Multi-robot Information Fusion
Considering spatial uncertainty models

Lars A. A. Andersson

Department of Management and Engineering
Linköping University, SE–581 83 Linköping, Sweden

Linköping 2008
Sparsity pattern of the resulting information matrix after joining of two robot maps. The map joining is done with C-SAM, which relies on inter-robot observations.
To Hanna

In theory, theory and practice are the same. In practice, they are not.

Lawrence Peter Berra
Abstract

The work presented in this thesis covers the topic of deployment for mobile robot teams. By connecting robots in teams they can perform a better job than each individual is capable of. It also gives redundancy, increases robustness, provides scalability, and increases efficiency. Multi-robot Information Fusion also results in a broader perspective for decision making. This thesis focuses on methods for estimating formation and trajectories and how these can be used for deployment of a robot team. The problems covered discuss what impact trajectories and formation have on the total uncertainty when exploring unknown areas. The deployment problem is approached using a centralized Kalman filter, for investigation of how team formation affects error propagation. Trajectory estimation is done using a smoother, where all information is used not only to estimate the trajectory of each robot, but also to align trajectories from different robots. Both simulation and experimental results are presented in the appended papers. It is shown that sensor placements can substantially affect uncertainty during deployment. When deploying a robot team the formation can be used as a tool for balancing error propagation among the robot states. A robust algorithm for associating rendezvous observations to align robot trajectories is also presented. Trajectory alignment is used as an efficient and cost-effective method for joining mapping information within robot teams. When working with robot teams, sensor placement and formation should be considered to obtain the maximum from the system. It is also of great value to mix robots with different characteristics since it is shown that using sensor fusion the robots can inherit each other’s characteristics if sensors are used correctly. Information sharing requires modularity and general models, which consume computational resources. Over time computer resources will become cheaper, allowing for distribution, and each robot will become more self-contained. Together with increased wireless bandwidth this will enable larger numbers of robots to cooperate.

Keywords:  Multi-robot, Sensor Fusion, Formation, Localization, Estimation, SLAM
Acknowledgements

This work has been carried out at the Division of Fluid and Mechanical Engineering Systems at Linköping University. There are a great number of people that I would like to thank for supporting me during the course of my research. First of all, I would like to thank my supervisor Prof. Karl-Erik Rydberg for his kindness and support over the years. I would also like to thank my co-writers Dr. Jonas Nygård, Marcus Berghult and Peter Nordin. Many of the results that are presented here come out of the long and inspiring discussions we have had over the years. I would also like to thank the head of the division, Prof. Jan-Ove Palmberg, for giving me the opportunity to do this research.

I pass a special thought to my parents, family and friends for all the support you have given me over the years. I am grateful to all members and former members of the Division of Fluid and Mechanical Engineering Systems for being there when in despair. A special thank-you to Magnus Sethsson and Anders Darander, with whom I have worked closely but not in the research mentioned in this thesis.

Linköping, August 2008

Lars A. A. Andersson
The following six appended papers are organized in chronological order of publication and will be referred to by their Roman numerals. All papers are printed in their originally published state.

In all papers the first author is the main author and editor, responsible for the work presented, with additional support from the co-authors.


The following papers are significantly related to the work presented but are not part of the work included in this thesis.


## Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>C-SAM</td>
<td>Collaborative Smoothing and Mapping</td>
</tr>
<tr>
<td>C-SLAM</td>
<td>Cooperative Simultaneous Localization and Mapping</td>
</tr>
<tr>
<td>CCD</td>
<td>Charge Coupled Device</td>
</tr>
<tr>
<td>CLAM</td>
<td>Cooperative Localization and Mapping</td>
</tr>
<tr>
<td>CML</td>
<td>Concurrent Mapping and Localization</td>
</tr>
<tr>
<td>EKF</td>
<td>Extended Kalman Filter</td>
</tr>
<tr>
<td>IRB</td>
<td>Industrial Robot</td>
</tr>
<tr>
<td>KF</td>
<td>Kalman Filter</td>
</tr>
<tr>
<td>LRS</td>
<td>Laser Range Scanner</td>
</tr>
<tr>
<td>MAP</td>
<td>Maximum a Posteriori</td>
</tr>
<tr>
<td>MCL</td>
<td>Monte Carlo Localization</td>
</tr>
<tr>
<td>ML</td>
<td>Maximum Likelihood</td>
</tr>
<tr>
<td>OG</td>
<td>Occupancy Grid</td>
</tr>
<tr>
<td>PC</td>
<td>Personal Computer</td>
</tr>
<tr>
<td>PDF</td>
<td>Probability Density Function</td>
</tr>
<tr>
<td>PF</td>
<td>Particle Filter</td>
</tr>
<tr>
<td>RBPF</td>
<td>Rao-Blackwellized Particle Filter</td>
</tr>
<tr>
<td>RHS</td>
<td>right-hand side</td>
</tr>
<tr>
<td>SAM</td>
<td>Smoothing and Mapping</td>
</tr>
<tr>
<td>SEIF</td>
<td>Sparse Extended Information Filter</td>
</tr>
<tr>
<td>SLAM</td>
<td>Simultaneous Localization and Mapping</td>
</tr>
<tr>
<td>SPmap</td>
<td>Symmetries and Perturbations map</td>
</tr>
<tr>
<td>SRIS</td>
<td>Square Root Information Smoothing</td>
</tr>
<tr>
<td>T-SAM</td>
<td>Tectonic Smoothing and Mapping</td>
</tr>
<tr>
<td>TCP</td>
<td>Tool Center Point</td>
</tr>
<tr>
<td>UGV</td>
<td>Unmanned Ground Vehicle</td>
</tr>
<tr>
<td>USB</td>
<td>Universal Serial Bus</td>
</tr>
<tr>
<td>WLAN</td>
<td>Wireless Local Area Network</td>
</tr>
</tbody>
</table>
Nomenclature

