for refining this graph embedding.

4.1 Rigidity Analysis

The *Rigidity Analysis* step of the proposed algorithm certifies whether or not a range-based localization problem has a single solution; if there is ambiguity in the solution space of the problem, there is little point in trying to it. Sec. 4.1.1 shows that this is the case when the encounter graph is generically globally rigid in $\mathbb{E}^d$. This can be certified in one of two ways. In $\mathbb{E}^2$ a polynomial-time deterministic algorithm exists, and is discussed in Sec. 4.1.2. Problems in $\mathbb{E}^3$ can only be certified probabilistically, using the algorithm outlined in Sec. 4.1.3.

4.1.1 Euclidean graphs and degrees of freedom

Any rigid body has d translational and $\frac{d(d-1)}{2}$ rotational degrees of freedom in d-dimensional space, which are referred to as its *external degrees of freedom.* For example, a complete structure in 2D can be translated arbitrarily along two orthogonal axes in the plane, and rotated arbitrarily about the vector orthogonal to this plane. In order to constrain these degrees of freedom, at least $d+1$ algebraically independent vertex coordinates must be known *a priori*, which motivates why anchors are required by sensor localization problems.

Consider a static sensor localization problem expressed as a graph. In this graph vertices represent the unknown sensor state, and edges represent measurements, or observations, that relate the sensors' state. Consider a special case in which the state encodes only position, and the measurements are simple pairwise distance estimates between states. Here, the localization problem is reduced to "finding a set of vertex coordinates that satisfies the observed edge lengths."

It is critical to note that not all localization problems afford a unique solution. Not only because their external degrees of freedom cannot be constrained, but because they may also have *internal degrees of freedom.* These non-trivial degrees of freedom manifest as ambiguity in the solution space of the problem. Critically, the internal degrees of freedom are determined by the presence or absence of an edge constraint (not the actual distance measurement) and there is no localization algorithm that can solve an under-constrained problem. A complete graph is always uniquely localizable, but measurement error makes this solution difficult or impossible to find.

Let $\mathbf{p} = (p_1, ..., p_n)$ be a *configuration* of n vertex positions, each existing in $\mathbb{E}^d$. A graph $G = (V, E)$ with a corresponding configuration $\mathbf{p}$ is referred to as a *framework,* and denoted $G(\mathbf{p})$. Let $\mathbf{p}$ and $\mathbf{q}$ be two configurations. The frameworks $G(\mathbf{p})$ and $G(\mathbf{q})$ are said to be *equivalent,* and denoted $G(\mathbf{p}) \equiv G(\mathbf{q})$, if the corresponding distances along all edges in E are equal in both configurations. That is,

$$\forall_{(i,j)} \in E : \|p_i - p_j\| = \|q_i - q_j\| \iff G(\mathbf{p}) \equiv G(\mathbf{q})$$

If the pairwise distances between *all points* (not just the edges in the graph) are equal for both configurations, then the two configurations are said to be *congruent,* and denoted $\mathbf{p} \equiv \mathbf{q}$. That is,

$$\forall_{i,j} \in V : \|p_i - p_j\| = \|q_i - q_j\| \iff \mathbf{p} \equiv \mathbf{q}$$

A framework $G(\mathbf{p})$ is *locally rigid* in $\mathbb{E}^d$ if there exists no continuous force that, when applied to the framework, causes it to flex. In other words, there exists an $\epsilon > 0$,

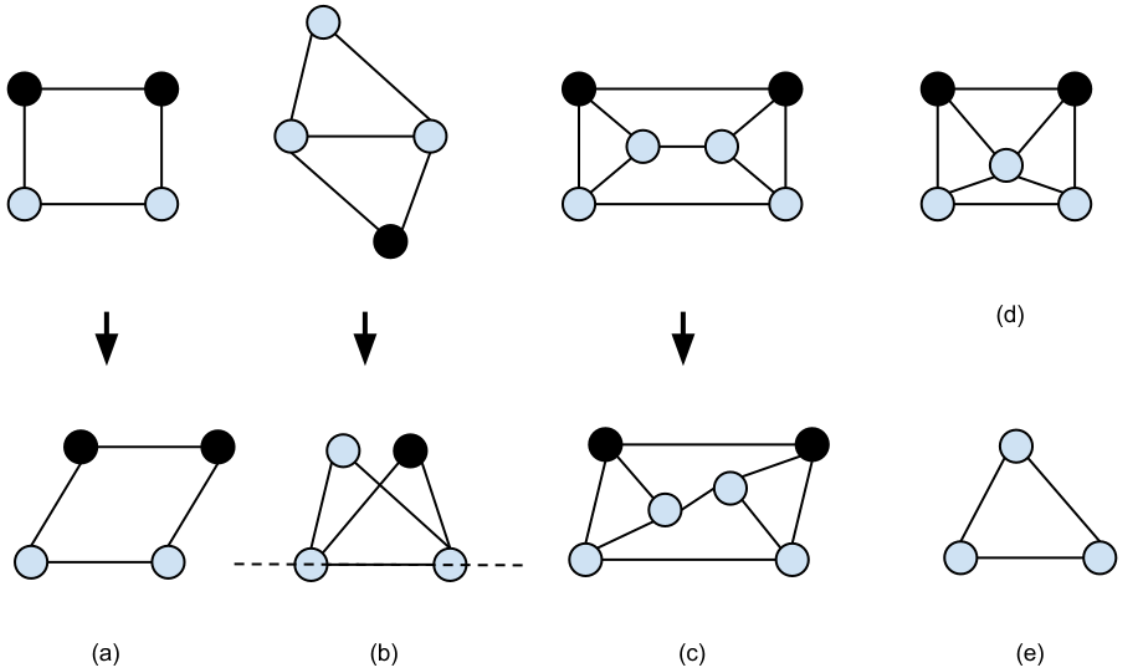


Figure 4.1: Degrees of graph rigidity: (a) a graph that is locally flexible; (b) a graph that is locally, but not globally, rigid; a dashed line shows an axis of reflection; (c) a graph that is locally, but not redundantly, rigid; (d) a framework that is globally rigid in the plane, but not universally rigid; (e) a framework that is universally rigid, and the basis for triangulation.

such that for any other configuration $\mathbf{q}$ in $\mathbb{E}^d$, if $\|\mathbf{p} - \mathbf{q}\| < \epsilon$ and $G(\mathbf{p}) \equiv G(\mathbf{q})$, then $\mathbf{p} \equiv \mathbf{q}$. A framework is called *redundantly rigid* if it remains locally rigid after the removal of any single edge in E, and *minimally rigid* if all edges are required to keep the framework locally rigid. A framework is *globally rigid* if for all configurations $\mathbf{q}$ in $\mathbb{E}^d$, $G(\mathbf{p}) \equiv G(\mathbf{q}) \Leftrightarrow \mathbf{p} \equiv \mathbf{q}$ holds. Finally, a framework $G(\mathbf{p})$ is *universally rigid* if $G(\mathbf{p}) \equiv G(\mathbf{q}) \Leftrightarrow \mathbf{p} \equiv \mathbf{q}$ for all configurations $\mathbf{q}$ in $\mathbb{E}^s$, where $s \geq d$. A non-rigid framework is said to be *flexible*, or have internal degrees of freedom. Fig. 4.1 shows several examples of rigid and flexible frameworks.

Note that rigidity is expressed entirely as a function of the edge constraints in a graph, making it independent of the actual measurement values. In the literature, a configuration in $\mathbb{E}^d$ is also referred to a *graph embedding* or *realization of the graph,* and the process of finding a configuration is referred to as either *graph realization* or *graph embedding.*

For localization problems expressed as Euclidean embeddable graphs, an assignment of values to states is simply a configuration of the graph vertices, forming a framework in $\mathbb{E}^d$. Certifying that a given localization problem yields a unique solu-

tion is therefore equivalent to certifying whether the corresponding graph is globally rigid. But, as shall be shown in the next section, this is complicated by the fact that there exists no exact polynomial-time algorithm for certifying generic global rigidity in $\mathbb{E}^d$, where $d > 2$.

4.1.1.1 Deterministic methods

Saxe [79] proved that the *decision problem* of certifying whether a given graph is embeddable in $\mathbb{E}^1$ is Strongly NP-Complete for the single dimensional case, and strongly NP-Hard for $\mathbb{E}^d$, where $d > 1$. Therefore, the majority of results in rigidity theory consider only those configurations that are in *general position*, a relaxation of the problem. A configuration is in general position if no $(d+1)$ points are algebraically dependent in $\mathbb{R}^d$. In other words, given a configuration $\mathbf{p}$, there exists no non-zero polynomial coefficients $\alpha = \{\alpha_1, ..., \alpha_{d+1}\}$, such that $\alpha_1\mathbf{q}_1 ... \alpha_{d+1}\mathbf{q}_{d+1} = 0$. For two dimensions this would be analogous to the constraint that no three or more points lie along a line, while in three dimensions that would be analogous to saying that no four or more points lie in the same plane.

Consider a framework $G(\mathbf{p})$, comprised of a configuration $\mathbf{p}$ in $\mathbb{E}^d$ and a graph G with v vertices and e edges. The $e \times dv$ *rigidity matrix* $M(\mathbf{p})$ is a set of coefficients describing the instantaneous velocities of all points in $\mathbf{p}$, induced by a smooth force acting on the framework. Row r of $M(\mathbf{p})$ represents the constraints edge r imposes on vertices i and j. In this row, columns that correspond to point i take on the value $p_i - p_j$. while columns that correspond to point j take on the value $p_j - p_i$. A d-dimensional framework $G(\mathbf{p})$ having $v \geq d+2$ points is said to be *infinitesimally rigid* if its rank is equal to the external degrees of freedom in $\mathbb{E}^d$, which equals $vd - \begin{pmatrix} d+1 \\ 2 \end{pmatrix}$. The intuition behind this is that v points in d dimensions have a total of vd degrees of freedom, some of which are trivial. The rows of the rigidity matrix must sufficiently constrain the columns of the rigidity matrix in order to be infinitesimally rigid.

Asimow and Roth [2] proved that d-dimensional infinitesimal rigidity is equivalent to d-dimensional generic local rigidity. This notion of "covering degrees of freedom" forms the underlying mechanism used by Laman [55] to prove that a graph G with v vertices and e edges is generically rigid in $\mathbb{E}^2$ if and only if $e' \leq 2v' - 3$ for every subgraph of G with v' vertices and e' edges. It is possible to prove this by evaluating all subgraphs, but it is more computationally-efficient to use a dynamic programming algorithm, called the *Pebble Game* by Jacobs and Hendrickson [42].

Proving generic local rigidity is insufficient to guarantee that a given graph is uniquely localizable. Doing so requires certifying that the graph is generically globally rigid. A graph G with v vertices is *k-connected* if the removal of any $i < k$ vertices results in a connected subgraph G' with $v - i$ vertices. Hendrickson [38] conjectured that, in order to be generically globally rigid in $\mathbb{E}^d$, a framework should have at most $d + 1$ vertices that are completely connected, or the graph must be $(d + 1)$-connected and redundantly rigid. Jackson and Jordán [41] proved this for $d = 2$. Connelly [13] showed by counterexample that Hendrikson's conjecture is false for $d = 3$ and currently there exists no deterministic, polynomial-time algorithm for certifying generic global rigidity in $\mathbb{E}^d$, where $d > 2$. For the case of $d = 2$, redundant rigidity certification can be achieved with a polynomial-time extension to the *Pebble Game*. Certification of 3-connectivity requires a SPQR Tree by Di Battista and Tamassia [20], which can be constructed in linear-time using an algorithm by Chimani et al. [10], based on a method by Hopcroft and Tarjan [40].

4.1.1.2 Probabilistic methods

Since there exists no deterministic algorithm for certifying generic global rigidity in $\mathbb{E}^d$ for $d > 2$, a probabilistic method must be used. Assume that some framework $G(\mathbf{p})$ has a corresponding stress vector $\omega = (..., w_{ij}, ...)$, which contains exactly one scalar weight $w_{ij} = w_{ji} \neq 0$ for every distinct pair of vertices (i, j) in the edge set E of G. A *stress matrix* Ω is constructed in the following way.

$$[\Omega]_{ij} = \begin{cases} -w_{ij} & i \neq j \\ \sum_{j \in E}^{i \neq j} w_{ij} & i = j \end{cases} \tag{4.1}$$

A framework is said to be in *equilibrium stress* if the stress e_i about each vertex i is zero. That is $e_i = \mathbf{0}$ for all $i \in V$, where e_i is calculated in Eqn 4.2.

$$\mathbf{e}_i = \sum_{j \in V} w_{ij}(\mathbf{p}_j - \mathbf{p}_i) \tag{4.2}$$

Connelly [13] proved that all generically globally rigid frameworks containing v points have an equilibrium stress matrix Ω with rank $(v - d - 1)$. Connelly also proved that this condition was sufficient up to two dimensions, and conjectured that it may also be true for higher dimensions. Connelly proposed a randomized algorithm for numerically solving the equilibrium equations, providing a probabilistic proof for certifying generic global rigidity in $d \leq 2$. Recently, Gortler and Thurston [32] proved Connelly's conjecture, providing the first probabilistic algorithm for certifying generic global rigidity in $\mathbb{E}^d$, for $d > 2$. A sketch of this algorithm is given in Appendix A.

Zhu et al. [97] showed how certification of generic universal rigidity can be expressed as a Semidefinite Programming (SDP), and solved in near-polynomial time by an Interior Point Algorithm (IPA) by Wright [94]. Universal rigidity is an unnecessarily restrictive condition in the context of 2D or 3D localization, so it will not be discussed in any further detail.

4.1.2 Deterministic certification in the plane

Chapter 2 showed that proving the uniqueness of a localization problem in $\mathbb{E}^2$ is equivalent to proving that the graph formed by its edge constraints is both 3-connected and generically redundantly rigid. At run-time the proposed localization algorithm uses two software libraries to assist in proving these properties. The first of these is the Open Graph Drawing Framework (OGDF) by Chimani et al. [10], which is open source C++ library for graph drawing. Of particular interest are the isConnected(G), isBiconnected(G) and isTriconnected(G) functions, which perform 1-connectivity, 2-connectivity and 3-connectivity certification respectively on a given graph G. The second software library is Kinari by Fox et al. [28], a closed-source framework for modelling and pebble game rigidity analysis. The PebbleGameWithCircuits class provides a method that allows one to calculate the redundantly rigid components of a graph, and hence whether the graph is generically redundantly rigid in the plane.

Fig. 4.2 summarizes the four step process that the proposed localization algorithm uses to certify whether a given localization problem (described by the edges in an *Encounter Graph)*, yields a unique solution. Each of the steps is outlined in the four subsections that follow. A brief description of how each algorithm is given, as well as its run-time complexity.

4.1.2.1 1-connectivity certification

The isConnected(G) function in the OGDF library is called to check whether all vertices are reachable in the graph. The implementation picks a starting vertex and performs a controlled breadth-first search of the graph. The algorithm starts by adding a random vertex to a queue. At each step of the search, the algorithm pops the first vertex of the queue and iterates over its neighbourhood. If a neighbour has already been searched, then it is ignored. Otherwise, it is added to the search queue and marked as 'visited'. The algorithm terminates when the queue is empty. The graph is 1-connected if the number of visited vertices is equal to the number of vertices in the graph, and has a run time complexity $O(v + e)$.

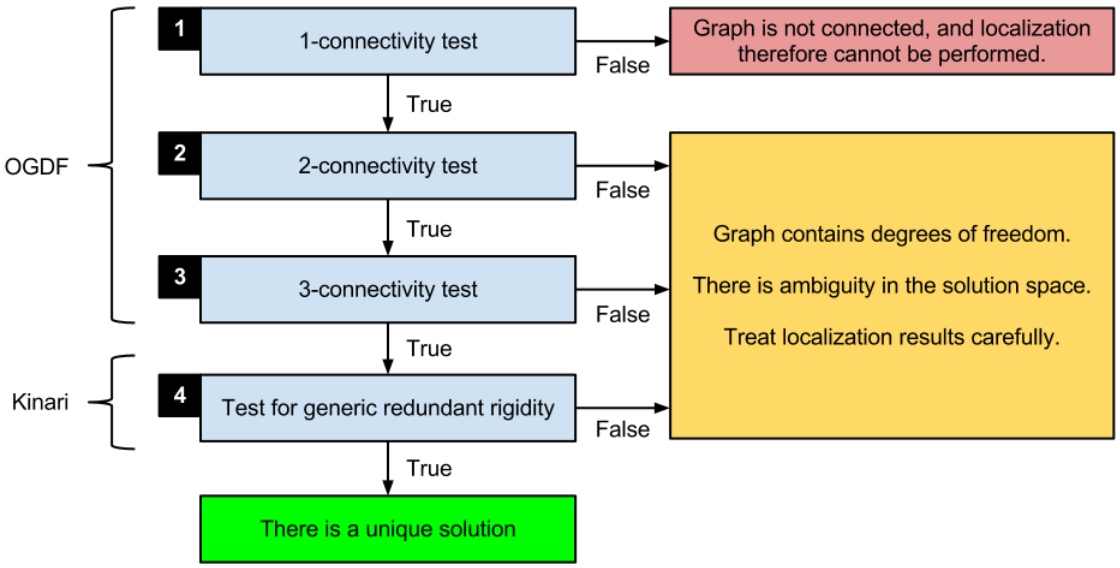


Figure 4.2: Logical flow for analysing the encounter graph

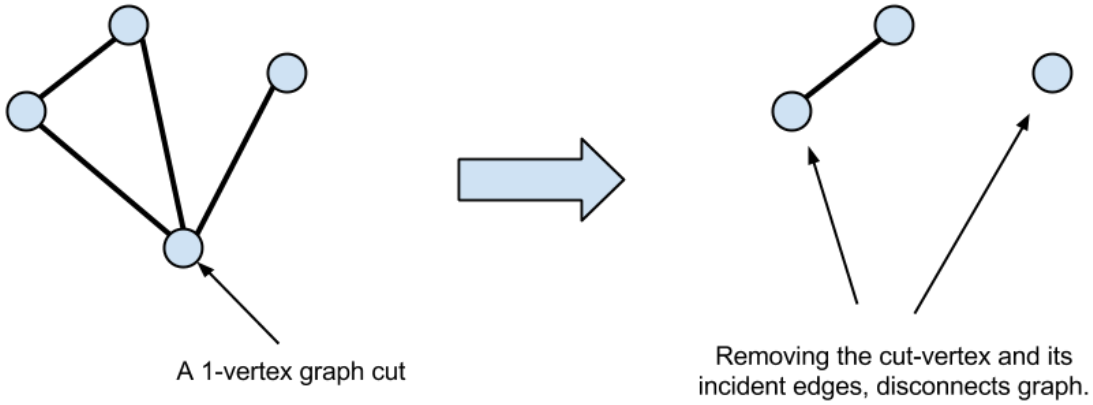


Figure 4.3: A 1-connected graph has at least one 1-vertex cut. Removing this vertex and all incident edges results in a graph with at least two components.

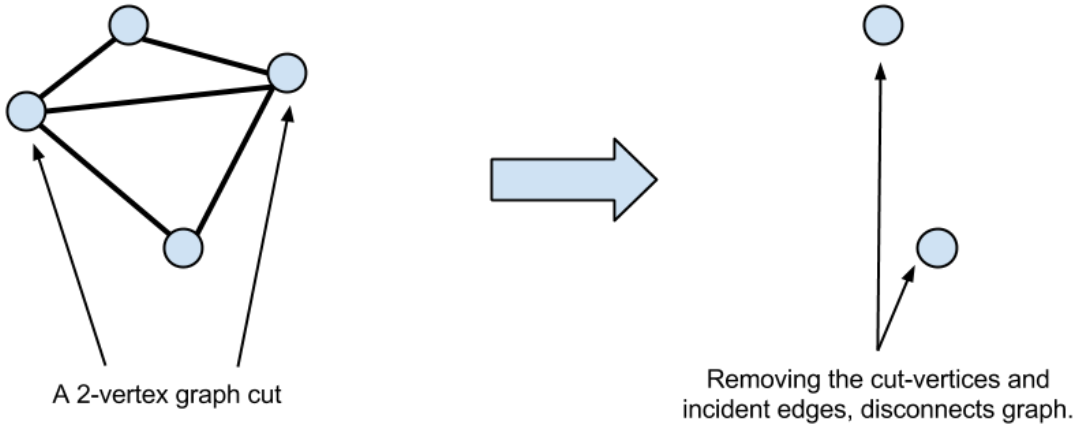


Figure 4.4: A 2-connected graph has at least one 2-vertex cut. Removing these two vertices and all incident edges results in a graph with at least two components.

If the graph is not 1-connected, it means that the relative motion of two or more connected components is unconstrained. A basic assumption of many localization algorithms is that the graph is connected, and violation of this assumption causes run-time errors. In the proposed cooperative localization algorithm, if the graph is not connected, then the problem is flagged and localization does not take place.

4.1.2.2 2-connectivity certification

The isBiconnected(G) function in the OGDF library assumes that the graph is 1-connected. Hopcroft and Tarjan [40] algorithm is used to split the graph into its biconnected components, at a run time complexity of $O(v+e)$. A graph is 2-connected if no cut vertex exists (a cut vertex split the graph into two connected components). The algorithm performs a depth-first search of the tree, maintaining the depth of the current node, as well as the *lowpoint* of each node. The *lowpoint* of a node is the lowest depth of all its descendant nodes. Importantly, any non-root target node is a cut-vertex if it has a child node with a *lowpoint* that is greater than the depth of the target node. A root node is a cut-vertex if it has more than one child.

A graph that fails 2-connectivity contains a continuous internal degree of freedom. In the proposed localization algorithm this manifests in a sensor's trajectory having two unconstrained degrees of freedom with respect to a single pivot point. The solution space is given by a circle, centered at the pivot point. The whole trajectory thus has two degrees of freedom – a rotation about the centre of the circle, and about the intersection point on the circumfrence.

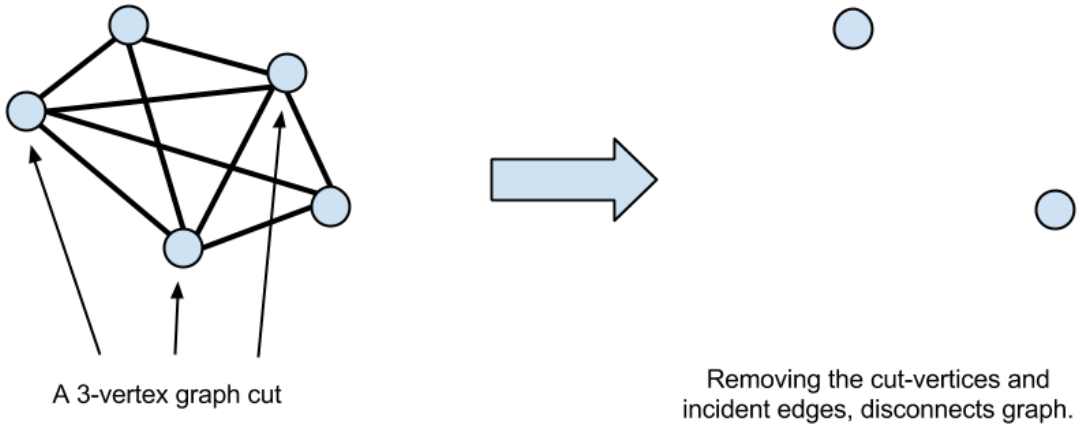


Figure 4.5: A 3-connected graph has at least one 3-vertex cut. Removing these three vertices and all incident edges results in a graph with at least two components.

4.1.2.3 3-connectivity certification

The isTriconnected(G) function in the OGDF library works by building an SPQR-Tree. The SPQR-Tree data structure was developed by Di Battista and Tamassia [20] for splitting a graph into its 3-connected components. The tree itself is built using an algorithm by Hopcroft and Tarjan [40] that was subsequently corrected by Gutwenger and Mutzel [35], and has a run-time complexity of $O(v + e)$.

A given graph is 3-connected if and only if it has exactly one R (rigid) node and zero S (serial) and P (parallel) nodes. If a graph fails 3-connectivity, it implies that there exists at least one 2-vertex cut, i.e. a pair of vertices that, when removed from the graph, splits the graph into two connected components. A graph that has a 2-vertex cut has an internal degree of freedom, and is therefore not rigid.

4.1.2.4 Generic redundant rigidity certification

The PebbleGameWithCircuits class in the Kinari library divides a given graph into its redundantly rigid components. Of particular interest is the $(2, 3)$-pebble game, which Jacobs and Hendrickson [42] proved efficiently certifies two-dimensional generic local rigidity. The idea behind this algorithm is that it assigns two pebbles to every vertex in the graph, representing degrees of freedom. The algorithm uses these pebbles to "cover" edges of the graph. At the end of the algorithm, if more than three pebbles remain on vertices, then the graph fails generic local rigidity. If exactly three remain, then the graph is minimally rigid, otherwise the graph is over-constrained. The pebble game algorithm follows two basic rules:

1. **Edge move** - If two vertices are neighbours in G with at least 4 pebbles, add a directed edge between them and remove a pebble from the endpoint.

2. **Slide mode** - If there is a pebble on the end vertex of a directed edge, move the pebble to start vertex and reverse the edge.

The classic (2,3)-pebble game requires e iterations to compute a solution. However, The PebbleGameWithCircuits class certifies a stronger condition, generic redundant rigidity. Recall that a graph is generically redundantly rigid if it remains generically locally rigid after the removal of any vertex. Therefore, to certify generic redundant rigidity, the pebble game must be repeated v times, with a final run-time complexity of $O(ve)$.

4.1.3 Probabilistic certification in higher dimensions

Sec. 4.1.1.2 showed that certifying whether a graph is uniquely embeddable in $\mathbb{E}^d$ involves evaluating the rank of the rigidity matrix, as well as the rank of the equilibrium stress matrix. However, constructing both matrices is non-trivial, and performing the rank calculation with symbolic equations is infeasible for large graphs.

Gortler and Thurston [32] showed that an efficient randomized algorithm can probabilistically prove generic global rigidity of a graph with v vertices and e edges in $\mathbb{E}^d$. A sketch of this algorithm is given in Appendix A. The high-level idea is to begin by randomly assigning integer coordinates to graph vertices. Each coordinate is drawn from a sufficiently large interval $[1, N]$, so as to bound the probability of vertices being algebraically dependent. Connelly [13] showed that if the rank of the rigidity matrix is equal to $vd + \binom{d+1}{2}$, then the graph is generically locally rigid: no continuous force would move any vertex in the graph. A stronger condition, generic global rigidity, guards against any discontinuous flexes, like reflections. Gortler and Thurston [32] showed that a linear system can be solved numerically to find an equilibrium stress matrix. And, if this equilibrium stress matrix has a rank equal to $v - d - 1$, then the graph is generically globally rigid with probability bounded by $1 - (ve)/N$. The run-time complexity of this algorithm is given by the solution to a linear equation of the form $Ax = b$, where A is of size $e + \binom{d+1}{2} \times e$, with a run-time complexity of approximately $O(e^3)$.

