Hybrid Topological-Metric Simultaneous Localisation and Mapping

Kirill Kouzoubov

11 October 2008
Declaration

This thesis is an account of research undertaken between March 2002 and June 2006 at The Department of Systems Engineering, Faculty of Engineering, The Australian National University, Canberra, Australia.

Except where acknowledged in the customary manner, the material presented in this thesis is, to the best of my knowledge, original and has not been submitted in whole or part for a degree in any university.
Acknowledgements

I would like to thank my supervisors Dr. David Austin and Prof. John Moore. Special thanks to Dr. Alex Zelinsky for encouraging me to take up the scholarship while I was a summer scholar in RSISE.

I express my gratitude to the organisers and hosts of the SLAM summer school of 2002. I have learned a lot about SLAM over that week. Presentations and code examples were extremely helpful. Sandwiches with cheese and capsicum slices, served in the afternoon, were of great help too.

I have visited Intelligent Robot Laboratory, University of Tsukuba in the winter of 2002. I have enjoyed three months in Japan, even though it was rather cold sometimes. I would like to thank Prof. Shinichi Yuta and Dr. Masahiro Tomono for the supervision during that time and I thank the team from Roboken for their hospitality and sake.

Dr. David Leibowitz, Dr. Nick Barnes and Dr. Matthew Ridley have helped with English in most parts of this thesis, readability has improved significantly thanks to their efforts. Thank you to Rosemary Shephered for her great help with the administrative side of this thesis.

Finally I acknowledge the contribution my wife Vlada has made to this thesis, her support during the final stages of this thesis was crucial.
Abstract

This thesis deals with the problem of simultaneous localisation and mapping (SLAM). The problem of SLAM has been studied over the past two decades by many researchers. Limitations of traditional approaches are well understood in the research community. Some of the more challenging aspects of the SLAM problem include: ambiguous data association in the presence of clutter, loop closing, large maps, dynamic environments.

Traditional approaches to SLAM cannot operate in large environments. Computational complexity is one of the limiting factors. A more important limitation is the accumulation of systematic approximation errors that ultimately lead to an inconsistent map. New map representations are needed to overcome these limitations.

This thesis presents a novel map representation that combines metric and topological information - hybrid topological SLAM (HTSLAM). The map is a graph: local maps form nodes of the graph, each local map has its own reference frame and edges of the graph maintain spatial information and coordinate transformations between adjacent regions. There is no global reference frame. FastSLAM is used for building local maps. The uncertainty in the coordinate transformations between adjacent maps is modelled using particles. This representation allows constant time map updates and robust loop closing. Since there is no global reference frame, this approach does not suffer from constantly increasing uncertainty.

The ideas of this thesis are tested on real-life data. Three different data sets are considered: a laser range scanner detecting corners in indoor environment, a laser range scanner detecting trees in the park (the well known Victoria Park data set), and a vision system detecting vertical edges in an indoor environment.
# Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acknowledgements</td>
<td>2</td>
</tr>
<tr>
<td>Abstract</td>
<td>3</td>
</tr>
<tr>
<td><strong>1 Literature review</strong></td>
<td>9</td>
</tr>
<tr>
<td>1.1 Lessons from Biology</td>
<td>9</td>
</tr>
<tr>
<td>1.2 Robotic Mapping</td>
<td>11</td>
</tr>
<tr>
<td>1.3 Localisation</td>
<td>12</td>
</tr>
<tr>
<td>1.3.1 Problem Definition</td>
<td>13</td>
</tr>
<tr>
<td>1.3.2 Common Approaches to Metric Localisation</td>
<td>14</td>
</tr>
<tr>
<td>1.4 Simultaneous Localisation and Mapping</td>
<td>15</td>
</tr>
<tr>
<td>1.4.1 SLAM Problem</td>
<td>15</td>
</tr>
<tr>
<td>1.4.2 Kalman Filter Approach</td>
<td>16</td>
</tr>
<tr>
<td>1.4.3 Particle Filter Approach</td>
<td>17</td>
</tr>
<tr>
<td>1.4.4 Effect of Ambiguous Data Association on Posterior Distribution</td>
<td>18</td>
</tr>
<tr>
<td>1.5 Loop Closing</td>
<td>20</td>
</tr>
<tr>
<td>1.6 Limitations of Metric Approaches to Mapping</td>
<td>22</td>
</tr>
<tr>
<td>1.7 Hybrid Approaches to SLAM</td>
<td>24</td>
</tr>
<tr>
<td>1.7.1 Atlas</td>
<td>25</td>
</tr>
<tr>
<td>1.8 Summary</td>
<td>26</td>
</tr>
<tr>
<td><strong>2 Notation and Background Theory</strong></td>
<td>27</td>
</tr>
<tr>
<td>2.1 Notation</td>
<td>27</td>
</tr>
<tr>
<td>2.2 Kalman Filter</td>
<td>27</td>
</tr>
<tr>
<td>2.2.1 Extended Kalman Filter (EKF)</td>
<td>29</td>
</tr>
<tr>
<td>2.3 Particle Filter</td>
<td>31</td>
</tr>
<tr>
<td><strong>3 HTSLAM Map Structure</strong></td>
<td>33</td>
</tr>
<tr>
<td>3.1 Hybrid Map structure</td>
<td>33</td>
</tr>
<tr>
<td>3.2 Modelling Uncertainty within a Local Map</td>
<td>36</td>
</tr>
</tbody>
</table>
3.3 Modelling Link Uncertainty ........................................... 37
3.4 Defining Region Boundaries ........................................ 39

4 Mapping Procedure ....................................................... 40
4.1 Initialisation .......................................................... 40
4.2 Local Mapping Procedure ............................................. 40
  4.2.1 Modifications to the Rao-Blackwellised Particle Filter .... 40
4.3 Monitoring Robot Location in the Topological Map .......... 41
4.4 Starting a New Local Map ............................................. 42
4.5 Switching Maps - Revisiting ........................................ 43

5 Multiple Hypothesis and Loop Closing ................................. 47
5.1 Loop Detection ........................................................ 47
  5.1.1 Computing Relative Map Poses ................................ 48
5.2 Map Matching ........................................................ 49
  5.2.1 Map Alignment from 2 Landmark Correspondences ......... 55
  5.2.2 Final Map Alignment ............................................. 56
5.3 Augmenting Global Map Structure .................................. 58
  5.3.1 Globally Consistent Approach ................................ 60
  5.3.2 Local Approach ................................................ 61
5.4 Confirming Loop Closing Hypothesis .............................. 61

6 Summary and Comparison ................................................. 63
6.1 Summary of HTSLAM .................................................. 63
6.2 Comparison ........................................................... 63
  6.2.1 Global Mapping Approaches ................................... 63
  6.2.2 Atlas ............................................................ 64

7 Experimental Results: Laser Scanner ................................ 66
7.1 Feature Detectors ...................................................... 66
  7.1.1 Corner Detector .................................................. 66
  7.1.2 Tree Trunk Detector ............................................. 69
7.2 Mapping ............................................................... 69
  7.2.1 Point Landmarks ................................................ 69
  7.2.2 XR4000: Odometry Model ....................................... 71
7.2.3 Truck: Odometry Model ........................................ 73
7.2.4 Region Model: Indoors ........................................ 73
7.2.5 Region Model: Victoria Park .................................. 73
7.3 Experiment Setup and Analysis ................................... 74
7.4 Results Indoors .................................................... 75
7.4.1 Execution time .................................................. 76
7.5 Results Victoria Park .............................................. 76

8 Experimental Results: Vision ....................................... 85
8.1 Sensor Description ................................................ 85
8.2 Feature Detector: Vertical Edges .................................. 85
8.2.1 Sensor Uncertainty ............................................. 88
8.3 Edge Landmarks .................................................... 89
8.3.1 Data Association ............................................... 90
8.3.2 Observation Update .......................................... 91
8.3.3 Genesis .......................................................... 92
8.4 Defining Map Region .............................................. 93
8.5 Results ............................................................. 93
8.5.1 Error Analysis .................................................. 99

9 Conclusion ............................................................. 100
9.1 Future work ....................................................... 101

Bibliography ............................................................ 103
# List of Figures

1.1 Example topological map. ........................................... 12  
1.2 The robot pose uncertainty, example ........................... 19  
2.1 Particle filter algorithm. .......................................... 32  
3.1 Example of an HTSLAM map structure. .......................... 34  
3.2 Example of an HTSLAM map for an indoor environment ....... 34  
3.3 Example of an HTSLAM map for an outdoor environment .... 35  
3.4 Modeling transition distribution ................................. 37  
4.1 Map regions used for triggering transition events .......... 42  
4.2 Example: revisiting map region. ................................. 44  
5.1 Breadth First Traversal ........................................... 48  
5.2 Reflection and alignment ......................................... 50  
5.3 Effect of unmodelled environment features on localisation 59  
6.1 Types of corner features ........................................ 66  
6.2 Trying to fit a line around the corner. .......................... 67  
6.3 Corner extraction from the laser scan. .......................... 68  
6.4 XR4000 odometry bias ............................................ 72  
6.5 Example of region evolution ..................................... 74  
6.6 Example of an inconsistent map .................................. 75  
6.7 Indoor environment: RSISE level 3. .............................. 75  
6.8 Laser indoors: Map of corners (100 particles) .................. 77  
6.9 Laser indoors: Map of corners (300 particles) ................ 78  
6.10 Laser indoors: Global Path over 100 test runs. ............... 79  
6.11 Laser indoors: Execution time 100 particles ................. 80  
6.12 Laser outdoors: Map of corners (100 particles) .............. 82  
6.13 Laser outdoors: Map of corners (300 particles) .............. 83  
6.14 Laser outdoors: Global Path over 100 test runs. .............. 84
LIST OF FIGURES

8.1 Barrel distortion ................................................. 86
8.2 Edge Detection Example ........................................ 88
8.3 Vision indoors: Map of edges (300 particles/map 52) .................. 94
8.4 Vision indoors: Map of edges (100 particles/map 89) .................. 95
8.5 Vision indoors: Jumped out, failed to come back (300 particles/map 40).
Topologically correct, but not “optimal” .......................... 96
8.6 Vision indoors: Failed transition 7 to 2 (300 particles/map 18) .... 97
8.7 Vision indoors: Global Path over 100 test runs. ....................... 98
1.1 Lessons from Biology

A lot of robotics research is biologically inspired. There are plenty of primitive tiny creatures in the natural world capable of very sophisticated behaviour. It is inspiring for a robotics researcher to know that complex tasks can be achieved with very limited resources, and we can learn how it can be done by studying biological systems.

Of particular interest to this thesis are the navigational skills that many organisms possess. It is generally assumed that sophisticated navigational behaviour can only be made possible if an organism maintains some form of map of the environment in their brain. A good overview of robotics research that mimics the navigational behaviour of biological systems is given in [27].

It is generally a difficult problem to decipher the exact representation an animal is using to describe the environment. There is only so much one can recover from observing the behaviour of an animal. We will therefore concentrate on the studies of the spatial representation in humans.

Early research in spatial representation in humans [77] assumed that environmental representations are actually stored in the brain in cartographic form. Piaget believed that a child’s intellectual development can be divided into four discrete stages. These are more than just convenient ways of breaking down a description of child development. He believed that each stage represents a qualitatively different type of understanding of the world from the previous one, and not just a quantitative increase in knowledge of the world. Piaget proposed the following stages:

1. Sensorimotor Intelligence

The child’s thought is almost entirely perceptual and is based on interactions with objects in the environment. Representation of the environment is relative to his own body only.
2. Preoperational Intelligence

The child's thought starts getting detached from the immediate environment. Objects and places in the external world become symbols in the mind, which can then be manipulated in a fairly limited way. Reasoning is intuitive rather than logical, and knowledge is relatively unstructured.

3. Concrete Operational Intelligence

The child is able to imagine multiple perspectives on a situation. He realises that a sequence of actions can be cancelled out by the reverse sequence, and that the same goal can be achieved by a number of different means.

4. Formal Operational Intelligence

Thinking becomes abstract and logical, free from direct experience.

Piaget’s theory of spatial development traces the evolution of understanding of two related concepts: projective relations and Euclidean geometry. As the child develops through successive stages of cognitive structure, these concepts develop in parallel, but also become increasingly coordinated. The child develops from early stages, where spatial thinking is tied to his own experience and images of the environment, to one where higher-order logical properties (e.g. the formal mathematical and geometrical properties of a Cartesian or Euclidean coordinate system) are applied to the relative locations of objects in space.

The rough robotic analogy of these stages would be

1. No map at all, reactive navigation with collision avoidance, odometry-based localisation.

2. Set of routes - topological maps without cycles.

3. Topological maps with cycles.

4. Global map with single reference frame. Relative locations of all landmarks are known with some certainty.

Similarly to Piaget, Siegel & White [82] proposed that adults construct a cognitive representation of the physical world at two levels. At the first level, sequences of landmarks are linked to the route. Each landmark has directional information attached to it: “turn left after passing 7/11 store”. Distances are left unspecified. As an
adult gains experience of the environment, he incorporates a sense of relative distances between landmarks. As this information gets incorporated into the map it becomes possible to recover bearing between any two points in the environment regardless of whether or not they are adjacent.

Siegel & White [82] see this process as the first step in the coordination of routes into a fully metric survey-like representation of the environment (equivalent of stage 4 in Piaget model).

However, many researchers disagree with the idea that the human brain stores a cartographic representation of the environment. Kuipers [59] argues that such a single metric representation would commit the system to a high degree of accuracy and would thus not support states of partial knowledge. He argues that a single mistake on such a representation would be very costly to rectify at a later date, requiring the readjustment of large portions of the representation.

Golledge [29] suggested that we may accurately represent the metric relations (i.e. distance and direction) between a limited number of major reference points, and that these then become fixed points of reference for more local groupings at a lower hierarchical level. We thus locate one of the minor landmarks by first orienting ourselves towards its nearest major reference landmark; from this we can then infer the route to the minor landmark. The relative orientations of the fixed systems are poorly specified and retain some subjective component; thus it is problematic to infer relative location directly between minor points in different systems. One must rather go up and down in the hierarchy.

1.2 Robotic Mapping

There are two main paradigms of robotic mapping: topological and metric. The topological map is essentially a graph. Nodes of the graph represent places in the environment. Edges of the graph correspond to routes between places. Every edge contains some information on how to get from one node to another. This information can be metric (e.g. go east for about 10m), but is generally too coarse, sufficient only to get a robot moving in the right direction. The robot is expected to rely on its path following (go down the corridor, for example) and place recognition skills for navigation. An example of a topological map used by people is a subway schematic (See Figure 1.1).

With a topological map, the robot is restricted to the routes along the edges of the
Map. The robot cannot take shortcuts even if they are physically possible.

Metric maps are generally more detailed and hence useful. The robot pose and the map features are defined in a single reference frame. Metric maps can have landmarks (unique or identical), or they can represent the environment as an occupancy grid. Each cell of an occupancy grid map is assigned a probability value of being “occupied by an obstacle”. A hybrid metric map, that combines landmarks and occupancy grids has also been proposed in [36].

1.3 Localisation

Robot localisation, estimating the pose of a robot, is one of the fundamental problems in the field of mobile robotics. It is considered one of the prerequisites for providing autonomous capabilities for a mobile robot [15]. The localisation algorithm can face three possible problems

- Position tracking
  The robot’s initial pose is known with relatively high certainty. The robot keeps track of its pose as it moves through a familiar environment.

Figure 1.1: An example of a topological map, the Moscow subway map.
• **Global Localisation**

The initial pose of the robot is unknown, or is given by a probability distribution \( p(x_0) \) with high variance. As the robot moves through the environment and collects observations, it determines its pose.

• **Kidnapped robot**

The robot is moving through the environment and is reasonably certain about its pose. Suddenly it gets moved by an external force (think of a robotic vacuum cleaner) to some other place within a known map. The robot should be able to detect that its estimate of the pose is no longer correct and re-localise.

Much research has previously concentrated on position tracking - in fact many existing algorithms address only the first problem (see review in [6]). Global localisation and the Kidnapped robot problems have been addressed using multi-hypothesis Kalman filters [45, 14], Markov localisation [25], and Monte Carlo methods [90].

### 1.3.1 Problem Definition

The problem of localisation can be thought of as a problem of estimating the probability density distribution of robot pose conditioned on the observation and control data. This posterior is typically called *Belief* and is denoted

\[
Bel(x_k) = p(x_k|U_{k-1}, Z_k),
\]

where \( U_{k-1} \) is a set of all control inputs up to time \( k - 1 \), and \( Z_k \) is a set of all observations up to time \( k \). It is usually desired to approximate the posterior recursively, otherwise one would run into computation problems very quickly. Making a Markov assumption that future data is independent of past data given the robot pose:

\[
p(z_k|Z_{k-1}, U_{k-1}) = p(z_k|x_k) \tag{1.2}
p(x_k|Z_{k-1}, U_{k-1}) = p(x_k|x_{k-1}, u_{k-1}). \tag{1.3}
\]

By applying Bayes’ rule, equation (1.1) is transformed into a recursive formula (see [90] for a derivation)

\[
Bel(x_k) = \eta p(z_k|x_k) \int p(x_k|x_{k-1}, u_{k-1}) Bel(x_{k-1}) dx_{k-1}, \tag{1.4}
\]
where \( \eta \) is a normalisation constant. To implement the recursive rule in (1.4) one needs to know the distributions \( p(z_k|x_k) \) and \( p(x_k|x_{k-1}, u_{k-1}) \). These distributions are generally referred to as observation (or sometimes perception) and motion models, respectively.