$\alpha$  Measurement of relative orientation
$\beta$  Baring measurement
$\Gamma$  Measurement error covariance
$\gamma$  Measurement error vector
$\Lambda$  Covariance of motion errors
$\Sigma$  Covariance of observation errors
$\theta$  Vector of optimization variables
$A$  Observation Matrix
$a$  Motion prediction error
$B$  Jacobian of observation model $c(.),$ evaluated for the base node $b$
$b$  RHS vector for least squares problem
$b_{pq}^c$  Base node between reference frame $p$ and $q$
$C$  Jacobian of observation model $c(.),$ evaluated for robot states $x$
$c$  Observation residual
$D$  Accumulated robot translation, between two sample times
$F_x$  Jacobian of motion model $f(.),$ evaluated for robot states $x$
$G_u$  Jacobian of motion model $f(.),$ evaluated for driving vector $u$
$H$  Jacobian of observation function $h(.),$ evaluated for robot states $x$
$J$  Jacobian of observation function $h(.),$ evaluated for landmark states $l$
$K$  Kalman gain matrix
$l$  Landmark state vector
$n$  Rendezvous observation residual
$P(x)$  Covariance of a vector $x$
$P(x,y)$  Cross covariance between two vectors, $x$ and $y$
$P(i|j)$  Covariance at time $i$ given all information until time $j$
$Q_u$  Covariance of control input errors
$Q_w$  Covariance of the linear process errors
$R(\phi)$  Rotation matrix with aspect of angle $\phi$
$S$  Innovation covariance
$U$  Robot pose change between two consecutive sample times
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$u$</td>
<td>Vector of control inputs</td>
</tr>
<tr>
<td>$v$</td>
<td>Vector of observation errors</td>
</tr>
<tr>
<td>$w$</td>
<td>Vector of linear process errors</td>
</tr>
<tr>
<td>$x$</td>
<td>Robot state vector, typically $[x, y, \phi]^T$</td>
</tr>
<tr>
<td>$z$</td>
<td>Observation vector</td>
</tr>
<tr>
<td>$\hat{x}$</td>
<td>Estimated conditional mean of a variable $x$</td>
</tr>
<tr>
<td>$\mu_{1/2}(x)$</td>
<td>Median of the elements in a vector $x$</td>
</tr>
<tr>
<td>$\nu$</td>
<td>Innovation</td>
</tr>
<tr>
<td>$\Omega$</td>
<td>Accumulated robot rotation, between two sample times</td>
</tr>
<tr>
<td>$\omega$</td>
<td>Driving rotational speed</td>
</tr>
<tr>
<td>$\rho$</td>
<td>Correlation coefficient</td>
</tr>
<tr>
<td>$\sigma_x$</td>
<td>Standard deviation for error of variable $x$</td>
</tr>
<tr>
<td>$\tilde{x}$</td>
<td>Estimation error of a variable $x$</td>
</tr>
<tr>
<td>$\xi_i$</td>
<td>Threshold for hypothesis test $i$</td>
</tr>
<tr>
<td>$c(.)$</td>
<td>Observation model for rendezvous observations</td>
</tr>
<tr>
<td>$D$</td>
<td>Mahalanobis distance</td>
</tr>
<tr>
<td>$f(.)$</td>
<td>Motion prediction model</td>
</tr>
<tr>
<td>$h(.)$</td>
<td>Observation model</td>
</tr>
<tr>
<td>$P(.)$</td>
<td>Probability density function, PDF</td>
</tr>
<tr>
<td>$r$</td>
<td>Distance measurement</td>
</tr>
<tr>
<td>$T$</td>
<td>Sample time</td>
</tr>
<tr>
<td>$v$</td>
<td>Driving forward speed</td>
</tr>
<tr>
<td>$x(k</td>
<td>k-1)$</td>
</tr>
<tr>
<td>$x(k)$</td>
<td>A variable $x$ evaluated at time $k$</td>
</tr>
<tr>
<td>$x^i_j$</td>
<td>Instant $j$ of variable $x$, referenced in frame $i$</td>
</tr>
</tbody>
</table>
# Contents

1 Introduction
   1.1 Background .................................................... 2
      1.1.1 Why Multiple Robots? .................................... 2
      1.1.2 Teams vs. Groups of Robots ............................... 4
      1.1.3 Localization and Mapping ................................ 5
      1.1.4 Formation and Deployment ................................. 12
   1.2 The Grand Vision ................................................ 15
   1.3 The Structure of the Thesis .................................... 15

2 Framework ................................................................. 17
   2.1 Stochastic Mapping ............................................. 18
      2.1.1 Kalman Filter ............................................. 19
      2.1.2 Map Representation ....................................... 20
      2.1.3 Adding Information ....................................... 21
      2.1.4 Moving Robots ............................................ 24
   2.2 Smoothing and Mapping .......................................... 25
      2.2.1 Representation ............................................ 26
      2.2.2 Adding Information ....................................... 28
      2.2.3 Collaborative Smoothing and Mapping .................... 30
      2.2.4 Base-node Estimate ....................................... 32

3 Modeling of Perception ............................................... 37
   3.1 Observation Model ............................................... 37
   3.2 Observation Prediction ......................................... 39
      3.2.1 Observation between Two Poses .......................... 41
   3.3 Characteristics for Common Sensors ............................ 41
      3.3.1 Laser Scanner ............................................. 43
      3.3.2 Camera .................................................... 44
      3.3.3 Stereo Camera ............................................. 44
4 Motion Models 47
   4.1 Holonomic and Non-holonomic Systems 48
   4.2 General Representation 49
   4.3 Time Discrete Constant Turn Model 51
      4.3.1 Basic Model 52
      4.3.2 With Lateral Slip 54
      4.3.3 With Timing Error 56

5 Contributions 61
   5.1 Results 62

6 Discussion and Future Outlook 65

7 Summary of Papers 71

Bibliography 75

Appended papers

I On Sensor Fusion Between a Pair Of Heterogeneous Robots 83

II On Utilizing Geometric Formations for Minimizing Uncertainty in 3 Robot Teams 93

III Effects on Uncertainty Utilizing Formation-Planning in Robot Teams 105

IV C-SAM: Multi-Robot SLAM using Square Root Information Smoothing 113

V Sensor Data Fusion for Terrain Exploration by Collaborating Unmanned Ground Vehicles 123

VI On Multi-robot Map Fusion by Inter-robot Observations 133
The work presented in this thesis deals with uncertainty in mobile robot teams. The work covers two major areas concerning how uncertainty can be affected during deployment and how trajectory alignment can be used as a tool for two robots to share uncertain spatial information about their surroundings.

Mobile robots and autonomous systems are starting to appear more frequently in industry and everyday life. Systems that have appeared in the marketplace in recent years are different types of cleaning robots such as the Trilobite, [1], and iRobot Roomba, [2]. These vacuum cleaners may seem to perform a trivial task but still constitute an important step by indicating the acceptance of robots in a home environment. This development phase has been seen in another technology field, which is also fundamental for the robot industry: the evolution of the Personal Computer (PC), introduced in the mid-1970s.

“*The challenges facing the robotics industry are similar to those we tackled in computing three decades ago.*”
(Bill Gates, Scientific American, January 2007)

This statement by Bill Gates, a pioneer in PC software development, indicates that the research field of robotics is still in its cradle.

Hitherto, it has been areas other than the consumer market that have driven robot development, such as industry, the military and space research. All of these players have an interest in autonomy in their systems. NASA have been sending semi-autonomous rovers to Mars for a number of years [3]; in a similar way the Russians were sending rovers to the moon in the late 1960s [4], see Figure 1.1. These systems need to be very robust and in some aspects self-contained, due to the great time delays when executing maneuvers.
Multi-robot Information Fusion

1.1 Background

Throughout this work a system is seen as a set of mechanical units that together execute a task deterministically, to reach a common goal or solve a given task. To make a system autonomous it needs to be completely self contained and handle all unexpected situations. From a design perspective a fully autonomous system is a utopia. Semi-autonomous systems are therefore more valid, since limitations from the design phase are included in the circumstances under which the system is autonomous or self-contained.

1.1.1 Why Multiple Robots?

This is a relevant question. In mobile robotics it is commonly accepted that it is better to have multiple-robots, given the required space and that each robot is capable of taking care of itself. Assume there are two distance measurement sensors needed to perform a certain task. To use information from both sensors they need to be spatially related. If both sensors are mounted on the same robot, the relation between them is unambiguous since they both move together with the robot. On the other hand, both sensors will more or less provide information from the same point of view. If sensors are mounted on separate robots, the sensors will not be as tightly coupled, due to uncertainties in relative robot location. On the other hand, it is possible to get different points of view from the two sensors, which in many cases is more valuable than a tight coupling and will therefore provide a better result. In some cases it is even critical to complete a task at all. This issue is to some extent discussed in all the appended papers.

A greater number of robots will most likely increase cost. There may in
many cases nonetheless be a cost benefit due to the fact that two robots will at a minimum do the job in half the time. In an exploration scenario each robot will cover half the total area of a single robot. There will also be the benefits of distributing sensor equipment, which results in redundancy and a better perspective.