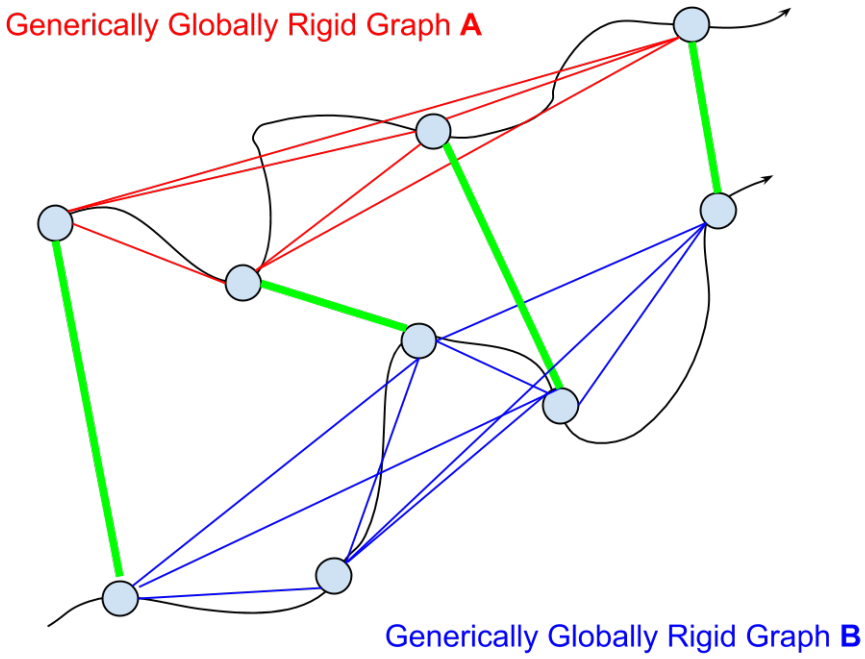


Figure 4.6: The odometry information allows the construction of a generically globally rigid graph in $\mathbb{E}^2$ for each sensor trajectory A and B. Since radio encounters are time-indexed only four radio measurements are required to make the joint graph generically globally rigid in in $\mathbb{E}^2$. The same idea extends to $\mathbb{E}^3$ with seven measurements.

4.1.4 Optimizing the graph for scalability

The time complexity of certifying rigidity in 2D and 3D is $O(ve)$ and $O(e^3)$ respectively. Clearly these algorithms do not scale well to large problems containing thousands of measurements. However, it is possible to leverage existing theory from the literature in conjunction with a special property of the encounter graph in order to substantially reduce the number of vertices in the graph.

Eren et al. [24] showed that if two separate graphs A and B are each generically globally rigid in $\mathbb{E}^2$, then the graph $A \cup B$ is also generically globally rigid in $\mathbb{E}^2$ if at least four edges are added, three of which are incident to distinct vertices in both A and B (see Fig. 4.6). The same idea extends to $\mathbb{E}^3$, but at least seven edges must be added, four of which must be incident to distinct vertices in A and B.

The encounter graph has two important properties that allow its complexity to be reduced. Firstly, each sensor's subgraph is generically globally rigid in $\mathbb{E}^d$ by virtue of the manner in which the graph is constructed. Secondly, encounters occur as a function of time and therefore, radio measurements connect distinct vertices in the sensors' subgraphs. Or, equivalently, each vertex in some sensor's subgraph is

connected to at most one vertex in another sensor's subgraph. These two conditions permit pruning all but $\begin{pmatrix} d+1 \\ 2 \end{pmatrix}$ edges between two sensors, retaining the rigidity of the graph. If one repeats this process for all sensor pairs, a significant proportion of the encounter edges are removed. Any vertex in the sensors' graph that is no longer incident to an encounter can be removed along with its incident edges, provided that the rigidity of the subgraph is retained by adding appropriate edges within the neighbourhood of the removed vertex.

4.2 Graph Embedding

The previous section discussed the *decision problem* of certifying whether a given graph is uniquely embeddable in $\mathbb{E}^d$. In this section it will be assumed that this is true, and the focus will be on the *computational problem* of finding an embedding.

The *Encounter Graph* $G = (V, E)$ contains a vertex set V and an edge set E, of size v and e respectively. Let matrix $[D]_{ij} = d_{ij}$ be a $v \times v$ symmetric matrix representing the set of edge distances, having infinite values for missing edges. i.e. $\forall_{i,j} \in V : (i,j) \notin E \Rightarrow d_{ij} = \infty$. As a reminder, the graph structure G emerges from the existence of measurements linking anchor and sensor positions in time, while the distances D are derived from inter-anchor distances that are known *a priori*, or by passing the measurement values through either a motion or radio model.

The remainder of this section outlines four competing graph embedding algorithms from the literature. These algorithms are typically used to solve the cooperative *static* sensor localization problem. But, by converting a localization problem into a graph these methods may be used to cooperatively localize *mobile* sensors. The objective of all four algorithms is to find an "embedding in dimension d", which is a set of v coordinates $X = \{x_i : \forall_{i \in V} x_i \in \mathbb{E}^d\}$ that best satisfies the observed distances D. Graph embedding is illustrated graphically with an example in Fig. 4.7.

Sec. 4.2.1 describes *Multidimensional Scaling* (MDS), a technique that finds graph vertex coordinates in $\mathbb{E}^d$ that minimizes the sum square difference between *all* predicted and observed pairwise distances. Then, Sec. 4.2.2 outlines *Distributed Multidimensional Scaling* (MAP), which stitches together results from 2-hop MDS patches across a network, thus performing better on irregularly-shaped networks. Both MDS and MAP require that unknown terms in the distance matrix D are calculated using a shortest-hop algorithm. Sec. 4.2.3 discusses *Spectral Graph Drawing* (SGD), a method that tries to find a solution that minimizes the sum squared distance between all vertices, weighted by a function that inverts distances. Finally, Sec. 4.2.4 discusses

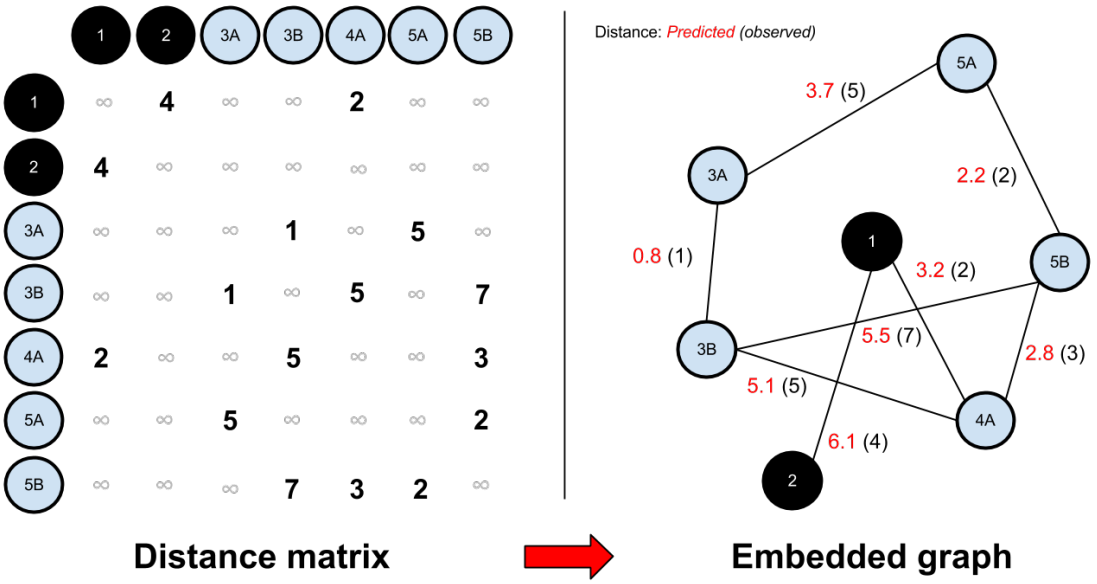


Figure 4.7: In the *Graph Embedding* step the distance matrix (left) is used to perform a one-shot estimation of the graph coordinates (right). The coordinates are calculated so that the *predicted* pairwise distances (red numbers) closely match the *observed* pairwise distances (black numbers)

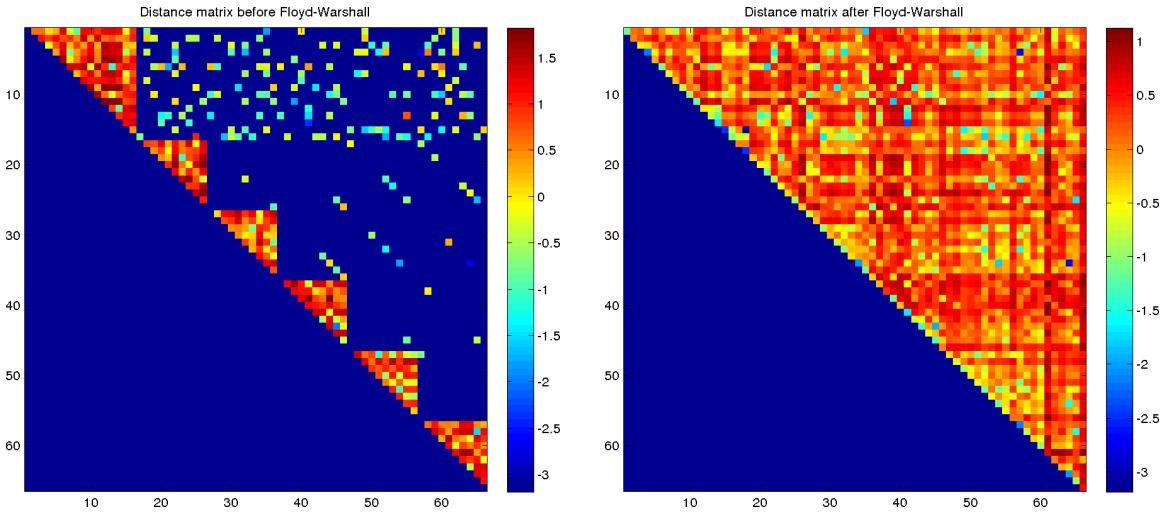
a variant of this method called *Degree Normalized Spectral Graph Drawing* (DNS), which takes into account the vertex degree when obtaining a solution.

4.2.1 Multidimensional scaling

Classical *Multidimensional Scaling (MDS)* is equivalent to *Principle Component Analysis* (PCA), where Euclidean distance is used in place of variance. MDS requires a complete distance matrix $\delta_{ij} =_{ij} [\Delta]$, with non-zero off-diagonal elements. However, only a subset of these distances are typically known. In the MDS-MAP algorithm by Shang et al. [83] unknown distances are first approximated using a shortest-path algorithm f_s, such as the well-known *Floyd-Warshall* or *Dijkstra* methods. These methods obtain a complete distance matrix $\Delta = f_s(D)$ from a sparse distance matrix D. Fig. 4.8 shows how this affects the distance matrix. Note that D is symmetric and upper-triangular, and only the upper-triangular part of the matrix is shown.

The goal of MDS is to find a set of vertices X that minimizes the stress function shown in Eqn 4.3.

$$X = \arg\min_X \sum_{(i,j)\in E} \left(\left\| \mathbf{x}_i - \mathbf{x}_j \right\| - \delta_{ij} \right)^2 \quad \delta_{ij} \in \Delta \tag{4.3}$$



(a) Distance matrix before shortest path (b) Distance matrix after shortest path

Figure 4.8: Effect of a shortest path algorithm on the distance matrix

Cox and Cox [15] provide a detailed description of MDS, which is summarized below. The inner product matrix B, centering matrix H and matrix A are defined as per Eqn 4.4.

$$B = XX^T, \qquad H = I - \frac{1}{n}\mathbf{1}_{n\times 1}\mathbf{1}_{1\times n} \quad [A]_{ij} = -\frac{1}{2}\delta_{ij}^2 \tag{4.4}$$

Using the above definition the inner product matrix can be rewritten as Eqn 4.5.

$$B = HAH \tag{4.5}$$

Matrix B is symmetric, positive semidefinite and of rank d, which implies that it has d non-zero and $v-d$ zero eigenvalues. Eqn 4.6 shows how B may be written in terms of its spectral decomposition, where V is a matrix of v eigenvectors and Λ is a matrix of their v corresponding eigenvalues.

$$B = V\Lambda V^T \tag{4.6}$$

The eigenvalues are now ordered in decreasing magnitude $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \geq 0$, along with their corresponding eigenvectors $v_1, \ldots, v_n$. Two matrices $V_d = [v_1 \cdots v_d]$ and $\Lambda_d = [\lambda_1 \cdots \lambda_d]$ are then constructed, containing the eigenvectors/values corresponding to the largest d eigenvalues. These matrices can be used to find a configuration X using Eqn 4.7. This is an orthogonal separation of the vertices in d-dimensional space that optimally solves the stress function given in Eqn 4.3, for complete error-free distance matrices.

$$X = V_d\Lambda_d^{\frac{1}{2}} \tag{4.7}$$

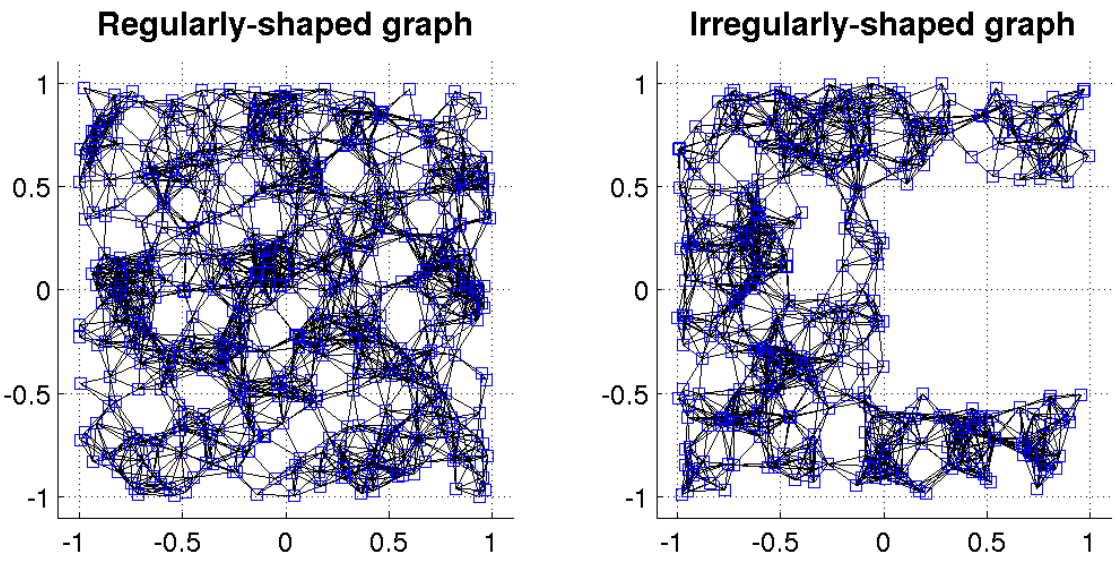


Figure 4.9: A regular (left) and irregular (right) graph. Vertices are shown as blue squares, while edges are shown as black lines.

4.2.2 Distributed multidimensional scaling

MDS is an optimal least squares estimator for *complete graphs*, where all of the pairwise distances are accurately known. The shortest-path algorithm inserts distances that are corrupted by a significant quantity of additive error. This causes MDS to pull vertices towards the centroid of the configuration, resulting in poor performance on irregularly-shaped (convex and equal in all directions) graphs. Refer to Fig. 4.9 for an example of a regular graph (left) and an irregular graph (right).

Shang and Ruml [82] extended MDS-MAP to apply MDS in subgraphs across the network. One subgraph is generated for each of the v vertices in the graph. Each subgraph i contains only the k-hop[1] neighbours of vertex i, as well as all known pairwise distances between neighbours. MDS is then applied to each subgraph to produce a *local map*. These *local maps* are then stitched together to form a *global map*. The stitching algorithm starts by picking a random *local map* to be the *global map*. It then searches through the remaining *local maps* for a candidate with the most overlap with vertices in the *global map*. A method similar to the *Procrustes Transformation* (see Sec. 5.2) is used to project the candidate *local map* into the *global map* in such a way that minimizes the sum square difference between correspondences. This stitching process continues until there are no more *local maps* to merge.

[1]The value $k = 2$ typically provides a good balance between complexity and accuracy.

4.2.3 Spectral graph drawing

Spectral Graph Drawing (SGD) is expressed by the minimization problem given in Eqn 4.8. The method attempts to push together those vertices i and j with a high weighting $f(d_{ij})$. It follows intuitively that the weighting should somehow be inversely related to distance. A common method is to set $f(d_{ij}) = e^{-d_{ij}}$. In so doing, unknown distances with infinite values are conveniently weighted zero.

$$X = \arg\min_{X} \sum_{(i,j)\in E} f(\delta_{ij}) \left(\left\| x_i - x_j \right\| \right)^2 \quad \delta_{ij} \in \Delta \tag{4.8}$$

The stress matrix S, degree matrix K and Laplacian matrix L are defined as per Eqn 4.9.

$$[S]_{ij} = f(d_{ij}), \quad [K]_{ii} = \sum_{j=N(i)} w_{ij}, \quad L = K - S \tag{4.9}$$

For $d = 1$ Eqn 4.8 is equivalent to Eqn 4.10.

$$\begin{aligned} X_1 &= \arg\min_{x} \left[x^T L x \right] \\ s.t &\quad x^T x = 1 \\ &\quad x^T \mathbf{1}_n = 0 \end{aligned} \tag{4.10}$$

The first constraint prevents the trivial solution $x = \mathbf{0}$. The second constraint $x^T \mathbf{1}_n$ sets the mean of the solution to zero, making the solution translation-invariant (its centroid will be the origin). Hall [36] proved that the general embedding X in $\mathbb{E}^2$ is obtained from the eigenvectors corresponding to the $d+1$ smallest eigenvalues of L. i.e. For $d = 1$, X_1 is given by the eigenvectors corresponding to the *second* smallest eigenvalue of the Laplacian matrix L, and for $d = 2$ the value for X_2 is given by the eigenvectors corresponding to the *third* smallest eigenvalue.

Importantly, SGD forces the spread of the embedding to equal 1. This means that a final scaling step is required. The scaling factor s is obtained by dividing the sum of observed distances by the sum of predicted distances (see Eqn 4.11.). The final embedding X is thus given by $X = s[X_1, ..., X_d]$.

$$s = \frac{\sum_{\forall_{(i,j)\in V}} \|x_i - x_j\|}{\sum_{\forall_{(i,j)\in V}} d_{ij}} \tag{4.11}$$

4.2.4 Degree-normalized spectral graph drawing

SGD performs poorly on irregular graphs, as it tends to draw nodes towards the centroid. In an effort to prevent this, *Degree Normalized Spectral Graph Drawing* (DNS) weighs the position of each graph vertex by its degree (see Eqn 4.12). Koren [51] proved that the solution to this constrained optimization problem is equivalent to finding the generalized eigenvectors of (L, K), where K and L are the degree and Laplacian matrices from the previous section (Eqn 4.9). For regular graphs, the degree matrix is close to a scaled identity matrix, which makes the constraint optimization problem equivalent to SGD.

$$\begin{aligned} X_1 &= \arg\min_x \left[x^T L x\right] \\ s.t &\quad x^T K x = 1 \\ &\quad x^T \mathbf{1}_n = 0 \end{aligned} \tag{4.12}$$

To see why this is a useful modification to the objective function, first let $a_{ij} = \exp(-\delta_{ij})$ and denote the neighbourhood of vertex i as $N(i)$. Eqn 4.13 captures the total energy $\mathcal{E}^i_{ene}$ of a single vertex i.

$$\mathcal{E}^i_{ene} = \sum_{j \in N(i)} a_{ij} \left(x_i - x_j\right)^2 \tag{4.13}$$

Eqn 4.14 shows the derivative of the total energy with respect to the node's position x_i.

$$\frac{\partial \mathcal{E}^i_{ene}}{\partial x_i} = 2 \sum_{j \in N(i)} a_{ij} \left(x_i - x_j\right) \tag{4.14}$$

This derivative is set to zero, in order to find the energy-minimizing x_i value. Eqn 4.15 shows that the location of x_i that satisfies this objective function is the weighted centroid of the vertices adjacent to the reference node i, which motivates the need to weight the vertex by its degree.

$$\begin{aligned} 0 &= 2 \sum_{j \in N(i)} a_{ij} x_i - 2 \sum_{j \in N(i)} a_{ij} x_j \\ x_i &= \frac{\sum_{j \in N(i)} a_{ij} x_j}{\sum_{j \in N(i)} a_{ij}} \\ &= \frac{\sum_{j \in N(i)} a_{ij} x_j}{\deg(i)} \end{aligned} \tag{4.15}$$

4.2.5 Semidefinite programming

Assume that there are v vertices in a graph comprising of a anchors and s sensors. Let $X = \{x_1, ..., x_s\}$ and $A = \{a_1, ..., a_a\}$ be the set of sensor and anchor positions respectively in $\mathbb{E}^d$. Similarly, let E_a and E_s be the set of edges that represent known distance measurements. Eqn 4.16 captures all of the constraints imposed on the unknown sensor positions X, given the observations.

$$\begin{aligned} \left\|x_i - x_j\right\|^2 &= d_{ij}^2, \quad \forall_{(i,j)} \in E_s \\ \left\|a_k - x_j\right\|^2 &= d_{kj}^2 \quad \forall_{(k,j)} \in E_a \end{aligned} \tag{4.16}$$

Eqn 4.16 can be rewritten as Eqn 4.17. Here, the matrix dot product $A \bullet B$ calculates the sum of the element-wise product $A \bullet B = \sum_{i,j} [A]_{ij} [B]_{ij}$ for all elements (i, j) of equal sized matrices A and B. The vector e_k has a value of one at index k, and zeros for all remaining $s - 1$ elements.

$$\begin{aligned} &\textbf{find} && X \in \mathbb{R}^{n \times d} \\ &\textbf{subject to} \\ \begin{bmatrix} \mathbf{0} \\ e_i - e_j \end{bmatrix} \begin{bmatrix} \mathbf{0} & e_j - e_i \end{bmatrix} \bullet \begin{pmatrix} I & X \\ X^T & Y \end{pmatrix} &= d_{ij}^2 \quad \forall_{(i,j)} \in E_s \\ \begin{bmatrix} -a_k \\ e_j \end{bmatrix} \begin{bmatrix} -a_k & e_j \end{bmatrix} \bullet \begin{pmatrix} I & X \\ X^T & Y \end{pmatrix} &= d_{kj}^2 \quad \forall_{(k,j)} \in E_a \\ Y &= X^T X \end{aligned} \tag{4.17}$$

The "SDP relaxation" of the problem defined in Eqn 4.17 requires that the condition $Y = X^T X$ be relaxed to $Y \succeq X^T X$. The objective function can now be solved in near-polynomial time by an *Interior Point Algorithm* (IPA), under the assumption that the distances are noise-free. In order for a solution to be obtained, SDP requires that the number of anchors is greater than the dimensionality of the problem. In cases where this requirement is not satisfied, the SDP algorithm first searches for a rigid *clique* of $d + 1$ vertices that contains any known anchor points. Trilateration is then used to calculate the relative positions for each vertex in the clique. The vertices in the clique are then treated as anchors to obtain a localization solution.

4.2.6 Summary

Fig. 4.10 illustrates the behaviour of MDS, MAP, SGD and DNS for a sample problem containing 200 sensors, randomly-placed in a 1×1 region of space. In the two right hand columns an irregular graph has been created by removing all sensors in the range $0 < x < 1$ and $0.3 < y < 0.7$. Columns 2 and 4 are sparsified versions of Columns 1 and 3, created by removing edges with a length greater than 0.15. Row 1 shows the truthful sensor positions, while rows 2 to 5 are the results from MDS, MAP, SGD and DNS respectively.

Firstly, it is clear that MAP[2] is the best candidate algorithm, in terms of error. However, it requires significantly more computational time, and especially for dense networks. Secondly, the regularity of the problem has a significant effect on the performance of the graph embedding algorithms. MDS provides a near-perfect estimate of regularly-shaped points, skewed only slightly at the edges. This is because the degree of perimeter vertices is lower, meaning that more edges are synthetically added by the shortest path algorithm. These edges are always overestimated, and therefore skew the embedding. As expected, MAP performed significantly better than MDS in the case of graph irregularity, albeit at a higher computational cost. SGD does not seems to produce as good quality embeddings, even for complete distance matrices. DNS appears to perform well only on complete graphs. Our results support the claim by Broxton et al. [7] that spectral methods are typically only useful when used in conjunction with a refinement algorithm.

4.3 Solution Refinement

The previous section showed how the distance matrix is used to embed the *Encounter Graph*. The result of this step is an assignment of d-dimensional coordinates X to all v vertices in the *Encounter Graph* vertices. This one-shot approach provides a good initial estimate of the true embedding. However, it can be refined further to improve localization accuracy, at the cost of additional computation time. The *Solution Refinement* step provides this functionality, and it is an optional step of the proposed localization algorithm. Fig. 4.11 shows how the distance and weighting matrices are used to make small changed to an embedding, so that the weighted square differences between the predicted and observed measurements is minimized.

[2]In the original paper, this algorithm was coupled with a refinement step. In our performance tests, this refinement step was completely disabled.