The motion model reflects the uncertainty in robot dynamics, so it is generally independent of time \( k \). The observation model is a function of robot sensors and of the environment. If sensor characteristics do not change with time and if features in the map are static (a common assumption), the same observation model can be used for every time step.

### 1.3.2 Common Approaches to Metric Localisation

Several approaches to localisation have been proposed. One family of algorithms use the Extended Kalman Filter\([44]\). Approaches based on the EKF linearise motion and observation models and approximate the posterior distribution with a Gaussian. While being computationally efficient, EKF based approaches have some limitations. They are not capable of global localisation, hence the initial pose of the robot should be known with relatively high confidence. An EKF is only capable of representing a unimodal posterior distribution, meaning that in an ambiguous situation only one of the position hypotheses can be considered. This can lead to localisation failure if the wrong choice is made.

A mixture of Gaussians model can be used to allow for multiple hypotheses \([45, 14]\), but the problems arising from linearisation are still present in these approaches.

Markov localisation uses a grid to model the posterior distribution \([25]\). This approach is capable of global localisation, and can deal with the kidnapped robot problem as well. Fox et al. \([26]\) have shown that Markov localisation works well, even in highly crowded environments. The main advantage of this approach is the ability to model any complex distribution. This makes the algorithm more robust and allows for global localisation. Drawbacks include finite grid size, computational demands and the need for a complex implementation to optimise the observation update and increase resolution “on the fly”.

Particle filter localisation (also known as Monte Carlo localisation) algorithms model the posterior distributions as a set of particles \([90, 43]\). A particle filter is capable of approximating any probability distribution, including multi-modal distribu-
§1.4 Simultaneous Localisation and Mapping

It can perform global localisation and can be used to solve the kidnapped robot problem. Unlike grid-based algorithms, particle filters have a floating point resolution for robot pose, and so can give a more accurate pose estimate. Although it is computationally more demanding than EKF approaches, the increased robustness of this algorithm makes it worth the additional computational resources. The computational complexity of the particle filtering algorithm is $O(N)$ in the number of particles used to represent the robot pose. Increasing the number of particles will result in greater accuracy and robustness. It is therefore straightforward to trade off the computational resources against performance.

1.4 Simultaneous Localisation and Mapping

1.4.1 SLAM Problem

This section introduces the problem of Simultaneous Localisation and Mapping [83, 87]. A robot starts at time 0 at location $x_0$, executes a series of motion commands $U_k = [u_0, u_1...u_k]$ and collects a set of observations at every step $Z_k = [z_0, z_1...z_k]$. In practice observation and motion commands are not synchronised, however this can be easily accommodated in the implementation. It is therefore commonly assumed that observation and motion happen at the same discrete time interval. Control and perception of the robot are subject to noise.

SLAM seeks to find the true path of the robot up to time $k$, $X_k = [x_0, x_1,...x_k]$, and the true model of the environment, $M_k$. Since there is an inherent uncertainty in the robot’s observation and motion, one generally cannot give an exact description of the robot’s path and a map of the environment. However, it is possible to construct a probabilistic model that assigns a likelihood to every path-map combination given all the available observations, control inputs and the initial pose of the robot:

$$p(X_k, M_k|Z_k, U_k, x_0).$$ \hspace{1cm} (1.5)

Assuming that odometry errors are independent of the observation errors and of the environment model, one can separate the model into an observation and motion parts and define:

1. The model of the robot’s motion $P(x_k|U_k)$.

If one makes a further assumption that past odometry errors have no effect on fu-
ture errors, one can represent robot motion as a Markov process, with probability distribution
\[ P(x_k | U_k) \equiv p(x_k | x_{k-1}, u_k). \] (1.6)

2. The model of the robot’s perception \( p(z_k | x_k, M_k, Z_k) \).

Here again an assumption is made that errors in the measurements are independent of past errors, hence
\[ p(z_k | x_k, M_k, Z_k) \equiv p(z_k | x_k, M_k). \] (1.7)

Equipped with the knowledge of these distributions one can in principle compute the exact posterior by considering every possible path, map and data association decision combination. In practice, however, computing such a distribution exactly is not possible due to the high dimensionality of the system. Reasonably good approximations can, however be obtained, and can be computed incrementally in real-time.

The two most common approaches to solving the SLAM problem are the EKF based algorithm [41] and a particle filter algorithm called “FastSLAM” [70].

1.4.2 Kalman Filter Approach

The Extended Kalman Filter approach treats the SLAM problem as a state estimation problem in which the state consists of the robot pose and locations of all landmarks. Usually the number of features is not known in advance, so most implementations have some provision for adding new elements to the map as new observations are obtained. EKF SLAM requires linearisation of the robot’s motion and perception models, so that the estimation error can be modelled by a Gaussian distribution. The computation complexity of the algorithm is \( O(K^3) \), where \( K \) is the number of features in the map [41].

Considerable research has addressed the computational performance of the EKF approach to SLAM [99, 101, 53, 31, 32, 2, 94, 84]. It is possible to exploit the fact that only a small subset of the whole map is observed at any given time to reduce the computational burden. Compressed EKF [32] is an optimal approach that reduces the computational cost of measurement update to \( O(K_a^2) \), where \( K_a \) is a number of features in the local region of the global map. This approach still requires a costly \( O(K^2) \) global registration stage when robot changes from one local region to the next.
A sub-optimal version of the same algorithm bounds the cost of the global registration stage.

Constrained Local Submap Filters [99] by Williams uses local maps for building the immediate environment around the robot, effectively forgetting about the rest of the world. The local map is then merged into the global reference frame. This keeps the computational requirements relatively constant, irrespective of the environment size.

The most important limitation of the EKF approach is the inability to model multiple data association decisions. Every observation has to be classified as noise, a new feature or an observation of an existing feature. It is possible to postpone the final decision by running multiple Kalman filters in parallel. Alternatively, one can try to improve the quality of data association decisions. Batch data association techniques like [72, 84] can reduce the ambiguity of data association, but it is impossible to eliminate this problem completely. It is impossible to undo incorrect data association in EKF, as a result EKF is quite sensitive to data association errors. Handling of large cycles is a very difficult problem for EKF SLAM, due to the inability to deal with ambiguous data association.

1.4.3 Particle Filter Approach

FastSLAM uses particles to approximate the posterior distribution [70]. Every particle maintains its own path, data association decisions and the resulting map. The map is a collection of independent landmarks. Particle locations are estimated using a Kalman filters for every landmark. FastSLAM requires linearisation of the perception model, but it samples directly from the motion model. As a result, no change to the motion model is required. FastSLAM can be implemented in $O(M \log K)$, where $M$ is the number of particles and $K$ is the number of landmarks.

Given enough particles, FastSLAM produces a reasonable estimate of the posterior distribution. Unfortunately it is not clear how many particles are needed for a particular problem. A modified version of the FastSLAM algorithm [71] uses an alternative sampling technique to generate particles where they are needed most, therefore reducing the number of particles needed. The authors [71], demonstrate that in some cases even one particle is enough.

The main advantage of FastSLAM is its ability to handle multiple data association decisions [69], which makes it a more robust algorithm. From the implementation
Perspective, FastSLAM is trivial to parallelise since computation per particle is independent of other particles. This allows implementations to exploit the recent trend towards larger numbers of processors, rather than faster processors, in emerging computational hardware.

1.4.4 Effect of Ambiguous Data Association on Posterior Distribution

One of the advantages of the EKF - representing a posterior distribution as a Gaussian, is also a weakness. Linearisation allows for an efficient implementation and provides a sound mathematical background for the algorithm. However, posterior distributions arising in SLAM can be distinctly non-Gaussian depending on the environment, robot dynamics and sensor characteristics.

One of the factors affecting the true posterior distribution is the quality of data association decisions. In the case of perfect data association, the posterior distribution tends to have a single peak and can be approximated with a Gaussian quite well. When data association is unreliable, multiple plausible hypothesis for data association arise. These multiple hypothesis produce a multi-modal posterior distribution.

To illustrate this point, consider a simple mapping example: the world consists of two point landmarks and a robot living on a two-dimensional plane, as shown in Figure 1.2. The robot pose is defined by its location on the plane only, ignoring the orientation of the robot (one can assume that the robot has an extremely accurate compass). The robot is equipped with a sensor that reports landmark location relative to the robot pose. The error of the sensor is zero-mean Gaussian.

The robot makes an observation at point (0,0), then moves one meter forward and makes another observation, see Figure 1.2(a). The robot dynamics are controlled by the following equation:

\[
\begin{bmatrix}
    x_k \\
    y_k
\end{bmatrix} =
\begin{bmatrix}
    x_{k-1} + r_k \cos(\Theta_k) \\
    y_{k-1} + r_k \sin(\Theta_k)
\end{bmatrix},
\]

here control inputs at time \( k \) are \( r_k \) and \( \Theta_k \) for distance and direction of travel, respectively. Gaussian noise in actuation produces a banana-shaped distribution over the robot pose, shown in Figure 1.2(b).

Consider first the scenario when data association is perfect. After execution of the motion command the robot observes the landmarks once more. Since the observations
Figure 1.2: In 1.2(a) a robot makes an observation at point (0,0), then moves 1 meter forward and makes another observation. The square shows the region for which pose uncertainty is computed. The resulting uncertainty in robot pose just before the second measurement is shown in 1.2(b). Two scenarios are considered: known and ambiguous data association (rows two and three respectively). The uncertainty due to the measurement is shown in 1.2(c) and 1.2(e). The robots pose uncertainty after incorporating the measurement is presented in 1.2(d) and 1.2(f).
were not certain for both first and second observation, the pose estimate that can be derived from the measurements has a significant uncertainty. This is illustrated in Figure 1.2(c). The posterior distribution is non-Gaussian in this case as well, although the difference is not as apparent as in the motion example. The posterior distribution of the robot pose given observations and control input is shown in Figure 1.2(d), while it is not strictly Gaussian, it can be argued that it can be approximated by a Gaussian reasonably well.

In the case when data association is ambiguous, the posterior distribution is slightly more complicated. Let’s assume the robot has three equally likely data association hypotheses.

- Observations 1 and 2 map to landmarks a,b
- Observation 1 maps to landmark b, observation 2 is spurious
- Observation 2 maps to landmark a, observation 1 is spurious

Each of these hypotheses by itself produces an almost Gaussian posterior distribution, but when combined the result is a multi-modal distribution shown in Figure 1.2(c). The posterior distribution of the robot pose given observations and control input is then as shown in Figure 1.2(f). Clearly this distribution differs significantly from a Gaussian.

The presented example shows that non-Gaussian distributions in robot posterior can arise as a result of an ambiguous data association. Approaches to SLAM that can deal with such distributions therefore might have an advantage in such situations.

1.5 Loop Closing

The problem of loop closing arises when mapping a large cyclic environment. When the robot returns to a previously mapped region via a large loop, rather than by backtracing its own steps, it needs to recognise the place as previously mapped. Failure to do so will result in inconsistent map. As the robot moves through an unexplored environment while building the map, it becomes more and more uncertain about its pose relative to the regions it mapped in the past. Increased uncertainty in robot pose makes it more difficult to detect a place the robot has already mapped.

In order to build a consistent map, the robot needs to be able to detect the places it has visited before and correct its estimate of the path along the loop. The problem of mapping large cyclic environments has been addressed by several researchers.
Lu and Milios [66] propose an algorithm they call “consistent pose estimation”. The algorithm uses laser scan matching and linearised global optimisation over scan poses to build a scan map of the environment in a single global frame. The approach builds a network of robot poses at which a laser scan has been taken. The poses are connected by “weak” and “strong” links. The links represent the spatial relations between poses. Weak links are derived from odometry and strong links are derived from the scan matches. The algorithm then finds a global configuration of the poses of the places by minimising a cost function. If one thinks of links as springs, the algorithm finds the configuration that reduces the sum of forces exerted by the springs. Spatial relations are modelled by Gaussians, and the cost function is the sum of the Mahalanobis distances.

The method by Lu and Milios is very sensitive to the initial estimate of the robot pose just before loop closing. Since the pose estimate is derived from local scan matching and odometry, it can, in principle, have unbounded error. If the initial estimate is too far from the true robot pose, the algorithm tends to converge to spurious local minima. The method also has high computational complexity, $O(N^3)$ in the number of poses.

Konolige and Gutman [55] provide a method based on the ideas of Lu and Milios, but try to address these problems. Instead of using single laser scan to find relations between places they use a map correlation technique [54]. This makes the system more robust to false correspondences. After detecting loop closing, the algorithm runs consistent pose estimation.

An interesting scan matching based mapping technique has been developed by Ferguson et al. [23]. In this approach loop closing can be undone if the observations further down the track do not support it.

Thrun et al.[91, 85, 89] suggest an approach that uses an Expectation Maximisation (EM) algorithm to construct maps of cyclic environments. The algorithm is given a relatively small set of observations of sparsely positioned non-unique landmarks and the set of all control inputs. In the expectation step the algorithm computes the most likely robot path given the map and the data. In the maximisation step the most likely map is computed given the robots path from the E-step. The Expectation and Maximisation steps are repeated until the map converges to the maximum likelihood solution. This algorithm cannot be implemented in real-time because it needs to have
access to all of the future and past data to determine the robot pose at any given time.

All of the methods presented so far build a map of the whole environment in a single global reference frame. Recently, Bosse et al. [7] suggest a mapping approach, *Atlas*, that does not require a global reference frame. The map structure used in this approach is a hybrid topological/metric map. Each node is a local map and every link in the topological map represents a spatial relationship between the two places it connects.

*Atlas* uses a combination of map matching and uncertainty projection to detect when the robot re-enters a previously explored area. Uncertainty projection computes the relative poses between the two non-adjacent local maps. If according to this estimate, the two maps might be overlapping, *Atlas* tries to find the landmark correspondences between the two maps. If the two maps match, the new link is added to the topological map. The pose relation for this link is computed from the landmark correspondences. In order to decrease the probability of incorrect data association, *Atlas* runs several hypotheses at any given time. If, after closing the loop, the observations do not match the map well enough, the loop closing hypothesis will be discarded in favour of the new map hypothesis.

### 1.6 Limitations of Metric Approaches to Mapping

While there exist several good methods for building metric maps, they do not scale up to big environments. The main restricting factors are:

1. **Increase in the computational complexity.** The larger the environment the longer it takes to execute an update with every new measurement.

2. **Increase of the robot pose uncertainty.** As you run the algorithm for longer, errors tend to accumulate, especially in large environments. No matter how accurate your mapping process is, there is still going to be some residual uncertainty, and it will accumulate as the robot travels longer distance. The effect of such errors is especially harmful when cycles are present.

3. **Increasing approximation error in the mapping procedure.**

The first point is fairly obvious: more landmarks require more computation and more memory. Since the environment is considered as a whole, one needs access to all
landmarks to continue mapping. As discussed earlier in Section 1.4.2, the problem of computational complexity has been addressed by many. In particular, of interest are compressed EKF [32] and constrained local submap filter [99].

Increasing uncertainty is another fundamental problem when a map is built incrementally, i.e. new measurements are added to the existing map as they arrive. In order to add a measurement the robot needs to solve a data association problem: is this observation from one of the features already in the map, is it new or is it noise? As the robot’s pose uncertainty in the existing map increases, data association becomes more difficult. An incorrect data association decision becomes more likely, leading to inconsistent maps. Even in the case when perfect data association is possible (when unique landmarks are used), increasing pose uncertainty will cause problems, since pose uncertainty also means map uncertainty. Even if the robot manages to re-localise itself, correcting the map can be a problem.

The problem of data association under uncertainty has been addressed by [72, 84]. Rather than finding the mapping of a single observation to a landmark, these approaches consider a set of observations simultaneously. While such approaches make data association more robust to sensor pose uncertainty, they do not completely resolve the problem of increasing uncertainty.

Algorithms that work on all of the data at once, like Expectation Maximisation [89], are not affected by this problem, but their computational requirements are huge and they cannot operate in real time, since all of the data has to be processed at once.

In the case of FastSLAM, data association is solved on a per particle basis. Data association is therefore not affected by the increasing uncertainty. FastSLAM can also correct the map after loop closing automatically and in real time, by simply keeping only the “good” particles. Nevertheless the increasing uncertainty is still a problem. In order to perform reliably, FastSLAM will require more and more particles to accurately represent the increasing robot pose uncertainty. Like any particle filtering algorithm, FastSLAM is affected by the particle depletion problem: if there are not enough particles in the right region, the algorithm will fail to detect a loop-closing.

EKF SLAM tends to underestimate robot pose uncertainty, and as a result it might fail to detect loop closing. Correcting the map and the path, even if the robot does re-localise itself, is also a big problem for EKF SLAM, and does require substantial computational resources.
Another important limitation comes from the approximation error. Every mapping algorithm is trying to compute the map as a joint posterior distribution of robot poses and a map given all the observations and control inputs. Since computing such a distribution in its true form is rather impractical, all existing approaches try to approximate it. As the environment size increases the effect of approximation becomes more significant.

1.7 Hybrid Approaches to SLAM

If one segments the environment into a number of smaller regions, and maps each region separately (build local maps), one can avoid the problem of unbounded complexity and also reduce the effects of approximation errors.

Hybrid topological/metric maps [23, 7, 85] allow for accurate local positioning by virtue of the metric component, yet are easier to construct since computational complexity per local map is bounded. It is also easier to close large cycles since closing a cycle in such a map is just a matter of adding a link between two places in the topological structure of the map.