An example of a task where two robots will perform better than a single robot is shown in Figure 1.2. Assume that a task is to measure the relative distances between four landmarks, or corners of a rectangle. To perform a distance measurement two sensors are required. The sensors used are not capable of measuring all four landmarks from a single pose. Transport is therefore necessary to cover all four landmarks. In figure 1.2(a) a single robot is equipped with both sensors. The relative distance between the two closest landmarks is measured at the start. The required motion to cover the other pair introduces uncertainty to the robot pose and the two later landmarks will therefore not be as accurately measured. In the latter case, figure 1.2(b), two robots are used with one sensor on each. The result of this is that the first two landmarks are not as accurately measured since the sensors are not as tightly coupled. In this case it is also necessary to travel to the other pair of landmarks. The motion of the robots introduces uncertainty in the same way as in the single robot case, while it can be supported with observations between the robots, giving leverage on the measurement. This results in the task being completed with a better result since the relative pose between the landmarks is measured with lower uncertainty.
Multi-robot Information Fusion

1.1.2 Teams vs. Groups of Robots

For the discussion a clear distinction is made between a robot team and a robot group. In a group each member will have to come up with a motion strategy based only on the information that they gather themselves and observations about other group members. An example in nature would be ants communicating their path to each other by leaving a pheromone trace for other ants in the hill to find [5]. A team is a subset of a group where the members share information by communicating in both ways. Given a fixed number of robots a team would be more powerful than a group, since the inter-communication enables the members to come up with a strategy, based on information from all members. However, for a large number of robots the communication overhead and the problem of fusing all information can be very expensive in terms of computer resources and the group strategy may therefore be an advantage.

An illustrative scenario is for two robots to negotiate an obstacle course.

Figure 1.3: Two strategies for robot cooperation. In a) and b) the two robots work as a group. Each robot has its own idea of the world and observes the other robot. In c), where the two robots work as a team, the sensor information is distributed and both robots get a better understanding of the surroundings.
Introduction

Each of the robots has an eye-like sensor for observing the surroundings. In the situation where the two robots operate as a group each robot observes the obstacles in the area closest to them, including the other group members, see figures 1.3(a) and 1.3(b). The extended information, gained by the group constellation, is that each member can observe actions made by the other members. In small groups this is of no more use than for predicting each other’s possible trajectories and avoiding collisions. In groups with large numbers of members this can however be useful since each member can observe a “trend” for the group and thereby decide to join the trend or not. When the group strategy is applied to a very large number of robots, this is in some literature referred to as swarm robotics, [5–8]. In a team constellation information is shared amongst the members and each member therefore gets information about what other members base their decisions on, see figure 1.3(c). Naturally there are different levels of cooperation. The highest level of cooperation is to share all collected information using radio communication or similar. When full information is shared, the system essentially becomes a distributed sensor system.

1.1.3 Localization and Mapping

Localization is what chiefly separates Industrial Robots (IRBs) from mobile robots. An IRB is in general installed in one position and all predefined tasks are related to this installation position. This results in a global operating positioning error directly related to the precision of installation, for which the IRB can be calibrated, [VIII]. Due to holonomic kinematics, the precision performance for repeated operations is also very good. The properties of holonomic and non-holonomic systems are further discussed in section 4.1. As the name indicates, a mobile robot is meant to change position. This can be a task all by itself since each change in position will introduce some type of error. Due to this, the localization problem is handled separately from the other tasks in many cases since the performance of the rest of the tasks is often directly related to the performance of the localization.

When localization is performed using landmarks in the surrounding environment and simultaneously building a map of these landmarks it is usually refereed to as Simultaneous Localization and Mapping (SLAM). Localization is where a robot is located relative to a common reference frame, later referred to as pose. Mapping is how the surroundings are spatially related to the robot. Over the last twenty years much research interest has been devoted to mobile robot localization and mapping. One of the first implementations of robot mapping is presented in [9]. This approach is based on Occupancy Grids (OG). The implementation shows the possibility to build environmental maps using a mobile robot, assuming the robot location is known. In figure 1.4 one can see what happens when uncertainties are not considered properly in a real scenario. A natural extension of this work is to find a way to solve the problem considering location, or odometry, error.
One of the first publications to address the problem of estimation of uncertain spatial relationships in robotics is [10]. This work proposed a method, later referred to as “The Stochastic Map”, which became a milestone in the research field of mobile robotics and has had a great influence on many recent methods. The filter update complexity of this naive method is $O(n^2)$ for $n$ landmarks. It is therefore not well suited for storing map information about a large environment. This framework is later implemented and used for outdoor experiments in [11]. An indoor experiment of a single robot localization with a time-of-flight laser is demonstrated in [12] where natural geometric landmarks, such as walls, are extracted and stored in a map framework together with the uncertainties. The wall segments are extracted from the laser data using the range weighted hough transform, presented in [13]. Figure 1.5 shows a map created with a SLAM algorithm and a CAD drawing of the same area. In the appended papers [I–III] the same framework is used but extended to work with multiple robots fusing data in a centralized manner.

A proof of an existing solution for the air-borne SLAM problem was introduced in [15]. It is proven that the uncertainty of a relative estimated map
Introduction

1.5(a) and 1.5(b)

Figure 1.5: By using a motion model, laser range data and a SLAM algorithm it is possible to create a map of an unknown environment. The map in a) is created incrementally as the robot deploys the area shown as a CAD drawing in b). This map shows a large open area in the Tech Museum in San Jose, CA, and was presented in [14].

monotonically converges to zero. Later it is shown that the absolute accuracy of the map converges to the absolute accuracy of the vehicle at the instant the first landmark is observed. The proof is only valid for linear motion and measurement models assuming uncorrelated gaussian noise. The problem is that in a real scenario the linear motion models and uncorrelated noise are not applicable. It is known that the inherent approximations due to linearizations of system and measurement models cause the Extended Kalman Filter (EKF) to diverge, [16]. In [17] it is proven that the EKF based SLAM algorithm always yields an inconsistent map, due to linearization errors in the sensor. However, these inconsistencies only become apparent after several hundred landmark updates. It is therefore questionable whether the framework in [10] provides a general robust and rigorous solution to the SLAM problem. This problem is further discussed in [18] where a robocentric mapping method is proposed to reduce the divergence effect. It is also claimed that a common misconception is that a non-zero initial level of uncertainty in the vehicle location may improve map consistency.

The Symmetries and Perturbations Map (SPmap) found in [19] is an interesting extension of The Stochastic Map. This implementation overcomes difficulties reported in earlier work dealing with singularities in the representation of geometric features. In contrast to earlier EKF based implementations, any type of geometric entity may be represented within the framework and is therefore well suited to deal with multiple sensor types. The experiments conducted
Multi-robot Information Fusion

also indicate the importance of keeping the correlations between entities (features) within the map to keep it consistent. Another key contribution for sensor data handling is [20], a stochastic mapping framework that introduces delayed decision making capability. This is not possible in the original framework since it did not explicitly represent robot positions and correlations over time. The solution is based on expanding the state vector to include estimates of prior vehicle states, thereby enabling the robot to utilize the information at a later stage. Delayed decision making is a required feature to be able to do single camera SLAM, addressed in [21]. The solution is based on an EKF framework using a constant turn model for predicting camera motion. When new features are detected a Particle Filter (PF), [22], is used together with a number of images to estimate the initial Z-component (depth in the image where the feature is located). The implementation does not cover the data association problem.

The issues with using an EKF based stochastic mapping solution lead to discussions about what to think of when implementing a SLAM algorithm. In [23], three postulates are presented for what should be considered to achieve an ideal solution to the SLAM problem. These are related to uncertainty in the solution, memory space for storing the map and the computational cost of solving the system. It is shown that by representing the SLAM problem based on sparse information matrices, the postulates are fulfilled, making the representation well suited for a SLAM algorithm. The Sparse Extended Information Filter (SEIF) is later implemented in [24] which yields a $O(n)$ computational cost for $n$ landmarks. Even though the actual mapping is linear the drawback is that the data association problem becomes more complex since the map content is represented in information form. However, in [25] real-world data from an outdoor experiment is used to show the performance of a Maximum Likelihood (ML) principle to handle data association. Even though the method used is approximative, it is claimed to perform favorably compared to the EKF due to the $O(n)$ computational complexity. Later, a novel approach for extracting consistent covariance bounds used for data association is presented in [26]. The complexity of the method scales asymptotically linear with map size and it is shown to provide a conservative approximation useful for landmark matching.