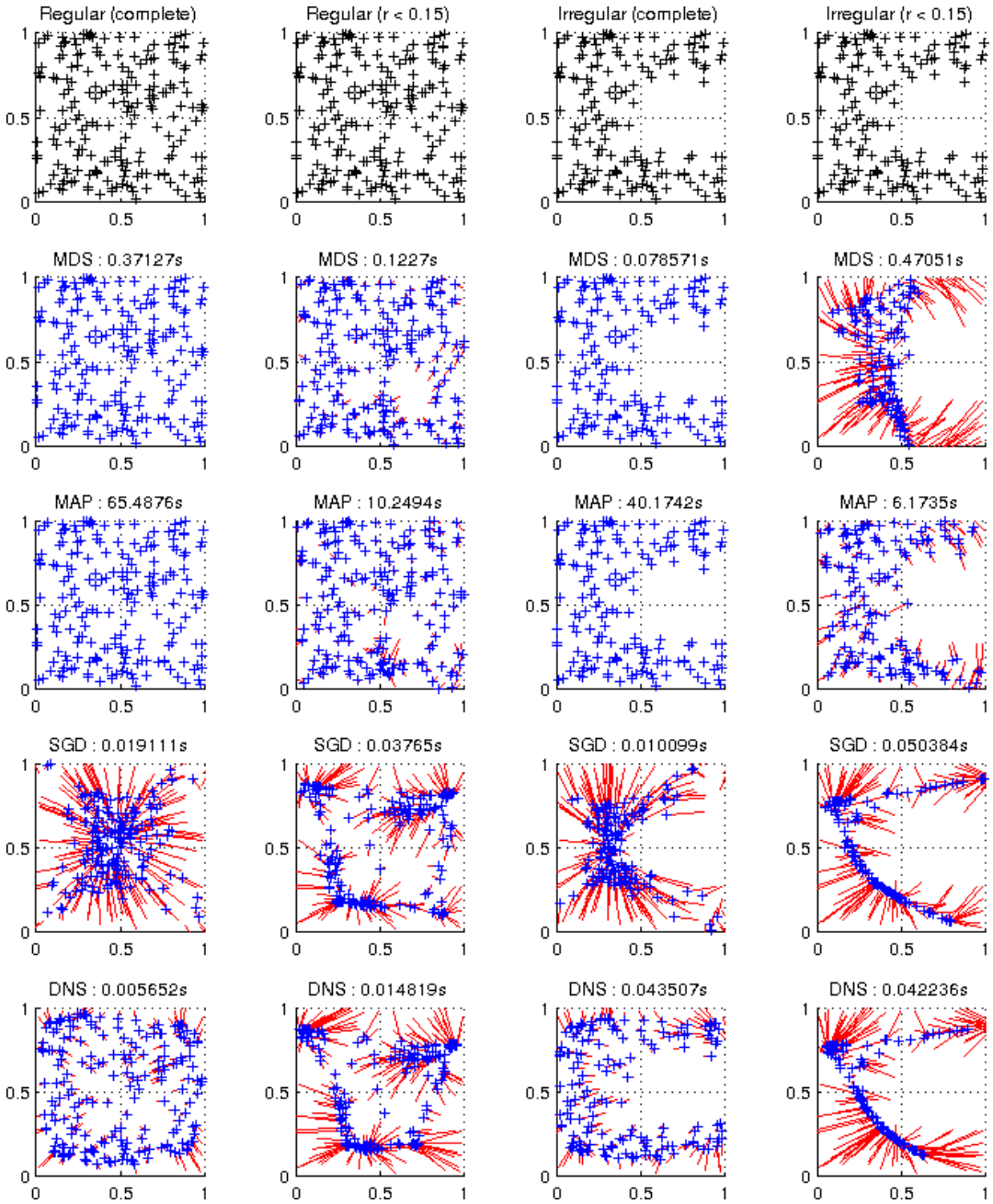


Figure 4.10: The image above illustrates the time and error performance of four graph realization algorithms. Column 1 shows a regularly-shaped problem with 200 vertices, having a complete distance matrix. Column 2 is a sparsified variant of Columns 1, created by removing all edges greater than 0.15. Column 3 is an irregular-shaped version of Column 1, created by removing with a square-shaped region of sensors. Column 4 is the sparsified variant of Column 2. The first row is the ground truth, and subsequent rows are the graph embedding results for MDS, MAP, SGD and DNS. Blue points are estimates, while red lines show error. The time in seconds taken to find an embedding is given in the graph title.

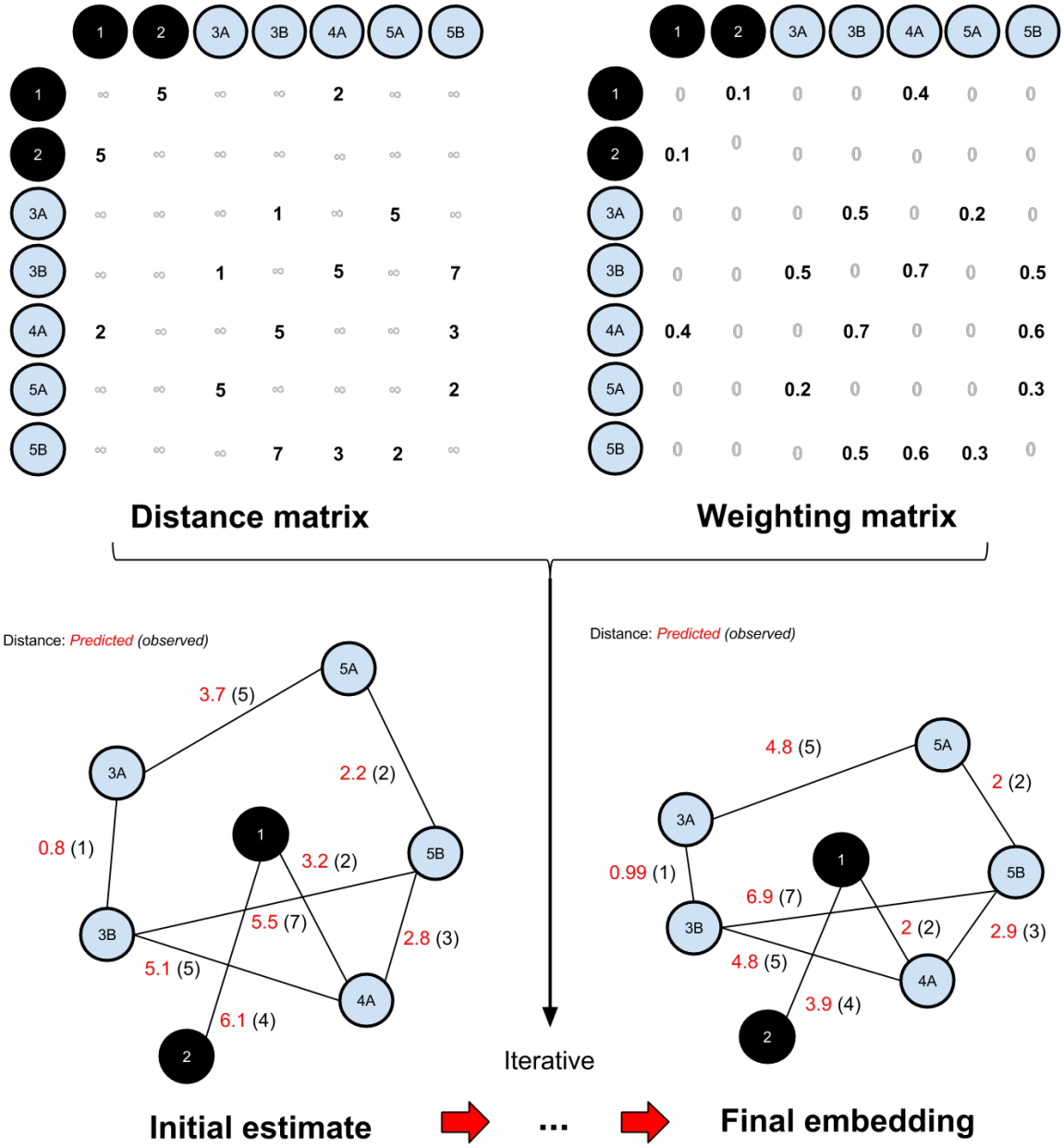


Figure 4.11: The Solution Refinement step of the proposed localization algorithm seeks to minimize the weighted error between the observed pairwise distances and the predicted pairwise distances (from the embedding).

Refinement methods distinguish themselves from realization methods in the sense that they are iterative, and take into account the amount of information provided by each measurement. The output coordinates of the previous step are denoted as an initial estimate X_0 to the *Solution Refinement* step. Let $[W]_{ij} = w_{ij}$ to be a $v \times v$ symmetric matrix of edge weightings, having zero values for missing edges. i.e. $\forall_{i,j} \in V : (i,j) \notin E \Rightarrow w_{ij} = 0$. The elements in W describe the variance associated with each measurement in the *Encounter Graph*.

The remainder of this section discusses two algorithms used for refinement. Both algorithms use the initial estimate X_0, pairwise distances D and weightings W to obtain an improved embedding estimate X_k after k iterations. The algorithms are iterative and terminate after ϵ_t iterations or when the improvement between two steps is less than ϵ_e. This provides a mechanism to balance time and error performance.

This section is structured as follows. Firstly, Sec. 4.3.1 discusses *Weighted Multidimensional Scaling* (WDS), a generalisation of MDS in which the objective function seeks to minimize weighted distance error. Then, Sec. 4.3.2 discusses *Non-linear Least-squares Optimization* (NLO). This approach refines the solution by choosing an update that maximizes measurement likelihood, given the current state.

4.3.1 Weighted Multidimensional Scaling

Weighted Multidimensional Scaling (WDS) is an extension to MDS that seeks to minimize the sum of weighted squared stresses. The stress function f is shown in Eqn 4.18, where Δ is a distance matrix, W is a weighting matrix, and $X = \{x_1, ..., x_v\}$ are a candidate set of vertex coordinates

$$f(X) = \sum_{<i,j>\in\Delta} w_{ij} \left(\left\| x_i - x_j \right\| - \delta_{ij} \right)^2 \tag{4.18}$$

$$\tilde{X} = \arg\min_X f(X) \qquad w_{ij} \in W, \quad \delta_{ij} \in \Delta \tag{4.19}$$

Unlike MDS, there exists no one-shot eigendecomposition that solves the stress minimization problem in Eqn 4.19. de Leeuw [18] developed a suitable iterative algorithm called *Scaling by Majorizing a Complicated Function* (SMACOF), the details for which are provided in Appendix B. The overall principle, however, is to carefully choose a second *majorizing* function $g(X,Y)$ which takes two arguments, X and Y, and satisfies $g(X,Y) \leq f(X)$ and $g(Y,Y) = f(Y)$. By virtue of the *sandwich equality*, a controlled selection of X and Y to minimize g also monotonically decreases $f(X)$.

4.3.2 Non-linear Least-squares Optimization

The overarching objective of *Non-linear Least-squares Optimization* (NLO) is to assign values to X so that the error function $e(X)$ expressed in Eqn. 4.20 is minimized. Each measurement i cherry-picks a subset of the elements in X and transforms them into a predicted measurement using a measurement function h_i. This prediction is compared to the observation z_i. In the localization problem both $h_i(X)$ and z_i are Euclidean distances. The diagonal matrix W^{-1} expresses the confidence in each distance estimate, which effectively weights the solution.

$$\begin{aligned} e(X) &= \sum_{i=1}^{|z|} \left(z_i - h_i\left(X\right)\right)^T W^{-1} \left(z_i - h_i\left(X\right)\right) \\ &= \|z - h(X)\|_{W^{-1}} \end{aligned} \tag{4.20}$$

The *normal equation* [84] used to find a δx that minimizes $e(x)$ is shown in Eqn. 4.21. It requires the first order linearization of the measurement model with respect to the state $H = \frac{\delta h}{\delta x}$. This equation is of the form $Ax = b$, where matrix A and vector b are known quantities. Solving this linear equation in a computationally-efficient manner has been the subject of much research [53, 58].

$$H^T W^{-1} H \delta x = H^T W^{-1} (z - HX) \tag{4.21}$$

4.4 Discussion

This chapter discussed the *Graph Realization* phase of the proposed localization algorithm, which is concerned with embedding the *Encounter Graph* in 2D or 3D space. Firstly, it was shown how to certify whether a given localization problem has a unique embedding. For 2D localization problems, uniqueness can be certified in polynomial time using a deterministic algorithm. However, no such algorithm exists for 3D localization problems, and so certification can only be carried out probabilistically. Then, it was shown how algorithms from static sensor localization can be used to embed the encounter graph. In the past, graph embedding has been restricted to static sensor localization. However the graph construction method used in the proposed localization algorithm allows these algorithms to be applied in a new way to mobile localization. Finally, this section discussed two methods for iteratively refining the embedding.

The next chapter assumes that the *Encounter Graph* has been embedded and refined, and the resulting vertex coordinates will be used to correct the *Inertial Trajectories*, and then project them into a global coordinate frame.

Chapter 5

Trajectory Reconstruction

The previous chapter showed how to certify whether a given range-based localization problem has a single solution, and then discussed estimation methods for calculating and refining solution. This chapter is concerned with *Trajectory Reconstruction* phase, in which the embedded graph vertices are used to back-correct the *Inertial Trajectories*. This phase is divided into two steps. In the *Drift Correction* step the Inertial Trajectories are threaded through a subset of the embedded graph vertices. In the *Trajectory Projection* step the embedded anchor positions and their corresponding global coordinates are used to obtain a transformation function. The transformation is applied to each point on all sensors' trajectories to obtain the *Projected Trajectories*.

The chapter is structured as follows. Sec. 5.1 discussed *Drift Correction* and provides two appropriate algorithmic solutions. One of these algorithms was taken from the literature, while the other is a novel contribution of this thesis. Sec. 5.2 then discusses *Trajectory Projection*, which uses Procrustes Analysis to find a suitable transformation that projects the trajectories into a global coordinate frame.

5.1 Drift Correction

Recall that the *Motion Model* produces an *Inertial Trajectory* for each sensor. The *Inertial Trajectory* is a high-resolution discrete-time estimate of a sensor's position, which drifts with time, as a result of of sensing error. The embedded *Encounter Graph* contains a course-grained discrete-time estimate of a sensor's position, which does not drift. The objective of the *Drift Correction* step is to use the vertex coordinates from the embedded *Encounter Graph* to correct the *Inertial Trajectory.* The correction itself is performed for each sensor in a piecewise fashion between encounter points.

The *Motion Model* provides a time-indexed state estimate x_t for each sensor. The *Drift Correction* step uses a subset of the elements of x_t that correspond to position,

denoted p_t. Recall that each vertex in the embedded graph represents the position of an anchor, or the position of a sensor at a given point in time. Now, consider evaluating the trajectory of a particular sensor from the perspective of the embedded *Encounter Graph*. To do this, a time-ordered set of graph vertices $Q = \{q_1, ..., q_T\}$ belonging to this sensor. As a direct result of the methodology used to construct the *Encounter Graph*, the first and last element in Q correspond to the sensor's position at time 0 and T respectively. The key intuition behind *Drift Correction* is to use two adjacent vertices $(y_i, y_j) \in Q$ to correct the corresponding *Inertial Trajectory* segment $p_{i:j}$. In so doing, we mitigate a portion of the drift over long periods is mitigated, while retaining the fine-grained detail that the *Inertial Trajectory* provides.

The sections that follow discuss two different piecewise drift correction algorithms. Sec. 5.1.1 outlines an algorithm from the literature, which will henceforth be referred to as *Linear Drift Correction* (LIN). This algorithm corrects each segment of the trajectory assuming that the *Inertial Trajectory* suffers drift as an additive fixed bias. One contribution of this thesis is an alternate piecewise drift correction algorithm called *Radial Drift Correction* (RAD), which assumes that the *Inertial Trajectory* suffers drift as a scaled rotational component.

5.1.1 Linear Drift Correction

Linear Drift Correction (LIN) was taken from Constandache et al. [14]. As its name suggests, this algorithm provides drift correction under the assumption that the *Inertial Trajectory* suffers a fixed additive linear drift per time unit. Fig. 5.1.1 shows the basic operation performed by the algorithm. The red and blue dashed circles in Box A represent p_i and p_j respectively, and the curved line is therefore the trajectory segment $p_{i:j}$. The objective is to map $p_{i:j}$ to start and end at the red and blue solid circles, which have coordinates y_i and y_j from the embedded *Encounter Graph*.

LIN begins by shifting the sensor's estimated trajectory to begin at y_i. This can be thought of as shifting the black curve from Box A to Box B in Fig. 5.1.1. We call this shift the *offset*. LIN then calculates the *correction* vector required to move p_j to y_j. This correction vector is shown by a dashed arrow in Box C in Fig. 5.1. The correction vector is then proportionally added to the shifted trajectory based on time. Eqn 5.1 shows how LIN corrects an *Inertial Trajectory*.

$$\hat{p}_t = p_t + \underbrace{(y_i - p_i)}_{\text{offset}} + \underbrace{\left(\frac{t-i}{j-i}\right)}_{\text{proportion}} \underbrace{(y_j - p_j + y_i - p_i)}_{\text{correction}} \qquad \forall_t : i \leq t \leq j \qquad (5.1)$$

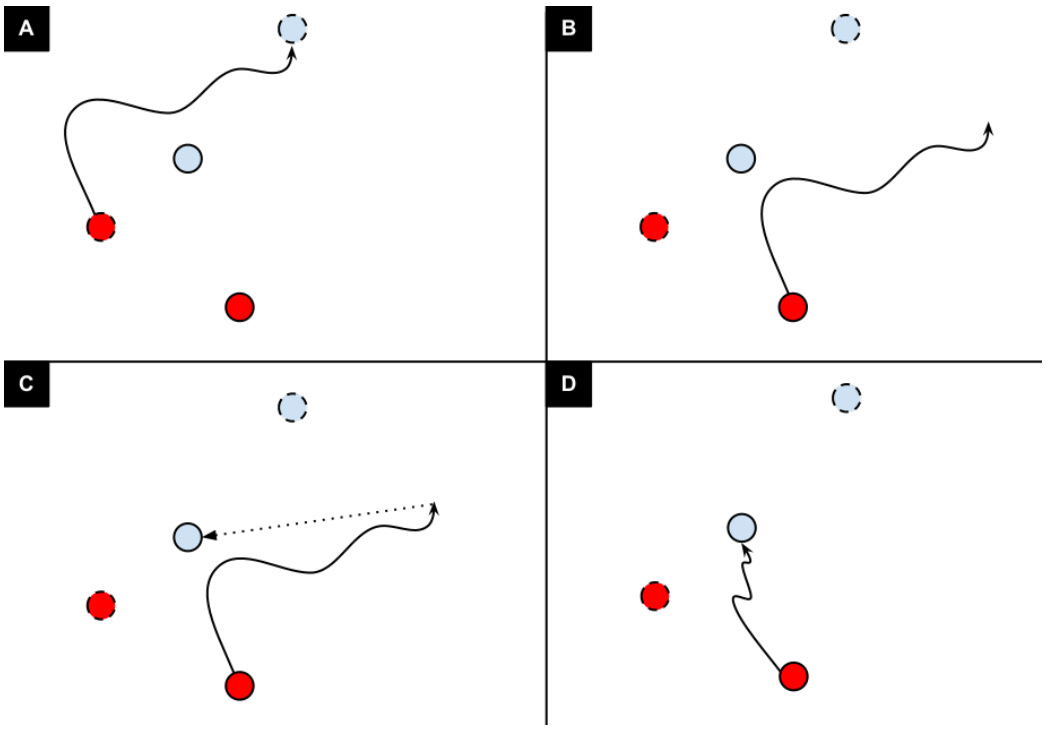


Figure 5.1: *Linear Drift Correction* (LIN) begins by shifting the Inertial Trajectory to begin at the first encounter (Panel A to B). A correction vector, shown as a dotted arrow in Panel C, is then added proportionally to the trajectory with respect to time. The resulting corrected trajectory is given in Panel D.

5.1.2 Radial Drift Correction

Radial Drift Correction (RAD) is a novel contribution of this thesis. RAD makes the assumption that the Inertial Trajectory suffers drift that can be expressed as a scaled rotation. Fig. 5.2 uses the same simple example from the previous section to illustrate how RAD works. Like LIN, RAD begins by shifting the *Inertial Trajectory* to start at y_i. Once the shift takes place, RAD calculates the angle α between the direction of the shifted trajectory and the encounter points (see the angle marked in Box B of Fig. 5.2). RAD then rotates the shifted trajectory so that it aligns with the encounter points. Then, RAD calculates a scale factor, that when multiplied with the trajectory, results in its end point aligning with y_j. This calculation is shown in Box C, while the final corrected trajectory is shown in Box D.

Eqn 5.2 shows the method RAD used to correct an *Inertial Trajectory* segment $p_{i:j}$, given two vertex coordinates y_i and y_j from the Encounter Graph. It relies on a matrix R, which rotates the current position about y_i by the angle shown in Box B of Fig. 5.2. Eqn 5.2 shows the general method RAD uses to correct a single point on the *Inertial Trajectory* p_t.

$$\hat{p}_t = y_i + \frac{\|y_j - y_i\|}{\|p_j - p_i\|} R\,(p_t - p_i) \qquad \forall_t : i \leq t \leq j \tag{5.2}$$

The 2D implementation of RAD uses the "perp dot product" of p and q to rotate the current point by the signed angle α about the origin. R is a 2D rotation matrix derived from α.

$$q = \begin{bmatrix} q_x \\ q_y \end{bmatrix} = y_j - y_i \tag{5.3}$$

$$p = \begin{bmatrix} p_x \\ p_y \end{bmatrix} = p_j - p_i \tag{5.4}$$

$$\alpha = \text{atan2}\,(p_x q_y - p_y q_x, p \cdot q) \tag{5.5}$$

$$R = \begin{bmatrix} cos\alpha & -sin\alpha \\ sin\alpha & cos\alpha \end{bmatrix} \tag{5.6}$$

The 3D implementation of RAD requires the use of quaternions. The function NORMALIZE divides a vector by its magnitude. RAD constructs a unit quaternion q from a rotation θ and an axis v, derived from the vector difference between the start and end points of the *Inertial Trajectory* and *Correction Vertices*. The matrix R is a

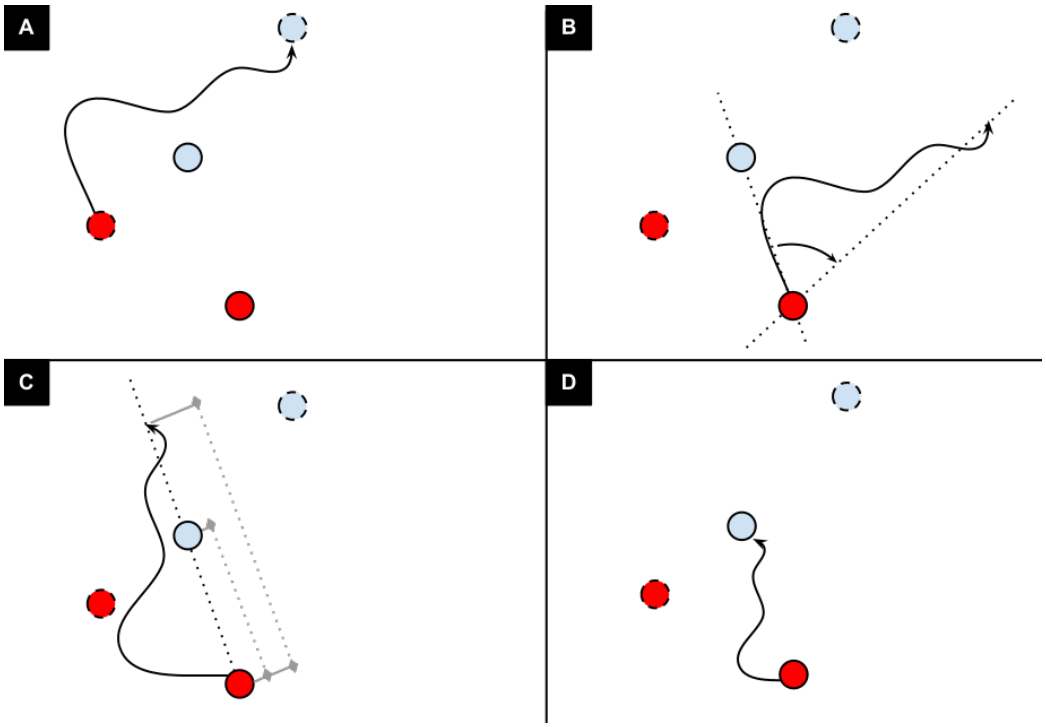


Figure 5.2: In *Radial Drift Correction* (RAD) the Inertial Trajectory segment is first shifted to begin at the first encounter (Panel A to B). The segment is then rotated into alignment with the displacement vector between the two encounter points (Panel B to C). Finally, the segment is scaled so that its end point aligns with the second encounter (Panel C to D).

3D rotation matrix, derived from the unit quaternion (see Schinstock [80]).

$$v = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} = \text{normalize}\Big([y_j - y_i] \times [p_j - p_i]\Big) \tag{5.7}$$

$$\theta = \arccos\left([y_j - y_i] \cdot [p_j - p_i]\right) \tag{5.8}$$

$$q = \begin{bmatrix} q_w \\ q_x \\ q_y \\ q_z \end{bmatrix} = \text{normalize} \begin{bmatrix} \cos\theta \\ v_x \sin\theta \\ v_y \sin\theta \\ v_z \sin\theta \end{bmatrix} \tag{5.9}$$

$$R = R_{r \leftarrow b}(q) \tag{5.10}$$

5.2 Trajectory Projection

At least $d+1$ algebraically-independent correspondences are required to project vertices from a local to a global coordinate frame, both in $\mathbb{E}^d$. Equivalently, a generically globally rigid Encounter Graph with at least $d+1$ anchors is required. If there are

too few anchors, then it is impossible to constrain all external degrees of freedom, and *Trajectory Projection* cannot take place. Nevertheless, in some cases this is an acceptable outcome. For example, in geographic routing algorithms [9].

In cases where $d+1$ algebraically independent correspondences exist, *Procrustes Analysis* (see Cox and Cox [15]) is used to find a transformation f that best maps the vertex coordinates from the embedded Encounter Graph Y to their corresponding global coordinate C. The transformation is calculated to minimize the mean square error between $f(Y)$ and C. The transformation comprises of a scale s, rotation R and translation T. Eqn 5.11 shows how the *Corrected Inertial Trajectories* are projected from local X to global X_G coordinates.

$$X_G = sXR + T \tag{5.11}$$

What follows is a brief description of how the transformation is obtained. Let Y and C be $m \times d$ matrices representing the d-dimensional coordinates of $m > d+1$ anchors in the local and global coordinate frame respectively.

1. **Calculating T** - Firstly, the translation vector T is calculated by subtracting the row mean of Y from the row mean of C. The result is a displacement vector that, when added to Y, maximizes its overlap with C.

2. **Calculating s** - The centroid of the matrices Y and C are then shifted to the origin, by subtracting their respective means. The shifted matrices are respectively denoted Y_c and C_c. Let $\text{ssd}(Y_c)$ and $\text{ssd}(C_c)$ be the sum square distance of all points from the origin for the respective point constellations. When multiplied to Y_c, the scale factor $s = \frac{\text{ssd}(C_c)}{\text{ssd}(Y_c)}$ disperses or compresses constellation Y_c, such that its overall spread matches C_c.

3. **Calculating R** - Finally, a rotation matrix R is calculated, which orients Y_c to maximize overlap with C_c. Schönemann [81] proved that this can be achieved using the Singular Value Decomposition (SVD) of $Y_c^T C_c = U\Sigma V^*$. The matrix that rotates sY_c into alignment with C_c is given by the equation $R = UV^*$.

5.3 Discussion

This chapter showed how the embedded encounter graph is used to correct the inertial trajectories provided by the *Motion Model*, hence correcting them for drift. The process is divided into two steps. In the first step, *Drift Correction*, the *Inertial*

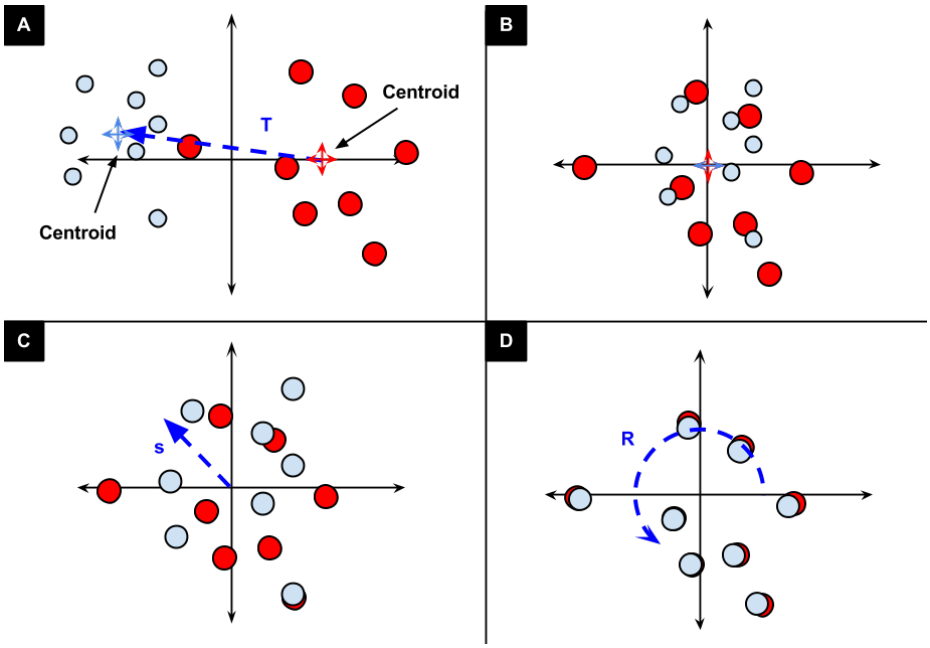


Figure 5.3: (A) The translation vector T shifts the centroid of points in Y (red) to C (blue); (B) Both centroids are shifted to the origin, resulting in constellations Y_c and C_c ; (C) Y_c is scaled by a factor s, so that its spread (sum squared distance of each point from the origin) equals the spread ofC_c ; (D) Singular Value Decomposition (SVD) is used to find the optimal rotation matrix R that brings sY_c and C_c into alignment.