Some of the hybrid methods desire to build a global metric map [85, 91]. These approaches use topological structure as an intermediate step only. The environment is split into local maps, each local map is explored separately, and all maps are then merged into a large, global map. Merging can happen incrementally or as a single post-processing step. Some optimisation steps might be used to enforce consistency of the global map. The main advantage of such methods is a more robust loop closing behaviour. This is due to the fact that loop closing is performed based on map matching.

Other methods build topological maps with independent metric maps for each node [12, 61]. Metric maps are built for every node of the topological map as an after thought. The relationships between nodes is purely topological. Metric information is only used inside the local maps.

More recently a tighter coupling of metric and topological information started to appear in robotic research: Atlas by Bosse et al. [7] incorporates metric information into the topological structure of the map. Every link in the topological structure stores the spatial relation between the two nodes (local maps) as a stochastic variable. Since the approach described in this thesis is using ideas similar to that of Atlas, it is worth
§1.7 Hybrid Approaches to SLAM

presenting it in greater detail.

1.7.1 Atlas

Atlas by Bosse et al. [7] is a hybrid topological/metric approach to SLAM. Atlas models environment as a set of local maps of bound size. There is no single global reference frame, only local transformations between adjacent local maps are maintained. Spatial relationships are approximated by a Gaussian, derived from the residual robot pose uncertainty of the mapping process used to build the local maps.

Atlas supports various underlying mapping modules for building local maps. The two mapping modules presented in [7] were based on EKF SLAM using line and point features extracted from either laser or sonar range data.

The main advantage of Atlas is the map structure it uses. Moving away from the single reference frame allows the mapping of large environments with huge cycles in real time. Even though there is no global reference frame, it is still possible to compute the relative poses of any two local maps with some certainty, hence it is possible to project the robot pose estimate from one reference frame to any other, along with the uncertainty.

When closing large cycles, Atlas relies on place recognition (map matching) and the robot pose estimate to detect a revisiting event. If the two maps match, and the robot pose derived from the match is likely, a new link can be added to the topological structure. In order to avoid false positives, a loop closing hypothesis is first tested by considering how well the observations match the map.

There are, however, some limitations to this approach. When, for instance, the robot returns to a previously mapped region it gains new knowledge about the relative positions of the local maps. The robot can use this knowledge to update the links in the map. Atlas does not perform this update because of the computational requirements involved, although the authors mention that such update can be performed after the mapping is complete. Atlas uses metrics to test loop closing hypotheses. These metrics do not necessarily correspond to the true likelihood, making it difficult to predict the behaviour of the algorithm in a new environment.
1.8 Summary

The past few decades of mobile robotics research provide powerful tools for mapping and localisation. Existing algorithms tend to be limited to relatively small environments. The main limiting factors are:

- The computational burden: as size of the environment grows, so do the computational requirements of existing algorithms.

- The data association problem in the face of uncertainty: in large environments the uncertainty of the landmarks far away from the origin is high. That makes it difficult to perform correct data association.

Various optimisation techniques and faster hardware can expand the capabilities of existing algorithms, but to overcome these limitations new map representations are required. In particular new ways to model uncertainty should be explored.
Chapter 2

Notation and Background

Theory

This chapter introduces the background theory necessary for development in subsequent chapters, in particular Kalman and Particle filtering and the application of these techniques to the problem of simultaneous localisation and mapping (SLAM). First, a notation that will be used throughout this document is introduced in Section 2.1. Section 2.2 gives a brief overview of the Kalman and Extended Kalman filters (Section 2.2.1). The particle filter is introduced in Section 2.3.

2.1 Notation

The pose of a robot is represented by $x^a_k$. Superscript $a$ indicates a reference frame and $k$ discrete time. A map of region $a$ at time $k$ is denoted $M^a_k$. Individual particles are distinguished by a particle index superscript $m$: $x^a_{k,m}$ and $M^a_{k,m}$. The control input at time $k$ is $u_k$ and an observation at time $k$ is $z_k$. Further notation will be introduced as needed, and is summarised in Table 2.1.

2.2 Kalman Filter

R.E. Kalman published his famous paper describing a recursive solution to the linear filtering problem in 1960 [52]. Since then the Kalman filter has been the subject of extensive research and application in various fields of science. It is used heavily in robotics for localisation and mapping purposes, and is briefly described below.

The Kalman filter addresses the general problem of estimating the state $x_k \in \mathbb{R}^n$ of a discrete-time controlled process that is governed by the linear stochastic difference
Table 2.1: Notation used throughout the thesis.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_{k,m}^{a,m}$</td>
<td>pose of the $m$-th particle in the $a$-th map frame at time $k$.</td>
</tr>
<tr>
<td>$X_k^a$</td>
<td>path of the $m$-th particle in the $a$-th map frame from time 0 to time $k$.</td>
</tr>
<tr>
<td>$u_k$</td>
<td>control input at time $k$.</td>
</tr>
<tr>
<td>$z_k^a$</td>
<td>subset of all observations up to time $k$, that apply to the map frame $a$.</td>
</tr>
<tr>
<td>$u_k$</td>
<td>subset of all observations up to time $k$, that apply to the map frame $a$.</td>
</tr>
<tr>
<td>$Z_k^a$</td>
<td>subset of all teleportations up to time $k$, that apply to the map frame $a$.</td>
</tr>
<tr>
<td>$\theta_i^{a,m}$</td>
<td>$i$-th map element from the $m$-th particle’s map in the map frame $a$.</td>
</tr>
<tr>
<td>$S_k^a$</td>
<td>$m$-th particle in the SLAM particle filter at time $k$, in the map frame $a$.</td>
</tr>
<tr>
<td>$t_{a\rightarrow b}^k$</td>
<td>a set of transition vectors from map frame $a$ to map frame $b$.</td>
</tr>
<tr>
<td>$T_{a\rightarrow b}$</td>
<td>a set of transition vectors from map frame $a$ to map frame $b$.</td>
</tr>
<tr>
<td>$a \sim p(A)$</td>
<td>$a$ is a sample from the distribution $p(A)$.</td>
</tr>
</tbody>
</table>

**equation**

$$x_k = A_k x_{k-1} + B_k u_{k-1} + w_{k-1},$$

(2.1)

where $A_k$ is a matrix that describes the evolution of the state from time $k-1$ to $k$ in the absence of control input or process noise, matrix $B_k$ relates control input $u_{k-1} \in \mathbb{R}^l$, to the change in the state vector and $w_k$ is process noise.

Indirect measurement of the state $z \in \mathbb{R}^m$ is of the form

$$z_k = H_k x_k + v_k,$$

(2.2)

where $H_k$ is a matrix that relates the state to the observation at time $k$ and $v_k$ is measurement noise. Both process and measurement noise are assumed to be zero mean Gaussian with known covariances $Q_k$ and $R_k$ respectively. Process and measurement noise are assumed to be independent:

$$w_k \sim N(0, Q_k),$$

(2.3)

$$v_k \sim N(0, R_k).$$

(2.4)

An estimate of the system state at time $k$, $\hat{x}_k$, can be computed recursively if an estimate of the initial state of the system is available. The uncertainty of the estimate is assumed to be Gaussian with covariance $\Sigma_k$. 
Kalman filtering consists of two stages: time update and measurement update. In the time update stage the filter predicts a new state based on the current state estimate and the process model. The measurement update stage corrects the estimate using the new measurement. The state update is given by (see [16] for detailed derivation):

\[
\dot{x}_k' = A_k \dot{x}_{k-1} + B_k u_{k-1} \tag{2.5}
\]

\[
\Sigma_k' = A_k \Sigma_{k-1} A_k^T + Q_{k-1} \tag{2.6}
\]

Equations (2.5) and (2.6) constitute the prediction stage of the Kalman filter. State \( \dot{x}_k' \) estimates system state at time \( k \) before incorporating measurement and \( \Sigma_k' \) is its covariance.

The measurement update equations are:

\[
K_k = \Sigma_k' H_k^T (H_k \Sigma_k' H_k^T + R_k)^{-1} \tag{2.7}
\]

\[
\dot{x}_k = \dot{x}_k' + K_k (z_k - H_k \dot{x}_k') \tag{2.8}
\]

\[
\Sigma_k = (I - K_k H_k) \Sigma_k' \tag{2.9}
\]

Equation (2.7) computes the Kalman gain, \( K_k \). The Kalman gain defines the contribution of the observation to the new state estimate. \( K_k \) is then used in (2.8) and (2.9) to update the estimate of the state and the corresponding uncertainty in accordance with the noisy observation \( z_k \) with covariance \( R_k \).

The Kalman filter provably converges to the true state given the assumptions of linearity and Gaussian noise [52]. These are fairly restrictive assumptions, since many real-life problems are non-linear and are subject to non-Gaussian noise.

### 2.2.1 Extended Kalman Filter (EKF)

The Kalman filter approach can be applied to non-linear systems by using first order Taylor series approximation of the process and measurement models. This approach is generally known as the Extended Kalman Filter.

Assume that state \( x_k \) evolves according to a non-linear stochastic difference equation, subject to noise \( w_k \):

\[
x_k = f(x_{k-1}, u_{k-1}, w_{k-1}). \tag{2.10}
\]
No direct measurement of the state is available. Instead the state is observed via a non-linear function \( h \), subject to some noise \( v_k \)

\[
    z_k = h(x_k, v_k).
\]  

(2.11)

Process and measurement noise are assumed to be independent and drawn from zero mean Normal distribution.

The update stage of the EKF computes a preliminary estimate of the state and updates the uncertainty of the estimate as follows:

\[
    \hat{x}'_k = f(\hat{x}_{k-1}, u_{k-1}, 0) \\
    \Sigma'_k = A_k \Sigma_{k-1} A_k^T + W_k Q_k W_k^T
\]

(2.12) \hspace{1cm} (2.13)

Matrices \( A_k \) and \( W_k \) in the equations above are partial derivatives of the state evolution function evaluated at \( \hat{x}_{k-1} \):

\[
    A_{k[i,j]} = \frac{\partial f[i]}{\partial x[j]}(\hat{x}_{k-1}, u_{k-1}, 0) \hspace{1cm} (2.14)
    W_{k[i,j]} = \frac{\partial f[i]}{\partial u[j]}(\hat{x}_{k-1}, u_{k-1}, 0). \hspace{1cm} (2.15)
\]

The measurement update stage requires a projection of the state uncertainty into the observation space. Since the observation function \( h \) is non-linear, the uncertainty of the projection is not Gaussian. The EKF approximates it by a Gaussian as follows:

\[
    K_k = \Sigma'_k H_k^T (H_k \Sigma'_k H_k^T + V_k R_k V_k^T)^{-1} \\
    \hat{x}_k = \hat{x}'_k + K_k(z_k - h(\hat{x}'_k, 0)) \\
    \Sigma_k = (I - K_k H_k) \Sigma'_k
\]

(2.16) \hspace{1cm} (2.17) \hspace{1cm} (2.18)

Here matrices \( H_k \) and \( V_k \) are partial derivatives of the observation function:

\[
    H_{k[i,j]} = \frac{\partial h[i]}{\partial x[j]}(\hat{x}_{k-1}, 0) \hspace{1cm} (2.19)
    V_{k[i,j]} = \frac{\partial h[i]}{\partial v[j]}(\hat{x}_{k-1}, 0). \hspace{1cm} (2.20)
\]

Unlike the Kalman filter, the EKF is not guaranteed to converge, and is likely to fail if the system is highly non-linear. However there are many successful applications
of the EKF in various fields of research. In robotics the EKF has been successfully used for localisation [44] and SLAM [19, 7, 35] problems.

2.3 Particle Filter

Particle filters have proven to be a useful tool in robotics, used for localisation [90, 88], mapping [70] and people tracking [81], to name just a few applications. In this thesis a particle filter is used for mapping and localisation.

The particle filter addresses a similar problem to that of the Kalman filter - to track a variable of interest as it evolves over time. Unlike the Kalman filter, a particle filter approximates the posterior distribution of the state by a set of weighted particles. Given enough particles, one can approximate any distribution, hence particle filters can be applied to non-linear problems and will generally provide a better approximation than an EKF.

In the present context of mapping and localisation, the task of a particle filter is to track the state of a controllable and partially observable discrete time Markov chain[28]. The state of the Markov chain at time $k$ is denoted by $x_k$. The state depends on the previous state $x_{k-1}$ and the control $u_{k-1}$ asserted in the time interval $(k-1; k]$, according to the probabilistic law

$$p(x_k|x_{k-1}, u_{k-1}). \quad (2.21)$$

Note that the Kalman filter and EKF state evolution equations (2.1),(2.10), can be expressed in the form of a probability like the one above. The state of the Markov chain is not directly observable, instead a stochastic projection $z_k$ of the true state $x_k$ generated via the probabilistic law is available

$$p(z_k|x_k). \quad (2.22)$$

The initial state $x_0$ is given by some distribution $p(x_0)$. The particle filter algorithm is presented in pseudo-code in the Figure 2.1 and briefly summarised below.

First the set of all particles $X_0$ is initialised by drawing $M$ samples from the initial distribution $p(x_0)$. For simplicity assume, without loss of generality, that control inputs and observations arrive at the same time. The particle filter first predicts the state by sampling from the distribution $p(x_k|x_{k-1}^m, u_{k-1})$. This sampling is applied to every
//Initialise
for \( m = 1 \) to \( M \)
\[
x_0^m \sim p(x_0)
\]
add \( x_0^m \) to \( X_0 \)
endfor
\( k = 0; \)

//Main Loop
while MoreDataIsAvailable

//Propagate particles
for \( m = 1 \) to \( M \)
\[
get x_{k-1}^m \text{ from } X_{k-1}
\]
\[
x_k^m \sim p(x_k|x_{k-1}^m, u_{k-1})
\]
add \( x_k^m \) to \( X_k' \)
endfor

//Update particle weights
for \( m = 1 \) to \( M \)
\[
w_k^m = p(z_k|x_k^m)
\]
add \( w_k^m \) to \( W \)
endfor

//Resample according to the weight.
\[
X_k = \text{resample}(X_k', W)
\]
k = \( k + 1 \)
end

Figure 2.1: Particle filter algorithm.

A particle in the set \( X_{k-1} \) and a new set \( X_k' \) is created. Particles in the new set are then assigned a weight, which is equal to the likelihood of the state represented by the particle, given the observation \( z_k \). Finally, the set of all particles \( X_k \) is computed by sampling with replacement from the set \( X_k' \) in proportion to the weight of the particles.

The resampling step selects particles that have higher likelihood associated with them with higher probability, and in doing so a new set \( X_k \) is created that approximates a random sample from \( p(x_k|Z_k) \), where \( Z_k \) is a set of all observations up to time \( k \). The resampling step is not required after every observation.
Hybrid topological SLAM (HTSLAM) models the map as a hybrid topological-metric structure: the environment is divided into a number of regions, and a local metric map is built for each region. The relative poses of the regions are stored in a topological structure. There is no global reference frame. Only relative poses of the nearby regions are stored.

The structure is motivated primarily by the following. First, dividing the world into a set of local maps appears to approximate the way that humans reason [59]. Second, it ensures that the mapping scheme scales to large environments. The scalability limitations of a single map SLAM methods are well known [36, 31, 32] and prohibit real-time SLAM for maps of practical size. The third and final motivation is that a robot does not require a single, global, metric description of the world for navigation. For example, in the “fetch me a beer from the refrigerator” task, there is no need for the robot to know the position of the fridge while it is in the living room. The robot only needs to know the position of the fridge when it is near to the fridge.

This chapter describes the HTSLAM approach to map modelling. Some aspects of the map design are motivated by the mapping procedures developed in Chapter 4, and will be clarified then. The following section describes the overall map structure. Section 3.2 presents the methods used for modelling uncertainty within each local map and section 3.3 describes the methods used to model the uncertainties of the topological links between maps. Finally, section 3.4 discusses methods for defining the shape and extent of each local region.

3.1 Hybrid Map structure

In HTSLAM, a map is a connected graph. Every node of the graph corresponds to a region within the environment. Each node has its own reference frame and stores a
§3.1 Hybrid Map structure

Figure 3.1: Example of an HTSLAM map structure. Each region has its own coordinate frame. A local map is built for every region (ellipses represent landmarks).

Figure 3.2: Example of an HTSLAM map for an indoor environment


Figure 3.3: Example of an HTSLAM map for an outdoor environment

local, metric map of the region and some representation of the extent of the region in the local coordinate frame. An example of an HTSLAM map is given in Figure 3.1. Note that there is no global reference frame. The edges of the graph represent the stochastic coordinate transformation between adjacent local maps only. Figure 3.2 shows an example of an HTSLAM map for an indoor environment, and Figure 3.3 for an outdoor environment.

The structure used in HTSLAM is similar to that of Atlas [7], however HTSLAM uses different uncertainty representations, both for the coordinate transformations between adjacent map frames and for the local maps. Atlas, as a framework, allows the use of an arbitrary mapping model, although the two implementations that have been published include EKF SLAM and scan matching. Atlas models the uncertainty in the coordinates of links or transitions between maps as Gaussian. It therefore requires that spatial uncertainty in robot pose is represented by a Gaussian distribution.

HTSLAM, on the other hand, uses FastSLAM for local mapping, permitting multimodal modelling of the local map. HTSLAM also models the transitions between maps using sets of particles.

A further difference between Atlas and HTSLAM is that HTSLAM maintains map extent information for every local map. This extra data allows HTSLAM to perform map transitions and loop closing in a more elegant manner (which will be discussed in Chapter 5). In addition, HTSLAM explicitly models the multiple hypotheses that arise during mapping, including during loop closing, giving a more intuitively appealing and robust approach.
The details of the uncertainty representation, both for each local map and for the transitions between maps, are discussed in the next two sections. Section 3.4 discusses methods for determining the extents of each local map.