Bayesian Belief Net is another way of representing the SLAM problem. The work presented in [27] uses this method for making globally consistent robot maps, similar to [28,29]. This approach is based on a relaxation algorithm which is shown to have $O(n)$ complexity and is therefore computationally highly efficient. The work is later extended in [30] where a more thorough explanation of the work is presented. Certain situations such as loop closing are identified to make the algorithm slower than usual. This is especially important if the accumulated odometric errors have become large. An accelerated Relaxation-based SLAM using a multi-grid approach is presented in [31]. The initial results were presented in [32]. This method overcomes the issues with the relatively slow convergence of closing large loops, identified in the original implementation. This is overcome by using so-called multi-grid methods used for solving partial
differential equations by optimizing the map at multiple levels of resolution. The O(n) property is retrieved by making iterative refinements to the existing solution at each step, instead of re-solving the equation system from scratch.

Smoothing is an alternative to filtering where all information is considered simultaneously, in contrast to incrementally. Square Root Information Smoothing (SRIS) is considered in [33] as a viable alternative to solve the SLAM problem. It is proposed that the SRIS approach is fundamentally better for these types of problems than the commonly used EKF. The first known publication of this work is [34] where SRIS applied to the SLAM problem becomes $\sqrt{SAM}$, later referred to as SAM. It is claimed that SRIS has several significant advantages over the EKF:

- Much faster when adding new information
- Can be used in either batch or incremental mode
- Better equipped to deal with non-linear process and measurement models. (In EKF early linearization errors can not be improved.)
- It has the possibility to yield the entire, smoothed, robot path trajectory, at lower cost.

One of the drawbacks of SRIS is that the computational complexity grows without bounds over time. However, in many typical scenarios the computational complexity of an EKF covariance matrix will grow much faster, due to its density. Also, as for all information matrix based approaches, it can be expensive to recover the covariance matrix governing the unknowns. On the other hand, the structure of the SAM observation matrix is inherently sparse, which simplifies the computation. One fundamental difference between the EKF and the SRIS is how the computational cost is affected for new, unexplored areas. While the computational cost of an SRIS grows continuously with the number of measurements made, the EKF grows quadratically for new areas and remains almost constant when closing the loop and re-entering previously visited areas. The reason for this is the fundamentally different approaches for collecting information. However, the structure of SAM has the same benefits when adding new information as delayed decision making introduced in [20].

The speed of the original SAM implementation is questioned by the author. This issue is later solved in [35] where more efficient optimization algorithms are used. It is also shown how the covariance for the exact values of interest can be extracted without having to calculate the complete dense covariance matrix. This is useful when there is a need to perform data association. The issue with computational complexity growth over time still remains and Tectonic SAM (T-SAM) is therefore introduced in [36]. This is a method for dividing large SAM maps into smaller sub-maps. This enables larger areas to be covered in real time without having the issues with computer resources. The work presents an out-of-core method for dealing with these large maps.
for each sub-map is also recovered when they are rejoined to larger maps, which saves calculation resources.

Monte Carlo Localization (MCL) using Particle Filter is also a way of solving the SLAM problem. A complete reference on how to implement an MCL algorithm is found in [37]. A predecessor to this work is [38] where Concurrent Mapping and Localization (CML) is presented. This successfully implemented an ML solution to the SLAM problem. An extension to the general MCL called Mixture Monte Carlo Localization is found in [39]. The Mixture-MCL overcomes a range of limitations that currently exist for different versions of MCL, such as the inability to estimate posteriors for highly accurate sensors, poor degradation to small sample sets, and the inability to recover from unexpected large state changes (robot kidnapping). An initial presentation of the Fast-SLAM algorithm is found in [29]. This work represents the SLAM problem as a belief net. An extension is found in [40] which applies a Rao-Blackwellized Particle Filter (RBPF) to estimate a posterior of the path for a robot, in which each particle has associated an entire map to it, as previously proposed by Murphy in [28].

Multi-robot

Multi-robot SLAM has not been studied as extensively as the single robot case. On the other hand, the results are most interesting and there is much work that indicates a major advantage of using more than one robot in many situations. The work that has been done can be divided into two categories: collaborative and cooperative. The collaborative multi-robot case refers to robots working as a team in realtime, continuously updating each other with the latest sensor information. The basic idea is also that the team is trying to solve a task together that none of the robots can solve individually. In the cooperative case the robots do not necessarily exchange information online, but can use external computer power to find a batch solution of the joint data. Usually the robots can operate individually but use information from each other to perform better. Some early work in this area is presented in [41] where the topic of Cooperative Localization and Mapping (CLAM) is covered.

The goal of the work is to show that multiple robots will in fact increase the accuracy of the resulting map, compared to using a single robot. Furthermore, the experiments indicate that the quality is sensitive to the chosen exploration strategy. This issue is further discussed in [42] where the lower performance, uncertainty, bound for a group of $n$ vehicles is derived. A method for determining the number of vehicles needed to construct a map to a desired accuracy is also presented together with a framework for how to perform multi-robot CML. It should be noted that the association problem is not covered and the convergence time may also be impractically large. Further, the improvement of cooperative localization accuracy per additional robot for large teams is shown in [43]. It is proven that the uncertainty for the team is inversely proportional to the number of robots though each additional robot follows a law of diminish-
ing return. It is also made conclusive that the uncertainty growth for the group depends only on the number of robots and the odometric and orientational uncertainty and not the accuracy of the relative position measurements. An extension to cover the analytical expressions for upper bounds of uncertainty for Cooperative Simultaneous Localization and Mapping (C-SLAM) is derived in [44]. A study of the properties of the Riccati recursion which describes the propagation of uncertainty through time, yields (i) the guaranteed accuracy for a robot team in a given C-SLAM application, as well as (ii) the maximum expected steady-state uncertainty of the robots’ landmarks, when the spatial distribution of features can be modeled by a known distribution. This, however, is not suited for an unknown environment where the distribution of landmarks is unknown. It also raises the concern of to what extent this is applicable with a more realistic, nonlinear, sensor model.

The multi-robot SLAM literature covers a number of different frameworks in the same way that the single robot case does. In [45] the Sparse Extended Information Filter from [25] is extended to the multi-robot case. The method works without initial correspondence and with landmark ambiguity. The method is based on aligning local maps from each robot into a single map. This is achieved by matching similar-looking local landmarks using a tree-based algorithm. This is paired with a hill-climbing algorithm that maximizes the overall likelihood by searching in the space of correspondence. A similar approach is further discussed in [46] where a decentralized SLAM algorithm for a team of collaborating vehicles is addressed. The focus is on how to communicate between the vehicles to acquire a joint map while coping with latency and limited bandwidth. The key idea is to represent maps in information form and communicate subsets of information, tailored to the available communication resources. It is shown that the communication scheme preserves consistency in the communicated information. The great benefits of using sparse representations of stochastic frameworks is undermined when closing large loops. Loop closing can cause radical increases in the density of the information matrix. In the single robot case this happens when a robot revisits areas, while in the multi-robot case this can happen any time information is shared within the team. It can therefore be beneficial to work with sub-maps of large environments as suggested in [IV]. This is the first known implementation of Collaborative Smoothing and Mapping (C-SAM). This approach uses rendezvous observations between robots to make local map alignments. The observations do not have to be synchronized nor do they have to be made in both ways. The approach is further studied in [VI] where initial experiments with the C-SAM method are presented for the two-robot case. A new algorithm for doing multi-robot mapping based purely on rendezvous observations is shown. The algorithm yields an estimate primarily to be used for eliminating spurious rendezvous observations but can also be used to initiate the linearization point when joining two maps. If more then two robots are studied, the same algorithm can be used to associate a set of rendezvous observations to a certain robot. This approach
can also be extended to use the Probability Density Function (PDF) of the rendezvous observations to decide whether a rendezvous results in enough information to actually perform a map-join or not. For this reason the method is very robust to false association. A similar approach is presented in [47] where rendezvous measurements between two robots are used to align the local coordinate systems. The rendezvous measurements are paired, one from each robot, requiring each measurement pair to be time-synchronized. If the maps overlap, duplicate landmarks are identified by calculating the Mahalanobis distance between them. The search for potential duplicate landmarks is done by means of nearest neighbor approach using a kd-tree. An indoor “hall-way experiment” is conducted using two robots. Due to the use of color cameras in the experiment, the association problem is basically overlooked. A similar approach with robot-robot measurement using a particle filter for the Multi-robot SLAM problem is presented in [48] which is derived from the Rao-Blackwellized particle filter described in [40]. The work initially extends the RBPF into the multi-robot case with all robots starting in the same pose. It is then extended into the case with unknown initial correspondence and it is assumed that the robots will eventually “bump into” one another to make relative pose estimates. It is noted that PF based mapping requires very accurate motion models; therefore, the raw odometry is supported with Laser-stabilized odometry in the conducted experiment. The work is further extended and thoroughly covered in [49].