Trajectories are threaded through a subset of the embedded graph vertices to produce *Corrected Trajectories*, in order to correct their relative structure. This thesis proposes a novel algorithm, *Radial Drift Correction*, which applies corrections to the trajectories as scaled rotation errors. This has the advantage of not shearing the trajectory, at the cost of introducing discontinuities at the encounter points. Finally, under the assumption that a sufficient number of anchors are available, the trajectories are projected into a global coordinate frame. It was shown how Procrustes Analysis can be used to find a suitable transformation, given anchors.

The proposed cooperative localization algorithm has now been completely described. The next chapter discusses the experimental setup for a simulation study conducted to evaluate the performance of the proposed algorithm for cooperative pedestrian localization.

Chapter 6

Experimental Setup

This chapter describes a simulation study that was carried out to measure the performance of the proposed localization algorithm. The simulation models several pedestrians (sensors) moving within a rectangular region in 2D or 3D space. These sensors periodically encounter each other or anchors by exchanging radio beacons. The resulting traces are corrupted using motion and radio error models, before being passed to the proposed localization algorithm as measurements. Parameters for these error models are derived in Sec. 6.1. Sec. 6.2 then outlines how the trajectories are generated by the simulator. Sec. 6.3 shows the methodology that is followed to evaluate the proposed localization algorithm. Sec. 6.4 concludes this chapter with specific implementation details, including referenced third-party software libraries.

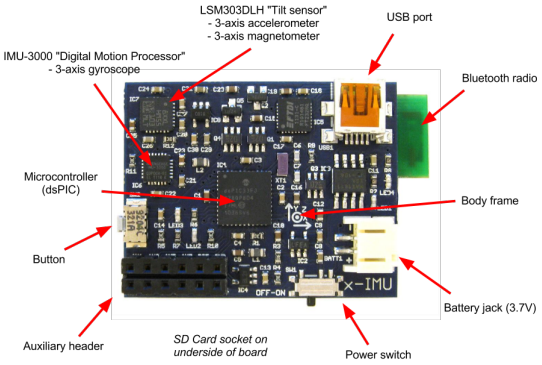
6.1 Error models

This section discusses how realistic inertial and radio error models were obtained for use in the simulation study. Sec. 6.1.1 reports on an experiment that was carried out to calculate gyroscope and accelerometer noise covariances, which are then used to calibrate the PDR algorithm. This algorithm was used in a second experiment to track a foot around a single floor of a building. The resulting trajectory was compared to a ground truth to calculate parameters for a motion model. Sec. 6.1.2 discusses the log-distance radio error model that is adopted from the literature.

6.1.1 Motion error

This chapter discusses an experiment that was conducted to calibrate the PDR algorithm in Sec. 3.2. In this experiment gyroscope and accelerometer measurements were

(a) The foot-mounted IMU (with enclosure)



(b) Description of the x-IMU hardware

Figure 6.1: The experimental setup used to calibrate the PDR algorithm.

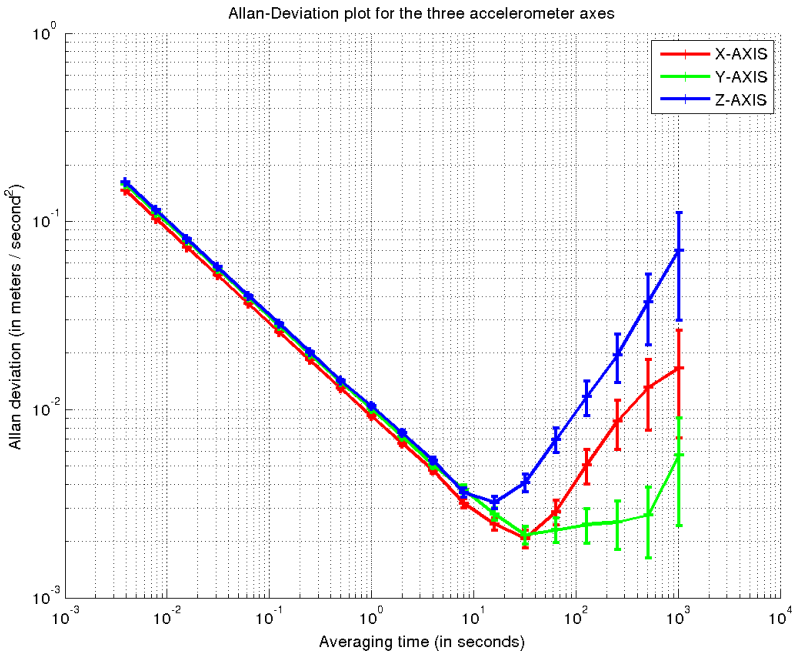
recorded at 256 Hz from the x-IMU[1] shown in Fig. 6.1a. The device was configured in "IMU mode", which implies the following. Both the accelerometer and gyroscope output factory-calibrated units, which account for device-specific fixed scaling and bias errors, as well as temperature-dependent gyroscope biases. However, the in-run bias of the gyroscope is not corrected as this would presume that the device is not undergoing any significant linear acceleration, which is not true for walking. [60].

Allan Deviation is a statistic that is used to identify the frequency noise stability of a given signal. It assumes that the signal is sampled at a fixed rate, resulting in a total of n sample points $x_1, \cdots, x_n$. This signal is then binned into fixed-width intervals of t seconds. Let m be the number of bins with at least nine data samples, and $x_i(t)$ be the average (mean) signal value in bin i. The *Allan Deviation* for a bin width of t seconds is the square root of the *Allan Variance* $\sigma_x^2(t)$, calculated in Eqn 6.1. An Allan Deviation plot shows log-log relationship between t and $\sigma_x^2(t)$.

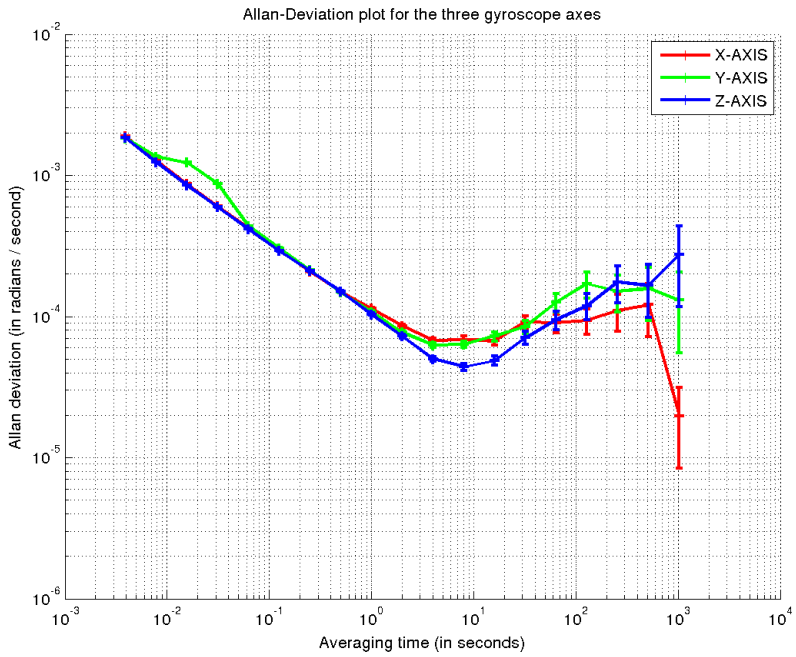
$$\sigma_x^2(t) \quad = \quad \frac{1}{2(m-1)} \sum_{i=1}^{m-1} \left(x_{i+1}(t) - x_i(t)\right)^2 \tag{6.1}$$

A short experiment was conducted to evaluate the frequency noise stability of the x-IMU. The device was placed on a stationary platform indoors, with the Z-axis roughly aligned with gravity. Ten minutes worth of calibrated gyroscope and accelerometer measurements were then recorded. Fig. 6.2a shows the Allan Deviation for the x-IMU unit's accelerometer. The initial linear slope for all axes indicates that measurements were corrupted by additive white noise; the inverse relationship between averaging time and variance indicates that averaging can be used to mitigate

[1] http://www.x-io.co.uk



(a) Allan-Deviation plot for 3-axis accelerometer



(b) Allan-Deviation plot for 3-axis gyroscope

Figure 6.2: Frequency stability of the x-IMU inertial sensors

some of this white noise. The X and Y noise floors are reached at 20 seconds (there is a turning point in their Allan Deviation curve). This means that wider averaging cannot help remove any additional noise. Fig. 6.2b shows the Allan Deviation for our x-IMU unit's gyroscope. It is immediately evident that all three axes suffer a random walk component at an averaging period of between 2 to 5 seconds.

Based on these frequency stability experiments, it was concluded that IMU gyroscope and accelerometer suffered white noise components with variance σ_ω and σ_α, given by the covariance matrices in Eqn 6.2 and 6.3 respectively. The noise covariance values for the velocity updates were assumed to be $\sigma_v = 1e-4$ ($1^{cm}/_s$).

$$\sigma_\omega = \begin{bmatrix} 1.6083e-08 & 1.3449e-08 & 3.5908e-09 \end{bmatrix} \quad (6.2)$$

$$\sigma_\alpha = \begin{bmatrix} 3.6258e-06 & 5.3334e-08 & 9.8073e-07 \end{bmatrix} \quad (6.3)$$

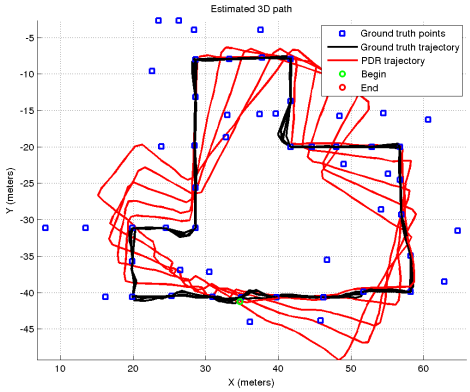
Following calibration, an experiment was carried out to model the position drift exhibited by a PDR filter that processes measurements from a foot-mounted x-IMU. In order to model the position error as a function of time, knowledge of the ground truth trajectory was required. To achieve this, 55 visual markers were placed on the ground along an office corridor. The test subject carried a downward-facing video camera, along with an x-IMU fixed rigidly to his right foot. Both the video camera and x-IMU were time synchronized, allowing for decimeter-accuracy positioning.

The test subject walked counter-clockwise in a closed loop, five times around Level 4 of the Oxford Department of Computer Science's Wolfson Building. Fig. 6.3a shows a sample frame of this video. Fig. 6.3b shows the ground truth visual markers as blue squares, the ground truth trajectory as a black line, and the inertial trajectory as a red line. The start and end positions are also indicated in the figure. The ground truth trajectory was obtained by performing piecewise drift correction on the inertial trajectory, using visual markers as correction points. For clarity, the inertial trajectory has first been shifted to start at the same point as the ground truth trajectory, and then rotated to align the trajectories in the first few seconds.. In reality, the inertial trajectory starts at an arbitrary position and orientation.

The power of the ZUPT tracking method is that velocity errors only grow open-loop during the stride phase of walking. When the foot hits the ground, this event is detected and used to reset the velocity to zero. As a result of this, foot-mounted IMUs clamp per-step position errors to within a few centimeters. However, this small error accumulates with time to bias the position estimate.

This thesis models position this bias as a Gaussian random walk. More specifically, it assumes that each of the X-Y-Z velocity components perturbed by errors

(a) Sample frame from the ground truth video



(b) Truth (black) and inertial (red) trajectories

Figure 6.3: Pedestrian dead reckoning results for a 110 second walk indoors.

that are drawn from independent, zero-centred Gaussian distributions. The ground truth and inertial data shown in Fig. 6.3 were used to find the standard deviations $\begin{bmatrix} 0.3788 & 0.3949 & 0.0469 \end{bmatrix}$ for the three independent distributions. Fig. 6.4 compares the observed displacement error for the first 200 seconds of the trace data to a synthetically generated displacement error using the parameterised distributions above.

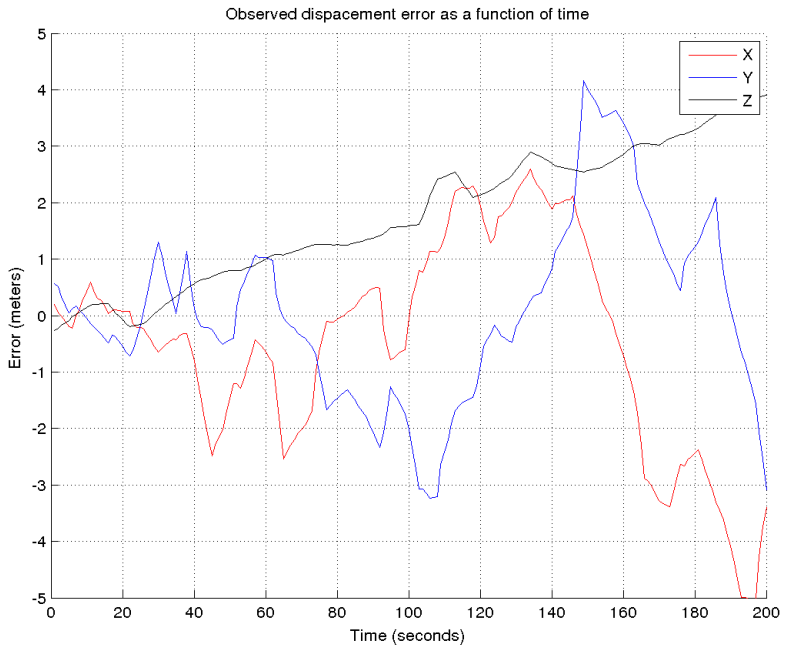
6.1.2 Radio error

Time of arrival (TOA) and received signal strength (RSS) are the only practical radio measurements that relate directly to range. However, TOA systems typically require specialised hardware[2]. For this reason this thesis considers RSS only. Chapter 3 outlined the *log-distance path loss model*, which is repeated in Eqn 6.4.

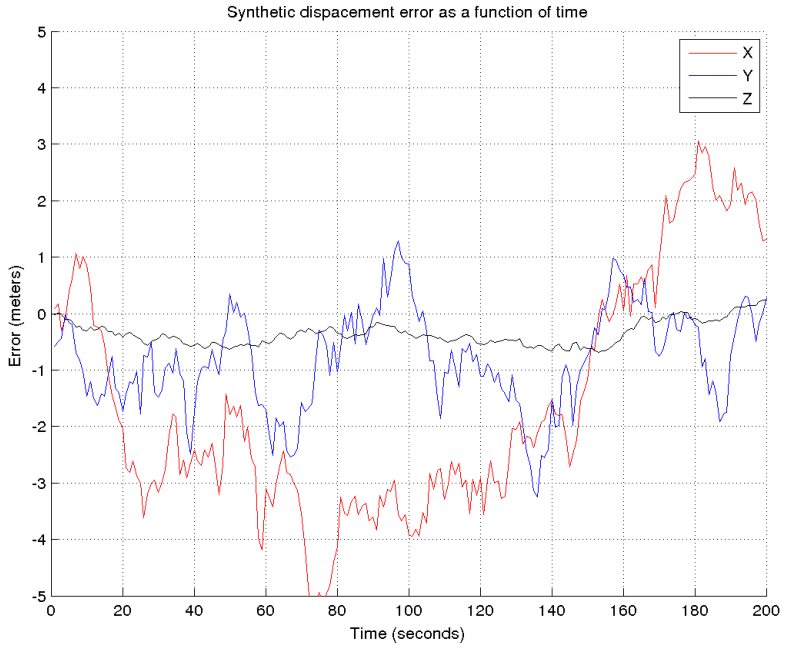
$$\gamma_r = \gamma_{r_0} - 10\eta \log_{10}\left(\frac{r}{r_0}\right) + e, \qquad e \sim \mathcal{N}\left(0, \sigma_r^2\right) \tag{6.4}$$

Dil and Havinga [21] conducted an experiment with 2.4GHz Zigbee radios in order to measure realistic parameters for the log-distance path-loss model, namely γ_{r_0}, η and σ_r^2. Radio packets were transmitted from ten different sensors and one receiver. The receiver was moved to 80 different locations in an open indoor environment. At each point 100 radio packets were transmitted in 38 different frequency bands. The fitted path loss model parameters are shown in Eqn. 6.5. This thesis will use these measured parameters for synthetically generating radio encounters.

[2]See http://www.ubisense.net and http://www.nanotron.de for example platforms



(a) Observed random walk



(b) Synthetic random walk

Figure 6.4: Observed and synthetic position drift

$$\gamma_{r_0} = -21.2\text{dBm}, \qquad \eta = 2.29, \qquad \sigma_r^2 = 3.16\text{dBm} \tag{6.5}$$

6.2 Synthetic trace generator

The simulator synthetically generates localization problems in order to test a wide variety of scenarios. The simulator is configured with an experiment region, duration, number of sensors, number of anchors and trace resolution. The trace resolution is a parameter that decides the resolution at which the encounter graph is built. For example, a trace resolution value of 5 seconds yields an encounter graph with measurements merged into 5 second bins, effectively trading off complexity (the size of the resulting encounter graph) for correction resolution.

The synthetic trace generator randomly assigns each anchor a fixed starting position, and each sensor a random starting position and orientation. Sensors are restricted to move at a fixed velocity forward or backward along any axis of the search space. The fixed velocity was taken as the average velocity for the trace data set (Fig. 6.3) which turned out to be 1.2m per second. The simulator creates a path for each sensor according to the following simple mobility model, which was designed to mimic movement through corridors. The sensor begins by selecting a direction vector orthogonal to the current direction vector. A positive distance is then chosen randomly from a range of positive real values, upper-bounded by the size of the region in the direction of motion. The trajectory is then synthetically created at 1Hz, by integrating a constant walking speed through the current direction. If a boundary is hit, a new direction vector is chosen. The entire trajectory is passed through a forward-backward filter in order to smooth out 90 degree direction changes.

A sample synthetic trace for a single sensor is shown in Fig. 6.5. The ground truth, shown as a dashed black line, was generated using the mobility model described above. Encounters between devices are synthetically generated using the radio model: if any predicted RSS between a sensor and another device – sensor or anchor – exceeds a threshold (-45dBm in the diagram) an encounter is recorded (the blue lines represent radio encounters). The RSS of each recorded encounter is perturbed with zero-mean Gaussian error, using parameters from the radio model. To generate the inertial trjectory, the ground truth trajectory is first perturbed with a random walk, and then shifted to the origin (0,0) and given a random starting orientation. The inertial trajectory is shown as the solid black line in the diagram.

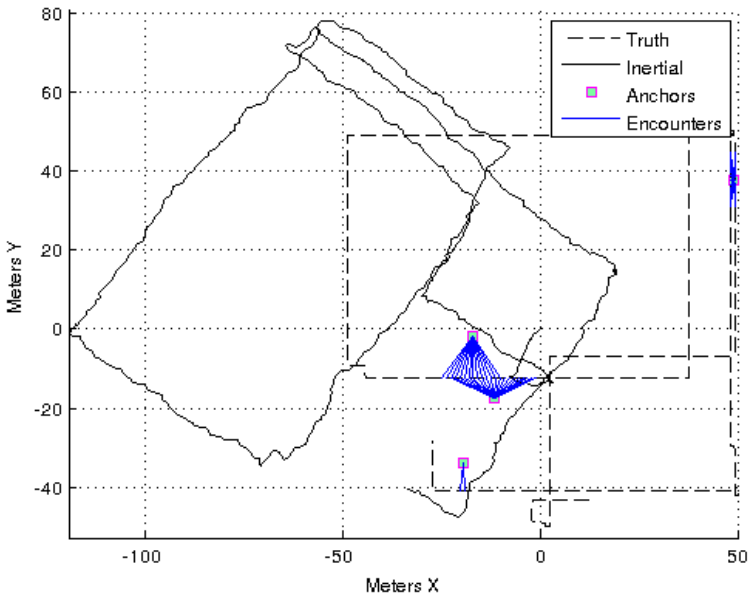


Figure 6.5: A synthetic trace for a single sensor moving in a 100×100 meter region, with a -45dBm SNR threshold. Cyan squares represent the known anchor points. The dashed black line is the synthetically-generated ground truth, while the solid black line is its drift-corrupted counterpart (the inertial trajectory). Blue lines show encounters between the sensor and anchors.

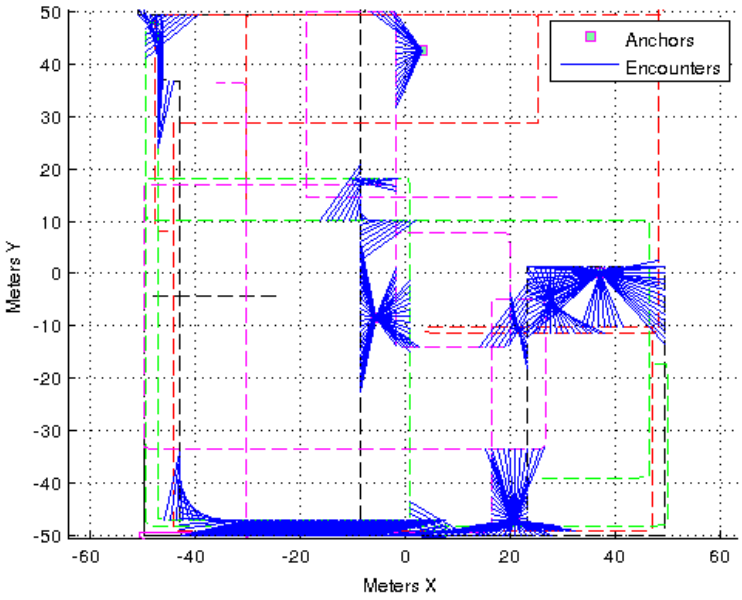


Figure 6.6: A more complex synthetic trace for five sensors moving within a 100×100 meter region, with a -45dBm SNR threshold. Cyan squares represent the four known anchor points. Each pedestrian's ground truth trajectory has been given a different colour. Dead reckoning trajectories are omitted for clarity.

Fig. 6.6 shows a more complex problem containing five sensors and four anchors. Again, 600 second traces are generated with a radio SNR of -45dBm. This time, both peer-to-peer and peer-to-peer radio encounters are shown by blue lines. Purely cooperative localization problems can be generated by setting the number of anchors in the simulator to zero. Since there are no anchors in such problems, the localization solution can only be represented in a relative coordinate frame.

6.3 Methodology

The objective of the simulation study is to evaluate the performance of th proposed localization algorithms for a variety of different network conditions. Sec. 6.3.1 lists the constant and variable parameters used by the simulator, while Sec. 6.3.2 describes the performance metrics used to evaluate components of the proposed model. Finally, Sec. 6.3.3 describes the experimental methodology that was followed.

Table 6.1: Simulation parameters

Parameter	Type	Value(s)	Unit
Experiment region	Variable	2D:100 × 100, 3D: 100 × 100 × 40	m
Number of sensors	Variable	1, 2, 4, 8	-
Experiment duration	Variable	450, 900, 1800, 3600	s
Number of runs	Constant	30	-
Number of anchors	Variable	0, 4, 8, 16	-
Graph resolution[3]	Constant	10	s
Sensor speed	Constant	1.2	m/s
Anchor-anchor variance	Constant	1e-4	m

6.3.1 Simulation parameters

The seven parameters used in the simulator study are listed in Tab. 6.1. The pedestrians (sensors) are assumed to move at a constant velocity within a fixed size region. The dimensionality (2D or 3D), number of sensors, number of anchors and experiment duration are varied. Each parameter combination is tested 30 times in order to obtain statistical significance. This resulted in 3840 simulations runs, which were used to evaluate the performance of the proposed localization algorithm.

6.3.2 Performance metrics

The following three performance metrics will be used to evaluate the performance of the proposed localization algorithm:

1. **Absolute Time Required** (ATR) - This is a measure of the total time in seconds required by an algorithm, or a component of the algorithm, to fulfill its task. It is measured in CPU clock cycles using Matlab's PROFILE tool, in order to mitigate the effect of background operating system tasks.

2. **Projected Position Error** (PPE) - For problems that are generically globally rigid with a sufficient number of anchors it is possible to express the sensor trajectories in a global reference frame. The *Projected Position Error* (PPE) simply describes the average distance of the final solution from the ground truth as the *Root Mean Square Error* (RMSE). This is defined as follows. Consider a set ε of errors, drawn in a consistent manner. Such errors may, for example, represent the Euclidean distance separating corresponding points (in time) along the estimated and ground truth trajectories. If the cardinality of the error set is denoted $|\varepsilon|$, then the RMSE is calculated by Eqn 6.6.

$$\epsilon = \sqrt{\frac{1}{|\varepsilon|}\sum_{e\in\varepsilon} e^2} \tag{6.6}$$

1. **Graph Realization Stress** (GRS) - This metric assesses how well graph realization performs, for both cooperative (no anchors) and infrastructure problems. It describes the average residual error of pairwise distances between all vertices in the realized graph and their true positions (from the FAK realization model). Eqn 6.7 shows how the GRS is calculated for a graph with edge set E and two realizations of the vertices, V and V', corresponding to the known ground truth positions and the graph realization estimate respectively.

$$\epsilon = \frac{1}{|E|}\sum_{(i,j)\in E} \left| \|V_i - V_j\|_2 - \left\|V_i^{'} - V_j^{'}\right\|_2 \right| \tag{6.7}$$

6.3.3 Experimental process

The proposed localization algorithm can be thought of as providing a means to (a) evaluate whether a given problem is localizable, (b) solve the problem using graph realization, and (c) use the solution to correct the trajectories. The experiment was designed to test these three components in isolation of each other. With this in mind, three individual experiments were carried out.

1. **Rigidity analysis** - In this step the rigidity of all 2D and 3D problems is analysed. The objective is to firstly determine the overall rigidity of the 3840 simulation runs, and to secondly conduct a sensitivity analysis to measure the effect of the number of sensors, anchors and experiment duration on rigidity.