## 3.2 Modelling Uncertainty within a Local Map

To understand how uncertainty is represented within the local map, one needs to be familiar with the FastSLAM [69] mapping scheme. An overview of FastSLAM appears in Chapter 2. In summary, FastSLAM uses particles to sample the possible paths of the robot. A map is built for each path. Particles are evaluated based on how well the measurements match the map. Regular resampling is used to prune unlikely paths.

The state $s^{a,m}_k$ of the $m$-th particle in the local map $a$ at time $k$ consists of the path of the particle within this local frame, denoted $X^{a,m}_k$, and the map of the local environment conditioned on the path, $M^{a,m}_k$:

$$
 s^{a,m}_k = [X^{a,m}_k, M^{a,m}_k] \\
 X^{a,m}_k = [x^{a,m}_0, x^{a,m}_1, \ldots, x^{a,m}_k] \\
 M^{a,m}_k = [\theta^{a,m}_1, \theta^{a,m}_2, \ldots, \theta^{a,m}_n],
$$

where $x^{a,m}_k$ is the pose of a robot for the $m$-th particle at time step $k$ in local map $a$ and $\theta^{a,m}_n$ is the position of an $n$-th landmark in local map $a$. The landmark position is a stochastic variable described by a Gaussian distribution. Note that one does not have to store the whole path of a particle, since only the most current pose of the particle is used in the mapping process.

A map is a probability distribution $p(M^{a,m}_k | X^{a,m}_k, Z^{a}_k)$. Assume that the map consists of landmarks and that observations are independent. It follows that a map can be represented by a set of independent landmarks, each conditioned on the path of a particle $p(\theta^{a,m}_n | X^{a,m}_k, Z^{a}_k)$. Note that there is a different map for each particle. FastSLAM approximates $p(\theta^{a,m}_n | X^{a,m}_k, Z^{a}_k)$ with a Gaussian. For point landmarks, the distribution is a two-dimensional Gaussian describing the location of the landmark on the plane. Other landmark characteristics can be incorporated into the stochastic variable as well. For example, if one builds a map of trees, the landmark might have one extra parameter for the diameter of the tree trunk. Note that since there are multiple particles, the actual posterior distribution of the landmark is a mixture of Gaussians. Only planar
3.3 Modelling Link Uncertainty

The spatial relationships between adjacent local maps are represented by a set of particles (see Figure 3.4). Whenever a new map is started, the assumption is made that the current location of the robot is precisely known in the coordinate frame of the new motion is considered in this thesis. The results, however, can be readily extended to three-dimensional motion. Note that, since the landmarks are estimated independently in FastSLAM, there is no need to maintain the global covariance matrix for the map.

Each particle makes its own data association decisions. As a result, FastSLAM captures not only the uncertainty arising from continuous variables like robot pose and sensor measurements, but also uncertainty due to discrete data association decisions. This uncertainty is not represented in EKF approaches [41]. The ability to capture data association uncertainty is advantageous since the data association is ambiguous in most real-life situations.

When mapping of a given local map is complete, the maps of all particles alive at the time are stored within the local map. This set of maps is effectively a sample from all the possible maps given the observations and odometry during the time robot was in the region. Due to use of importance sampling, the sample from the maps is expected to contain the most probable paths and their associated maps (which are expected to be the better maps).

**Figure 3.4:** Relative locations of the adjacent maps are represented by a set of particles.
Thus, the transition between the old and the new map can be most directly computed from the estimate of the robot pose in the old map (a set of FastSLAM path particles) and the pose in the new map (known by assumption). The transition is then a stochastic variable which is easily represented by particles. This is the chief motivation for using a particle representation for the transitions: the choice of representation makes the process of adding new maps straightforward.

Note that when closing loops or, more generally, revisiting maps, there is no similarly easy method for transition determination, and samples must be taken from other distributions. When closing loops in HTSLAM, for example, transition samples are taken from the results of map matching. In map matching, landmark matches are used to provide an estimate of the transition function.

Define $\mathbf{t}^{a \rightarrow b}$ to be a coordinate transformation between coordinate frames $a$ and $b$, and the coordinate transform operator $\oplus$

$$\mathbf{x}_k^b = \mathbf{t}^{a \rightarrow b} \oplus \mathbf{x}_k^a.$$ 

This operator projects the pose of the robot from one coordinate frame to another.

The exact transformation between two maps is never known. It is therefore a stochastic variable. In general, transitions and maps are not independent, hence a joint probability needs to be stored. For adjacent maps $a$ and $b$ this joint probability distribution is $p(\mathcal{M}^a, \mathbf{t}^{a \rightarrow b}, \mathcal{M}^b)$. Effectively, the transition between the two adjacent maps captures a three-way relationship between the two maps and the relative pose of the reference frames of the two maps in a probabilistic manner. The exact mechanism used to store the probability densities is described in more detail in Chapter 4.

It is worth noting that it is possible to derive the joint probability $p(\mathcal{M}^a, \mathbf{t}^{a \rightarrow b}, \mathcal{M}^b)$ even for non-adjacent maps $a$ and $b$ by merging probability densities along the topological path joining the two maps. The presence of loops in the topological structure implies multiple paths between some of the local map pairs. It follows that $p(\mathcal{M}^a, \mathbf{t}^{a \rightarrow b}, \mathcal{M}^b)$ can be derived in several possible ways from the HTSLAM structure. This adds a constraint to the HTSLAM structure which will be discussed in more detail in the chapter on loop closing (Chapter 5).

\footnote{Often it is assumed that the robot is at the origin. Here the position of the robot is chosen to place the origin of the coordinate frame suitably for describing the local map extents.}
3.4 Defining Region Boundaries

Each region has its own “area of influence”. This area is defined by a grid map in the local reference frame. The size of the grid cells is of the order of the robot footprint. Landmarks are allowed to be outside the “area of influence”. However, if the robot leaves the region, it will not be able to add new landmarks to the corresponding map. All landmarks visible from the region are mapped within this region. This implies that some landmarks will be mapped in multiple maps.

By limiting the area of the local map, the HTSLAM algorithm achieves two major goals:

1. Computational requirements per local map are bounded, since there is a finite number of landmarks visible from a finite area.

2. The residual uncertainty of the robot pose within the local region is bounded.

Note that these grid maps define the area of influence of a particular region. They are not meant to serve as occupancy grids, although when operating indoors it is advantageous to align map regions with the structure of the building.

Regions which are neighbours in the topological sense have their region boundaries cropped to minimise the overlap. Grid maps are defined in local coordinates, and must be transformed from one reference frame to another in order to perform such cropping. Since relative poses of the neighbouring maps are known with high certainty, the transformation is performed using the mean of the relative poses of the two regions.

Whenever a new map is started, an initial region for the map is defined. The size and shape of the region depends on the environment and the available sensors. For the indoor experiment with the laser range sensor described in Section 7.1.1, free space information from the laser sensor is used to initialise the region. For the outdoor data set, the region is set to a rectangle of a fixed size aligned with the initial robot pose.

When mapping is complete for a given map, the region of influence is cropped to minimise area that was not actually mapped. This is achieved by computing a convex hull for all landmarks in a map, mean of a landmark location is used and any uncertainty in landmark location is ignored. Area that falls outside the convex hull is removed from the region.
This chapter explains the mapping process of HTSLAM. First, the technique used for building local maps is presented in section 4.2. Section 4.3 deals with the problem of tracking of robot pose in the topological structure of the HTSLAM map. The procedure for starting a new local map is described in section 4.4. Section 4.5 explains how revisiting previously mapped regions is handled in HTSLAM.

4.1 Initialisation

It is generally assumed in the SLAM literature that a robot starts with absolutely no knowledge of the environment. The robot does not know how many landmarks there are or where they are located: its map is initially empty. Similarly, in HTSLAM the initial map contains just one region and an empty map.

4.2 Local Mapping Procedure

4.2.1 Modifications to the Rao-Blackwellised Particle Filter

In this work mapping within a local map is performed using FastSLAM. However, due to the nature of HTSLAM, some modifications have to be made to the original FastSLAM algorithm. The changes affect mainly notation; modifications to the actual implementation of a local map building algorithm are minimal. Unlike the global approach, in HTSLAM each local map is updated only when the robot is within the region of the map. A local map at time $k$ is conditioned on a subset of all observations and control inputs. $Z_k$ denotes a set of all observations up to time $k$; $Z_k^a$ is a subset of $Z_k$ and contains only observations relevant to the region $a$ (i.e. taken when the robot was in the region $a$). Similarly, $U_k$ is a set of all control inputs issued up to time $k$, and $U_k^a$ is a subset relevant to region $a$. The set $X_k^a$ denotes all robot poses within a
local map \( a \) from time 0 up to time \( k \).

In global SLAM the pose of the robot is continuously being tracked by the filter. The set of all observations and control inputs up to time \( k \) and an initial pose at time 0 is sufficient to estimate robot pose at time \( k \). In HTSLAM the robot disappears from the local map at some point, to reappear later in a completely different spot. To a mapping module this “teleportation” of the robot from one spot at time \( k_1 \) to another at time \( k_2 \) appears as a special kind of control input \( \hat{u}_{k_1} \), where the robot position at time \( k_2 \) is taken to be

\[
x_{k_2}^{a} \sim p \left( x_{k_1}^{a} \mid \hat{u}_{k_1} \right).
\]

The set of all such teleportations is denoted \( \hat{U}_k^a \). For each local map HTSLAM estimates the probability density

\[
p \left( \mathcal{M}_k^a, \mathbf{X}_k^a \mid \mathbf{Z}_k^a, \mathbf{U}_k^a, \hat{U}_k^a, x_{k_0}^{a} \right),
\]

where \( \mathcal{M}_k^a \) is a map of the region \( a \) and \( x_{k_0}^{a} \) is an arbitrary initial robot pose (typically at the origin).

### 4.3 Monitoring Robot Location in the Topological Map

At every point in time the robot maintains a map of neighbourhood region. A neighbourhood consists of the current map and a set of maps that are close to a current one in a topological and Cartesian sense. The neighbourhood region is a union of the current map region and projections of neighbouring regions onto the current coordinate frame. Two regions are not considered neighbours if their relative pose uncertainty is too high. Two regions that are close in Cartesian sense but far apart in topological structure are not considered neighbours until their proximity is verified by loop closing procedure. The procedure for dealing with loop closing is discussed in the next chapter.

The robot constantly checks how much of the sensor field of view falls within the current region (see Figure 4.1), and how much falls in the neighbouring regions. When the robot detects that one of the neighbouring regions has a significantly large overlap with the sensor field of view, it switches to that region. The goal is to maximise the proportion of observations that match to an existing map. If one assumes uniform distribution of landmarks in the environment and uniform detection rates for a sensor view (i.e. a landmark straight ahead is as easily detectable as one on the robot’s
4.4 Starting a New Local Map

The pose of a robot is represented by a set of particles: at any time instant HTSLAM maintains a set of path particles, and the corresponding maps conditioned on the path. When a new map is started the relative pose of the current and new map reference frames is different for every particle. Since the pose of the robot in the new map is known exactly (and so is equal for every particle), it is possible to compute the relative transformation between the two maps for every particle. A set of particles is treated as a sample from the distribution \( p\left(t^{A\rightarrow B}, M^A\right) \). The new empty map that is created is independent of the previous path of the robot, hence

\[
p\left(M^A, t^{A\rightarrow B}, M^B\right) = p\left(M^A, t^{A\rightarrow B}\right) p\left(M^B\right).
\]

Furthermore, tuple \((M^A, t^{A\rightarrow B})\) is actually a single variable. Since the map is conditioned on the path of the particle, there exists a one to one non-random relationship

Figure 4.1: The overlap between the sensor range and the area of the local map is used to decide when to start a new map or transfer to a neighbouring region.

left), then this approach actually maximises the number of re-observed landmarks. In the real world landmarks are not distributed uniformly, and the detection rates for different views of a landmark are different. This approach nevertheless provides a good, computationally inexpensive measure for determining map transitions.

A sensor view based approach also makes the decision to start a new map trivial. If no region overlaps sufficiently with the sensor field of view, the robot assumes it is entering an unexplored area and initiates a new map. In order to avoid oscillations between two regions when the robot is near the boundary, thresholds are set to slightly favour staying in the current region.
between the transition and map $A$, that is $t^{A\rightarrow B} = f(M^A)$. Therefore

$$p(M^A, t^{A\rightarrow B}, M^B) = p(M^A, M^B).$$

At the time when $M^B$ is started $p(M^A, M^B)$ is assumed to be a discrete uniform distribution - a pair of map instances from $M^A$ and $M^B$ is considered to be equally likely. This is due to the fact that $M^B$ is empty, and so any sample from $M^B$ is equally “compatible” with any sample from $M^A$. As $M^B$ is explored, more information, that can be used to evaluate the relative compatibility of the $(M^A, M^B)$ sample becomes available. If, for example, a set of landmarks appears in both $M^A$ and $M^B$, and their correspondence is known, one can evaluate the likelihood of the map pair using covariance intersection [51] of the common landmarks. However there is no need to continuously update the transition distribution; it is assumed to be unchanged until it is time to sample from it. The only information that needs to be stored at the time of creating a transition is a mapping $M^A \Rightarrow t^{A\rightarrow B}$.

The process of creating new map is as following:

1. A new node is added to the graph, and is connected to the current map.
2. The current reference frame is set to that of the new node.
3. The pose of all particles is set to some initial value (0), their maps are set to empty, and the mapping process is restarted.

As the robot continues mapping the new region, any error in the map and robot pose estimate remains conditionally independent of errors in any previous maps. This independence remains until the region is revisited.

### 4.5 Switching Maps - Revisiting

When the robot detects that there is a significant overlap between one of the neighbouring regions and the sensor field of view, it switches to the corresponding local map. Typically, the neighbouring region will be adjacent to the current region, however two regions that are separated by several links in the graph can also be considered neighbours if the compound coordinate transformation from one to the other is sufficiently certain\(^1\). In the case when the regions are not directly connected, the transition is

\(^1\)Loop closing is discussed in the next chapter
performed along the shortest path between the two regions and is equivalent to performing several transitions sequentially. The procedure for computing the shortest path between two local maps is discussed in the chapter on loop closing (Chapter 5).

Consider the transition under the assumption that the two regions are adjacent. Let $A$ be the region mapped first, and $B$ the region created subsequently. As described previously (Chapter 3), each transition maintains joint probabilities that capture the three-way relationship of the two maps joined by the transition and their relative pose. It is therefore possible to sample a map-transition-map triple $\{\mathcal{M}^A, t^{A\rightarrow B}, \mathcal{M}^B\} \sim p(\mathcal{M}^A, t^{A\rightarrow B}, \mathcal{M}^B)$.

It is best to describe the process of transition from one map to another with an example, referring to Figure 4.2. In this example the robot has a car-like driving system and lives on a plane. Three particles are used to approximate the robot’s motion. The robot starts in region $A$ and travels some distance forward, taking observations. Some time later the robot reaches the boundary of region $A$ and starts a new map $B$. Again three particles are used to model the uncertainty of the robot’s path in map $B$. After performing a three point turn the robot returns to region $A$. At this point there are 9 possible paths the robot could have taken. These paths are illustrated in Figure 4.2(c).

By starting a new map $B$, HTSLAM effectively conserves the uncertainty at that time instant. Now, when the robot is returning to the region $A$ there are more possible paths to consider. More hypotheses are beneficial. The number of hypotheses is akin
to resolution in images - more hypotheses provide a clearer picture of the observed uncertainty. HTSLAM uses only 3 particles for mapping but has 9 possible paths. A global mapping approach would have had to use 9 particles (that is $N^2$) to get the same “resolution” on the uncertainty. While having higher resolution is desirable, computational resources are limited. Therefore in this example only 3 paths have to be sampled.

One can assume that all paths are equally likely and sample appropriately. This approach is, however, wasteful. If the two maps share common landmarks, it should be possible to evaluate each path based on some map overlap criteria that approximate $p(M^A, M^B)$.

Alternatively one can use some form of adaptive particle filter [24] that adjusts the number of particles based on the spatial uncertainty in the robot’s pose. In the case of a transition from one map to another, the uncertainty of the robot’s location increases and hence the number of particles would be temporarily increased. After some observations of map A, bad paths would be pruned out through the process of re-sampling.

Once the robot transfers from B to A the two maps are no longer independent. Any further update to map A will depend on the pose of the robot in map A. Robot pose in map A depends on the transition and on robot pose in map B at the time of transition, which in turn depends on map B. All three variables, map A, map B and the transition between them are therefore dependent variables. This dependency is represented in the HTSLAM structure by a sample from the $p(M^A, t^{A\rightarrow B}, M^B)$.

In practice this means that each particle now knows which map to update in A and in B, and which transition to use when transferring from one to the other. From that point onwards no sampling will be required when transferring from A to B and back. If the robot continues operating only within these two maps, the mapping process will be equivalent to a normal FastSLAM, except that the environment is split into regions of bounded size, and hence computation requirements are bounded.

Due to the process of re-sampling, the overall uncertainty in the transition between maps A and B will decrease over time, as more observations become available. This is due to the fact that particles that ended up with “bad” transition hypotheses won’t match observations to the map as well as the ones with “good” transition hypotheses, and will be pruned out during the re-sampling stages of the particle filter. While this
is generally a good thing, there is a drawback: the particle deprivation problem. Since there is no process in place that will introduce new particles to the transition, over time the transition estimate will collapse to just a few or even one particle. This will not necessarily cause major problems, since by the time that happens the relative poses of the local maps are already known and the topological structure has been initialised.