A somewhat different approach to multi-robot mapping can be found in [50]. The manifold representation presented takes the two-dimensional maps out of a plane and onto a two-dimensional surface embedded in a higher-dimensional space. The key advantage of the manifold representation is that it is self consistent. It is shown that this representation does not suffer from the “cross-over” problem that planar maps exhibit in environments containing loops. Experimental results are included.

1.1.4 Formation and Deployment

Deployment is the change of relation between a robot and the surroundings while formation is the spatial relation between robots. Consequently, a robot group can not be deployed without a formation. However, in some cases the formation is more crucial than in others. Any type of deterministic formation for a number of robots requires cooperation. As mentioned earlier, this can be achieved through a group or team constellation.

Assume that a number of circular robots with a diameter $D$ are to move from one position $\{A\}$ to another position $\{B\}$ over obstacle-free terrain, see Figure 1.6(a). If all the robots deploy separately, without considering the other robots, there are likely to be some collisions. To avoid collisions, a typical formation constraint would be for each robot to keep a distance $d > D$ to any other robot. This can be solved by letting the robots operate in a group, as discussed in the section above. If the group is supposed to deploy through a
complex passage as shown in Figure 1.6(b) there is great risk of deadlock. This can be eliminated by performing resource allocation, such as Bankers Algorithm [51] or similar. However, this would require communication and therefore a team constellation. In most cases the formation constraints are dependent on external factors such as kinematics, sensor types, assigned task and terrain. In some cases even the non-collision constraint may not be of interest since the collision itself is a way for team members to communicate.

Formation control is essential for a robot group to be efficient. In [52] results for relative positioning maintenance using motor schemas, or primitive behaviors, are presented. The work is an extension from [53] into the multi-robot case. Four basic formations are studied together with different leading strategies. It is found that different formations should be considered depending on the choice of leader strategies. For a cooperative behavior like localization and mapping, a common reference frame for the group is needed. Such a framework is presented in [54] where a common x – y reference for a number of mobile robots under a number of constraints is presented. Among these, uncertainty of robot motion is ignored. The robots are also seen as point objects, not causing occlusion, for the eye-like sensor. On the other hand, it is not assumed to be possible to separate one particular robot from another during an observation and is therefore to some extent robust to the association problem. The special case of cooperative localization of three robots is covered in [55]. The work presents a model for the team geometry as well as a closed loop stable dilation-control strategy for the robots. This enables the user to scale the formation using only one parameter. Uncertainties for the formation are not discussed and therefore the dynamics of the system are ignored when proving stability.

The deployment problem is valid for both single robots and multiple robots. A thorough dynamic model of a three-wheeled mobile robot is presented in [56], together with a set of algorithms for path planning and path tracking for a single robot. The path planning uses an optimization algorithm based on robot geometric constraints while the trajectory tracker is based on a sliding mode
control algorithm. The technique is designed to work in an obstacle-free environment. Cooperative deployment with multiple robots is done with two sets of robots in [57]. Each set moves, one at a time, using the stationary set as landmarks. This enables both sets of robots to move in uncharted terrain and still do a better job, with lower uncertainty, than by using dead-reckoning from each individual robot. Simulations performed indicate great benefit from cooperation. An extension of this work is found in [58] with an enhanced algorithm. Both experiments and simulations are presented for three “optimum” moving strategies, or formations. Analytical error solutions are derived for all three solutions using weighted least squares. Deployment of a robot group in a dynamic environment is more challenging. A multi-robot planner which is assumed to work effectively in both static and dynamic environments is found in [59]. It is well-suited to non-holonomic platforms since mechanical constraints of the platform are integrated to generate only kinematically consistent plans. The speed of the actual planning is fast enough to handle real time and is demonstrated in experiments using up to 15 robots. When deploying multiple robots it is important to consider the sensor characteristics. In [60] a strategy for sensor deployment using mobile robots is presented. The method is basically a way to distribute laser scanners over an area. The algorithm is designed to maximize the total area covered by the robots’ sensors by deploying one robot at a time, while simultaneously maintaining line-of-sight contact with one another. This method can be useful when doing surveillance or similar.

In contrast to what was presented earlier, the work in [II] covers the topic of how the formation of a 3-robot team can and will effect the uncertainty of the robots during deployment. Simulated results show that in some cases it is beneficial to change the formation of the team during deployment. It is shown that the formation has an impact on the balance of the covariance matrix. By choosing different formations it is possible to minimize the uncertainty in, for example, position at the cost of uncertainty in orientation. This work is further extended in [III] where the same formations are used and further investigated in respect of what formation is better suited for what purpose. The problem of choosing formation based on a given balance of uncertainty is studied; what formation should be chosen if one is interested in low orientation uncertainty or position uncertainty respectively. How the formation should be chosen is dependent on robot kinematics and sensor characteristics. The results show that if one travels long distances with a bearing-only sensor the formation should be seriously considered. An interesting result is that it seems to be possible to stabilize the orientation uncertainty at a fixed level and then balance out all new uncertainty in position instead. In some cases this can be beneficial since the non-linear approximation of motion is mainly related to the orientation.
1.2 The Grand Vision

Mobile robots can be seen as an extension to include physical interaction in a computer network. Due to mobility the operating space is not necessarily limited to the robot’s physical size nor fixed to a given location. The rovers shown in figure 1.1 are a typical example where mobile robots increase the range of physical interaction for humans by means of a mechanical system. This will enable totally new situations where humans will be able to interact on distance on a totally different level than what is possible today.

A mobile robot can be very useful when collecting data since it can provide different points of view for the sensors. This is a powerful tool when working with information fusion. In many areas parallelism has become popular and has been shown to be more cost-efficient, faster and better performing than single operating machines. The main thrust in this thesis is the belief that in most situations a cooperating robot group, if used correctly, will perform better than each robot would individually. Using multiple robots will yield a better result than using a single robot since the distribution itself can be used as leverage when extracting information from the collected data.

Scalability and redundancy are two of the greatest benefits from working with multiple robots in contrast to single robots. The possibility to scale a system is also of great benefit especially from a cost perspective where investments can scale with the productivity. Since the redundancy eliminates a weakest link, a multi-robot system is more robust to production interference than a single robot system. When the networking idea is applied to mobile robots the result is mobile robot teams. Mobile robot teams are for the robot industry what Wireless Local Area Networking, WLAN, was for the PC industry. It not only enables distributed information collection but also distributed physical interaction.

The possibility to mix different types of robots will generate great benefits, especially in a sensing and fusion perspective. The scalability yields the possibility to extend an existing team with new features as time goes on. It is not necessary to redesign the single robot from scratch only because a new sensor device or other new features are available. This results in faster time to market and less initiation cost for new functionality of a robot team.