2. **Graph realization** - In this step only localizable graphs will be taken into consideration. The objective is to measure the time and error performance of graph embedding and refinement algorithms. Tab. 6.2 shows the four realization and two refinement algorithms that were implemented, as well as two additional control algorithms described below:

    - *Fake Tracking* (FAK) - This is an artificial graph realization algorithm that mimics perfect graph realization by drawing vertex coordinates from the ground truth. While not useful in a real world setting, this model provides a good mechanism for evaluating the performance of drift correction algorithms, as it mitigates any error introduced by *Graph Embedding.*

Table 6.2: List of algorithms tested in the simulation

Step	Acronym	Full name	Ref.
Realization	MDS	Multidimensional Scaling	Sec. 4.2.1
	MAP	Distributed Multidimensional Scaling	Sec. 4.2.2
	SGD	Spectral Graph Drawing	Sec. 4.2.3
	DNS	Degree-normalized Spectral Graph Drawing	Sec. 4.2.4
	FAK	Control	Sec. 6.3.3
Refinement	WDS	Weighted Multidimensional Scaling	Sec. 4.3.1
	NLO	Non-linear Least-squares Optimization	Sec. 4.3.2
	IGN	*Ignored (nothing is done)*	Sec. 6.3.3
Correction	LIN	Linear Drift Correction	Sec. 5.1.1
	RAD	Radial Drift Correction	Sec. 5.1.2

- *Ignore* (IGN) - This just signifies that either the *Graph Realization* or *Solution Refinement* is skipped completely. This will be useful for testing the *Graph Embedding* performance without *Solution Refinement.*

3. **Drift correction** - In this step it will be assumed that realization is carried out perfectly, and the resulting vertex coordinates are used to correct the inertial trajectories. The objective is to critically compare the two piecewise corrections algorithms in Tab. 6.2 for both 2D and 3D localization problems.

6.4 Software implementation

The simulator ran on a computer with an Intel i7 dual-core processor and 8GB of RAM. The simulation itself was written in MATLAB 2013a in a 64-bit Ubuntu Linux 13.10 environment. The simulation makes use of the following third-party software:

- The *Spectral Graph Drawing*, *Degree Normalised Spectral Graph Drawing*, *Non-linear Least-squares Optimization*, *Multidimensional Scaling*, *Weighted Multidimensional Scaling* and *Procrustes Transform* were implemented in MATLAB[4].
- The algorithm for deterministically certifying generic global rigidity in 2D uses the pebble game implementation from KINARI[5], and connectivity testing implementations from the OPEN GRAPH DRAWING FRAMEWORK[6].

[4]http://www.mathworks.co.uk
[5]http://kinari.cs.umass.edu
[6]http://www.ogdf.net

- The MDS-MAP graph realization algorithm was adapted from PARC's implementation[7].
- The CERES-SOLVER [8] by Sameer Agarwal et al. was used to solve large, sparse non-linear least-squares problems. CXSPARSE was used because SPARSESUITE had BLAS and LIBLAPACK conflicts with MATLAB libraries.

6.5 Discussion

This chapter described a simulation that was carried out to assess the performance of the proposed localization algorithm. It showed how parameters were obtained for suitable motion and radio models, which were then used by a synthetic trace generator to create sample localization problems. The chapter also described the experimental methods and materials, which included a discussion of the simulation setup, performance metrics, methodology third party libraries.

The next chapter focuses entirely on the results from the simulation study.

[7] https://code.google.com/p/sim4j
[8] https://code.google.com/p/ceres-solver

Chapter 7

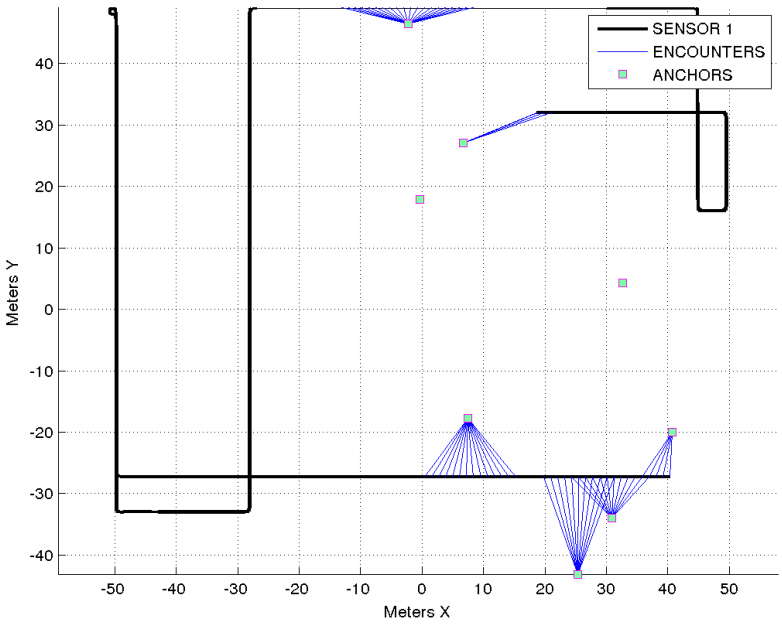
Results and Discussion

This chapter discusses results from the simulation study that was carried out to evaluate the performance of the proposed offline, range-based localization algorithm. The accuracy with which one is able to localize is affected by the following three sources of uncertainty: (i) the rigidity of the graph formed by the measurements, (ii) the quantity of error introduced by realizing the graph, because of noisy measurements or solution approximation, and (iii) the quantity of error introduced by trajectory reconstruction. This chapter will be divided into three sections; one devoted to discussing each source of error.

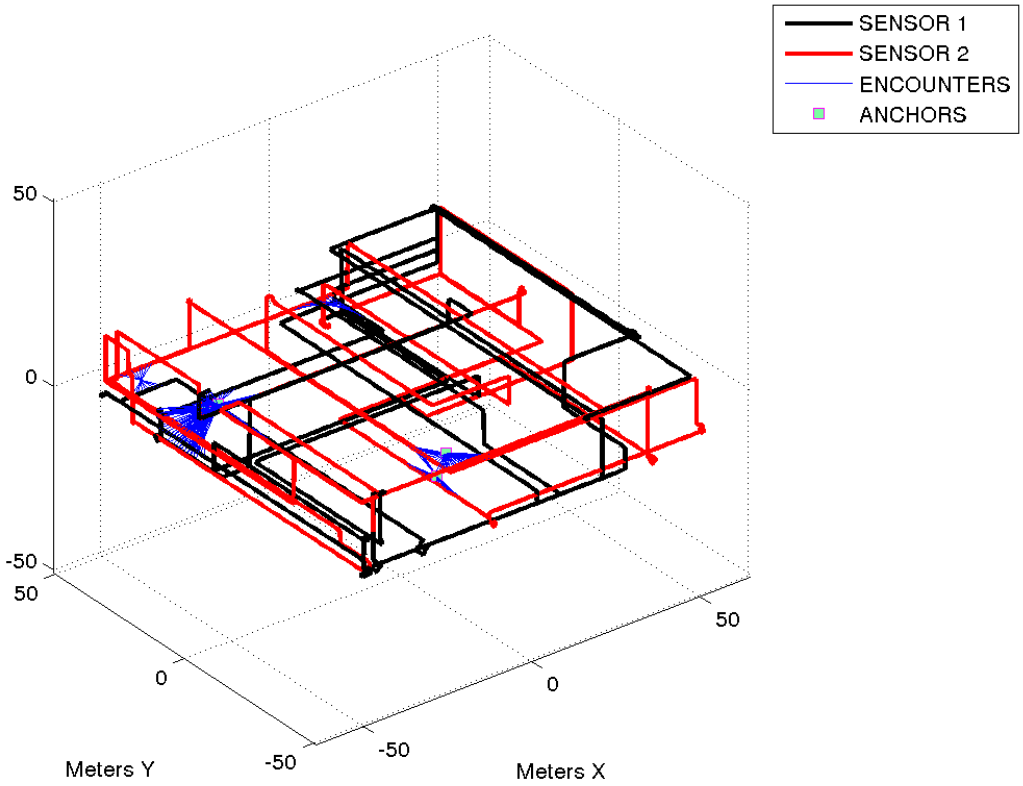
A total of 3840 simulations were carried out: 1920 of these in 2D (100×100 meter region) and 1920 in 3D ($100 \times 100 \times 40$ meter region). In all simulations the transmission power is taken as constant and equal to the transmit power used to fit parameters to the log-distance path loss model. For range-based indoor localization it is desirable to keep only those frames with a high receive signal strength, as they are more likely to be line-of-sight; non line-of-sight packets are problematic to work with, as the relationship between the received signal strength and the distance is a function of clutter topology. The receive sensitivity was therefore set to a threshold of -45dBm, resulting in a communication range of around 10 meters.

Fig. 7.1 shows example simulation runs for both 2D and 3D. Each sensor's ground truth trajectory is shown, but the inertial trajectories are omitted for clarity. Radio encounters are shown as blue lines, and anchors are shown as cyan squares. Note that in both examples encounters do not occur at a fixed rate, but occur in short bursts over time. This is primarily due to anchors being non-uniformly distributed over space (the cluster noticeably in Fig. 7.1a). However, it is also because sensor mobility patterns are uncorrelated, so sensors meet opportunistically for brief periods.

Sec. 7.1 is devoted to graph rigidity. In particular, it shows that the number of anchors and experiment duration have a significant effect on problem localizability.



(a) Example 450 second 2D problem with 1 sensor and 8 anchors.



(b) Example 1800 second 3D problem with 2 sensors and 4 anchors.

Figure 7.1: Example problems from the simulated set.

It is also shown that the optimization technique employed to reduce the complexity of the graph prior to rigidity analysis is essential to making the problem solvable in polynomial time. Rigidity analysis is therefore shown to provide a method by which researchers can predict localizability with high probability, given a fixed region, radio link budget and a suitable sensor mobility model.

Sec. 7.2 evaluates the localization step of the proposed algorithm. The results show that all graph realization algorithms scale at least quadratically with vertex count and therefore cannot be used to solve problems with a thousand or more vertices. It is shown that vertex count scales linearly with experiment duration, and more than linearly with the number of anchors and sensors. The only feasible approach is a non-linear least-squares optimizer that exploits measurement sparsity. However, it is shown that this approach suffers convergence issues if initialized poorly.

Sec. 7.3 discusses the trajectory reconstruction step of the proposed localization algorithm. The results suggest that in the general case radial drift correction, a novel contribution of this research, outperforms linear drift correction, a widely-used approach from the literature. The reason for this is that it preserves the shape of the trajectory, at the expense of piecewise discontinuities around correction points. It is also shown that there exist special, but identifiable, cases where the proposed drift correction algorithm under-performs.

7.1 Certifying localizability

This section focuses specifically on localizability, as opposed to localization. Sec. 7.1.1 first summarizes the proportion of 2D simulations that yield unique solutions. It then discusses the relationship between the number of anchors and experiment duration on the localizability of a problem. Sec. 7.1.2 performs the same analysis on the 3D problem set, highlighting differences from the 2D problems. Finally, Sec. 7.1.3 shows how the time performance of rigidity analysis is significantly improved using the graph optimization technique in Sec. 4.1.4.

7.1.1 Localizability in 2D

Of the 1920 2D simulation runs, 50 (3%) failed generic local rigidity certification. A total of 20 of these 50 runs failed because of a disconnected graph. This is caused by at least one sensor not encountering another device over the time period. The remaining 30 of these 50 runs produced a graph that was not generically local rigid. In these problems an insufficient number of encounters are made by at least one

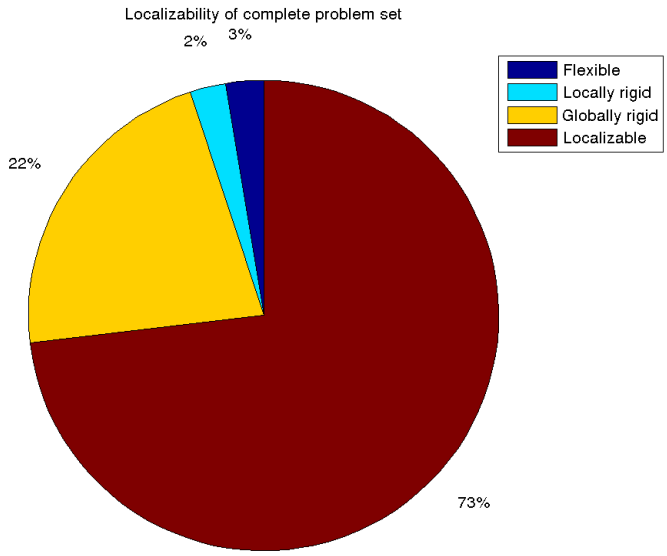


Figure 7.2: Overview of the 2D simulation dataset

sensor in order to constrain its subgraph relative to the rest of the graph. The net result is that at least one continuous degree of freedom, meaning that there is an infinite number of valid solutions to the localization problem. A further 47 (2%) of the problems failed generic global rigidity certification. These problems contained at least one discontinuous internal degree of freedom, in the form of either a reflection or superposition. The result is that there is an integer number of valid localization solutions. Of the remaining generically global rigid graphs, 419 (22%) had fewer than three anchors. These graphs result from problems that don't contain a sufficient number of anchors, or problems in which fewer than three distinct anchors were encountered by all sensors. The result is a graph with at least one external degree of freedom, in which all trajectories cannot be projected precisely into an external reference frame. The remaining 1404 (73%) graphs were classed as being "localizable" – generically redundantly rigid with a sufficient number of anchors to project the solution into a global coordinate frame. This result is summarized by Fig. 7.2.

An investigation of the relationship between experiment duration, number of anchors and number of sensors on localizability was carried out. Region size, transmission power and receive sensitivity are the three remaining variables that affect rigidity, but they are held constant by the simulation. This is reasonable, since transmission power is bounded by regulatory or energy constraints, and region size is constrained

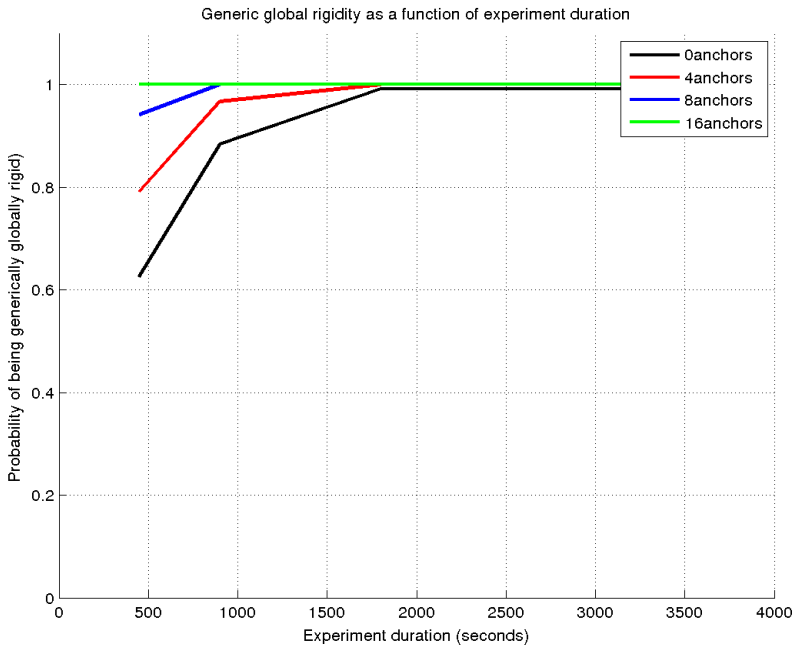


Figure 7.3: Effect of anchors and experiment duration on 2D generic global rigidity

to within a building of fixed dimensionality. Note that measurement noise does not affect localizability, as rigidity is a product of the existence of a constraint between vertices, and not the value or error distribution of the constraint itself.

In pure cooperative problems (no anchors) localizability cannot be expected, since there is no means by which the solution may be projected into a global reference frame. Generic global rigidity is a weaker form of localizability, which verifies that the sensors can be localized with in a relative coordinate system. Fig. 7.3 summarizes the effect of the number of anchors an experiment duration on this weaker type of localizability. It is clear from this diagram that, irrespective of the number of anchors, if the experiment is run for more than 1800 seconds there is a very high probability that the problem becomes weakly localizable. This is a useful tool that allows network designers to select a minimum number of anchor points to deploy (provided they are deployed in a uniformly random manner) to ensure that weak localization is possible within a target amount of time.

7.1.2 Localizability in 3D

Fig. 7.4 shows the same breakdown as in the previous section, but for the 3D problem set. It is immediately evident that there are substantially fewer localizable runs in the 3D set. This is firstly because the region of space over which the experiment takes

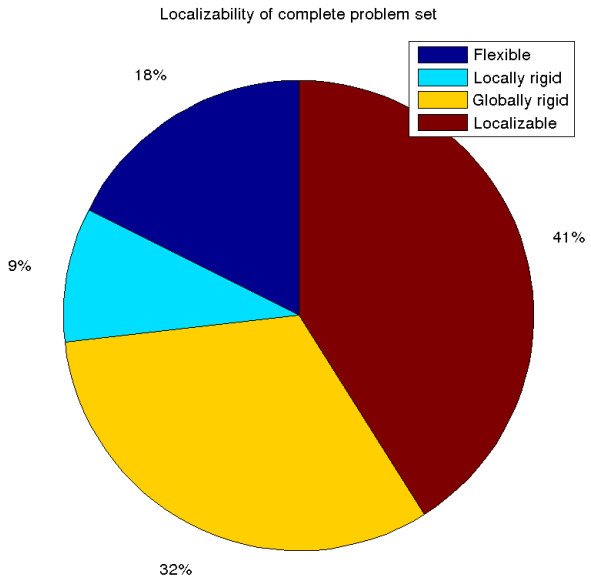


Figure 7.4: Overview of the 3D simulation dataset

place is substantially larger, while the transmission range and sensor mobility remain the same. The second reason for this is that there are more degrees of freedom in 3D, and therefore more constraints are required to enforce rigidity. The outcome is that only 1405 (73%) of the problems are generically globally rigid, with only 788 (41%) of these involving a sufficient number of encounters with anchors to be fully localizable. Since 3D rigidity certification is probabilistic, and solutions were obtained with a 99% confidence, one would expect that 1% of the 515 remaining problems were incorrectly classified as false negatives, and so the localizable set is larger than stated.

Fig. 7.5 again shows the relationship between experiment duration and number of anchors of the probability that the resulting graph is generically globally rigid (weakly localizable). It is clear that the number of anchors has a significant effect on the localizability of problems: 20% of problems involving fewer than 9 anchors don't result in, at least, a weakly localizable graph after 30 minutes. The results also show the counter-intuitive result that having more anchors does not necessarily increase the weak localizability of the graph (the red curve lies below the black curve). This is due to the single-sensor runs. In the cooperative case (black curve, no anchors) the graph is by construction generically globally rigid, because it contains only motion measurements between points along a single sensor's trajectory. In the case of 4 anchors (red curve) there are two generically rigid subgraphs – one representing the

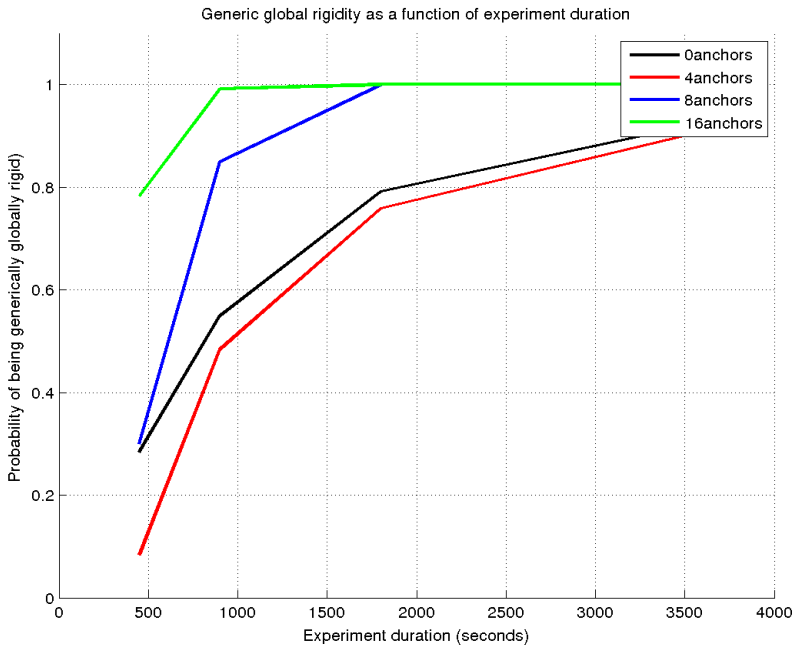
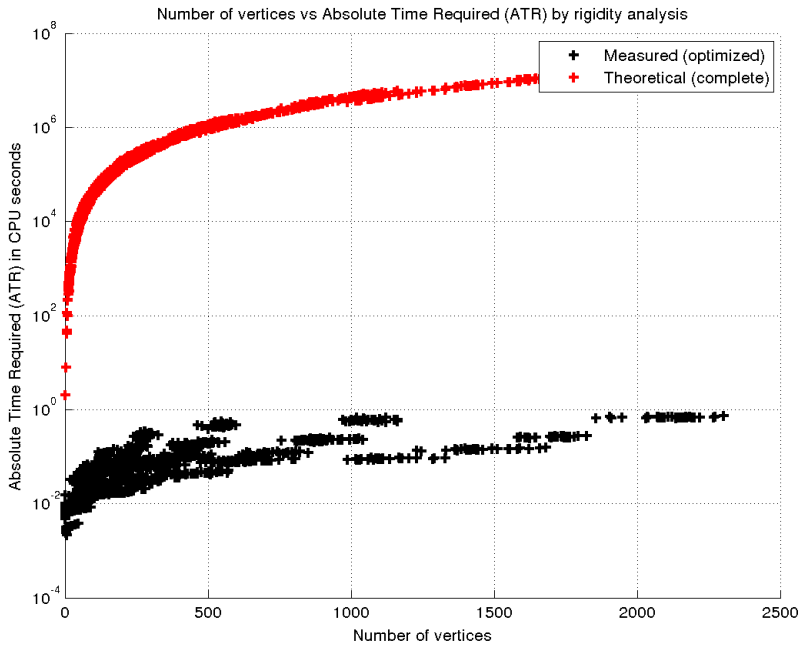


Figure 7.5: Effect of anchors and experiment duration on 3D generic global rigidity

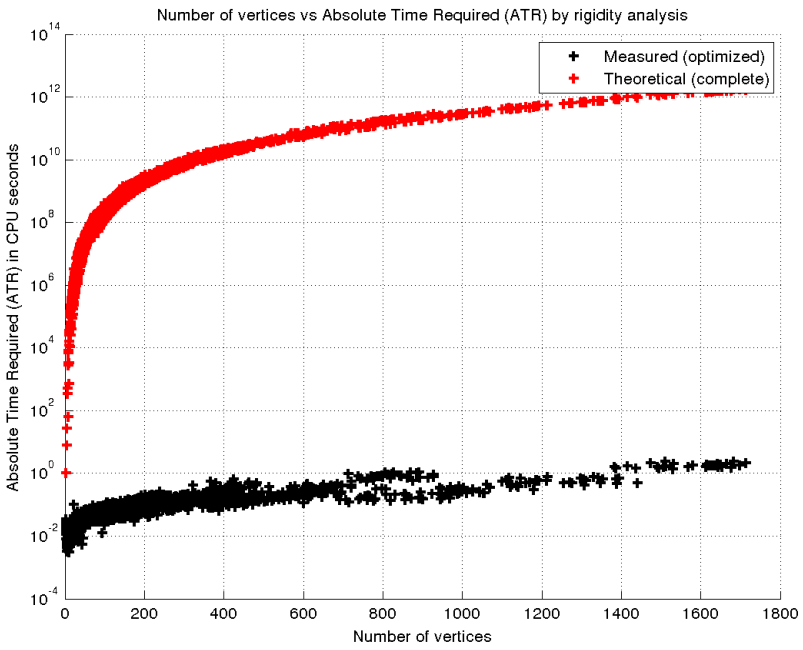
single sensor and one representing the fully-connected anchors. The joint rigidity of these two graphs is decided by the number and configuration of encounters, which may be poor for short experiment durations.

7.1.3 Optimization and time performance

Fig. 7.6 summarizes the time performance advantage brought by the optimization algorithm that was used to reduce the size of the encounter graph prior to rigidity analysis. The x-axis value refers to the number of vertices in the original, unoptimized graph. In both sub-figures, the black curve shows the total log-scaled CPU time required to reduce the complexity of the graph and carry out rigidity analysis. The red samples show the CPU time required to solve the same problem, but using the original unoptimized graph. Note that the red curves in Fig. 7.6a and Fig. 7.6b are different, because the complexity of 2D and 3D rigidity certification is $O(ve)$ and $O(e^3)$ respectively, where v and e represent the number of graph vertices and edges. The results show clearly that significant benefit is brought by reducing the number of vertices and edges in the graph prior to rigidity analysis.



(a) CPU time for 2D rigidity analysis



(b) CPU time for 3D rigidity analysis

Figure 7.6: Time performance of rigidity analysis

7.2 Solving problems

In this section it will be assumed that a given localization problem is provably generically globally rigid, and the objective is to obtain a suitable embedding of the graph. The proposed localization algorithm does so by first obtaining an estimate of that embedding by graph realization. Several variants were proposed, and their time-performance is compared in Sec. 7.2.1. The proposed localization algorithm then refines this estimate, taking into account the amount of information each measurement provides. Sec. 7.2.2 provides a comparison of the time-performance for the refinement approaches.

The results show clearly that the complexity of the realization and refinement algorithms is such that they cannot be used in practice to solve mobile localization problems involving more than several hundred vertices. Roboticists typically solve these types of problems use non-linear least-square solvers with initial estimates drawn from dead-reckoning. Such solvers exploit the sparsity of the Jacobian matrix, which expresses the linearized gradient of the non-linear relationship between state values. Sec. 7.2.3 shows that such methods are nevertheless highly sensitive to initialization – even when the supplied graph is provably rigid, and the measurements are perfect, such an approach fails to converge with a random initialization. In theory, graph realization could be used to find a suitable initialization, but there exists no such algorithm with a favourable complexity.

7.2.1 Complexity of graph realization

Fig. 7.8 shows the relationship between the size of the graph and the time taken to realize the graph in 2D (as measured by the ATR metric). Four different graph realization algorithms were tested – Multidimensional Scaling (MDS), Spectral Graph Drawing (SGD), Degree-Normalised Spectral Graph Drawing (DNS) and Distributed MDS (MAP). The results include graphs of up to 200 vertices to illustrate scalability; attempting to solve larger problems with thousands of nodes substantially increases simulation time with little extra information from which to draw conclusions.