If the robot starts a new region $C$ and then returns to region $A$, a similar process of sampling will occur. This time, however, every map in $A$ is linked to a map in $B$ and to the transition between regions $A$ and $B$. 
Multiple Hypothesis and Loop Closing

In this chapter an approach to loop closing is presented. Section 5.1 describes how loops in the environment are detected. Map matching is described in section 5.2. The update to the global structure needed after loop closing is presented in section 5.3. Then section 5.4 describes how HTSLAM deals with multiple hypothesis arising from the loop closing.

5.1 Loop Detection

When mapping new terrain a robot periodically checks whether it has returned to a region that was mapped previously. Since the HTSLAM structure does not have a global reference frame, a robot pose is only defined for the region currently being mapped. It is however possible to estimate a relative alignment of any two given regions. Using this information, the robot pose can be transformed from one reference frame to another. The exact mechanism for doing that is described in Section 5.1.1.

There are three levels of tests for loop detection. Higher level tests are only considered if all the previous tests have passed.

- Robot pose: is the robot likely to be within the region?
- Map similarity (map matching): does the other map look like the current map?
- Consistency with current observations: assuming the robot is revisiting a region, are the observations consistent with the map of that region? (I expect to see a coffee machine around the corner: is it there?)
5.1 Loop Detection

Figure 5.1: Breadth First Traversal is used to find a shortest paths (in the number of transitions required) from one map frame to all the other map frames.

5.1.1 Computing Relative Map Poses

Since HTSLAM maps in general can contain cycles, there might be multiple paths between any two regions. To resolve this ambiguity the graph structure of the map is converted into a tree with the root being the current node. This can be achieved using either breadth-first traversal of the graph or with Dijkstra’s algorithm for finding the shortest path in the graph [18].

Breadth First Traversal

One sensible approach is to minimise the number of transitions needed to compute the relative pose of two map frames. As it happens, it is possible to compute the shortest path between one of the nodes of the graph and all the other nodes of the graph by converting a graph into a tree using Breadth-First Traversal (see Figure 5.1). The computational complexity of the BFT approach is $O(|V|+|E|)$, where $|V|$ is the number of nodes (vertices) and $|E|$ is the number of edges in the graph.

Dijkstra’s Shortest Path Algorithm

Dijkstra’s shortest path algorithm computes a path between two nodes of the graph that minimises some metric. In the case of HTSLAM, it is desirable to find a path between two regions, that has a most certain compound transition. To implement this approach one requires an accurate estimate of the uncertainty of the compound transition. One possible solution is to approximate the error in each transition with a Gaussian. This way compound transitions can be computed recursively. Dijkstra algorithm can then be modified to minimise the determinant of the covariance of the


5.2 Map Matching

Given two local maps, a method is required to assess the probability that they correspond to the same region in the environment and find their relative alignment. The choice of map matching algorithm depends on the type of local mapping module. This thesis deals with feature based maps (other types include: occupancy grids, dense point clouds and laser scans). In the case of feature based maps, the map correspondence problem is split into a number of landmark correspondence problems. To solve the map correspondence problem, one needs to find the most likely set of common landmarks between the two local maps. It is a requirement of HTSLAM that the correspondence problem can be solved without accurate knowledge of the relative map alignment. The nearest neighbour gating technique, commonly used for data association problems, will not work in this situation since the uncertainty in the relative alignment of the two maps is large. There are a number of batch data association techniques. One of them is the joint probability branch and bound (JPBB) [72] method, which is more robust to pose uncertainty, than nearest neighbour, however it still assumes that the rough alignment is known. JPBB performs a search in a correspondence decision tree for a branch with the best match.

Another batch data association technique is the graph theoretic approach [84]. Essentially this algorithm performs a search for a clique (the largest fully connected sub-graph of a graph) in a sparse correspondence graph. The correspondence graph
is constructed from the geometric feature matches (distances between landmarks, in the case of point landmarks). If individual landmarks can be differentiated in some way (for example a door landmark cannot match a chair landmark, or a pink door cannot match a green door), such a constraint is referred to as an absolute constraint. Absolute constraints are also taken into account when constructing the correspondence graph. Unlike JPBB, the graph based approach can work without prior knowledge of the relative alignments of the two sets being matched: however if a prior is available it is used.

Both of the batch methods mentioned above were developed for data association in EKF SLAM. As a result they assume that a set of uncorrelated observations is matched to a set of correlated landmarks. However they can be easily extended to match two sets of landmarks instead. In HTSLAM landmarks are uncorrelated, and so some computation can be simplified significantly.

The approach used in this thesis is similar to the graph approach. Like the graph approach it looks for a maximum clique of the correspondence graph, but without constructing the graph explicitly. This approach computes joint compatibility in a slightly different way. Geometric and absolute constraints are used as in the graph approach, but in addition this approach also exploits constraints arising from the alignment of the two sets. Generally, geometric constraints alone are not sufficient to disregard all impossible sets of correspondences. Any two sets of landmarks which are mirror images of one another will satisfy the geometric constraints, but the landmarks in these sets do not correspond (see Figure 5.2).

![Figure 5.2](image-url)

**Figure 5.2:** Two incompatible landmark sets, that do satisfy geometric constraints. One set is a mirror image of another, the distances between landmarks match, but the sets cannot be aligned by a Euclidean transform.

The most direct way to evaluate a correspondence hypothesis is to perform a two-view reconstruction of the scene, and then compute the likelihood of the reconstruction.
Let $\theta_i$ be a landmark pose in the reference frame of map $a$, let $\theta_i^a, \theta_i^b$ be an estimate of the landmark $i$ in the reference frames $a$ and $b$ respectively. Let $R, t$ be a rotation and translation from frame $a$ to frame $b$. Assume also that a prior $p(R, t)$ is available. The reconstruction process needs to find $[\theta_1, ..., \theta_n, R, t]$ that maximises

$$p(R, t) \prod_{i} p(\theta_i | \theta_i^a) p(R\theta_i + t | \theta_i^b).$$

(5.1)

The maximum of (5.1) can be used as a measure of the likelihood of a particular landmark correspondence configuration. While there is no closed form solution to this problem, an iterative numerical optimisation can be used to find the maximum of (5.1). This process is outlined in Section 5.2.2. Such an operation is computationally expensive and cannot be used to search for the most likely configuration. However it can be applied to compare the best few hypotheses.

Preprocessing step: let $E_{i,k}^a$ be the edge between landmarks $i$ and $k$ in the map $a$. For every landmark correspondence compute an upper bound on the size of the correspondence set it can belong to

$$u_{ij} = \sum_{k \neq i} \sum_{l \neq j} \text{MATCH}(E_{i,k}^a, E_{j,l}^b).$$

Here function MATCH returns 1 if edges match and zero otherwise. Edges match if the landmarks on either side of the edge satisfy absolute constraints AND their lengths are sufficiently similar AND the alignment arising from the edge correspondences is consistent with the prior. When implemented naively the computational complexity of the preprocessing step is $O(N_eM_e)$, where $N_e, M_e$ are number of edges in the two sets. However, if one first sorts the edges by their lengths, one does not need to match each edge to every other. The complexity in that case will be $O(N_e \log N_e + M_e \log M_e + N_eK)$ where $K \leq M_e$ but in practice $K \ll M_e$ typically, and depends on the search window width and the distribution of edge lengths.

Let $S$ be a set of landmark correspondence hypotheses. Each element of $S$ is a triple $(u_{ij}, i, j)$, where $u_{ij}, i, j$ mean the same as above. Set $S$ is sorted in descending order according to $u_{ij}$. Let $M$ be a set of jointly compatible correspondences. $P_{ab}$ is a prior on relative alignment of the two sets being matched. MATCH is a lookup table: inputs are edges from map $a$ and map $b$, and the output is a boolean indicating whether the two edges match. $M_{\text{max}}$ is a set of all current best matches. Matches with
equal number of common landmarks are considered equally likely. $N_{\text{max}}$ is a number
of common landmarks in the current best match or matches. $T^a, T^b$ are boolean arrays
indicating which landmarks from $a$ and $b$ respectively, have been matched.

The proposed matching algorithm is recursive. The level of recursion is limited by
the number of landmarks in the smaller of the two maps. The algorithm is presented
in pseudo-code.

Algorithm 5.2.1: MATCHMAPS($L_a, L_b, P_{ab}$)

\begin{itemize}
\item \textbf{comment}: $L_a, L_b$ set of landmarks to be matched
\item \textbf{comment}: $P_{ab}$ prior on relative alignment
\item global $E^a, E^b, \text{MATCH}, S$
\item global $M_{\text{max}}, N_{\text{max}}, T_a, T_b$
\end{itemize}

\begin{align*}
E^a &= \text{COMPUTEEDGES}(L_a) \\
E^b &= \text{COMPUTEEDGES}(L_b) \\
(S, \text{MATCH}) &= \text{MATCHEDGES}(E^a, E^b, P_{ab}) \\
N_{\text{max}} &= 3 \quad \text{comment: At least three landmarks have to match.} \\
M_{\text{max}} &= \{\} \\
T_a &= [\text{false}, \text{false}, \cdots, \text{false}] \\
T_b &= [\text{false}, \text{false}, \cdots, \text{false}] \\
\text{RecursiveSearch}(S, \{\}, P_{ab}) \\
\text{return } (M_{\text{max}})
\end{align*}
Algorithm 5.2.2: \texttt{RECURSIVESEARCH}(S, M, P_0)

\texttt{global} \ M_{\text{max}}, N_{\text{max}}, T_a, T_b

\texttt{if} \ size(M) > N_{\text{max}} \ \texttt{then} \ \begin{cases} M_{\text{max}} = \{M\} \\ N_{\text{max}} = size(M) \end{cases}
\texttt{else if} \ size(M) = N_{\text{max}} \ {\{M \rightarrow M_{\text{max}}\}}

\texttt{while} \ size(S) + size(M) \geq N_{\text{max}} \text{ and } S \neq \{} 
\begin{cases} (u_{ij}, i, j) \leftarrow S \\ \text{if} \ u_{ij} < N_{\text{max}} \ \text{then} \ \{\text{return}\} \\ \text{if} \ \texttt{ISCOMPATIBLE}(i, j, M, P) \\ \begin{cases} (i, j) \rightarrow M \\ T_a[i] = T_b[j] = \text{true} \\ P = \texttt{UPDATEALIGNMENT}(M, P_0) \\ \texttt{RECURSIVESEARCH}(S, M, P) \\ (i, j) \leftarrow M \\ T_a[i] = T_b[j] = \text{false} \end{cases} \end{cases}

Algorithm 5.2.3: \texttt{ISCOMPATIBLE}(i, j, M, P)

\texttt{global} \ T_a, T_b, \texttt{MATCH}, E_a, E_b

r = \neg (T_a[i] \text{ or } T_b[j])
\texttt{while} \ r \text{ and } M \neq \{} 
\begin{cases} (k, l) \leftarrow M \\ \text{if} \ \texttt{MATCH} \left[ E_{i,k}^a, E_{j,l}^b \right] = \text{true} \\ \texttt{then} \ \begin{cases} A = \texttt{ROTATIONFROMEDGES}(E_{i,k}^a, E_{j,l}^b) \\ r = \texttt{COMPATIBLEROTATION}(A, P) \end{cases} \texttt{else} \ \{r = \text{false}\} \end{cases}
\texttt{return} \ (r)
Distance Between Landmarks

The distance between two landmarks is not known exactly, and the uncertainty is a non-Gaussian distribution. It is approximated by a Gaussian using first order Taylor series expansion. An alternative is to use an unscented transform [98]. The unscented transform provides a more accurate estimate, with similar computation requirements. In this work however, Taylor series approximation was used.

The mean and variance of the distance is computed according to the following approximation:

\[
\begin{align*}
\mu_d & = f([x_1, y_1, x_2, y_2]^T) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \\
\sigma_d^2 & = \nabla f \left[ \begin{array}{cc} \Sigma_1 & 0 \\
0 & \Sigma_2 \end{array} \right] \nabla f^T.
\end{align*}
\]

(5.2)
(5.3)

Here \( \nabla f \) is a Jacobian of \( f \), evaluated at \([x_1, y_1, x_2, y_2]^T\). \( \Sigma_1 \) and \( \Sigma_2 \) are the two landmark covariances. Equations (5.2) and (5.3) can be simplified by introducing variables \( \Delta x = x_1 - x_2 \), and \( \Delta y = y_1 - y_2 \), the uncertainty of \( \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} \) is equal to a sum of uncertainties of the two landmarks \( \Sigma_\Delta = \Sigma_1 + \Sigma_2 \):

\[
\begin{align*}
\mu_d & = \sqrt{\Delta x^2 + \Delta y^2} \\
\sigma_d^2 & = \nabla f \Sigma_\Delta \nabla f^T
\end{align*}
\]

(5.4)
(5.5)

where

\[
\nabla f = \left[ \frac{\Delta x}{\sqrt{\Delta x^2 + \Delta y^2}}, \frac{\Delta y}{\sqrt{\Delta x^2 + \Delta y^2}} \right].
\]

The distance between landmarks is not the only transform invariant feature. It might be possible to differentiate individual landmarks to some degree. In the case of corner landmarks, for example, a convex corner should not match a concave corner. Landmark type is therefore used as a binary measure to disregard impossible landmark correspondences. In the case of vision landmarks, cross-correlation of the landmark templates can be used to compute the likelihood of the two landmarks corresponding.

Using landmark appearance or type is beneficial to the task of map matching, as it reduces computation time and increases the likelihood of finding correct correspondences. In the situation when landmarks do not have any attributes that can be used
to differentiate them, any landmark correspondence is considered to be possible and equally likely. The proposed algorithm can work without this extra information, but will use it if it is available.

5.2.1 Map Alignment from 2 Landmark Correspondences

Generally when closing the loop there is a priori knowledge of the relative alignment of the two maps. No matter how uncertain this knowledge is, it is still beneficial to use it to speed up the matching process by disregarding landmark correspondences that are in disagreement with the prior. In the case of a point landmark, a single landmark to landmark correspondence does not have a rigid constraint on the relative map pose, however a pair of point landmark correspondences is sufficient to compute the relative pose of the two maps. Since the landmark locations are not known exactly, the relative pose will also have some uncertainty. This uncertainty, however, is likely to be significantly less than that of a prior, and is therefore used to assess the possibility of the correspondence being correct.

Assume $\theta_1^a$ and $\theta_2^a$ are true poses of the two landmarks expressed in the reference frame of the map $a$. Assume $R, t$ are true rotation and translation from frame $a$ to frame $b$, such that $R\theta_1^a + t$ is a landmark pose expressed in frame $b$. $\theta_1^a, \theta_2^a$ are estimates of the first landmark in frames $a$ and $b$ respectively, and $\theta_1^b, \theta_2^b$ are observations of the second. Ideally, relative map pose and the locations of landmarks have to be computed simultaneously to find the configuration that maximises

$$p(\theta_1^a, \theta_2^a, R, t|\theta_1^b, \theta_2^b, \theta_1^a, \theta_2^a) = p(\theta_1^a|\theta_1^a)p(R|\theta_1^a + t|\theta_1^b)p(\theta_2^a|\theta_2^a)p(R|\theta_2^a + t|\theta_2^b).$$

In this thesis an approximation is used instead. Rotation from frame $a$ to frame $b$ is approximated to be

$$\Delta \phi = \theta_1^a - \theta_1^b,$$

where $[\Delta x^a, \Delta y^a]^T = \theta_1^a - \theta_1^b$ and $[\Delta x^b, \Delta y^b]^T = \theta_2^b - \theta_2^b$. The uncertainty of the rotation is approximated using first order Taylor series expansion. Alternatively one can use unscented transform [98].

For a fixed rotation, translation between two sets of landmarks is exactly Gaussian, and can be computed analytically. The exact process is outlined in Section 5.2.2 on map alignment.
5.2.2 Final Map Alignment

After landmark correspondence is established HTSLAM requires an estimate of the relative alignment of the two maps in a probabilistic manner in order to model the coordinate transformation between the two maps. This section describes an approach to map alignment used in the experiments presented in chapters 7 and 8.