1.3 The Structure of the Thesis

The following chapters will explain the frameworks, methods and models used in the appended papers. In chapter 2 two different frameworks for how to store spatial information for mobile robots are described. The first framework is used in papers [I–III] while the second is used in papers [IV–VI]. Chapter 3 describes the perception modeling. A general sensor representation is derived and its use in the different frameworks explained. The sensor models used in the appended papers are also derived and characterized. In chapter 4, the modeling
of robot motion is presented. A distinction is made between two categories of motion models together with a general representation. Also, the constant turn model used in all appended papers is fully derived. In the following chapters the authors’ contributions to the research field are explained together with possible extensions to the contributions. The final chapter, 7, summarizes all the appended papers and gives the reader some insight into what to expect from each of the papers.
Framework

Robot exploration is beginning to become a well studied area. Since the world is not ideal it will never be possible to find the true state of a mobile platform and/or objects in a surrounding environment. This is due to many different things such as slippage, tolerance in robot construction, tire pressure, modeling errors and measurement uncertainty, among others. One way of dealing with these errors is to reduce them to negligible limits, by tailoring the working environment, using high precision sensors and manufacturing techniques. It is also necessary to individually calibrate each robot and sensor-object combination. However, this totally fails the idea of having a robot team trying to navigate in an unknown environment. Even if a well known environment is used, there will still be uncertainty related to approximations made in the motion models and sensor measurements. A better way of dealing with the problem is to incorporate the spatial uncertainties in the representation of the states.

Equation (2.1) presents a general description of a non-linear motion model and observation model in discrete time, used in this work.

\[
\begin{align*}
\mathbf{x}(k) &= f(\mathbf{x}(k-1), \mathbf{u}(k)) + \mathbf{w}(k) \\
\mathbf{z}(k) &= h(\mathbf{x}(k)) + \mathbf{v}(k)
\end{align*}
\]  

(2.1)

The system states \( \mathbf{x}(k) \) are based on the earlier states \( \mathbf{x}(k-1) \) and the input signal \( \mathbf{u}(k) \) applied between these two samples. Since the input signal is a measurement provided by proprioceptive sensors it has the same timing index as the prediction. \( \mathbf{w}(k) \) is some external noise that is applied directly onto the system states. For the exteroceptive measurements made by different sensors to be fused in the same framework, these are transformed into an observation. The observation represents the states for a certain landmark in the framework. A landmark observation \( \mathbf{z}(k) \) is dependent on the state where the observation is made. Therefore, the observation model \( h(\cdot) \) is introduced, where \( \mathbf{v}(k) \) is the observation noise. These models can be simple or complex, depending on
what the user is interested in. In some cases one can use a linear model to
describe a non-linear system, although in others this is not good enough. On
the other hand, it may not always be the best idea to have a complex model
since that will increase computational cost. This is one of the major problems
when dealing with mobile robots; the balance between computational cost and
complexity.

As discussed in chapter 1, many different frameworks have been implemented
and used successfully for sharing information amongst robots. This chapter
presents two different methods used in the appended papers for storing spatial
information collected by a robot team. Section 2.1 focuses on “The Stochastic
Map” initially introduced in [10]. This method of storing spatial information
is used in the first three appended papers [I–III]. The method is well adopted
in the research field and is common in other published work [11, 15, 61, 62].
For a better understanding of the appended papers, the complete framework is
presented.

In the last three appended papers, [IV–VI], a different method for storing
the spatial information is used. This was introduced in [34] and is referred to
as Smoothing and Mapping (SAM). Instead of a filter, a smoother is used to
fuse data from different robots.

2.1 Stochastic Mapping

In this section the estimation problem is presented using the Kalman Fil-
ter (KF) approach [63], [64]. There are several other ways of estimating the
pose of a mobile robot and two of the most common are the Maximum Like-
lihood (ML) [65] and Particle Filter (PF) [37]. The latter has been used with
success in recent years and will most likely be more common in the future. The
basics of PF can be found in [22] where Monte Carlo simulation techniques are
used to solve optimization problems. Similar ideas are also presented in [66],
but not with the same depth. More recent work that addresses the problem
can be found in [67] and [68]. These approaches are not covered here since the
methods are not related to the KF.

This section presents a version of “The Stochastic Map” suited to systems dis-
cussed in this thesis. In section 2.1.2 uncertain spatial relationships will be tied
together in a representation called “The Stochastic Map”. It contains estimates
of the spatial relationships, their uncertainties, and their inter-dependencies.
The following section, 2.1.3, describes how new information is added to the
map while section 2.1.4 shows how the map is updated as the robots move.
Throughout the discussion it will be assumed that the objects are static and
do not move. The original representation of the map described how the map
is transformed into different viewpoints. That is not covered here since those
operations are not used in the appended papers. A more general and thorough
description of the method can be found in [10].
2.1.1 Kalman Filter

Kalman Filtering is a common method for performing state estimation. There are a number of different ways of doing Kalman filtering and this section will cover the most widely used approach when dealing with mobile robots. In this framework the KF is used to update map information by merging information from different observations. The updated estimate is a weighted average of the estimated observation and the predicted motion of the robot, where the weighting factor (computed in the weight matrix \( K \)) is proportional to the prior covariance in the state prediction, and inversely proportional to the conditional covariance of the observation. If the observation covariance is large compared to the state covariance, then \( K \to 0 \), and the observation has little impact when revising the state estimate. Conversely, when the prior state covariance is large compared to the noise covariance, then \( K \to I \), and almost the entire difference between the observation and its expected value is used in updating the state. There are many books describing different aspects of the Kalman filter. One of the most classic references is [64]. A recent collection of methods in the estimation field can be found in [69] and [70].

The most common filtering approach when dealing with mechanical systems is the Steady State Filter. This can be used to make estimations of a process that can be approximated to have linear dynamics. In these systems the state matrix does not need to be recalculated between the time steps. When mobile robots are being studied this is not an adequate assumption since the state matrix is heavily non-linear and historically dependent and therefore needs to be recalculated constantly. For this type of system there are other approaches that are more suitable, such as the Extended Kalman Filter (EKF).

**Extended Kalman Filter**

If the motion transition function \( f(.) \) and observation model \( h(.) \) in (2.1) are non-linear in the state variables, which is always the case when dealing with mobile robots and realistic sensor models, then \( f(.) \) and \( h(.) \) will have to be re-evaluated constantly. The Extended Kalman Filter, a sub-optimal non-linear estimator can be used in such cases. It is one of the most widely used non-linear estimators because of its similarity to KF, the optimal linear filter, its simplicity of implementation, and its ability to provide accurate estimates in practice. The problem of estimating a position of a mobile robot is heavily non-linear, although the noise can in many cases be approximated as Gaussian with some “stabilizing noise” added. Usually the system is also discrete, since the sensors and actuators are sampled with computers. The EKF for the discrete time is discussed in more detail in [16,64]. The EKF uses a locally linear approximation of the system. This linearization is usually done in each time step.
2.1.2 Map Representation

This section will present the “The Stochastic Map” with similarities to the representation in [10]. Some changes have been made to suit the purpose of this work better. In the general case, a state vector is described as a vector of \( n \) spatial variables:

\[
x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}
\] (2.2)

The idea of describing the position and orientation, the pose, of an object or mobile platform with spatial variables is well adopted. Since this work is presented in 2D/3DOF the state vector for a single robot is described by:

\[
x = \begin{bmatrix} x \\ y \\ \phi \end{bmatrix}
\] (2.3)