Those methods based on eigendecomposition (MDS, SGD, DNS) reduce to a Cholesky decomposition of a symmetric matrix. The complexity of this step is approximately $O(v^3)$, which was verified empirically in MATLAB for matrices up to 300×300 in size. In the case of MDS, an additional all-pairs shortest path step for completing the distance matrix must be carried out beforehand. This problem is solved in our simulation using *Floyd's algorithm*, which has a run-time complexity of $O(v^3)$. The

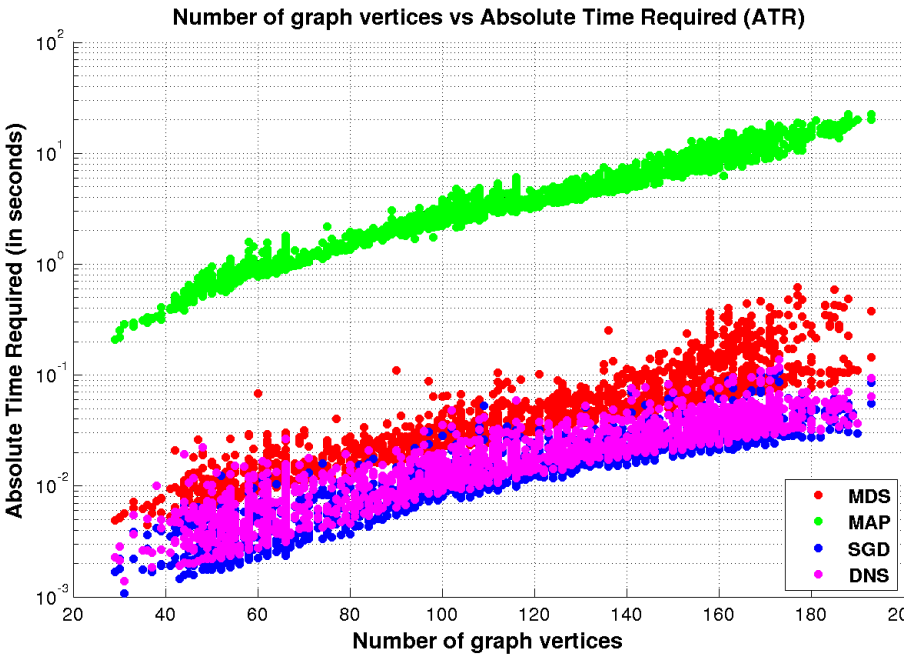


Figure 7.7: Scalability of graph realization algorithms

MDS, SGD and DNS time performances scale similarly, but MDS has approximately twice the baseline cost of SGD or DNS, because of the all-pairs shortest path step. This is consistent with the results shown in Fig. 7.8.

MAP first applies MDS to local patches over the network. The resulting configurations are then stitched together. Assuming that there are, on average, k nodes in each local map, then the run-time complexity of solving a single local map is $O(k^3)$. Since a local map is built for each vertex, it follows that the complexity of the map-building stage is $O(vk^3)$. Shang and Ruml [82] provided a stitching method with a run-time complexity of $O(vk^3)$, implying that the overall complexity of the algorithm is bounded by $O(vk^3)$. Therefore in the worst case – where each vertex is less than two hops away from every other vertex – the run-time complexity of MAP is $O(v^4)$. Fig. 7.8 confirms that, in practice, this worst-case performance is seldom seen, and MAP's run-time complexity is similar to the competing three algorithms, but the stitching process adds just under a second of overhead.

7.2.2 Complexity of solution refinement

Fig. 7.8 shows how the relationship between the size of the encounter graph and the run-time complexity of solution refinement. Recall from Sec. 6.3.3 that the IGN model shows the time required to carry out graph realization without refinement,

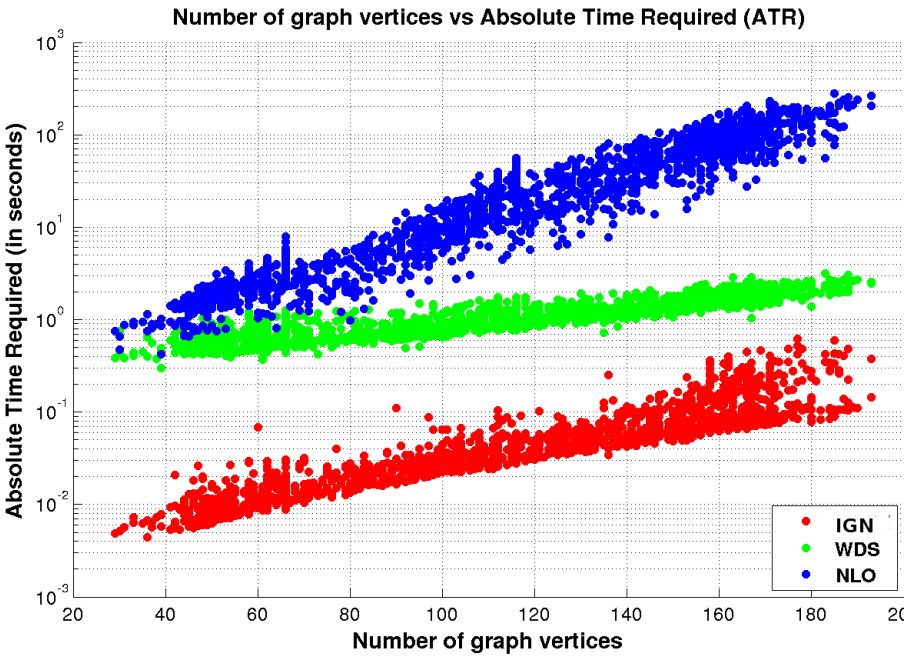


Figure 7.8: Scalability of solution refinement algorithms

and can be ignored. The proposed localization algorithm offers Weighted Multidimensional Scaling (WDS) and Non-Linear Least-squares Optimization (NLO). It is evident that both models exhibit very different time performances. As in the previous section, the number of graph vertices and edges are denoted by v and e respectively.

Since no *Solution Refinement* step is carried out in the IGN model, its complexity is decided entirely by MDS, which we have shown to have a run-time complexity of $O(v^3)$. The complexity of the WDS algorithm is determined by the Guttman transform / stress computation of SMACOF, which share a run-time complexity of $O(kv^2)$, where k is the number of iterations. Under the assumption that k is constant, and that significantly fewer iterations are required than vertices in the graph, then the run-time complexity of WDS is $O(v^2)$. The performance of WDS will slowly converge to IGN as v increases, since the complexity of its realization algorithm $O(v^3)$ – used to initialize WDS – dominates the complexity of the refinement algorithm itself.

The complexity of NLO is decided by the step in which a linear system $Ax = b$ solved, where A is of size $vd \times vd$, where d is the embedding dimension. This operation therefore requires approximately $O((vd)^{2.3})$ arithmetic operations. Assuming that k iterations are required to obtain an NLO solution, then the run-time complexity of NLO is thus $O\left(k(vd)^{2.3}\right)$. The number of iterations is usually significantly lower than vd, and so NLO's complexity is thus $O\left((vd)^{2.3}\right)$. Importantly, this naive implemen-

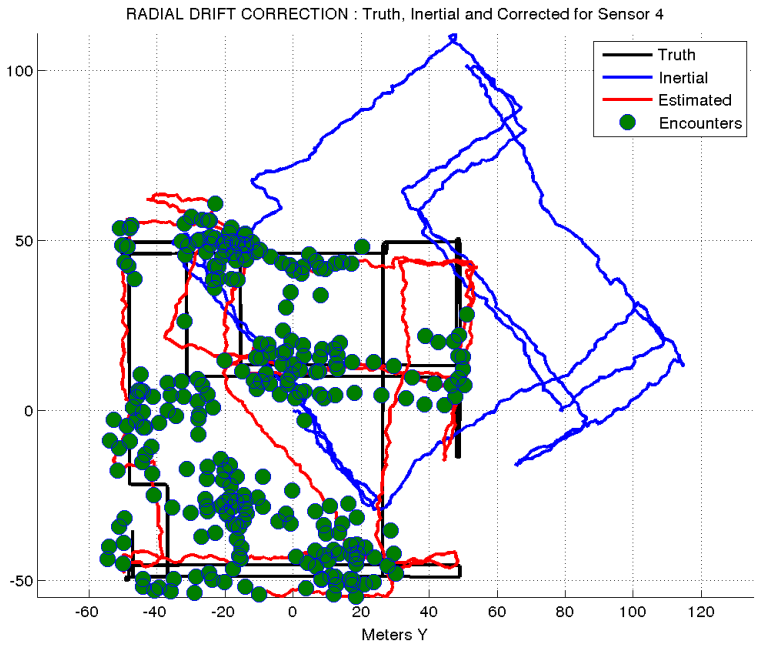


Figure 7.9: The figure above illustrates the performance of NLO for a 900 second problem containing 4 sensors and 8 anchors. The inertial trajectory is shown in blue, ground truth trajectory in black and the sparse NLO trajectory (using RAD correction) in red, for sensor 4 of 4. Encounter points are shown as green circles.

tation of NLO does not exploit the fact that relationship between vertices is sparse, which is why its performance scales so poorly. The next section shows how a sparse implementation of NLO can be used to solve large localization problems.

7.2.3 Effect of anchor and sensor count on performance

In the field of robotics the full localization problem is typically solved using non-linear least-squares optimization. However, these solvers differ from the NLO method described in the previous section in that they exploit the fact that the matrix A in the linear system $Ax = b$ is sparse. To test the performance of these sparse methods, a Matlab MEX wrapper was created to pass range-only localization problems to CERES-SOLVER, a C++ library for solving sparse non-linear optimization problems. Fig. 7.9 shows an example run for 4 sensors and 8 anchors, which lasted 900 seconds. The solver performs substantially better than the previous naive NLO method (which used the LSQNONLIN function in Matlab) obtaining the result in just over 100ms.

The objective of this section is to determine how the error performance of sparse NLO (the model described in the previous section) changes, as the problem scales. To

achieve this a series of 450 second experiments were conducted, with each parameter combinations run 30 times to obtain a 1σ confidence interval. Since sparse NLO is capable of solving large problems, the graph resolution was increased from 10 seconds to 5 seconds to improve localization accuracy. The solver's objective function minimizes the weighted difference between the square observed predicted and square observed distances. This is done to avoid differentiating a function with a square root, which is discontinuous at zero. Two different initialization methods were evaluated: The first FAK variant is sparse NLO initialized with the correct solution, and it represents the lower bound achievable for the method, given the measurement noise. The second PDR variant uses the dead reckoning trajectories to obtain an initial estimate. The GRS error metric captures the average residual between the observed and predicted edges, and is thus invariant to any error introduced by trajectory reconstruction.

Fig. 7.10 shows an example run containing a single sensor and eight anchors. The ground truth trajectory is shown by a solid magenta line. Truthful graph vertex positions are shown as black circles on this line, or square anchor points. Note that encounters cluster around the anchor points, as there is only one sensor in this problem and therefore no peer-to-peer encounters. The red circles visible at the top of the figure show that FAK-initialised sparse NLO performs poorly only at the end of the trajectory, because there are no radio measurements to tie the positions to an external reference point. The net effect is that the solver iteratively distorts the end of the trajectory to match the observed motion measurements, which are noise-perturbed and hence inaccurate. PDR-initialized NLO typically converges to bring graph vertices to within 5m of their actual positions. Decreasing the tolerance and increasing the maximum number of iterations of spare NLO has the net effect of pushing the blue vertices closer to the red vertices.

Fig. 7.11 shows that each subsequent anchor added to the problem improves the expected performance of sparse NLO. This result is intuitive: the collection of anchors forms a strongly rigid structure, and each additional encounter fixes a portion of the inertial trajectory to this rigid structure. The addition of an anchor also one unknown vertex (two or three states in 2D or 3D respectively) to the solver, but because all anchors are fully-connected by edge measurements with a low variance, this state is strongly localizable. Several of the FAK runs failed to converge, despite being initialised with the correct solution. This is due to the fact that the solver's cost function defines a different 'correct solution' because of measurement noise.

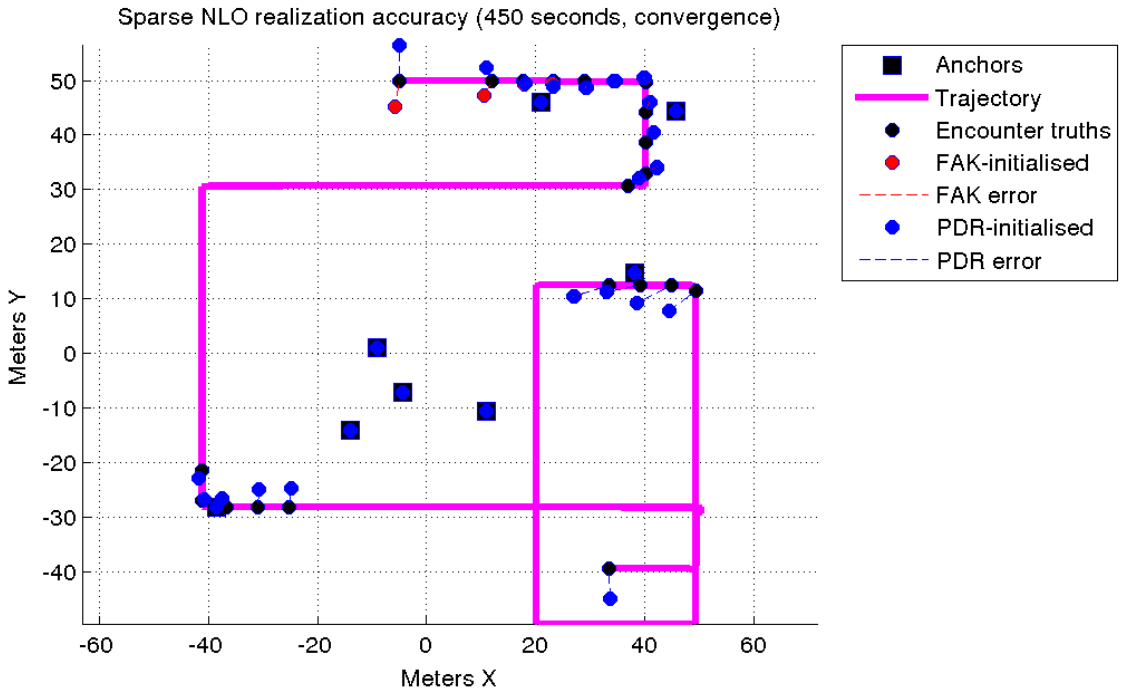


Figure 7.10: The image above shows an example 450s simulation run with a single sensor (path shown by a pink line) and eight anchors (black squares). In general FAK performs very well (red circles are occluded by the black circles), the exception being at trajectory endpoints. PDR provides a good approximation too correct solution.

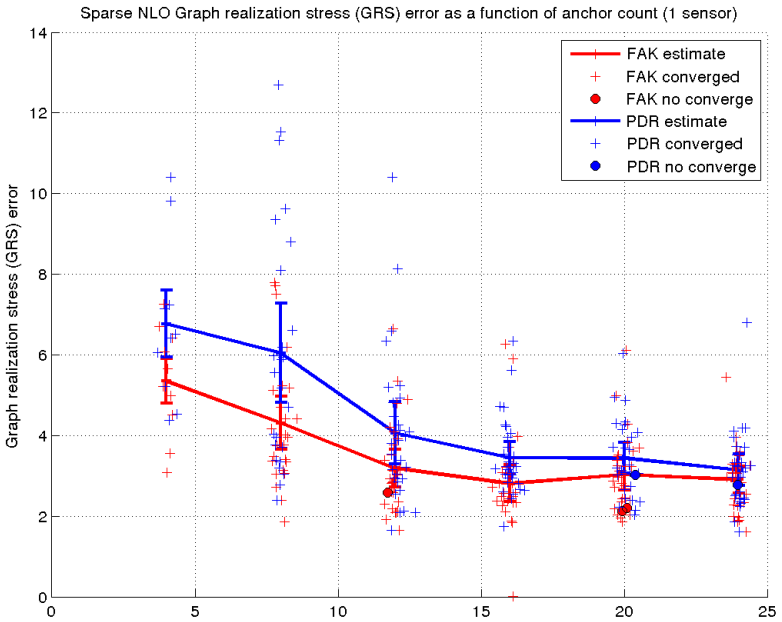


Figure 7.11: Graph summarizing the relationship between anchors and graph realization stress, for 450 second problems. Each problem contains one sensor and 1σ confidence intervals are shown for 30 runs. The red curve represents the sparse NLO performance when initialised close to the correct solution, while the blue curve represents its performance when initialised with dead reckoning trajectories. Crosses and circles represent sample runs, which converged and did no converge respectively.

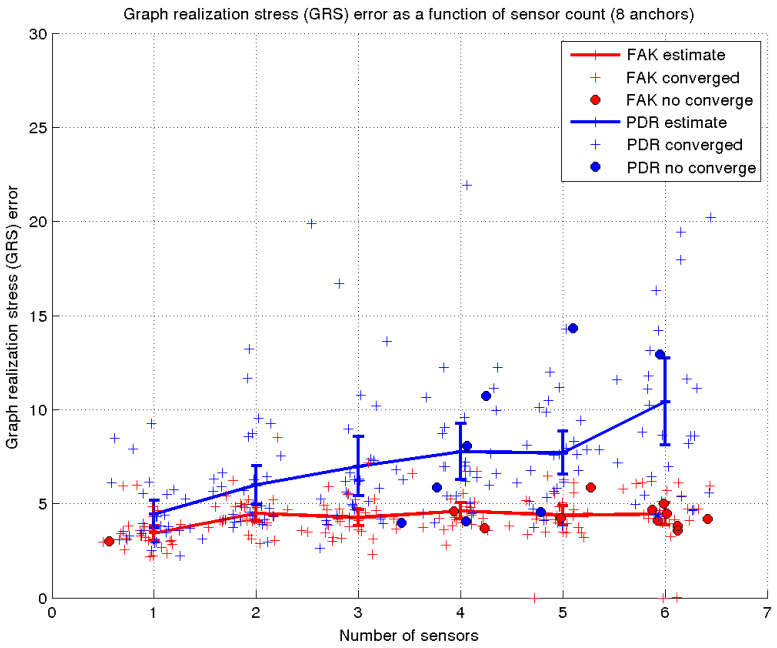


Figure 7.12: Graph summarizing the relationship between sensors and graph realization stress, for 450 second problems. Each problem contains one sensor and 1σ confidence intervals are shown for 30 runs. The red curve represents the sparse NLO performance when initialised close to the correct solution, while the blue curve represents its performance when initialised with dead reckoning trajectories. Crosses and circles represent sample runs, which converged and did no converge respectively.

Intuitively, one might expect that increasing the number of sensor would similarly lead to a drop in realization error. However, as Fig. 7.12 illustrates, the opposite trend is observed. The graph takes a similar format to the previous graph, but represents how the number of sensors affects localization performance, in problems containing eight anchors. It was observed that PDR-initialised sparse NLO diverges from its FAK-initialised counterpart. This suggests that the addition of subsequent sensors actually reduces the performance of sparse NLO.

To investigate further why the addition of sensors reduces localization performance, the same experiment was conducted firstly without any sensor-to-sensor encounters (Fig. 7.13) and secondly without any sensor-to-anchor encounters (Fig. 7.14). In addition, a wider range of sensors were tested, and the solver was permitted up to 1000 iterations to mitigate any convergence issues. The results show clearly that it is the existence of peer-to-peer radio encounters that causes error to increase with sensor count. The underlying cause of this divergence is that each subsequent sensor increases the state space of the problem substantially. Critically, by copying initial

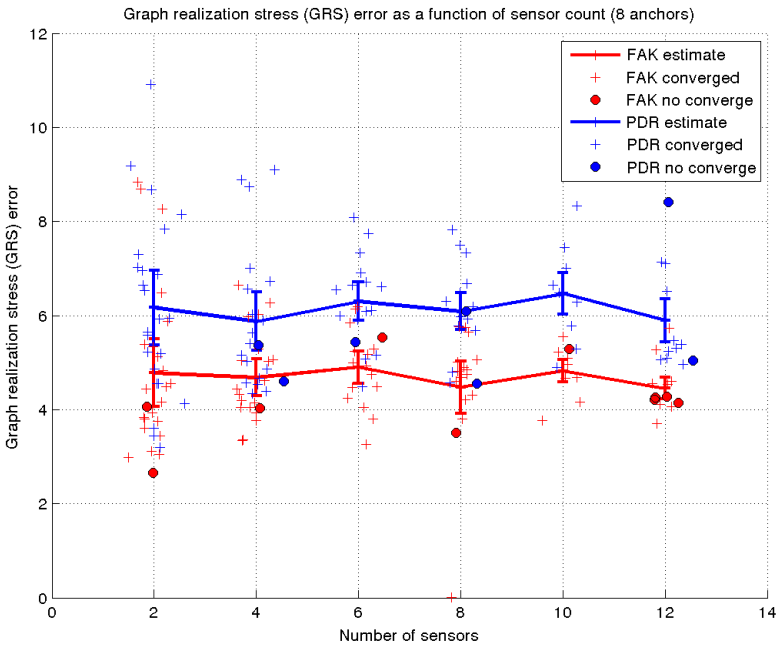


Figure 7.13: Summary of the relationship between the number of sensors and localization performance, for problems containing only encounters with anchors (peer-to-peer encounters were disabled). Again, a 1σ confidence interval is shown for 30 runs.

values from PDR these additional states are relatively well-initialized with respect to themselves, but poorly initialized with respect to other sensors. When the only encounters are from anchors (Fig. 7.14) there is a clear gradient that the solver follows to bring the sensors' trajectories into alignment with the ground truth. However, when sensor-to-sensor encounters are used (Fig. 7.13) the constellations of points representing each trajectory exert attractive forces on each other. The resulting cost gradient therefore contains many local minima into which the solver converges.

Looking just at the FAK-initialised result in (Fig. 7.13) it can be seen that the addition of each sensor does in fact increase the amount of information, which reduces localization error. This suggests that sensor-to-sensor radio encounters can be used to improve localization performance. However, actually realizing this advantage requires a better initial estimate than PDR provides.

7.3 Reconstructing trajectories

This section assumes that the graph has been realized, and the goal is to provide piecewise corrections to the inertial trajectories using the vertex coordinates from

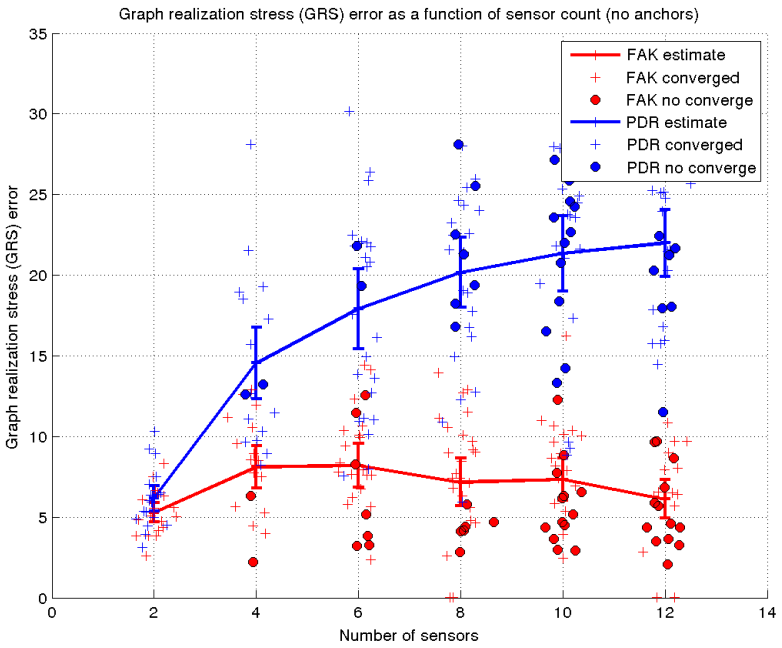
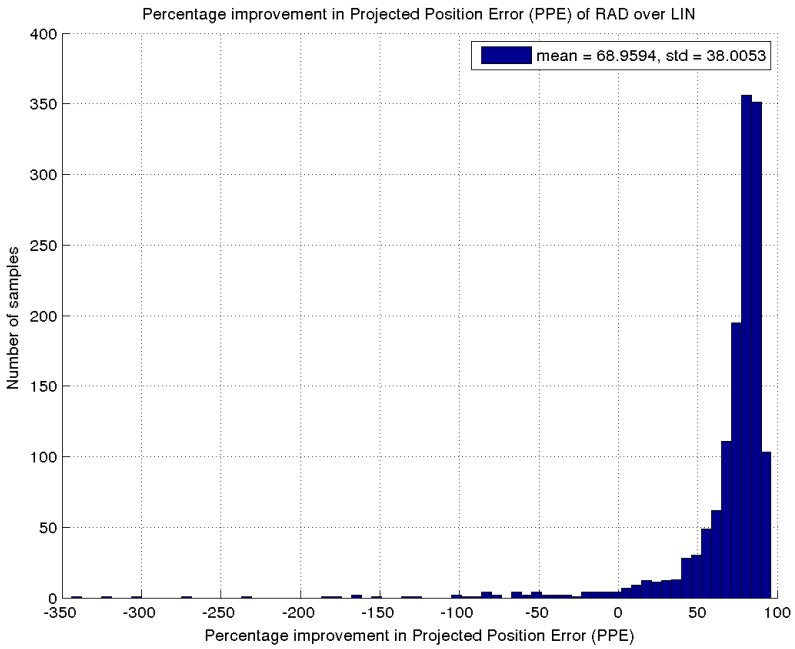


Figure 7.14: Summary of the relationship between the number of sensors and localization performance, for problems containing only encounters between sensors (anchor-free problems). Again, a 1σ confidence interval is shown for 30 runs.

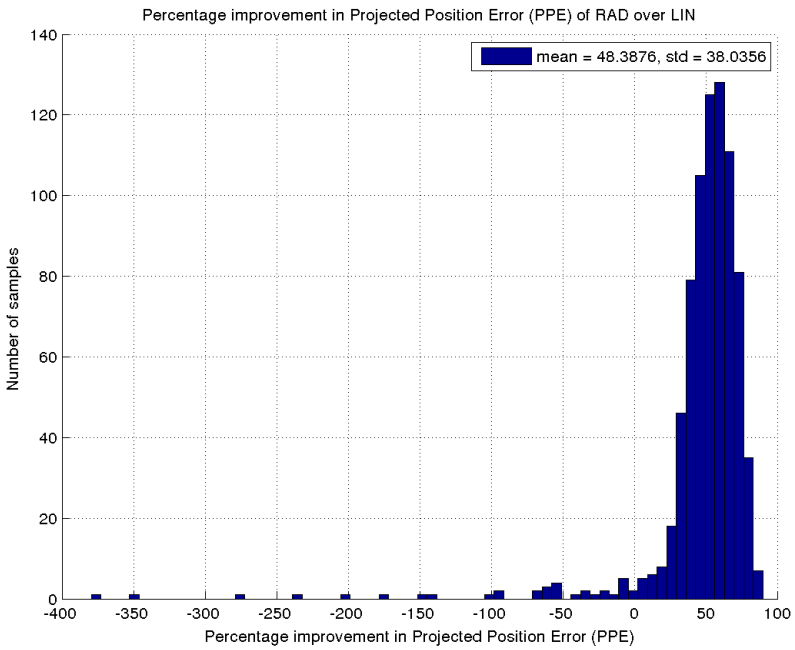
the realized graph. This section therefore seeks to compare the performance of two competing algorithms: *Linear Drift Correction* (LIN) by Constandache et al. [14], and a novel algorithm proposed by this research, called *Radial Drift Correction* (RAD). In order to carry out this comparison in a fair manner, it is necessary to assume that the preceding steps produce a perfect realization of the graph. If this is the case, then the PPE metric measures how close the corrected trajectory lies to the ground truth, and hence how much error is introduced only by piecewise correction.