HTSLAM requires a sample from the following distribution

\[ p(R, t) = \prod_{l=1}^{n} \int p(\theta_l^a)p(\theta_l^b|R, t)d\theta \]  

Taking logs of both sides

\[ \log p(R, t) = \sum_{l=1}^{n} \log \int p(\theta_l^a)p(\theta_l^b|R, t)d\theta. \]  

Let \( \theta_i \) be a true location of the landmark \( i \) expressed in the coordinate frame of map \( a \). Let \( R, t \) be a true rotation and translation from frame \( a \) to \( b \), such that \( R\theta_i + t \) is a true location of the landmark \( i \) in the reference frame of \( b \). Let \( M^A = [(a_1, A_1), ...(a_n, A_n)] \), \( M^B = [(b_1, B_1), ...(b_n, B_n)] \) be a set of common landmarks in the two maps. The mean of the landmark \( i \) is denoted by \( a_i \) and \( A_i \) is the inverse of the corresponding covariance. It is required to find \( R, t, \theta_1, \theta_2...\theta_n \) that maximise

\[ p(\theta_1, \ldots, \theta_n, R, t|M^A, M^B) = \eta \prod \exp(-(\theta_i - a_i)^T A_i (\theta_i - a_i)/2) \exp(-(R\theta_i + t - b_i)^T B_i (R\theta_i + t - b_i)/2), \]  

where \( \eta \) is a normalisation constant, independent of \( R, t, \theta_1, \theta_2...\theta_n \). Taking logs:

\[ \log p(\theta_1, \ldots, \theta_n, R, t|M^A, M^B) = \log \eta - 0.5 \left[ \sum_i (\theta_i - a_i)^T A_i (\theta_i - a_i) + \sum_i (R\theta_i + t - b_i)^T B_i (R\theta_i + t - b_i) \right]. \]  

The term that needs to be minimised is

\[ F = \left[ \sum_i (\theta_i - a_i)^T A_i (\theta_i - a_i) + \sum_i (R\theta_i + t - b_i)^T B_i (R\theta_i + t - b_i) \right] \]

\( F \) can be rewritten in matrix form

\[ F = x^T G x - 2H x + c \]
where $x^T = [\theta_1^T, ..., \theta_n^T, t^T]$ and matrices $G, H$ and scalar $c$ are:

$$G = \begin{bmatrix}
A_1 + R^T B_1 R & 0 & \cdots & 0 & (B_1 R)^T \\
0 & A_2 + R^T B_2 R & & (B_2 R)^T \\
\vdots & \ddots & \ddots & \vdots \\
0 & \cdots & 0 & A_n + R^T B_n R & (B_n R)^T \\
B_1 R & B_2 R & \cdots & B_n R & \sum_i B_i
\end{bmatrix}$$

$$H = \left[a_1^T A_1 + b_1^T B_1 R, \ldots, a_n^T A_n + b_n^T B_n R, \sum_i b_i^T B_i \right].$$

$$c = \sum_i a_i^T A_i a_i + \sum_i b_i^T B_i b_i$$

Note that matrices $A_i, B_i$ are symmetric and hence $G$ is a symmetric matrix as well.

For a fixed rotation, a configuration that minimises $F$ is found by solving

$$2x^T G - 2H = 0.$$ 

Which is equivalent ($G$ is symmetric, $G = G^T$) to solving

$$Gx = H^T.$$ 

Due to the structure of matrix $G$, $x$ can be solved for by back-substitution. First, define

$$M_i = A_i + R^T B_i R$$

$$Y_i = B_i R$$

$$S = \sum_i B_i$$

$$K_i = \left[a_i^T A_i + b_i^T B_i R \right]^T$$

$$W = \left[\sum_i b_i^T B_i \right]^T$$

The problem can then be rewritten as a series of linear equations

$$M_1 \theta_1 + Y_1^T t = K_1$$

$$M_2 \theta_2 + Y_2^T t = K_2$$

$$\vdots$$

$$M_n \theta_n + Y_n^T t = K_n$$

$$\sum_i Y_i \theta_i + St = W.$$
Expressing $\theta_i$ in terms of $t$, substituting it in to (5.15), after some rearranging

$$\theta_i = M_i^{-1}K_i - M_i^{-1}Y_i^T t$$  \hspace{1cm} (5.16)

$$t = \left( S - \sum_{i}^{n} Y_i M_i^{-1} Y_i^T \right)^{-1} \left( W - \sum_{i}^{n} Y_i M_i^{-1} K_i \right).$$  \hspace{1cm} (5.17)

Having solved for $t$, it is possible to back-substitute to solve for every landmark location as well. Denoting the solution as $\mu, H = \mu^T G$. $F$ can then be rewritten as

$$F = x^T G x - 2\mu^T G x + c$$

$$= (x - \mu)^T G (x - \mu) - \mu^T G \mu + c.$$  \hspace{1cm} (5.18)

For a fixed rotation $R$

$$\log p \left( [\theta_1, ..., \theta_n, t]^T | \mathcal{M}^A, \mathcal{M}^B, R \right) = \log \eta - 0.5 (x - \mu)^T G (x - \mu) + 0.5\mu^T G \mu - 0.5c.$$

From this it follows that under a fixed rotation the error in translation and landmark locations is a scaled Gaussian

$$p \left( [\theta_1, ..., \theta_n, t]^T | \mathcal{M}^A, \mathcal{M}^B, R \right) = \alpha N(\mu, G^{-1})$$

The rotation that maximises $\mu^T G \mu$ is the most likely rotation and can be found numerically, for example using Golden Ratio search for minima algorithm [79].

Rather than using one optimal value to describe the relative poses of the two maps, HTSLAM needs a sample of the poses instead. A direct sample of $(R, t)$ pairs from the distribution given in (5.9) is not possible. Since it is possible to find rotation independent of translation, it is possible to sample rotation first, and then sample translation condition on a given rotation. For a given $R$, $t$ is distributed normally $N(\mu, G^{-1})$.

### 5.3 Augmenting Global Map Structure

Assume for the moment a simple scenario: a robot starts in region $A$, explores new regions $B, C, D, E$, in that order, then discovers that region $A$ and $E$ overlap.

From landmark correspondences it is possible to infer the relative pose of the two maps, allowing for an update of the global map structure. Map alignment is used to
compute a new link between the two maps. The robot then transfers into map $A$ and continues mapping. However, this new information also provides information about the path of the robot along the loop. Before map matching all paths are equally likely (by construction), subsequently the robot re-localises itself in map $A$.

If there are no loops in the HTSLAM structure, transitions are independent variables. This follows from the process by which transitions are constructed. However, when the loop is closed one can no longer think of transitions as independent. New evidence provides the means to evaluate the joint probability of all transitions along the path, and this joint probability is generally not equal to the product of individual probabilities, invalidating the independence hypothesis.

The big question is whether the fact of loop closing provides information about the path along the loop. Is there enough information to evaluate the joint probability of the transitions along the path? Model approximations introduce errors that accumulate and introduce biases in the robot pose estimate.

![Figure 5.3: Robot starts at point $A$, and travels on a loop $ABCDA$. Since the segment $BC$ passes over the hill, its length is longer than that of $DA$. As a result, the flattened path does not close the loop. By the time robot returns to point $A$, it estimates its position to be at $A'$.](image)

The effects of model approximation errors can be significant, as Figure 5.3 illustrates. In this example the robot assumes a planar surface; while this assumption might work well enough for local observations, it breaks globally. The more accurate the robot’s sensors are, the more obvious model errors become. If your observations tell you that to get from $A$ to $C$ you either go 100 km north, then 30 km west, OR 30 km west then 110 km north, then maybe both these observations are right, but your assumption of a planar surface is wrong.

If one assumes that the impact of model approximation errors is insignificant along
the path, one can attempt to incorporate all information along the path when closing the loop. This is however problematic due to practical considerations. Given there are $K$ transitions along the loop, and there are $N$ particles per transition, the joint distribution of transitions will be a $K$-dimensional lookup table, and will contain $N^K$ elements. A direct computation of joint distribution is clearly not possible.

Traditional metric mapping techniques make an assumption that it is possible to obtain an exact map of the environment, given an infinite number of observations. They assume that all dimensions of the problem are observable. Quite often this is not the case, as was illustrated in the example above. The limitations of the perception system of an entity can indeed limit the certainty of the world model that can be achieved.

### 5.3.1 Globally Consistent Approach

In this approach every particle has to commit to a path at the time of loop closing. From $N^K$ possible paths $N$ should be chosen before the robot can continue mapping. Ideally, these $N$ paths should be representative of the original distribution - the first and second moments of the sample should be close to that of the true distribution. The higher the value of $K$ the more difficult it is to achieve this goal.

If the loop was long (large $K$), there will be a great variety of likely paths. It might be impossible to represent the full range of paths with just $N$ samples, especially since the original distribution is likely to be multi-modal. Furthermore, for large $K$, it is impractical to generate all the paths, since there is such a large number of them. As a result one would have to resort to some form of approximation in order to obtain a sample. For example a genetic algorithm:

1. Start with a small random sample of paths.
2. Generate new samples by mutating paths (mutation is performed by changing one transition at a time).
3. Sample with replacement “good” paths.
4. Repeat from step 2, until convergence is reached

Such an approach is likely to generate an overly certain sample, which can lead to inconsistencies later on in the mapping process.

---

1The number of particles needed to represent a given distribution is an open research question.
5.3.2 Local Approach

In the case of a local approach to loop closing only local relations between regions are modelled. It is assumed that unobservable unmodelled errors prevent accurate estimation of alignments between regions far apart.

In order to create a link between the maps a sample from the following distribution is needed

$$p \left( M^A, t^{A-E}, M^E \right).$$

This is however quite problematic. There are $N^2$ possible map pairs and the relative pose of each pair is not directly available. The relative pose can be computed through the process of map alignment, as described earlier. While it is theoretically possible to sample relative poses for every map pair, in practice $O(N^2)$ computation will take too much time.

A simplification used in this work is to assume that the relative alignment is independent of the maps. It is therefore possible to estimate the alignment using just one of the pairs, and then sample map-transition-map triples, based on the independence assumption. An approach like this works well when maps are relatively similar. When maps between particles differ significantly, samples generated this way will contain a large proportion of very poor matches. A particle filter will eventually get rid of the poor matches, however more particles will be needed to represent the uncertainty accurately.

5.4 Confirming Loop Closing Hypothesis

No matter how accurate place recognition capabilities might be, false positives are inevitable due to the self-similarity of the environment. One should therefore not rely on place recognition alone to perform loop closing. Some mechanism should be in place to evaluate the likelihood of the hypothesis that the two local maps are the two views of the same region.

In HTSLAM such a mechanism is built into the local mapping module. Several mapping hypotheses can be compared directly in a particle filtering framework. FastSLAM samples over robot paths in one global reference frame. HTSLAM extends particle filtering to sample over robot paths and different reference frames. If observations can be better explained by a map in region $A$ than by a map in region $B$, particles
in region \( B \) will gradually die out.

Generally HTSLAM can run multiple hypotheses at any time. For clarity of explanation assume that there is only one hypothesis running at the time the loop is detected. Let the robot be exploring region \( E \). At this time every particle in the filter has the same global map structure. The differences are in the path through the region \( E \) and the local maps of the region. Assume a correspondence with region \( A \) is detected. At this time the current hypothesis is “cloned”. The particle set is duplicated. In one of the sets the global map is updated with a link between regions \( A \) and \( E \), and the particles are transferred into region \( E \). The other set remains as it was. At this time there are two types of particles: ones with the closed loop and ones without. To the particle filter they appear as one uniform set. The particle filter continues to operate with twice the number of particles until one of the hypotheses dies out, at which time the set is reduced.

There is no theoretical limit on the number of hypotheses within an HTSLAM framework, however there are some practical constraints. The number of hypotheses should be limited in order to bound the computational effort. Measures should be in place to prevent creating “essentially equivalent” hypotheses, as they would run in parallel wasting computational resources with only limited benefit. An example of “essentially equivalent” hypothesis would be closing the loop a second time while the first hypothesis is still running. In this case local maps of the two hypotheses would be slightly different, but the global structure would be the same. On the positive side, a situation like that does not invalidate the mapping approach, it is merely a practical nuisance that wastes resources.
Summary and Comparison

This chapter provides a brief summary of the HTSLAM approach. A comparison with other existing mapping techniques is given.

6.1 Summary of HTSLAM

HTSLAM is a novel mapping technique that addresses some of the limitations of traditional mapping approaches. HTSLAM decomposes the environment into a number of interconnected regions. Each region is mapped separately. There is no global reference frame, only metric relationships between neighbouring regions are stored in the HTSLAM map. Local mapping is performed using a particle filter based approach called FastSLAM [70, 71]. The particle filter allows HTSLAM to deal with mapping ambiguities in a rigorous probabilistic framework.

In a general case of unknown data associations (non-unique landmarks), probability density over robot pose is often multi-modal, due to ambiguities arising in the data association decisions. Particle filters deal with multi-modal distributions natively. HTSLAM benefits from this ability of particle filters. This advantage is especially apparent when closing large cycles. When closing the loop HTSLAM runs multiple mapping hypotheses in parallel. Pruning of unlikely hypothesis happens automatically as part of a resampling step of the particle filter.

6.2 Comparison

6.2.1 Global Mapping Approaches

Global mapping approaches include EKF SLAM and FastSLAM. These methods and their limitations were described in detail in Chapter 1.

The main advantage of HTSLAM over global methods is scalability. HTSLAM
performance does not degrade as the robot moves further away from the origin; the quality of the local maps is a function of the environment and the sensors, and is independent of the distance from the origin. Assuming a sensible exploration strategy that avoids unnecessary long loops, or venturing into feature-less spaces, the area that can be covered is only limited by the robot storage capacity. A smart implementation that swaps out unused maps from memory to disk can indeed cover large areas with this approach.

Since HTSLAM only updates a small local region (or sometimes several small local regions) at any given time, the computational complexity is bounded.

HTSLAM provides a robust method for loop closing. In HTSLAM loop closing can be postponed until enough information supporting it is available. In fact it is possible to post-pone loop detection and closing until mapping is complete and update the map at a later time. In contrast, EKF and FastSLAM have to rely on observation to map correspondence alone when closing the loop.

### 6.2.2 Atlas

HTSLAM and Atlas share a similar hybrid map structure. Both approaches do without a global reference frame and instead rely on local metric information only. There are also significant differences between the two approaches

- Choice of the local mapping module.
- Map transition process.
- Loop closing procedure.
- Representation of uncertainty in coordinate transformations between map frames.

*Atlas* uses the EKF as its local mapping module\(^1\), while HTSLAM uses FastSLAM. FastSLAM has a number of advantages over EKF. FastSLAM is computationally more efficient and also provides a richer representation of uncertainty as it can model multi-modal distributions, while the EKF is restricted to a single mode. For a local mapping module computational complexity is not as important, as map size is bounded. The EKFs inability to handle multi-modal distributions makes it unusable in some environments, even if the size of the map is limited. In high clutter environments data

\(^1\)The authors of *Atlas* do mention the possibility of using mapping modules other than EKF, however as of time of the writing no such implementations have been reported.
association errors are more likely. Incorrect data association is known to lead to inconsistent maps when using EKF. FastSLAM can deal with data association ambiguity by sampling over all plausible data association decisions [70, 69, 74].

HTSLAM maintains map extent information for every local map. This extra data allows HTSLAM to perform map transitions in a simple way. In HTSLAM the estimate of the robot pose in a current reference frame is all that is needed to make a decision on whether to stay in the current frame, perform a transition into a neighbouring region or to start a new map. In contrast, Atlas runs multiple localisation hypotheses in all neighbouring regions in order to determine which region provides a better explanation to current observations. Atlas uses some quality metric to judge the fitness of each alternative. Generally, when traversing a region that has been mapped previously, the robot is well localised, there is no ambiguity with regards to which local map the robot should be updating. It is therefore unnecessary to run multiple hypotheses in such a situation. HTSLAM only generates multiple hypothesis if there is significant ambiguity in the robot pose.

Both Atlas and HTSLAM use multiple hypotheses to prove or disprove the validity of loop closing. Atlas uses some performance metrics to compare different hypotheses, and to prune out the weak ones. In HTSLAM mapping hypotheses are compared directly with each other within a common particle filter framework. Importance sampling is used to prune out the unlikely hypotheses. There is no need to invent performance metrics. HTSLAM provides a more formal approach to management of uncertainty arising due to loop closing.

Atlas does not maintain correlations between local maps, nor does it keep correlations between maps and transitions. By construction the transition is a dependent variable of a local map. In HTSLAM this dependence is captured in a sample.
Chapter 7
Experimental Results: Laser Scanner

7.1 Feature Detectors

7.1.1 Corner Detector

Corners are very common features in the indoor environment. They can be easily detected using a laser range finder. Corners are generally sparse, which makes the data association process more reliable. A single observation of a corner is sufficient to completely define the pose of the sensor. Using corners as features allows algorithms to be tested on a clean simulation-like data, while still being realistic.

The corner detector algorithm distinguishes between three possible types of corners: convex, concave and "jump". A "jump" corner is defined as two parallel lines spaced a small distance apart. For example doors in a hallway are classified by the algorithm as jumps. It is possible to detect a convex corner even when only one side of it is visible. Such an observation is referred to as "convex hidden" corner. Figure 7.1 shows the different types of corners.

Figure 7.1: Corner types, from the left to the right: concave, convex, jump
Algorithm

The corner detector algorithm consists of two stages. During the first stage the algorithm looks for regions in the laser scan that might correspond to corners. During the second stage each corner hypotheses is analysed more closely. Hypotheses that do not pass the tests are discarded. Those that remain are classified to be either convex, concave, jump or convex hidden corners.

![Image](image.png)

**Figure 7.2:** Trying to fit a line around the corner.

During the first stage the algorithm looks at a narrow portion of the laser scan (7 readings wide, that is only 3 degrees) and computes the probability that this portion of the scan has been reflected from a planar surface. This is achieved by fitting a line to the portion of the laser scan and computing the average error. High error implies low likelihood of the planar surface within the region. The algorithm starts at the right most region of the laser scan and moves counter-clockwise at one laser reading increment.

As the window slides along the planar region of the scan the error remains low. However when the region includes corners and other irregularities, the error increases significantly, and hence the line quality drops, refer to Figure 7.2. Regions where the line quality falls from high to low are suspects for possible corners, see Figure 7.3(b).

Some laser rays fail to reflect back to the sensor, either because there is no obstacle in the sensor range, or due to the reflective properties of the surface. If there are any “no return” rays within the window, the quality of the line for this window is set to the lowest. One can see in Figure 7.3(c), the effect it has. While most hypothesis actually correspond to true corners in the environment, there maybe false hypothesis arising from sensor range limitations. However, the second stage of the algorithm successfully removes any false hypothesis. Figure 7.3(d) demonstrates this point.