If more states are required to describe a pose one simply extends the state vector in (2.3) with the additional states. Because of the uncertainty in the states they will be presented as the first two moments of the probability distribution. The mean \( \hat{x} \) will represent an estimated pose and the corresponding covariance \( P(x) \) will represent the uncertainty of the states

\[
\hat{x} \triangleq E(x) \\
\tilde{x} \triangleq x - \hat{x} \\
P(x) \triangleq E(\tilde{x}\tilde{x}^T)
\] (2.4)

where \( E \) is the expectation operator, and \( \tilde{x} \) the deviation from mean \( \hat{x} \). In the system discussed, these are described by

\[
\hat{x} = \begin{bmatrix} \hat{x} \\ \hat{y} \\ \hat{\phi} \end{bmatrix}, \\
P(x) = \begin{bmatrix} \sigma_{xx} & \sigma_{yx} & \sigma_{x\phi} \\ \sigma_{yx} & \sigma_{yy} & \sigma_{y\phi} \\ \sigma_{x\phi} & \sigma_{y\phi} & \sigma_{\phi\phi} \end{bmatrix}
\] (2.5)

The diagonal elements of the covariance matrix are the variances of the spatial variables, while the off-diagonal elements are the covariances between the spatial variables. One can think of the covariances in terms of their correlation coefficients \( \rho_{ij} \):

\[
\rho_{ij} \triangleq \frac{\sigma_{ij}}{\sigma_i \sigma_j} = \frac{E(\tilde{x}_i \tilde{x}_j)}{\sqrt{E(\tilde{x}_i^2)E(\tilde{x}_j^2)}}, \quad -1 \leq \rho_{ij} \leq 1
\] (2.6)

To present the problem with multiple robots and objects a common reference frame is needed. If each robot, at a certain time-step, is seen as an object
with static uncertainty it is possible to create a topological map. The map is
described by the matrixes in equation (2.5) which need to be evaluated for all
robots and objects. Assume that the system, at a certain time, consists of \( m \)
robots and objects. Similar to a system of \( n \) uncertain spatial relationships, a
vector of all spatial variables is constructed in the same way as equation (2.2).
This vector is referred to as the system state vector and contains the system
matrixes for all robots and objects currently in the system. The size of this
state vector will be \( m \times n \), where \( n \) is the number of states in each of the system
vectors. The stochastic map is defined as the conditional mean estimate \( \hat{x} \)
of the state vector \( x \) and the corresponding system covariance matrix \( P \).

\[
\hat{x} = \begin{bmatrix} \hat{x}_1 \\ \hat{x}_2 \\ \vdots \\ \hat{x}_m \end{bmatrix},
P = \begin{bmatrix} P(x_1) & P(x_1, x_2) & \cdots & P(x_1, x_m) \\ P(x_2, x_1) & P(x_2) & \cdots & P(x_2, x_m) \\ \vdots & \vdots & \ddots & \vdots \\ P(x_m, x_1) & P(x_m, x_2) & \cdots & P(x_m) \end{bmatrix}
\]

(2.7)

where

\[
P(x_i, x_j) \triangleq E(\tilde{x}_i \tilde{x}_j^T)
\]

\[
P(x_j, x_i) = P(x_i, x_j)^T
\]

(2.8)

\( x_i \) is the vector of the spatial variables for each individual robot or object and
\( P(x_i) \) are the associated covariance matrices. The matrices \( P(x_i, x_j) \) represent
the cross-covariance between different objects or robots. These off-diagonal sub-
matrices encode the dependencies between the estimates of the different states
in the map.

### 2.1.3 Adding Information

In most situations when working with a single robot, the starting pose of the
robot is used as reference (origin) and all observed landmarks are then related to
that state. In a multi-robot environment some common reference frame between
the robots is necessary. This can be chosen in different ways but a common
solution is to define a “master” robot whose starting point is used as origin.
This requires great care when adding new robots to the team.

The following section discusses how new information is added to the map.
This can be done in several ways, depending on how the information was ob-
tained. The first part covers the problem of adding totally new objects or robots.
If a priori information is available, such as initial correspondence between the
robots, this should be added during the initiation of the map. The final section
covers the robot motion. As the robots move around and make measurements
in the partially known environment the information contains new constraints
about existing objects or robot states.
New Feature

Initially the system state vector is empty; the known robots and objects therefore need to be added before any updates can be made. A priori information has to be inserted into the estimated state vector $\hat{\mathbf{x}}$ and the covariance matrix $\mathbf{P}$ correctly as shown in equation (2.9).

$$\hat{\mathbf{x}}^G_+ = \begin{bmatrix} \hat{\mathbf{x}}^G \\ \hat{\mathbf{x}}^G_{n+1} \end{bmatrix}$$

(2.9)

$$\mathbf{P}(\hat{\mathbf{x}}^G_+) = \begin{bmatrix} \mathbf{P}(\hat{\mathbf{x}}^G) & \mathbf{M}^T \\ \mathbf{M} & \mathbf{N} \end{bmatrix}$$

where

$$\mathbf{N} = \mathbf{P}(\hat{\mathbf{x}}^G_{n+1})$$

$$\mathbf{M}_i = \mathbf{P}(\hat{\mathbf{x}}^G_{n+1}, \hat{\mathbf{x}}^G_i) = 0 \quad \forall i \in \{1 \ldots n\}$$

(2.10)

The state vector $\hat{\mathbf{x}}^G$ relates to the global coordinate frame $\{G\}$. All information being inserted needs to be presented in the same frame. If a priori information is already described in $\{G\}$ it can be inserted directly into the system as described in equation (2.10), given that the new and existing states are independent.

If new information is not in the global reference it has to be transformed before being added. The same procedure is used when a new object or robot occurs, observed by an existing robot. For such observations a robot is seen as an object. In equation (2.3) the states for each object are presented. When a robot observes a new object for the first time, the pose will be represented in the frame of the observing robot $\{i\}$. Equation (2.11) describes the procedure of transforming the new object from $\{i\}$ into $\{G\}$.

$$\hat{\mathbf{x}}^G_{n+1} = \hat{\mathbf{x}}^i_{n+1} + \mathbf{R}(\phi) \tilde{\mathbf{z}}^i_{n+1}$$

(2.11)

where

$$\mathbf{R}(\phi) = \begin{bmatrix} \cos(\phi) & -\sin(\phi) & 0 \\ \sin(\phi) & \cos(\phi) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

and $\tilde{\mathbf{z}}^i_{n+1}$ is the observed relation between the robot and the object. The corresponding observation covariance $\mathbf{\Sigma}$ is also represented in $\{i\}$ and therefore needs to be transferred into $\{G\}$ using equation (2.12). The observation will correlate the object with the robot. Since the uncertainty of the observing robot has correlations to the map, the new object will also be correlated to the same
states.

\[
M = \mathbf{GP}(\hat{x}^G) \\
N = \mathbf{GP}(\hat{x}^G)\mathbf{G}^T + \mathbf{R}(\hat{\phi}_i)\Sigma\mathbf{R}(\hat{\phi}_i)^T \\
G = \begin{bmatrix} \cdots & \frac{\partial x^G_{n+1}}{\partial x^G_i}(\hat{x}^G_i) & \cdots \end{bmatrix}, i = 1 \ldots n
\]

(2.12)

where \(M\) describes the cross-correlation between the states of the new object and the states of the existing objects and robots. \(N\) represents the internal covariance and correlations for the new states in \(\{G\}\). The results from equation (2.12) and (2.11) are then inserted into (2.9). The observation covariance \(\Sigma\) is different for each sensor and needs to be derived for each sensor type, further described in chapter 3.

New Constraints

When robots make observations of objects already in the map there is no need to expand the state vector and covariance matrix. Only the new information needs to be integrated into the existing system states. It should be noted that new constraints can be added even though no measurement is made. It can, for example, be information about how objects are physically linked, and therefore heavily correlated, or any other type of constraint that is known from the environment. Both cases are mathematically similar. However, in this work only observations based on measurements are considered. To read more about this please see [10].