Recall that the PPE metric looks at the root mean square distance between all ground truth points and their corresponding points along the estimated trajectory. Assume that the PPE error of linear drift correction and radial drift correction are given by ε_l and ε_r respectively. The percentage by which RAD outperforms LIN is thus given by the equation $(\varepsilon_l - \varepsilon_r)/\varepsilon_l \times 100$. Fig. 7.15a and Fig. 7.15b are histograms of this percentage for 2D and 3D problem sets respectively.

From the 2D results in Fig. 7.15a it can be seen that, in the general case, RAD significantly out-performs LIN. The underlying reason for this is lack of a common global bearing. PDR algorithms that make use of magnetometers to provide bearing corrections ensure that all trajectories are oriented correctly within the global



(a) Percentage improvement of RAD over LIN in 2D



(b) Percentage improvement of RAD over LIN in 3D

Figure 7.15: Comparative performance of RAD versus LIN

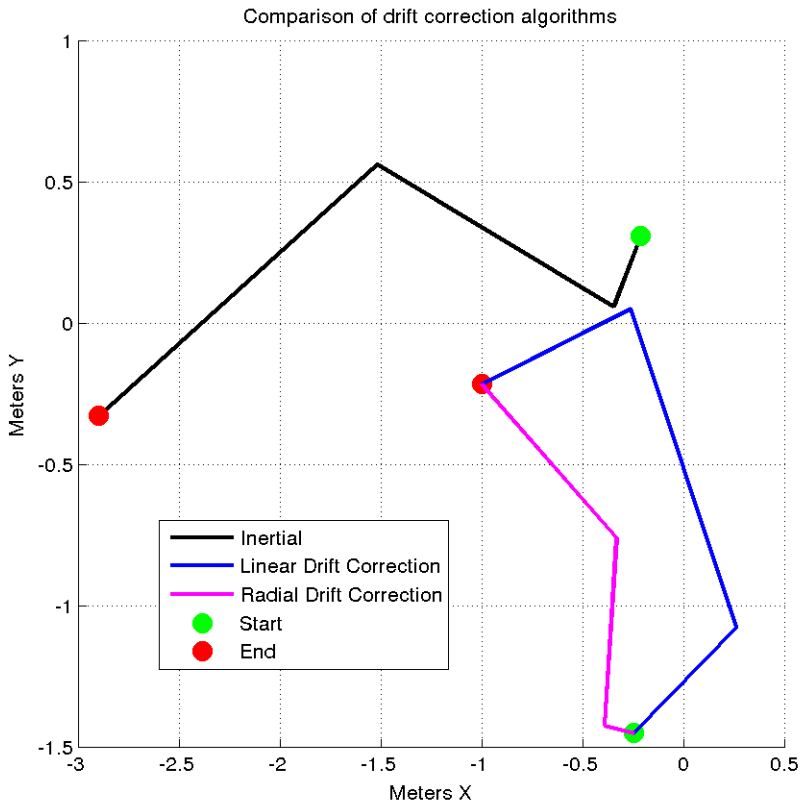
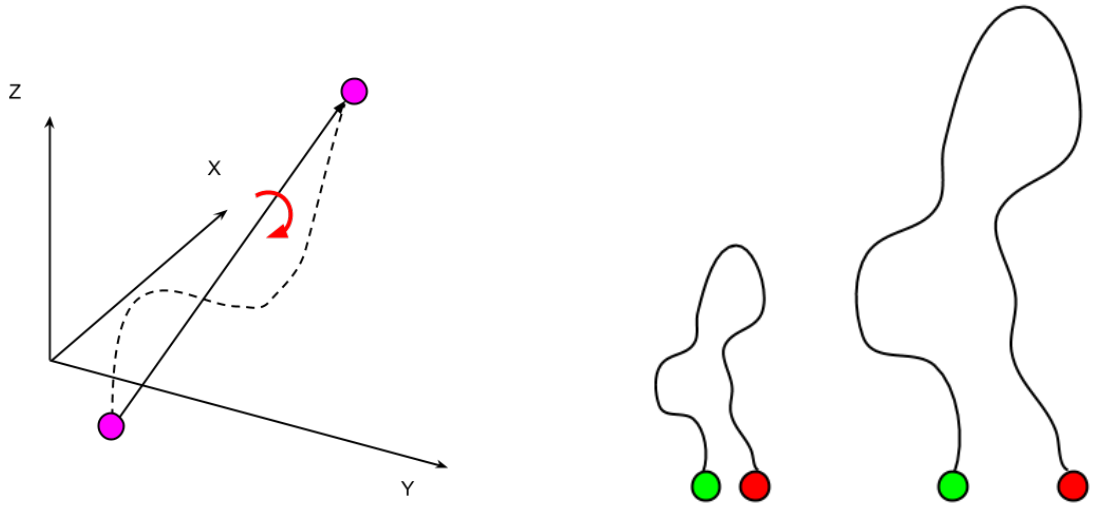


Figure 7.16: How Linear Drift Correction (LIN) distorts trajectories

reference frame. However, in applications without magnetometers or initial global bearing, the PDR trajectories may be substantially misaligned with the global reference frame. In these cases LIN models rotation as a distortion to the trajectory shape, whereas RAD performs a controlled rotation to first bring the trajectories into alignment. Fig. 7.16 illustrates this with a simple trivial example in 2D. The inertial trajectory (black curve) must be mapped to lie between the corresponding red and green encounter points. The two red-green direction vectors are strongly orthogonal, indicating a rotation is required. Observe how RAD (purple curve) preserves the shape of the curve, compared to LIN (blue curve).

The 3D results in Fig. 7.15b show again that RAD outperforms LIN, but by a lesser quantity. Again, this performance improvement is brought by the fact that RAD is shape-preserving during rotations. However, the advantage is significantly diminished (an reduction from 68% to 48%) by the fact that 3D piecewise correction requires an additional degree of freedom to be constrained. This is illustrated by Fig. 7.17a. Here, the dashes trajectory segment is mapped to lie between two purple



(a) Constrain degrees of freedom in 3D

(b) Magnification of scaling errors

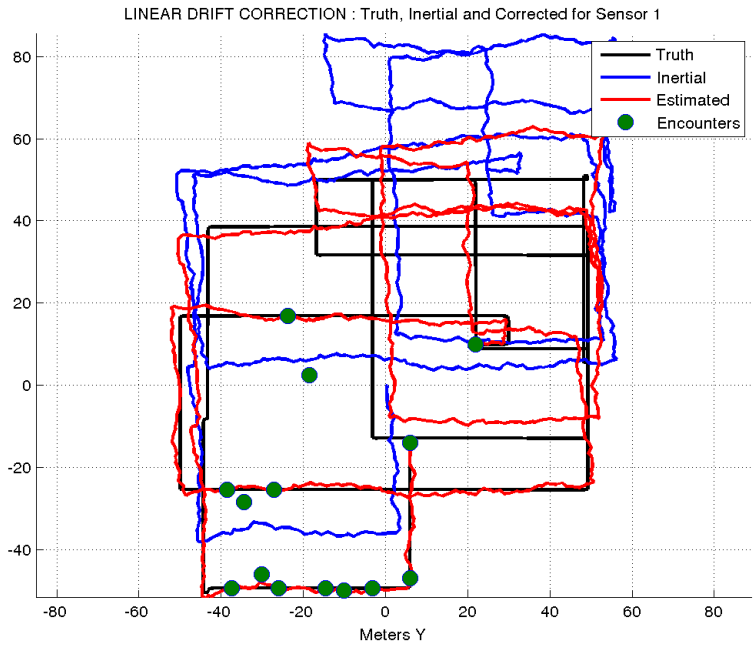
Figure 7.17: Cases where radial drift correction fails

encounter points. Since RAD does not preserve continuity at the encounter points, this trajectory segment can be arbitrarily rotated about the correction vector. This degree of freedom is shown by a red arrow. This problem has a limited effect in the simulation because the synthetic pedestrians walked mostly in the X-Y plane. Had this been a truly random walk in 3D, LIN would have outperformed RAD.

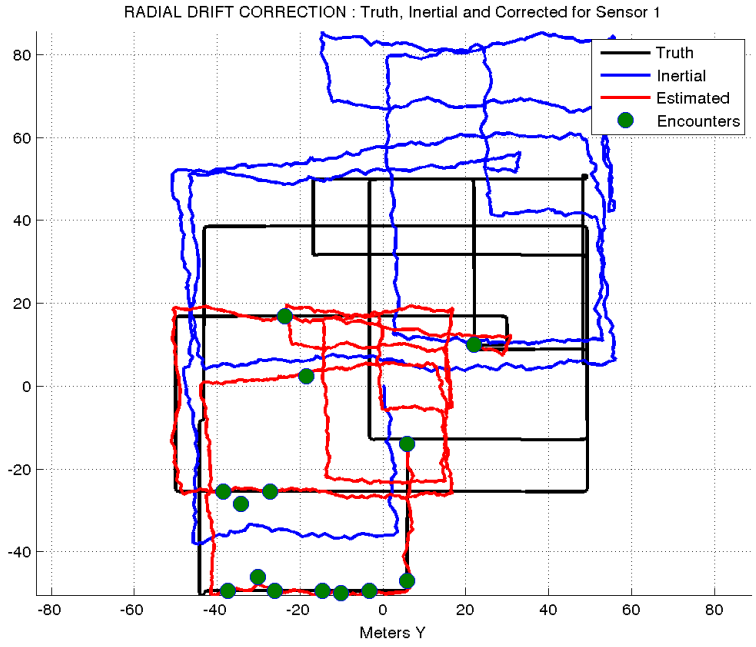
A further issue experienced with RAD is that its scaling mechanism periodically magnifies trajectory segments, and particularly when the correction requires no rotation. This effect is exacerbated when time difference between sequential encounters is large, but the encounters themselves are relatively close together. Fig. 7.17b illustrates this problem. Here, the trajectory on the left needs to be mapped to lie between the two corresponding points on the right. Since no rotation is needed, the trajectory is scaled to accommodate the difference, and the curve is magnified.

Fig. 7.18 shows an example simulation run where LIN outperforms RAD by the greatest amount. Fig. 7.18a shows the LIN solution to the problem, while Fig. 7.18b shows the RAD solution to the problem. What is critical to note is that the inertial trajectory (blue curve) and ground truth trajectory (black curve) differ mostly by a translation; the respective headings were aligned by chance. LIN performs very well in this scenario, corroborating the claim that it is shape-preserving with aligned global orientation.

Fig. 7.19 shows another example where RAD outperforms LIN. Again, the LIN solution is shown in Fig. 7.19a, and the RAD solution is shown in Fig. 7.19b. In



(a) High performance: Linear Drift Correction (RAD)



(b) Low performance: Radial Drift Correction (RAD)

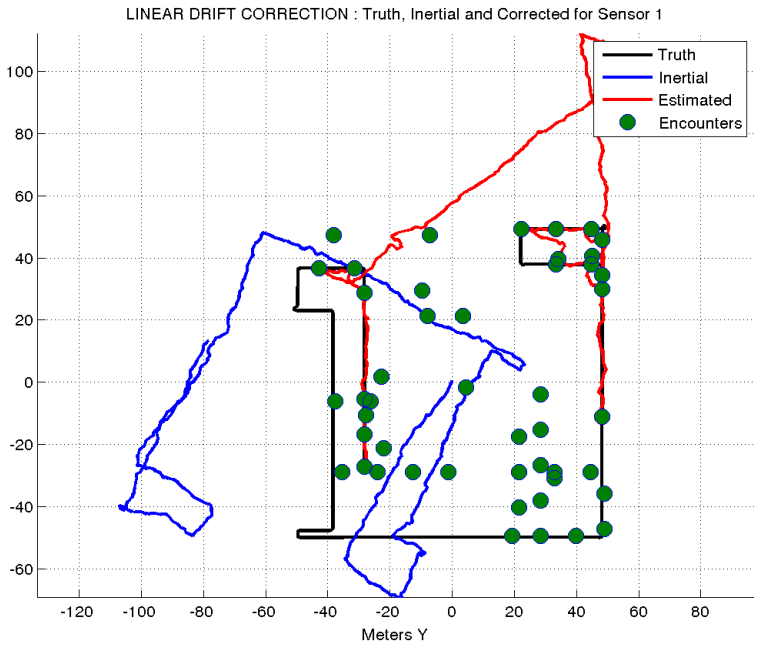
Figure 7.18: An example case where LIN outperforms RAD.

this run there is a significant misalignment between the ground truth and the inertial trajectory. The result is that LIN distorts the shape of the trajectory to preserve piecewise continuity about its correction points, whereas RAD rotates the segment into its correct position.

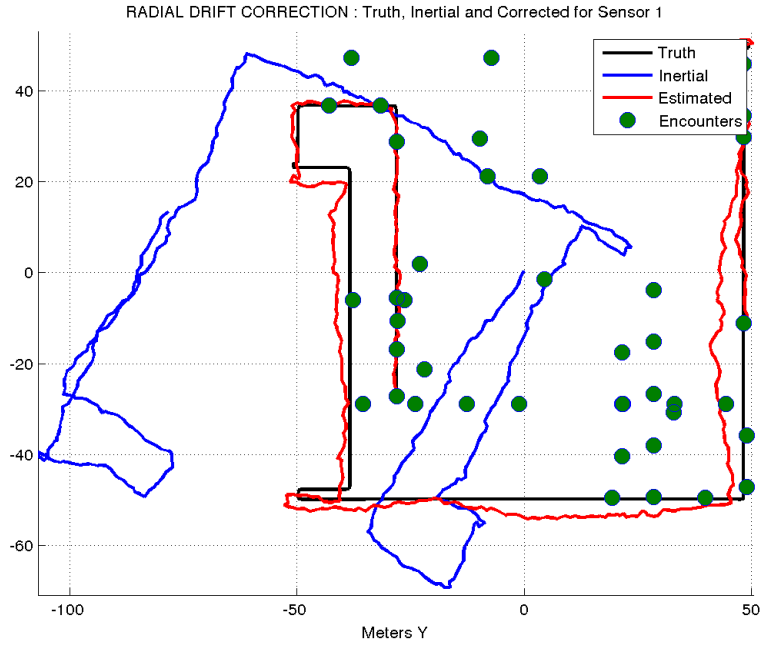
7.4 Discussion

This chapter presented the results from a simulation study, in which the performance of the proposed localization algorithm was evaluated. Firstly, it was shown that rigidity theory provides an effective method of certifying whether a 2D or 3D localization is solvable, provided that an optimization step is run on the graph in advance to reduce its complexity. Secondly, it was shown that all graph realization and refinement algorithms considered in this work have at least quadratic run-time complexities in the number of vertices. Since vertex count scales approximately linearly with time, these methods cannot be used for long-term mobile localization. A more general solution is to use a non-linear least-squares optimization algorithm that exploits the block sparsity of the problem, initialized with the PDR solution. Using this method it was confirmed that increasing the number of anchors reduces localization error. However, increasing the number of sensors does not. The underlying cause was found to be that PDR trajectories do not share a common reference frame, and so while individual sensor trajectories are well-structured, the join initial state estimates are greatly misaligned. The peer-to-peer encounters yield a complex cost gradient having many local minima into which the solver converges. Finally, it was shown that radial drift correction outperforms linear drift correction in problems where there is no global alignment of the inertial trajectories. However, radial drift correction suffers from three major deficiencies: piecewise discontinuities at encounter points, problematic scaling in certain rare cases, and an inability to constrain a degree of freedom in 3D tracking problems.

The next chapter concludes the thesis and outlines directions for future research.



(a) Low performance: Linear Drift Correction (RAD)



(b) High performance: Radial Drift Correction (RAD)

Figure 7.19: An example case where RAD outperforms LIN.

Chapter 8

Conclusion and Future Work

Having discussed the results of our simulation study, the thesis will now be brought to a conclusion. Sec. 8.1 begins with a summary of the research. Then in Sec. 8.2 the key contributions of the work are highlighted, and then tied back to the motivating applications in Sec. 8.3. Finally, Sec. 8.4 discusses limitations of the research, which are used to motivate avenues for future research.

8.1 Research summary

The work in this thesis is motivated by the need to localize sensors when GPS or other precise positioning systems cannot be used, because of resource or infrastructure constraints. Example applications include indoor pedestrian navigation or underground badger tracking. A solution is proposed that fuses motion and radio measurements into a connected graph. This graph can be analysed to determine whether a sufficient number of measurements were recorded in order to define a unique localization problem. From a theoretical perspective it is also possible to realize this graph in 2D or 3D, and use the resulting coordinates to provide piecewise corrections to the sensors' trajectories. This algorithm follows the six step sequence below:

1. **Graph construction** - Motion and radio measurements are combined into a graph that models sensor mobility. Vertices in the graph correspond to key points on the sensors' trajectories that correspond to radio encounters. Each edge in the graph represents a pairwise distance estimate between two vertices, comprising of a mean and variance. Each distance estimate abstracts from many smaller inertial or radio measurements, which are fused together to reduce the size of the graph. The result is a compact graph that models the high-level spatial relationship between device positions.

2. **Rigidity analysis** - In some cases an insufficient number of radio encounters are recorded for the relative sensor trajectories to be uniquely determined. Such problems do not afford a unique solution and cannot be solved, irrespective of how well a candidate localization algorithm performs. The proposed algorithm provides an optimization technique for reducing the size of the graph, such that the uniqueness can be certified within a feasible amount of time. While it is possible to deterministically certify rigidity in 2D, it is currently only possible to probabilistically certify rigidity in 3D.

3. **Graph embedding** - The goal of graph embedding is to find a set of vertex coordinates that best matches the observed pairwise distances. The approach proposed four candidate graph realization algorithms from the literature – *Multidimensional Scaling*, *Distributed Multidimensional Scaling, Spectral Graph Drawing and Degree-normalised Spectral Graph Drawing*. However, the design is extensible and so other methods can be used.

4. **Solution refinement** - The objective of solution refinement is to further improve the embedded graph. The refinement algorithms distinguish themselves from embedding algorithms in the sense that they are iterative, take measurement error into account and require an initial estimate. In this step the proposed algorithm can use one of two solution refinement algorithms - *Weighted Multidimensional Scaling* and *Non-linear Least-Squares Optimization*. Again, the proposed algorithm is extensible, and so other methods can be used.

5. **Drift correction** - The objective of the drift correction step is to remap each sensors' drift-corrupted inertial trajectory to pass through select vertex coordinates from the embedded graph. The proposed approach offers two piecewise drift correction algorithms - *Linear Drift Correction* and, a novel contribution of this thesis, *Radial Drift Correction*.

6. **Solution projection** - In the final step of the proposed approach the drift-corrupted trajectories are projected out of a relative coordinate system, and into a global coordinate system. This step is optional, as it requires at least $d+1$ algebraically independent anchors in order to project the solution between coordinate frames in $\mathbb{E}^d$. The proposed approach uses the well-known *Procrustes Transform* to find a suitable mapping.

8.2 Key contributions

This thesis makes the following key contributions:

1. **A modular algorithm for offline, range-based sensor localization** - The proposed offline range-based localization algorithm is modular, extensible and application-agnostic. Importantly, it does not presume the existence of an infrastructure of base stations and can therefore be used to localize in infrastructure-free problems.

2. **An extensive experimental evaluation of the proposed localization algorithm for cooperative pedestrian tracking** - In the experimental phase of this research radio and inertial traces were used to build error models. These error models were then used to synthetically create cooperative pedestrian dead reckoning problems to evaluate the proposed algorithm. The key findings of this experimental study were the following:

   (a) **Graph rigidity theory can be used effectively to certify whether an offline, range-based localization problem affords a unique solution** - The proposed algorithm uses the existence of measurements to determine whether sensor positions are sufficiently constrained to make the problem localizable. In the sensor networking literature, graph rigidity has been used to certify the uniqueness of 2D static localization problems.

   (b) **The time complexity of widely-used graph realization algorithms precludes their use in offline mobile localization over long periods** - Efficient sparse graph realization remains an unsolved problem. For small static localization problems with tens or hundreds of devices, approximation methods work well. However, the time complexity of these methods scale greater than quadratically with the number of graph vertices. In mobile localization problems the number of vertices scales linearly with time, and therefore even simple problems contain thousands of vertices and cannot be realized.

   (c) **The addition of more sensors can adversely affect the performance of sparse non-linear optimization if the solver is initialized poorly** - As a result of the poor time complexity of graph realization, the only feasible method of carrying out localization is to use sparse non-linear least-squares optimization initialized with inertial trajectories. Simulation

results confirmed the intuition that the addition of anchors increases the quantity of information, improving localization performance. However, the addition of sensors has the opposite effect. This behaviour results from the fact that, compared an anchor, the addition of a sensor significantly increases the state space of the problem. More importantly, however, the state values are initialized from dead reckoning, which implies that they are initially severely misaligned. Unlike the single-sensor case where there is a smooth gradient that draws the sensor into alignment with the anchors, in the cooperative localization problem the error function has many local minima into which the localization algorithm converges.

(d) **A novel drift correction algorithm that performs well when the inertial and ground truth trajectories are not orientated within a global coordinate frame** - The objective of piecewise drift correction is to remap segments of a drift-corrupted inertial trajectory to lie between correction points. We showed that linear drift correction, an existing approach from the literature, fails to perform well in cases where the inertial trajectory is not orientated correctly with respect to the ground truth.

8.3 Relation to motivating applications

Recall that three motivating applications were listed in the first chapter of this thesis: customer tracking for retail, office space usage monitoring and underground badger tracking. The findings of this thesis relate to these applications in the following ways:

1. **Planning** - Under the assumption that localization is to be carried out over a finite region using sensors with a fixed radio communication range that move according to some unknown mobility model, the results in this thesis showed that localizability is related to the number of anchors and experiment duration. These two factors provide a tuning method for planning sensor deployments. Using an appropriate simulation, one may determine a conservative number of anchors to deploy in order to ensure that the graph becomes localizable with a certain probability within some time threshold. For example, researchers might use this approach to decide on the number of anchors to deploy in badger tracking application to ensure localizability.

2. **Energy** - The per-bit energy cost of storing data in sensor networks is usually large. Applications such underground badger tracking or office space usage

monitoring may require that the respective collars or tags operate for several months. Storing all inertial and radio measurements might therefore be impossible because of limited energy budgets. Reducing these measurements to range estimates substantially lowers the dimensionality and frequency at which information is recorded, making localization a feasible objective over long periods.

3. **Privacy** - The additional information brought by peer-to-peer radio encounters provides information that gives a theoretical lower bound on localization error. Although, for reasons mentioned earlier, in practice this may be difficult to achieve. Irrespective of this, obtaining peer-to-peer encounters may be difficult. For example in retail settings there are clear ethical issues with distributing an application that uses customers to cooperatively track one another. Although monitoring office space usage may be good for optimizing business processes, tracking employees again creates ethical issues.

4. **Sensing** - The simulation study in this thesis presumed that it is possible to collect motion and radio measurements from foot-mounted sensors. For real pedestrian dead reckoning deployments it is seldom possible to use foot-mounted sensors, and one must rely rather on data sensed from waist-mounted sensors, watches or mobile phones. The further away from the foot the sensor, the more difficult it is to reliably detect step events, and the less accurate the assumption of the zero-velocity update technique. As a result, pedestrian dead reckoning systems may exhibit substantially more error.

8.4 Limitations and future work

The results presented in this thesis motivate the following directions for future work:

1. **Investigate lower-complexity graph realization algorithms** - The simulation study in this thesis highlighted that the commonly-used graph realization algorithms scale more than quadratically in terms of time-complexity. This is acceptable for static localization problems, where the problem state space is small. However, for mobile localization where the state space increases linearly with time, this poses a large problem. Sparse graph realization is an open problem in mathematics and new algorithms, such as 3D-ASAP by Cucuringu et al. [16], should be investigated.

2. **Develop an online implementation** - Although offline tracking is useful for certain time-insensitive applications, many applications require real-time localization. Since radio range is limited, connectivity cannot be guaranteed and the proposed algorithm must therefore be modified to operate in a distributed fashion. One idea is to have each sensor act as a data relay, ferrying measurements opportunistically between peers. Each sensor would maintain its own best-effort solution, using local measurements and those ferried by peers. An immediate benefit of this distributed version is that communication overhead is low. Recall that the graph encodes only abstract pairwise distances, rather than all measurements. This means that sensors exchange a comparatively small quantity of information in order to perform localization. In sensor networking this is important, as the per-bit energy cost of transmission is high.

3. **Improve drift correction** - Although it performs well in general, the radial drift correction algorithm proposed in this thesis is limited in the sense that it causes discontinuities about the vertices, suffers scaling magnitude issues and fails to constrain one degree of freedom in 3D. One idea is to investigate using a filtering or smoothing approach step to apply corrections to the trajectories. For our example application, the pedestrian dead reckoning filter could be re-run on the raw inertial data using embedded vertices as periodic location corrections.

4. **Incorporate more information**- The proposed algorithm only takes into account radio and motion information, although other types exist and are regularly available for use by the algorithm. Examples include:

   (a) **Maps** - Map information may provide three key benefits to cooperative localization. Firstly, in localization problems without generically globally rigid graphs, map-matching can be used to resolve embedding ambiguities. Secondly, maps may also be used to find the most likely projection of a collection of trajectories into a global coordinate system, when no anchors are available. Finally, maps may potentially be used to refine the projected trajectories, based on rules like "no trajectory can pass through a wall".

   (b) **Location probability priors** - It is seldom the case that movement is truly random. Consider a prior likelihood distribution of sensor's position over the search space, possibly derived as a function of time. For instance, during normal working hours an office employee is expected to be at, or

close to, their desk. This information could be used to either refine a solution or resolve ambiguities in the solution space.

(c) **Negative information** - Assuming that radio beacons are transmitted at a fixed rate, the absence of a beacon indicates that two devices are likely separated by a minimum distance. Like map-matching, negative information may help refine certain localization problems.

8.5 Closing remarks

The offline, range-based localization algorithm proposed in this thesis fuses ideas from sensor networking and robotics. Although it was ultimately found that the graph realization algorithms do not scale well, the exercise was nonetheless useful. It was shown that rigidity theory, which has classically been used to certify the localizability of 2D static problems, can be used in 3D and mobile localization problems. A sparse non-linear least-squares optimization approach was in stead implemented, and its sensitivity to increasing the number of sensors and anchors in the problem was assessed. While an increase in the number of sensors theoretically provides more information and ability to localize to a greater accuracy, it is difficult to achieve this in practice. The underlying reason is that each additional sensor significantly increases the state space of the problem which, when coupled with PDR as an initialization strategy, causes sparse non-linear least-squares optimization to converge to local minima. Finally, it was shown that radial drift correction outperforms linear drift correction when the inertial trajectories are not oriented within the same global reference frame. However, radial drift correction causes piecewise discontinuities at encounter positions, fails to constrain a degree of freedom in 3D and under-performs in rare, but identifiable, situations because of scaling errors.