The second stage of the corner detection algorithm classifies potential corner observations as one of the: convex, concave, jump or convex hidden corner, and discards those that do not fit either of the models. Two intersecting lines to the left and to
(a) Scan is taken from right to left, counter-clockwise.
(b) Negated total error squared is used as a measure of the line quality. Horizontal line indicates good line threshold. Red lines mark suspected corners.
(c) Falls in line quality are suspect for corners. Circle marks the beginning of the suspect region, square marks the end.
(d) Further tests confirm good corners and discard bad ones. Corners are classified into several types.

**Figure 7.3:** Corner extraction from the laser scan.
the right of the suspected corner correspond to either convex or concave corner. Two closely spaced collinear lines to the left and right of the corner correspond to the "jump" corner. One line intersected by the laser ray corresponds to the "convex hidden" corner.

**Uncertainty Representation**

A corner is considered to be a point landmark with additional attributes. The additional attributes include the type of the corner and the orientations of the corner arms. Overall there are four metric parameters that define the corner: two parameters for the location and two for the orientation of the corner arms. While it is possible to compute the uncertainty directly from the laser readings, it is not very obvious how to propagate the uncertainty of individual laser beams through a line fitting process. In this work errors in the estimation of the corner position and the orientation of the corner arms are assumed to be independent zero mean Gaussians.

Observation of the corner location is expressed in polar coordinates. Range and bearing measurements are assumed to be independent.

Three standard deviations in the range measurement error is set to 5 per cent of the measured distance to the corner plus some constant. Three standard deviations in the bearing measurement error is set to 2 degrees. Error in the estimate of the orientation of each arm is set to 3 degrees for three standard deviations. While this is a very simplistic model of the uncertainty, in practice it performs surprisingly well.

### 7.1.2 Tree Trunk Detector

Locations of tree trunks were extracted from the laser data using the matlab code available from the ACFR website [33].

### 7.2 Mapping

This section describes the implementation of FastSLAM algorithm. It defines the landmark update equations. Odometry models of vehicles used in the experiment are also described.

#### 7.2.1 Point Landmarks

Both trees and corners are treated as point landmarks. The extra parameters, trunk diameter in the case of the tree landmark, and orientation of the arms in the case of the
corner landmark, are assumed to be independent of the landmark position, and hence can be estimated separately. The extra parameters are used for the data association and for the computation of the probability of an observation, given a landmark.

The point landmark is defined by its pose in the local reference frame

\[ x = \begin{bmatrix} x \\ y \end{bmatrix}. \]

The true state of the landmark is not known, the estimate of the landmark state is maintained instead. The estimate of the landmark state at time \( k \) is \( \hat{x}_k \). The covariance for the landmark state is also maintained and is denoted \( P_k \).

The landmark state is related to the observation by the following equation

\[ z = h(x, v) = \begin{bmatrix} \sqrt{(x - x_s)^2 + (y - y_s)^2} + v, \\ \tan^{-1} \frac{y - y_s}{x - x_s} + \theta_s \end{bmatrix}, \]

here \( x_s, y_s \) and \( \theta_s \) denote the location and heading of the sensor in the local reference frame and \( v \) is the observation noise, assumed to be zero mean Gaussian.

The Jacobian of partial derivatives of \( h \) with respect to landmark state is

\[ H_{[i,j]} = \frac{\partial h_{[i]}}{\partial x_{[j]}} (\hat{x}_k, 0). \]

For clarity lets define the estimate of the landmark location in the sensor centric coordinate frame as \( \hat{x}' = \hat{x} - x_s \) and \( \hat{y}' = \hat{y} - y_s \). The Jacobian above is then equal to:

\[ H = \begin{bmatrix} \frac{\hat{x}'}{\sqrt{\hat{x}'^2 + \hat{y}'^2}} & \frac{\hat{y}'}{\sqrt{\hat{x}'^2 + \hat{y}'^2}} \\ \frac{-\hat{y}'}{\hat{x}'^2 + \hat{y}'^2} & \frac{\hat{x}'}{\hat{x}'^2 + \hat{y}'^2} \end{bmatrix}. \]

The Jacobian of partial derivatives of \( h \) with respect to noise is also required by the EKF, in this case it is trivial

\[ V_{[i,j]} = \frac{\partial h_{[i]}}{\partial v_{[j]}} (\hat{x}_k, 0) = I. \]

These Jacobians evaluated at the current state are used to update the state of the landmark when a new observation becomes available.
Observation Update

Every time a landmark is observed, its pose estimate is updated to take into account the new information. This update is performed using the standard Extended Kalman filter equations.

Genesis

When an observation does not match any of the existing landmarks, a new landmark is added to the map. A range and bearing observation is converted into a Cartesian coordinate space. The covariance of the landmark is computed from the covariance of the measurement using first order Taylor series approximation.

7.2.2 XR4000: Odometry Model

The robotic platform used for the indoor experiment is an XR4000 robot from Nomadic Technology. This robot has a holonomic drive system with four wheels. Controllers on-board the robot integrate distance travelled by each of the wheels to compute the location of the robot in some global reference frame. This estimate is subject to drift due to the accumulation of errors in the process of integration. It is difficult to model such a system from the physical principles, instead a simplified model is devised.

The state of the robot at time $k$ is its position and orientation in a local reference frame

$$x_k = \begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix}.$$

Being a holonomic robot, the XR4000 can move in any direction and rotate at the same time, just like an office chair. Control input $u_k$ is defined as an instantaneous translational and rotational velocity of the robot at time $k$. Translational velocity is expressed in polar coordinates $[v_k, \psi_k]^T$ in the robot centred frame. $\omega_k$ is a rate of rotation of the robot centred frame relative to the reference frame

$$u_k = \begin{bmatrix} v_k \\ \psi_k \\ \omega_k \end{bmatrix}.$$
In the case of the XR4000, the state evolves according to the following rule

\[ x_k = f(x_{k-1}, u_{k-1}) = x_{k-1} + \Delta t \begin{bmatrix} v_k \cos(\theta_{k-1} + \psi_{k-1}) \\ v_k \sin(\theta_{k-1} + \psi_{k-1}) \\ \omega_{k-1} \end{bmatrix} . \]

Since \( u_k \) is not available directly it has to be derived from the integrated robot poses returned from the on-board wheel controllers. \( u_k \) is computed by the differentiation of the controller readings.

**Figure 7.4:** XR4000 Odometry, before correcting for bias (light line) and after (dark line)

**Bias Correction**

The XR4000 used in this experiment has a significant drift to the right, when travelling on a straight line. Experiments revealed that, when the robot is instructed to travel on a straight line forward, it actually follows an arc of 100 meter radius, that corresponds to a rotation of \( 0.57^\circ \) to the right per meter travelled. This bias is corrected by adjusting rotational velocity \( \omega_k \), and the heading of the translational velocity \( \psi_k \) accordingly.

When the robot is travelling on a curve, the bias is likely to be different, however no experiments were conducted to estimate it. It is assumed to be equal to that of the straight line.
7.2.3 Truck: Odometry Model

An Ackerman model is used to model the odometry of the truck used to collect the Victoria Park data set. Sensor input consists of the velocity of the left rear wheel and the angle of the front wheel. Velocity of the centre of the vehicle $v_c$ is computed from the velocity of the left wheel $v_e$ using the following:

$$v_c = \frac{v_e}{1 - \tan \alpha \frac{H}{L}}.$$  

The state of the truck is defined by

$$\begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix} = f(x_k, u_k) = x_k + v_c \begin{bmatrix} \cos \theta_{k-1} \\ \sin \theta_{k-1} \\ \tan \alpha \frac{L}{H} \end{bmatrix} \Delta T.$$  

In the above: L=2.83m - the distance between the front and rear axles, H=0.76m - half the distance between the rear wheels. Truck coordinate system is set at the middle of the rear axle with $x$ axis pointing forward and $y$ axis pointing left. The laser sensor is mounted at 3.78m, 0.5m.

7.2.4 Region Model: Indoors

For the indoor experiment region extent was computed directly from the laser scans. The first five scans are used to compute the free space in front of the robot. This region defines the extent of the local map. The occupancy grid with a cell size of 10cm is computed by a standard ray tracing algorithm. Any uncertainty in the robot pose is ignored when building the occupancy grid.

7.2.5 Region Model: Victoria Park

In outdoor environments free space dominates. It is therefore possible to assign any region to the local map. A grid map with a resolution of 50cm is used. The initial map region is a rectangle aligned with the robot heading direction. At the end of the mapping process the region is trimmed, as shown in Figure 7.5. The final region is set to the intersection of the initial region and the region formed by the convex hull of the map. When computing the convex hull of the map any uncertainty in landmark estimates is ignored.
Figure 7.5: Example of region evolution: start with a polygon regions, after mapping is complete fit convex hull to the map, set region to the overlap between the initial region and the convex hull.

7.3 Experiment Setup and Analysis

Particle filters produce different output for the same input (it is the nature of the filter). As a result, the performance of a particle filter can not be judged based on just one execution of the algorithm.

Traditionally a RMS (Root Mean Squared) distance from the estimate to the true state is used to judge the quality of the PF. To use this metric one needs to know the true state of the system. In the case of a SLAM problem the path of the robot needs to be known. RMS is not a very useful metric for the purposes of this thesis, since HTSLAM does not produce a global estimate of the path. Even for global mapping approaches this metric is of little use, since the RMS error is likely to increase as the robot travels further away from the starting point. High RMS error on the fringes of the map does not always suggest a bad map, while even relatively low RMS error near the origin of the map may indicate an inconsistent map.

Self-consistency is a more important measure of the quality of the map, though it is harder to define and measure. In the ideal world the definition can be simple: the map should contain only those landmarks that exist in the environment and have been observed during the experiment. All observations should be assigned to the correct landmark. In real life experiments this condition will never hold. Sensors do produce erroneous readings. Data association is often ambiguous, especially in the environments where landmark density is higher than the sensor uncertainty. Furthermore finding the true correspondences between all landmarks and all observations is a prohibitively laborious task.

The map self-consistency requirement needs to be relaxed to accommodate for a realistic situation. In this work the following definition is used: if the filter experiences systematic data association errors, and diverges from the ground truth path, then the map is considered to be inconsistent. Figure 7.6 shows an example of an inconsistent
Figure 7.6: Example of an inconsistent map. On a second pass through the region (dashed line), data association failed and the robot erroneously added new landmarks (squares).

map. Errors such as these are easy to spot for a human operator with a priori knowledge of the environment.

The HTSLAM map is considered to be consistent if all local maps are consistent, and if all relative map transformations are consistent. Inconsistencies in relative local map poses can arise during loop closing, when incorrect map correspondences are established. Generally, inconsistent relative poses lead to inconsistent local maps, and are therefore easy to detect. Failure to close the loop does not result in the inconsistent map, however such a map provides less information about the environment, and is therefore inferior to a proper map.

7.4 Results Indoors

Indoor experiments were performed on the third floor of the RSISE building in ANU. A schematic of the environment is shown in Figure 7.7.

The HTSLAM algorithm was run 100 times on each data set. The map from each run was classified as one of: consistent and proper (closed loop), consistent but improper (failed to close the loop) or inconsistent.

Figure 7.7: Indoor environment: RSISE level 3.

The same data set was processed with 100 and 300 particles. One hundred runs,
each with a different random seed, were taken to discard the possibility of a “lucky run”. All of the runs have produced correct maps on this particular data set.

For each run HTSLAM produces one map for every particle. It is not practical to plot the maps for each particle, instead a single sample from the final distribution is displayed. There is no global reference frame in HTSLAM, however for display purposes a robot starting pose was used as a reference frame. All the local maps for a single particle are plotted in that reference frame.

Figure 7.8 shows one of the maps built with one hundred particles. Local maps are projected into the reference frame of the first map. Note that no post-processing was performed to align the maps. Figure 7.9 shows one of the maps produced with 300 particles. The paths of all runs projected into the reference frame of the first map are presented in Figure 7.10. As expected, the path estimate differs more between runs further from the origin.

7.4.1 Execution time

Figure 7.11 shows the plot of execution time per laser reading for one of the runs (100 particles were used). The spikes in the execution time correspond to the creation of the new region. Most of this time is spent computing the occupancy grid of the region. It is possible to do this computation in the background, while robot is still processing new observations. Note that the graph never exceeds 0.2 seconds, which is the time between observations in case of the SICK laser range finder. The algorithm runs faster than real-time. An important thing to note is that computation time per observation remains constant throughout the experiment. As the robot explores larger and larger area and the size of the map grows, the time to process an observation remains roughly the same.

7.5 Results Victoria Park

The Victoria park data set has been processed in a similar manner to the indoor data set: 100 randomised runs of 100 and 300 particles. While the majority of the runs have produced a correct map, some have failed. Table 7.1 shows the proportion of successful runs for 100 and 300 particles.

The result for a run with 100 particles is quite disappointing - every fourth execution of an algorithm failed to build a consistent map. However when the number of particles
§7.5 Results Victoria Park

(a) Projection of the HTSLAM map in the reference frame of map 1.

(b) Topological path of the robot.

Figure 7.8: Laser indoors: Map of corners (100 particles)
§7.5 Results Victoria Park

(a) Projection of the HTSLAM map in the reference frame of map 1.

(b) Topological path of the robot.

Figure 7.9: Laser indoors: Map of corners (300 particles)
Figure 7.10: Laser indoors: Global Path over 100 test runs.
Figure 7.11: Laser indoors: Execution time 100 particles.
Table 7.1: Laser outdoors: Summary of the experiments on the Victoria Park data set.

<table>
<thead>
<tr>
<th>Num. Particles</th>
<th>Consistent</th>
<th>Consistent but improper</th>
<th>Inconsistent</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>75</td>
<td>0</td>
<td>25</td>
</tr>
<tr>
<td>300</td>
<td>93</td>
<td>0</td>
<td>7</td>
</tr>
</tbody>
</table>

is increased to 300 the failure rate falls to 7 per cent. This a positive sign as it indicates that most failures are probably due to insufficient number of particles used. Manual inspection of the failed maps suggests that most of the failures happen straight after a first transition in to a previously mapped region. Better sampling strategy for the first transition might improve the performance of the algorithm.

Figure 7.12 and Figure 7.13 show an example of a correct map for 100 and 300 particles respectively. Figure 7.14 shows paths of all runs projected in to the reference frame of the first map.
(a) Projection of the HTSLAM map in the reference frame of map 1.

(b) Topological path of the robot.

Figure 7.12: Laser outdoors: Map of corners (100 particles)
Figure 7.13: Laser outdoors: Map of corners (300 particles)
Figure 7.14: Laser outdoors: Global Path over 100 test runs.
8.1 Sensor Description

The sensor used in the experiment consists of a stereo pair of cameras. The cameras are mounted on an XR4000 robot facing forward, parallel to the ground plane. The two cameras are aligned parallel to one another. The stereo baseline is approximately 30 cm. The relative pose of the cameras were computed by stereo calibration [102]. Internal camera parameters were determined by calibration for each of the cameras [102].

The cameras are capable of capturing colour images at a maximum resolution of 640x480 at the rate of 30 frames per second with interlacing, however the recording system on-board of the robot can sustain a much lower throughput. The capturing system records only odd fields (320x480) at an average rate of 11.5 frames per second per camera.

The cameras used in this experiment have a noticeable barrel distortion that is corrected in software. The correction is applied off-line. Figure 8.1 shows an example of original and corrected images.

8.2 Feature Detector: Vertical Edges

Vertical edges are common in human built environments, and can be detected fast and reliably. A matched pair of edges in two images corresponds to a line in world coordinates. A matched pair of vertical edges corresponds to a vertical line in the world if both cameras are parallel to the ground plane. A vertical line in a three dimensional space has only two degrees of freedom and can be defined completely by a point of intersection with the ground plane. Similarly, a vertical line in an image plane can be defined completely by a single parameter: its distance from the left side of an image.


A camera detecting vertical edges is effectively a planar bearing-only sensor.

By reducing the dimensionality of the sensor from 3d to 2d the observation model is simplified, which in turn simplifies uncertainty modelling. Since the robot motion is restricted to the plane, the vertical dimension can be safely ignored.

The most problematic part of the bearing only SLAM is landmark creation. One needs at least two bearing observations from different poses to compute an estimate of landmark location. There are a number of techniques commonly used [3]. Since the robot in this experiment has two cameras, the problem of creating a new landmark is simplified significantly. Having two cameras also assists in the data association process.

The first step of the algorithm is the detection of vertical edges in the left and right images using vertical Sobel edge detector [40]. To make feature detection more robust, short edges are discarded as these are more likely to be spurious. Edges that are few pixels apart are also removed at this stage.

In the second step correspondences between edges in the left and right images are established. The cameras on the robot are aligned to be parallel, and as a result epipolar lines are along the $x$ axis[40]. Normalised cross-correlation is used to establish initial
matches. Matches with correlation higher than a predefined threshold are accepted. If too many matches (> 5) are found for a particular edge, this edge is removed. Edges that had no correspondence are also removed. For the remaining edges the best three matches are considered.

It is clearly impossible that two distinct edges in the left image appear in the same spot in the right image, without one being hidden by another, in which case the depth of the edges cannot be determined. The matching process has to consider all pairwise matches to avoid matching the same edge in one image to several edges in the other.

One can visualise this problem as a graph: every pair of matched edges defines a node in the graph. Nodes of the graph that do not share a common edge in both images are connected. The largest fully connected sub-graph of this graph (commonly referred as “clique” or “maximal clique” of a graph) corresponds to the largest set of compatible edge matches. The problem of finding the clique of a graph is NP complete [13], which is intractable for a large set of features. There are, however, methods that can find an approximate solution [4, 75]. The approach adopted in this work is a simple greedy search algorithm with the following steps:

1. Sort matches in descending order according to their normalised cross-correlation results.
2. Set output set to empty.
3. Take one item from the top of the sorted input set.
4. If compatible with all elements in the output set, add it to the output set, else discard.
5. Repeat 3 and 4 until the input set is empty.