In the future it may be possible to realize graphs in a computationally-efficient manner, and provide sparse non-linear optimization with a strong initial estimate from which to converge. Although evaluated on indoor pedestrian localization problems, the proposed algorithm is not restricted to this application. For example, cooperative localization of a swarm of aerial vehicles may also be possible using the localization algorithm presented in this thesis. However, if one is to use the proposed model beyond pedestrian dead reckoning, careful attention must be paid to measurement noise, and comprehensive experimentation must be carried out to evaluate the suitability of the algorithm on a per-application basis.

References

[1] M Angermann and P Robertson. FootSLAM: Pedestrian simultaneous localization and mapping without exteroceptive sensors – hitchhiking on human perception and cognition. *Proceedings of the IEEE*, 100:1840–1848, 2012.

[2] L Asimow and B Roth. The Rigidity of Graphs. *Transactions of the American Mathematical Society*, 245:279–289, 1978.

[3] J Aulinas, YR Petillot, J Salvi, and X Lladó. The SLAM problem: a survey. *CCIA*, 2008.

[4] P Bahl and VN Padmanabhan. RADAR: An in-building RF-based user location and tracking system. In *The 19th Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 2, pages 775–784, Tel Aviv, 2000.

[5] P Biswas and Y Ye. Semidefinite programming for ad hoc wireless sensor network localization. In *Information Processing in Sensor Networks, The 3rd International Symposium on*, pages 46–54, Berkeley, CA, 2004. ACM New York, NY.

[6] M Bosse and P Newman. An Atlas framework for scalable mapping. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–25, Taipei, Taiwan, 2003.

[7] M Broxton, J Lifton, and J A Paradiso. Localization on the pushpin computing sensor network using spectral graph drawing and mesh relaxation. *ACM SIGMOBILE Mobile Computing and Communications Review*, 10(1):1–12, 2006.

[8] L Bruno and P Robertson. WiSLAM: Improving FootSLAM with WiFi. In *International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, number September, pages 21–23, Guimarães, Portugal, 2011.

[9] F Cadger, K Curran, J Santos, and S Moffett. A Survey of Geographical Routing in Wireless Ad-Hoc Networks. *IEEE Communications Surveys & Tutorials*, 15 (2):621–653, January 2013.

[10] M Chimani, C Gutwenger, M Jonger, K Klein, P Mutzel, and M Schulz. Open Graph Drawing Framework (OGDF). In *The 15th International Symposium on Graph Drawing*, Sydney, 2007.

[11] K Chintalapudi, AP Iyer, and VN Padmanabhan. Indoor localization without the pain. In *Proceedings of the sixteenth annual international conference on Mobile computing and networking (MobiCom)*, page 173, New York, USA, 2010. ACM.

[12] J Collin, O Mezentsev, and G Lachapelle. Indoor positioning system using accelerometry and high accuracy heading sensors. In *International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GPS/GNSS 2003)*, pages 1–7, Portland, OR, 2003.

[13] R Connelly. Generic Global Rigidity. *Discrete & Computational Geometry*, 33 (4):549–563, October 2005.

[14] I Constandache, RR Choudhury, and M Azizyan. Did You See Bob? : Human Localization using Mobile Phones. In *The 16th annual international conference on mobile computing and networking (MobiCom 2010)*, pages 149–160, Chicago, IL, 2010.

[15] M Cox and T Cox. Multidimensional scaling. *Handbook of data visualization*, pages 315–347, 2008.

[16] M. Cucuringu, a. Singer, and D. Cowburn. Eigenvector synchronization, graph rigidity and the molecule problem. *Information and Inference*, 1(1):21–67, December 2012.

[17] A Cunningham, KM Wurm, W Burgard, and F Dellaert. Fully distributed scalable smoothing and mapping with robust multi-robot data association. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1093–1100, St. Paul, MN, May 2012. IEEE.

[18] J de Leeuw. Applications of convex analysis to multidimensional scaling. *Recent developments in statistics*, pages 133–145, 1977.

[19] F Dellaert and M Kaess. Square Root SAM: Simultaneous Localization and Mapping via Square Root Information Smoothing. *The International Journal of Robotics Research*, 25(12):1181–1203, December 2006.

[20] G. Di Battista and R. Tamassia. Incremental planarity testing. In *The 30th Annual Symposium on Foundations of Computer Science*, number v, pages 436–441, Durham, NC, 1989. Ieee.

[21] BJ Dil and PJM Havinga. On the calibration and performance of RSS-based localization methods. *Internet of Things (IOT), 2010*, pages 1–8, November 2010.

[22] J Djugash, S Singh, G Kantor, and W Zhang. Range-only slam for robots operating cooperatively with sensor networks. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2078 – 2084, Orlando, FL, 2006. IEEE.

[23] H Durrant-Whyte and T Bailey. Simultaneous Localisation and Mapping (SLAM) : Part I The Essential Algorithms. *Robotics and Automation Magazine*, pages 1–9, June 2006.

[24] T Eren, BDO Anderson, W Whiteley, AS Morse, and PN Belhumeur. Merging globally rigid formations of mobile autonomous agents. In *Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1260–1261, Washington, DC, 2004. IEEE.

[25] RM Faragher. Opportunistic radio SLAM for indoor navigation using smartphone sensors. In *Position Location and Navigation Symposium (PLANS)*, pages 120–128, Myrtle Beach, SC, April 2012. IEEE/ION.

[26] B Ferris, D Hähnel, and D Fox. Gaussian processes for signal strength-based location estimation. In *Robotics Science and Systems*, Philadelphia, PN, 2006.

[27] B Ferris, D Fox, and N Lawrence. WiFi-SLAM using Gaussian process latent variable models. In *20th international joint conference on artifical intelligence (IJCAI)*, pages 2480–2485, Hyderabad, India, January 2007. Morgan Kaufmann Publishers Inc.

[28] N Fox, F Jagodzinski, Y Li, and I Streinu. KINARI-Web: a server for protein rigidity analysis. *Nucleic acids research*, 39:W177–83, July 2011.

[29] E Foxlin. Pedestrian tracking with shoe-mounted inertial sensors. *IEEE Computer Graphics and Applications*, 25(6):38–46, 2005.

[30] U Frese. Treemap: An O(log n) algorithm for indoor simultaneous localization and mapping. *Autonomous Robots*, 21(2):103–122, August 2006.

[31] U Frese, P Larsson, and T Duckett. A multilevel relaxation algorithm for simultaneous localization and mapping. *IEEE Transactions on Robotics*, 21(2):196 – 207, 2005.

[32] SJ Gortler and DP Thurston. Characterizing the universal rigidity of generic frameworks. Technical report, Department of Computer Science, Harvard University, Cambridge MA, 2010.

[33] PD Groves. *Principles of GNSS, Inertial, and Multisensor Integrated Navigation Systems, Second Edition*. Artech House, London, 2013.

[34] J Guivant and E Nebot. Optimization of the Simultaneous Localization and Map Building Algorithm for Real Time Implementation. *Robotics & Automation*, 17 (3):242 – 257, 2001.

[35] C Gutwenger and P Mutzel. A linear time implementation of SPQR-trees. In *Graph Drawing*, volume 1984, pages 77–90, 2001.

[36] KM Hall. An r-dimensional quadratic placement algorithm. *Management Science*, 17(3):219–229, 1970.

[37] R Harle. A Survey of Indoor Inertial Positioning Systems for Pedestrians. *IEEE Comm. Surveys & Tutorials*, 15(3):1–13, 2013.

[38] B Hendrickson. The Molecule Problem: Exploiting Structure in Global Optimization. *SIAM Journal on Optimization*, 5(4):835–857, November 1995.

[39] JD Hol, TB Schon, and F Gustafsson. On resampling algorithms for particle filters. *Nonlinear Statistical Signal ...*, 2006.

[40] JE Hopcroft and RE Tarjan. Dividing a graph into triconnected components. Technical report, Cornell University, Ithica, N.Y., 1974.

[41] B Jackson and T Jordán. Connected rigidity matroids and unique realizations of graphs. *Journal of Combinatorial Theory, Series B*, 94(1):1–29, 2005.

[42] D J Jacobs and B Hendrickson. An algorithm for two-dimensional rigidity percolation: the pebble game. *Journal of Computational Physics*, 137(2):346–365, 1997.

[43] A Javanmard and A Montanari. Localization from incomplete noisy distance measurements. In *Information Theory, 2011 IEEE International Symposium on*, pages 1584–1588, Saint Petersburg, 2011.

[44] SJ Julier and JJ Laviola. On Kalman Filtering With Nonlinear Equality Constraints. *IEEE Transactions on Signal Processing*, 55(6):2774–2784, 2007.

[45] SJ Julier and JK Uhlmann. A New Extension of the Kalman Filter to Nonlinear Systems. In *Signal Processing, Sensor Fusion, and Target Recognition*, Orlando, FL, 1997.

[46] M Kaess, A Ranganathan, and F Dellaert. iSAM: Incremental smoothing and mapping. *IEEE Transactions on Robotics*, 24(6):1365–1378, 2008.

[47] M Kaess, H Johannsson, R Roberts, V Ila, J Leonard, and F Dellaert. iSAM2: Incremental smoothing and mapping with fluid relinearization and incremental variable reordering. In *Robotics and Automation, The 2011 IEEE International Conference on*, pages 3281–3288, Shanghai, China, 2011.

[48] N Kern, B Schiele, and A Schmidt. Multi-sensor activity context detection for wearable computing. In *First European Symposium on Ambient Intelligence (EUSAI)*, pages 220–232, Veldhoven, The Netherlands, 2003.

[49] B Kim, M Kaess, L Fletcher, J Leonard, A Bachrach, N Roy, and S Teller. Multiple relative pose graphs for robust cooperative mapping. In *Robotics and Automation, 2010 IEEE International Conference on*, pages 3185–3192, Anchorage, AK, May 2010.

[50] S Kim, M Kojima, H Waki, and M Yamashita. SFSDP: A Sparse Version of Full SemiDefinite Programming Relaxation for Sensor Network Localization Problems, June 2010.

[51] Y Koren. On spectral graph drawing. *Computing and Combinatorics*, 2697: 496–508, 2003.

[52] B Krach and P Robertson. Integration of Foot-Mounted Inertial Sensors into a Bayesian Location Estimation Framework. In *5th Workshop on Positioning, Navigation and Communication (WPNC)*, volume 2008, pages 55–61, 2008.

[53] R Kummerle, G Grisetti, H Strasdat, K Konolige, and W Burgard. G2o: A general framework for graph optimization. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3607–3613, Shanghai, China, May 2011.

[54] B Kusy, A Ledeczi, and X Koutsoukos. Tracking mobile nodes using RF Doppler shifts. In *The 5th international conference on Embedded networked sensor systems*, pages 29–42, Sydney, Australia, 2007.

[55] G Laman. On graphs and rigidity of plane skeletal structures. *Journal of Engineering mathematics*, 4(4):331–340, 1970.

[56] N Lawrence. Probabilistic Non-linear Principal Component Analysis with Gaussian Process Latent Variable Models. *Journal of Machine Learning Research*, 6 (Nov):1783–1816, 2005.

[57] M Lourakis and A Argyros. The design and implementation of a generic sparse bundle adjustment software package based on the levenberg-marquardt algorithm. Technical report, Institute of Computer Science, FORTH, Greece, 2004.

[58] M Lourakis and A Argyros. SBA: a software package for generic sparse bundle adjustment. *ACM Transactions on Mathematical Software*, 36(1):1–30, March 2009.

[59] F Lu and E Milios. Globally consistent range scan alignment for environment mapping. *Autonomous robots*, 1997.

[60] SOH Madgwick, AJL Harrison, and A Vaidyanathan. Estimation of IMU and MARG orientation using a gradient descent algorithm. In *The International Conference on Rehabilitation Robotics*, volume 2011, page 5975346, Zurich, Switzerland, January 2011.

[61] A Markham, N Trigoni, DW Macdonald, and SA Ellwood. Underground Localization in 3-D Using Magneto-Inductive Tracking. *IEEE Sensors Journal*, 12(6): 1809–1816, 2012.

[62] M Maróti, P Völgyesi, S Dóra, and B Kusý. Radio interferometric geolocation. *Proceedings of the 3rd ...*, pages 1–12, 2005.

[63] DW Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the society for Industrial and Applied Mathematics*, 11(2): 431–441, 1963.

[64] E. Menegatti, A. Zanella, S. Zilli, F. Zorzi, and E. Pagello. Range-only SLAM with a mobile robot and a Wireless Sensor Networks. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 8–14, Kobe, Japan, May 2009. IEEE.

[65] M Montemerlo, S Thrun, D Koller, and B Wegbreit. FastSLAM: A factored solution to the simultaneous localization and mapping problem. In *Eighteenth National Conference on Artificial Intelligence*, Edmonton, Canada, 2002.

[66] P Newman and J Leonard. Pure Range-Only Sub-Sea SLAM. In *IEEE Conference on Robotics and Automation (ICRA)*, pages 1921—-1926, Washington, DC, 2002. IEEE.

[67] D. Niculescu and B. Nath. Ad hoc positioning system (APS). In *Global Telecommunications Conference*, volume 5, pages 2926–2931. IEEE, 2001.

[68] E. Olson, J. Leonard, and S. Teller. Fast iterative alignment of pose graphs with poor initial estimates. In *IEEE International Conference on Robotics and Automation (ICRA)*, number May, pages 2262–2269, Orlando, FL, 2006. Ieee.

[69] N Patwari and AO Hero. Using proximity and quantized RSS for sensor localization in wireless networks. In *The 2nd ACM international conference on Wireless sensor networks and applications (WSNA)*, page 20, San Diego, CA, 2003. ACM Press.

[70] N Patwari, JN Ash, S Kyperountas, AOII Hero, RL Moses, and NS Correal. Locating the nodes: cooperative localization in wireless sensor networks. *Signal Processing Magazine*, 22(4):54–69, July 2005.

[71] AR Pratama and R Hidayat. Smartphone-based Pedestrian Dead Reckoning as an indoor positioning system. In *International Conference on System Engineering and Technology*, pages 1–6, Bandung, Indonesia, September 2012. Ieee.

[72] NB Priyantha, A Chakraborty, and H Balakrishnan. The Cricket location-support system. In *The 6th annual international conference on Mobile computing and networking (MobiCom)*, pages 32–43, New York, NY, 2000. ACM Press.

[73] A Rai and K Chintalapudi. Zee: Zero-effort crowdsourcing for indoor localization. In *The18th annual international conference on Mobile computing and networking*, pages 293–304, Instanbul, Turkey, 2012.

[74] CE Rasmussen. *Gaussian Processes For Machine Learning*. MIT Press, 2006.

[75] P Robertson, M Angermann, and M Khider. Improving Simultaneous Localization and Mapping for pedestrian navigation and automatic mapping of buildings by using online human-based feature labeling. In *IEEE/ION Position, Location and Navigation Symposium*, pages 365–374, Indian Wells, CA, May 2010. Ieee.

[76] P Robertson, MG Puyol, and M Angermann. Collaborative pedestrian mapping of buildings using inertial sensors and FootSLAM. In *ION GNSS*, Portland, OR, 2011.

[77] D Roller, M Montemerlo, S Thrun, and B Wegbreit. Fastslam 2.0: an improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In *Sixteenth International Joint Conference on Artificial Intelligence (IJCAI)*, Acapulco, Mexico, 2003.

[78] A Savvides, C Han, and MB Strivastava. Dynamic fine-grained localization in ad-Hoc networks of sensors. In *The 7th annual international conference on Mobile computing and networking (MobiCom)*, pages 166–179, New York, NY, 2001. ACM Press.

[79] JB Saxe. Embeddability of weighted graphs in k-space is strongly NP-hard. In *The 7th Annual Allerton Conference on Communication, Control and Computing*, pages 480–489, Monticello, IL, 1979.

[80] DE Schinstock. GPS-aided INS Solution for OpenPilot. Technical report, Kansas State University, Manhattan, KA, 2010.

[81] PH Schönemann. A generalized solution of the orthogonal procrustes problem. *Psychometrika*, 31(1):1–10, March 1966.

[82] Y Shang and W Ruml. Improved MDS-based localization. In *The 23rd Conference of the IEEE Computer and Communications Societies*, volume 4, pages 2640–2651, Hong Kong, 2004.

[83] Y Shang, W Ruml, Y Zhang, and MPJ Fromherz. Localization from mere connectivity. In *The 4th ACM international symposium on mobile ad hoc networking and computing (MobiHoc)*, pages 201–212, Annapolis, MD, 2003.

[84] D Sibley, C Mei, I Reid, and P Newman. Adaptive relative bundle adjustment. In *Robotics: science and systems*, pages 1–8, Seattle, WA, 2009.

[85] F Sivrikaya and B Yener. Time Synchronization in Sensor Networks: A Survey. *IEEE Network*, 18(4):45–50, 2004.

[86] I Skog, P Händel, J Nilsson, and J Rantakokko. Zero-velocity detection - an algorithm evaluation. *IEEE transactions on biomedical engineering*, 57(11):2657–2666, July 2010.

[87] A Symington and N Trigoni. Encounter based sensor tracking. In *The 13th ACM international symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, page 15, Hilton Head, SC, 2012. ACM Press.

[88] S Thrun and M Montemerlo. The GraphSLAM Algorithm with Applications to Large-Scale Mapping of Urban Structures. *The International Journal of Robotics Research*, 25(5-6):403–429, May 2006.

[89] S Thrun, W Burgard, and D Fox. *Probabilistic Robotics*. MIT Press, Cambridge, MA, 1 edition, 2005.

[90] B Triggs and PF McLauchlan. Bundle adjustment – a modern synthesis. *Vision Algorithms: Theory and Practice, LNCS*, 34099:298–372, 2000.

[91] S Ungarala, E Dolence, and K Li. Constrained Extended Kalman Filter. In *8th International IFAC Symposium on Dynamics and Control of Process Systems*, volume 2, pages 63–68, Cancún, Mexico, 2007.

[92] Harvey Weinberg. Using the ADXL202 in pedometer and personal navigation applications. Technical report, Analog Devices, Norwood, MA, 2002.

[93] O Woodman and R Harle. Pedestrian localisation for indoor environments. In *Proceedings of the 10th international conference on Ubiquitous computing (UbiComp)*, pages 114–123, New York, NY, 2008. ACM Press.

[94] MH Wright. The interior-point revolution in optimization: History, recent developments, and lasting consequences. *Bulletin of the American Mathematical Society*, 42(01):39–57, September 2004.

[95] Shuozhi Yang and Qingguo Li. Ambulatory walking speed estimation under different step lengths and frequencies. In *2010 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, pages 658–663. IEEE, July 2010.

[96] M Youssef and A Agrawala. The Horus WLAN location determination system. In *Proceedings of the 3rd international conference on Mobile systems, applications, and services (MobiSys)*, pages 205–218, Seattle, WA, 2005.

[97] Z Zhu, AM So, and Y Ye. Universal Rigidity: Towards Accurate and Efficient Localization of Wireless Networks. In *The IEEE Conference on Computer Communications*, pages 1–9, San Diego, CA, March 2010. Ieee.

Appendix A

Generic Global Rigidity in 3D

Gortler and Thurston [32] show that a randomized algorithm can be used to certify generic global rigidity in $\mathbb{E}^d$, given any connected graph with v vertices and e edges. The overarching objective is to create a framework ρ with random integer coordinates. This framework is then used to create a *rigidity matrix*, which describes the instantaneous velocities of vertices with respect to their neighbourhood. If the rank of the rigidity matrix is sufficiently large, then the graph is generically locally rigid in $\mathbb{E}^d$. A linear system is then solved to calculate an *equilibrium stress matrix* of the framework ρ. Gortler and Thurston [32] proved that certifying generic global rigidity in $\mathbb{E}^d$ is equivalent to checking whether the rank of this matrix is sufficiently large. This algorithm never returns a false positive, but returns a false negative with probability related to the range from which the random framework's coordinates were drawn.

Given a graph $G = (V, E)$ having v vertices and e edges, generic global rigidity in $\mathbb{E}^d$ is certified in the following way:

1. Given that $p = 1 - ve/N$, select an appropriate target confidence p, and calculate N, given v and e.

2. Select a random framework ρ by drawing v random d-dimensional vertex coordinates uniformly from the range $[1, N]$.

3. Use ρ to calculate the rigidity matrix R. A rigidity matrix is an $e \times vd$ matrix, describing the velocity of each vertex with respect to its neighbours. For example, assume we have a framework in $\mathbb{E}^d$ with edge k linking vertices i and j. Let $p_i^x$and p_j^x be the coordinate of i and j along coordinate axis x. Row k of the rigidity matrix will therefore contain a value of $p_i^x - p_j^x$ in column $(i-1) \times d + x$, and a value of $p_j^x - p_i^x$ in column $(j-1) \times d + x$, for all $x = 1, ..., d$.

4. Construct E and b in the following way. First, define H to be a $e - vd + \binom{d+1}{2} \times e$ matrix of random integers drawn from the range $[1, N]$. Append H to the bottom of the transposed rigidity matrix R' to form matrix E of size $e + \binom{d+1}{2} \times e$. Let b a zero vector of length $e + \binom{d+1}{2}$, except for a single $\mathbf{1}$ value in any of the last $e - vd + \binom{d+1}{2}$ elements.

5. Solve the linear system $E\omega = b$, where. The vector ω is the equilibrium stress vector for the randomly-chosen framework ρ.

6. By removing the last $e - vd + \binom{d+1}{2}$ elements from the vector ω, we convert it into a equilibrium stress vector. This vector can then be converted into an equilibrium stress matrix Ω in the following way. The kth element in ω corresponds to an edge between two vertices i and j. We begin by initializing $\Omega = \mathbf{0}_{v \times v}$ For every element k in ω, we find the corresponding i and j and set $[\Omega]_{ij} = [\Omega]_{ji} = -\omega_k$. Finally, we set each element of the diagonal of Ω to the negative sum of its row elements. i.e. For each $i \in V$ we set $[\Omega]_{ii} = \sum_{j=1}^{v} -\Omega_{ij}$.

If the rank of R is equal to $vd - \binom{d+1}{2}$, then the graph is *generically locally rigid* in $\mathbb{E}^d$ with a false negative probability bounded by $1 - ve/N$. Furthermore, if the rank of Ω is equal to $v - d - 1$, then the framework is *generically locally rigid* in $\mathbb{E}^d$ with a false negative probability bounded by $1 - ve/N$.

Appendix B

Weighted Multidimensional Scaling

The objective of Weighted Multidimensional Scaling (WDS) is, given some distance matrix Δ and weight matrix W, to find a set of coordinates $\tilde{X} = \{x_1, ..., x_v\}$ that solves the stress minimization problem below.

$$\begin{aligned} f(X) &= \sum_{<i,j>\in\Delta} w_{ij} \left(\left\| x_i - x_j \right\| - \delta_{ij} \right)^2 \\ \tilde{X} &= \arg\min_X f(X) \qquad w_{ij} \in W, \quad \delta_{ij} \in \Delta \end{aligned}$$

de Leeuw [18] proposed a suitable iterative stress majorization algorithm called *Scaling by Majorizing a Complicated Function* (SMACOF), and proved that it monotonically decreases $f(X)$. SMACOF uses a principle known as majorizing, which requires a surrogate function $g(X, Y)$ that majorizes the function $f(X)$. The key idea is that it may be difficult to find an analytical solution to minimizing $f(X)$, and so we use this surrogate function as a proxy. The surrogate function must satisfy the following two properties.

$$\begin{aligned} g(X, Y) &\leq f(X) \\ g(Y, Y) &= f(Y) \end{aligned}$$

The value of Y is fixed, and known as the *supporting point*. This leads to the following *sandwich inequality*, where $\tilde{X}$ is the minimizing value.

$$f(\tilde{X}) \leq g(\tilde{X}, Y) \leq g(Y, Y) = f(Y) \tag{B.1}$$

Majorization involves a controlled selection of a X and Y values to minimize $g(X, Y)$. By virtue of the sandwich inequality, this selection mechanism also minimizes the objective function $f(X)$. In order to iteratively solve the WDS problem SMACOF finds an appropriate support function for $f(X)$. To do this, we begin by

construct scalar $s_{ij}(X)$ as per Eqn B.2, and a matrix A_{ij} to take a value of 1 at a_{ii} and a_{jj}, -1 at a_{ij} and a_{ji}, and zeros elsewhere.

$$s_{ij}(X) = \begin{cases} \delta_{ij}/d_{ij}(X) & \text{if } d_{ij}(X) > 0 \\ 0 & \text{if } d_{ij}(X) = 0 \end{cases} \tag{B.2}$$

From these definitions we use Eqn B.3 two construct further matrices, V and $B(X)$.

$$\begin{aligned} V &= \sum_{i<j} w_{ij} A_{ij} \\ B(X) &= \sum_{i<j} w_{ij} s_{ij}(X) A_{ij} \end{aligned} \tag{B.3}$$

The weighting matrix may be multiplied by a constant factor without affecting the minimization result. A such, we are able to assume that $\sum_{i<j} w_{ij}\delta_{ij}^2 = \frac{v(v-1)}{2}$, where v is the number of vertices in the graph. We may expand the objective function and replace the summations with matrix operations. The function $\mathrm{tr}(K)$ calculates the sum of the elements along the diagonal of matrix K.

$$f(X) = \sum w_{ij} \left(\delta_{ij} - d_{ij}(X)\right)^2 \tag{B.4}$$

$$= \sum w_{ij}\delta_{ij}^2 + \sum w_{ij} d_{ij}^2(X) - 2\sum w_{ij} d_{ij} d_{ij}(X) \tag{B.5}$$

$$= \frac{v(v-1)}{2} + \mathrm{tr}\left(X'VX\right) + 2\mathrm{tr}\left(X'B(X)X\right) \tag{B.6}$$

The Cauchy-Swartz inequality leads to the following sandwich inequality

$$f(X) = \frac{v(v-1)}{2} + \mathrm{tr}\left(X'VX\right) + 2\mathrm{tr}\left(X'B(X)X\right) \tag{B.7}$$

$$\leq \frac{v(v-1)}{2} + \mathrm{tr}\left(X'VX\right) + 2\mathrm{tr}\left(X'B(Y)Y\right) = g(X,Y) \tag{B.8}$$

The support function $g(X,Y)$ is a quadratic function, which is minimized by setting its derivate equal to zero.

$$\frac{\partial g(X,Y)}{\partial X} = 2VX - 2B(Y)Y = \mathbf{0}$$

Solving this equation requires the use of the Moore-Penrose pseudoinverse $V^+ = (V + \frac{1}{v}\mathbf{1}\mathbf{1}')^{-1} - \frac{1}{v}\mathbf{1}\mathbf{1}'$, where $\mathbf{1}$ is a vector of v ones. The value $\overline{X}$ that minimizes $g(X,Y)$ is thus given by $\overline{X} = V^+B(Y)Y$. SMACOF performs majorization over given number of iterations k. The support point Y is initialised with a configuration

$\overline{X}_0 = X_0$, from the *Graph Embedding* step of EBT. The stress at time $t = 0, \ldots, k$ is calculated as $f(\overline{X}_t)$. In each step the support function $g(X, Y)$ is minimized to obtain a new estimate $\overline{X}_t$. The algorithm terminates after T_i iterations, or when $f(\overline{X}_t) - f(\overline{X}_{t-1}) < T_e$.