The compatibility step in the algorithm above can be implemented to have a constant time computational complexity. The overall computational complexity of the algorithm is therefore $O(N \log N)$ - the complexity of the sort.

An example of vertical edge detection is shown in Figure 8.2. Edge correspondences are indicated by colour: a blue edge in the left corresponds to a blue edge in the right image, green to green, and so forth. Locations of the features in the reference frame of the left camera are also shown. Landmark uncertainty is also shown. Note that
8.2 Feature Detector: Vertical Edges

Figure 8.2: Edge Detection Example: the two video frames from the left and right cameras are shown in the upper row, the plot below shows location of the edges in the sensor centric reference frame and their corresponding uncertainties.

Cartesian uncertainty is only used for initialisation of landmarks. Landmark update equations assume a bearing-only model.

The appearance of a feature provides important information to the mapping module by assisting in data association. This is especially useful in situations when two edges are very close spatially (like the cyan and magenta edges (edges 4 and 5, counting from the left) in the Figure 8.2).

Since vertical edges are generally uniform along the vertical dimension, the edge template can be “compressed” by computing the average intensity along the vertical dimension. The experiments presented use a template width of 7 pixels. Since all three colours per pixel are considered, each template is represented by 21 integer values. Using only 7 pixels per feature (21 values) greatly reduces computational effort for template matching. Templates are also normalised to have zero mean, making correlation computation fast (21 multiplications and 20 additions in this case). Normalisation also makes template matching less sensitive to variations in lightning conditions.

8.2.1 Sensor Uncertainty

There are several sources of uncertainty in the sensor. Some uncertainty comes from the fact that camera calibration is not perfect. The camera focal length $F$ and principal point $C$ are not known exactly and might also vary slightly during the experiment. A second source of uncertainty is the feature extraction mechanism. As point of view
changes, an edge might be detected on a slightly different part of a physical object. Furthermore, discretisation errors might add noise as well.

Bearing observation is computed from the column pixel $p$ using the following equation

$$z = f([p, C, F]^T) = \frac{p - C}{F}.$$ 

The Jacobian of $f$ with respect to $[p, C, F]^T$, $\nabla f$, is used to compute the approximate Gaussian uncertainty of $z$ from the uncertainties of $[p, C, F]^T$:

$$\nabla f = \begin{bmatrix} \frac{1}{F} & \frac{1}{F} & -\frac{p + C}{F^2} \end{bmatrix}.$$ 

The uncertainty of the bearing is then

$$\sigma_z^2 = \nabla f \begin{bmatrix} \sigma_p^2 & \sigma_C^2 & \sigma_F^2 \end{bmatrix} \nabla f^T.$$

### 8.3 Edge Landmarks

An edge landmark may be treated as a point landmark with some additional attributes. The extra attributes include the normalised edge template (7 pixel x 3 colours = 21 values) and a list of all observations of the landmark. The normalised average template is used to aid in data association and also during the map matching when closing loops. A list of observations is also used during data association if the result of matching with the average template failed to produce a definitive answer.

The data structure used to hold the list of observations can be copied in constant time, so there is no undue computational burden for keeping this information. Memory requirements however grow as more observations are collected. It is not absolutely necessary to keep all this information, as the average template is usually sufficient. The main motivation for doing so is to allow accurate template matching in situations where the appearance of the landmark changes significantly with observation angle.

The goal of keeping templates is to provide a comprehensive representation of the appearance of the landmark from various view points. This can be achieved with relatively few templates. It is not necessary to store all of them. An approach like principal component analysis [47] can be used to find a sufficient set of templates for a given edge feature. This problem is, however, beyond the scope of this thesis.
The true landmark state $x = [x, y]^T$ represents its location in the local reference frame. The camera pose in the local frame is denoted $x_c, y_c$ and camera orientation is $\theta_c$. For clarity, define $x' = x - x_c$ and $y' = y - y_c$. Observation $z_k$ and landmark states are related by the non-linear function $h$, defined by ($v = N(0, \sigma_v^2)$ is a measurement noise)

$$z_k = h(x, v) = \frac{-x' \sin \theta_c + y' \cos \theta_c}{+x' \cos \theta_c + y' \sin \theta_c} + v.$$

The true state of the landmark is not known, but the estimate of the landmark state at time $k$, $\hat{x}_k$, is available. The estimated covariance of the landmark is maintained by the EKF and is denoted $P_k$. For clarity, define also $\dot{x}_k' = \dot{x}_k - x_c$ and $\dot{y}_k' = \dot{y}_k - y_c$.

Additionally, define a Jacobian of partial derivatives of $h$ with respect to the landmark state:

$$H_{[i,j]} = \frac{\partial h_i}{\partial x_j}(\hat{x}_k, 0).$$

It is equal to

$$H = \begin{bmatrix}
\frac{-\dot{y}_k'}{x_k'^2 \cos^2 \theta_c + 2x_k' \dot{y}_k' \cos \theta_c \sin \theta_c + y_k'^2 \sin^2 \theta_c} \\
\frac{-\dot{x}_k'}{x_k'^2 \cos^2 \theta_c + 2x_k' \dot{y}_k' \cos \theta_c \sin \theta_c + y_k'^2 \sin^2 \theta_c}
\end{bmatrix}.$$

These Jacobians are used by the EKF, and also during data association, to project landmark uncertainty into the observation space.

The Jacobian of partial derivatives of $h$ with respect to noise is also needed by the EKF, and in this case it is trivially

$$V_{[i,j]} = \frac{\partial h_i}{\partial v_j}(\hat{x}_k, 0) = 1.$$

### 8.3.1 Data Association

First, all landmarks are projected into the observation space, both into the left and right camera. Those landmarks that fall outside the sensor range are discarded. For each observation the most likely landmark is found using the nearest neighbour search.

The quality of a match between a landmark and a single bearing observation is computed using the following:

$$w_k = \int p(z_k|\hat{x}_k)p(\hat{x}_k)d\hat{x}.$$

Here $\hat{x}_k$ is a landmark projected into the camera plane, and $p(\hat{x}_k)$ is approximated
to be a Gaussian, EKF style:

\[ p(\tilde{x}_k) = N(h(\tilde{x}_k, 0), H P_k H^T). \]

The term \( p(z_k | \tilde{x}_k) \) represents the measurement uncertainty and is also a Gaussian distribution \( N(z_k, \sigma^2_{zk}) \). Since both distributions are Gaussians, the integral of their product can be computed analytically.

Both left and right bearing have to match the landmark. The total quality of the match is a product of the left and right weights \( w^L_k w^R_k \).

Template matching is used to aid data association as a binary pass or fail operator. Template matching is only performed if the spatial tests have passed.

Nearest neighbour is the most common approach to data association, due to its simplicity, however there are some drawbacks to this method. The fact that observations were extracted from the same video frame provides an important piece of information - that these observations are of different physical entities. Nearest neighbour search discards this information. Different observations from the same video frame can be assigned to the same landmark.

Ideally the data association process should take all available information into consideration. One can use the joint compatibility data association [72], or one might explore the complete set of possible data associations directly by sampling from the set of all possible data association decisions [74]. In this work nearest neighbour search assisted by template matching proves sufficient, however a better data association approach is likely to improve the success rate of the existing approach.

### 8.3.2 Observation Update

An observation is treated as two independent bearing-only observations, first left then right. The standard EKF update equations are applied to update the estimate of the landmark state and its covariance.

The landmark template is also updated. It is simply an average of all observation templates. Each landmark maintains a count of the number of times it has been observed and a sum of all templates. The average template is computed from the sum by simply dividing by the total number of observations. After that, the template is normalised such that the mean is equal to zero and the sum of deviations from the mean squared is equal to 1. The normalisation is pre-computed to speed up the computation.
8.3.3 Genesis

When an edge pair does not match any of the existing landmarks in the map, a new landmark is added to the map. The initial landmark pose and the corresponding uncertainties are computed from the observations using the equations described below.

Let $x_r, y_r$ be the location of the right camera with respect to the left one and $\theta_r$ be the orientation of the right camera relative to the left one. Let $u_l, u_r$ be the bearing observations (tangent of the angle) from the left and right cameras respectively. Observation is defined to be

$$z = \begin{bmatrix} u_l \\ u_r \\ x_r \\ y_r \\ \theta_r \end{bmatrix}.$$

The landmark state expressed in the coordinate frame of the left camera is computed from the observation using the following:

$$x = f(z) = \begin{bmatrix} y_r - x_r \tan(\tan^{-1} u_r + \theta_r) \\ u_l - \tan(\tan^{-1} u_r + \theta_r) \\ y_r - x_r \tan(\tan^{-1} u_r + \theta_r) \\ u_l - \tan(\tan^{-1} u_r + \theta_r) - u_l \end{bmatrix}.$$

Assume that the uncertainty of the observation is a Gaussian with the following block-diagonal covariance matrix:

$$C_z = \begin{bmatrix} \sigma_{ul}^2 & \sigma_{ur}^2 & \sigma_{uy}^2 & \sigma_{uy}^2 \\ \sigma_{uy}^2 & \Sigma_{xy} & \Sigma_{xy} & \Sigma_{xy} \\ \sigma_{uy}^2 & \Sigma_{xy} & \Sigma_{xy} & \Sigma_{xy} \\ \sigma_{uy}^2 & \Sigma_{xy} & \Sigma_{xy} & \Sigma_{xy} \end{bmatrix}.$$

Computing the Jacobian of $f$ with respect to each of the observation elements,

$$J_{[i,j]} = \frac{\partial f_{[i]}}{\partial z_{[j]}}(z),$$

the initial uncertainty of the feature is then:

$$P = J C_z J^T.$$
The landmark template is initialised to that of the observation adjusted by the pose of the sensor in the local reference frame.

### 8.4 Defining Map Region

Unlike laser range sensors, vision sensors cannot provide free space information easily. It is therefore assumed that no free space information is available. The approach for defining region boundaries is identical to that used for the Victoria park data set, with an exception of the size of the initial region and the size of grid cells, which were set to be smaller.

### 8.5 Results

Vision data has been collected together with the laser corner data so that a comparison can be made between vision and laser experiments. The vision data set was processed with 100 and 300 particles. One hundred runs of the algorithm were taken to judge the robustness of the algorithm (100 for 100 particles and 100 for 300 particles).

Figures 8.3 and 8.4 show the map produced for one of the runs using 300 and 100 particles respectively. Local maps are projected into the reference frame of the first region.

Vision is a much more challenging sensor than laser. While most of the runs have produced good map, many runs failed. Table 8.1 lists the success/failure rates for the experiment.

<table>
<thead>
<tr>
<th>Num. Particles</th>
<th>Consistent</th>
<th>Consistent but improper</th>
<th>Inconsistent</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>67</td>
<td>1</td>
<td>32</td>
</tr>
<tr>
<td>300</td>
<td>71</td>
<td>5</td>
<td>24</td>
</tr>
</tbody>
</table>

*Table 8.1: Vision indoors: Summary of the experiments.*

Increasing the number of particles makes the algorithm more robust, but only slightly. Figures 8.5 and 8.6 show examples when HTSLAM did not succeed to build a proper map. Figure 8.7 shows the paths of all 100 experiments projected into the reference frame of the first region.
§8.5 Results

(a) Projection of the HTSLAM map in the reference frame of map 1.

(b) Topological path of the robot.

Figure 8.3: Vision indoors: Map of edges (300 particles/map 52)
Figure 8.4: Vision indoors: Map of edges (100 particles/map 89)
Figure 8.5: Vision indoors: Jumped out, failed to come back (300 particles/map 40). Topologically correct, but not "optimal"
§8.5 Results

(a) Projection of the HTSLAM map in the reference frame of map 1.

(b) Topological path of the robot.

Figure 8.6: Vision indoors: Failed transition 7 to 2 (300 particles/map 18).
§8.5 Results

8.5.1 Error Analysis

There are a number of reasons for a poorer performance of the vision SLAM when compared to a laser detector in the same environment.

- Data association at scale. Collisions with this type of sensor.
- Pose estimation of the sensor.

Figure 8.7: Vision indoors: Global Path over 100 test runs.

(a) 100 particles

(b) 300 particles
8.5.1 Error Analysis

There are a number of reasons for a poorer performance of the vision SLAM when compared to a corner detector in the same environment:

- Data association is quite challenging with this type of sensor.
- Poor calibration of the sensor.
- Unmodelled sensor error correlations.
- Sub-optimal local region assignment

Data association is especially difficult when revisiting the region when travelling in an opposite direction. The appearance of landmarks can vary significantly when viewed from a different angle. Furthermore, a very different set of landmarks is visible when traversing a region in opposite direction. This makes transitions back into a mapped region from a different direction prone to data association errors.

Camera provides a rather accurate bearing measurement. As a result the performance of the sensor is sensitive to calibration errors. The more accurate your sensor is, the more obvious the impact of biases in the error model will be.

Another important factor to consider is the effect of unmodelled correlations between errors in individual observations. All features detected in the same video frame share common source of uncertainty: camera calibration parameters. Yet errors are assumed independent for practical reasons.

As mentioned previously vision and corner data sets were collected from the same platform at the same time, yet one can see that local region assignment is significantly different between the two data sets. In corner based SLAM occupancy grid was used to define a local region. In the case of vision based experiment no free space information was available, hence regions were assigned to be of fixed size. Using free-space information for local region assignment is advantageous as it partitions the space in a more sensible way. Consequently, local map transitions are less reliable for a vision data set. This suggests that a better model for local region assignment might be necessary.
This thesis has presented a novel approach to mapping that combines metric and topological information to build maps of medium to large environments. In HTSLAM metric and topological representations are equally important. HTSLAM does not aim to build a global metric map with a single reference frame. Instead, each local region has its own reference frame. Relative poses of neighbouring regions are maintained in the topological structure of a HTSLAM map. The proposed map structure facilitates computation of relative poses between any two local regions, effectively allowing a “global” reference frame to be selected in any of the regions.

A common problem for global SLAM is an increasing uncertainty as the robot moves further away from the origin. The HTSLAM map structure moves the origin into the current local region, providing a more efficient representation of spatial uncertainty. HTSLAM uses particles to represent spatial uncertainty between local regions. To the best of my knowledge there is no other published research that combines topological mapping with particle filter metric mapping. The main advantage of using particles over more traditional Gaussian representation is the ability to efficiently represent multi-modal non-linear distributions. Approaches that assume Gaussian uncertainty, suffer from accumulation of approximation errors. The effect of linearisation becomes especially significant in large scale environments.

It is generally impossible to partition a realistic environment into a set of completely separate regions, some neighbouring regions are likely to share common landmarks. Local maps cannot be truly decorrelated. HTSLAM captures these inter region correlations in a sample of map particles.

Loop closing is one of the difficult problems for SLAM. This thesis presents an elegant solution that deals with the inherent ambiguities of loop closing situation in a formal rigorous manner. HTSLAM does not rely on any “performance metrics” that might be sensor/environment specific, instead multiple mapping hypotheses are
evaluated against each other directly as part of a single particle filter. The hybrid structure of HTSLAM makes it possible to delay loop closing until there is sufficient evidence that the region has been visited previously. In fact, loop closing can be delayed indefinitely, since failure to close the loop does not result in an invalid map, only a sub-optimal one.

The performance of HTSLAM has been tested on three distinct data sets:

- Laser range finder mounted on XR4000 robot, operating in an indoor environment. Corners and doorways were used as landmarks.

- Laser range finder mounted on a utility vehicle, operating in a park (the well-known Victoria Park data set [33], made publicly available by the team from ACFR, Sydney). Tree trunks were used as landmarks.

- Stereo cameras mounted on an XR4000 robotic platform, operating in an indoor environment. Vertical edges were used as landmarks.

The results show that HTSLAM is capable of building maps of medium sized environments. Computational requirements stay constant due to the structure used by the algorithm, therefore potentially mapping of large environments should be possible. HTSLAM is capable of using vision sensors. Vision sensors are generally considered to be challenging for SLAM due to highly ambiguous data association.

9.1 Future work

There are a number of improvements to the current algorithm that can make mapping more robust and efficient. Experiments with real data have shown that most failures occur after the transition between local regions. This is because the process of transition between regions generates a large proportion of weak particles. The process of resampling will generally favour good particles and prune out weak ones, however there is a danger that there will not be any particles in the right place. A smarter sampling strategy that takes into account recent observations, or overlap between local maps, can provide a significant improvement over the current approach.

At the moment there is no obvious answer to how many particles are needed to build a map of a particular environment. It would be beneficial to automatically adjust the number of particles during mapping, increasing the number of particles when the
situation is ambiguous (loop closing, or transition), and decreasing it when things are going smoothly. Automatic adjustment of sample size has been used for the problem of localisation [24], it would be interesting to see if such an approach can be applied to Rao-Blackwellised particle filters as well.

Most environments contain static, semi-static and dynamic features. Static features are always present, for example walls in the indoor environment. Dynamic features include moving objects like humans or other robots. Semi-static features are those that change their position from time to time, a couch or a desk for example. Traditional approaches to mapping assume a static world. While these approaches can tolerate a small number of dynamic objects, changes to the environment due to semi-static objects will render the map unusable eventually, requiring a complete rebuild. HTSLAM map structure provides a good framework for a long term map management system.


[64] B. Kuipers, J. Modayil, P. Beeson, M. MacMahon, and F. Savelli. Local metrical and global topological maps in the hybrid spatial semantic hierarchy. In *IEEE*