For each sensor type there is an observation model. In chapter 3 the observation model is derived together with a discussion of the characteristics for a number of different sensor types. The information gained from all sensor measurements will automatically be distributed among the members, due to correlation within the map. It is assumed that the observation is made in Cartesian coordinates relative to the measuring robot’s coordinate frame \(\{i\}\). How this is accomplished for a sensor measuring in polar coordinates is discussed in section 3.1.

Assume that robot \(i\) at pose \(x^G_i\) makes an observation \(z^j_i\) of an object at a pose \(x^G_j\).

\[
z^j_i = h(x^G) + \mathbf{v}
\]

(2.13)

where

\[
E[\mathbf{v}(k)] = \mathbf{0} \\
E[\mathbf{v}(k)\mathbf{v}(k)^T] = \Sigma
\]
Given the observation model, the conditional mean estimates of the sensor pose and their uncertainties, and an actual observation, we can update the state estimate using the EKF equations.

\[
\hat{x}^G(k|k) = \hat{x}^G(k|k-1) + K \left[ z_i(k) - h(\hat{x}^G(k|k-1)) \right]
\]

\[
P(x^G(k|k)) = P(x^G(k|k-1)) - KHP(x^G(k|k-1)) \tag{2.14}
\]

\[
K = P(x^G(k|k-1)H^T \left[ HP(x^G(k|k-1))H^T + \Sigma(k) \right]^{-1}
\]

where \( H = \nabla h \) is the Jacobian of the observation model \( h(.) \). The observation covariance \( \Sigma \) is directly related to the actual sensor making the measurement.

### 2.1.4 Moving Robots

If any of the robots move between two time steps it is necessary to update the spatial information in the map. This section will discuss how to perform such an update. In chapter 4 the transition function equations used to calculate the kinematic motion for each individual robot are presented. Due to different kinematics the robots in the team do not necessarily all have the same transition function or noise parameters. Assume that the motion of robot \( R \) between two time steps can be described as

\[
x^G_R(k) = f(x^G_R(k-1), u(k)) + w(k) \tag{2.15}
\]

The errors of the motion will be different for each robot and heavily dependent on the transition function. The following assumptions are made about the errors induced into the system:

\[
E[w(k)] = E[\tilde{u}(k)] = 0, \forall k
\]

\[
E[w(i)w^T(j)] = \delta_{ij} Q_w(i)
\]

\[
E[\tilde{u}(i)\tilde{u}^T(j)] = \delta_{ij} Q_u(i)
\]

\[
E[w(i)\tilde{u}^T(j)] = 0, \forall i,j \tag{2.16}
\]

where \( \delta \) is the dirac delta.

\[\delta_{ij} = 1, i = j \]

\[\delta_{ij} = 0, i \neq j \tag{2.17}\]

The motion prediction is derived from the existing state estimate and the current input vector \( \tilde{u}(k) \).

\[
\hat{x}^G_R(k|k-1) = f(\hat{x}^G_R(k-1|k-1), \tilde{u}(k)) \tag{2.18}
\]

This is then used to update the predicted state vector \( \hat{x}^G(k|k-1) \). The state vector is not expanded since no new spatial variables need to be inserted, only
updates of the existing ones.

\[
\hat{x}^G(k|k-1) = \begin{bmatrix}
\vdots \\
f(\hat{x}^G_R(k-1|k-1), \hat{u}(k)) \\
\vdots 
\end{bmatrix} 
\]  

(2.19)

Since the robot uncertainty changes as the robots move, the covariance also needs to be updated. This procedure is further explained for each of the robots in chapter 4. However, when the robots work as a team they are correlated to each other and objects through earlier observations. These are represented as cross-covariances in the map, as explained in section 2.1.2. The models are presented in such a way that the process noise is uncorrelated with the robot states. Therefore, the simplified equation (2.20) can be used to update the covariance matrix.

\[
P(\hat{x}^G(k|k-1)) = \begin{bmatrix}
\vdots \\
M_i^T \\
\vdots \\
\cdots M_i \cdots N \cdots \\
\vdots 
\end{bmatrix}
\]

where

\[
M_i = P(\hat{x}^G_R(k|k-1), \hat{x}^G_i(k-1|k-1))
= F_x P(\hat{x}^G_R(k-1|k-1), \hat{x}^G_i(k-1|k-1)), \forall i \neq R
\]  

(2.20)

\[
N = P(\hat{x}^G_R(k|k-1))
= F_x P(\hat{x}^G_R(k-1|k-1)) F_x^T + G_u Q_u(k) G_u^T + Q_w(k)
\]

\(F_x = \nabla f_x\) and \(G_u = \nabla f_u\) are the Jacobians of the motion model \(f(\cdot)\), further explained for different motion models in chapter 4. Since objects are assumed to be stationary there is no difference between a prediction and an estimate. These states can therefore be transferred directly to the prediction vector.

### 2.2 Smoothing and Mapping

In the final three appended papers a different stochastic framework is used. Instead of filtering, a smoothing approach is used to fuse information shared amongst robots in a team. This framework is based on Square Root Information Smoothing (SRIS) and since it is applied onto the feature based mapping problem it is in the original implementation, [33], referred to as Square Root...
Smoothing and Mapping, or simply √SAM, later referred to as SAM. The original implementation was applied onto the single robot case and has been extended to be used with multiple robots in [IV] and is referred to as Collaborative Smoothing and Mapping (C-SAM).

In contrast to raw feature matching, such as in [25], the C-SAM approach is developed to fuse map information by initially aligning the robot trajectories and then matching features. By doing this the computation burden for feature matching does not become as high. The SAM framework is well suited for this purpose since it has the inherent property of extracting the robot trajectory. This is the main reason for choosing SAM as a base for this work. Like many other frameworks it also scales linearly with time. In C-SAM a base node is introduced to represent an estimate of global pose difference between two robot coordinate systems, or reference frames. This node represents the transformation between the two coordinate systems. The base node estimation presented in section 2.2.4 also apply on other frameworks although robot trajectory estimates needs to be available.

2.2.1 Representation

SAM represents the SLAM problem as a belief net, or Bayesian network, in the same manner as many others, [28,29,71]. The SLAM problem is described as finding the Maximum a Posteriori (MAP) estimate for an entire trajectory.
\[ X = \{ x_i \} \] and the mapped landmarks \[ L = \{ l_j \} \], given the observations \[ Z = \{ z_k \} \] and the control inputs \[ U = \{ u_i \} \], see figure 2.1.

\[
P(X, L, Z) = P(X_0) \prod_{i=1}^{M} P(x_i|x_{i-1}, u_i) \prod_{k=1}^{K} P(z_k|x_{i_k}, l_{j_k})
\]

(2.21)

\[
\theta^* \overset{\Delta}{=} \arg \max_\theta P(X, L|Z) = \arg \min_\theta - \log P(X, L, Z)
\]

(2.22)

where all unknowns \( X \) and \( L \) are collected in \( \theta \Delta = \{ X, L \} \). As for the prior \( P(x_0) \), it is assumed that \( x_0 \) is given. This is often an adequate assumption since the origin of the coordinate system is arbitrary and therefore we can just as well fix \( x_0 \) at the origin.

The MAP is found by minimizing the Mahalanobis norm for all nodes in the network by solving the following non-linear least squares problem:

\[
\theta^* = \arg \min_\theta \left\{ \sum_{i=1}^{M} \| f_i(x_{i-1}, u_i) - x_i \|_\Lambda_i^2 + \sum_{k=1}^{K} \| h_k(x_{i_k}, l_{j_k}) - z_k \|_\Sigma_k^2 \right\}
\]

(2.23)

In practice a linearized version of the problem is considered. Therefore we assume that these models provide a good linearisation point or that we are working on one iteration of a non-linear optimization method, such as Gauss-Newton or Levenberg-Marquardt. The observation prediction model \( h(.) \) is the same as derived in section 3.2 while the motion prediction model \( f(.) \) is derived in section 4.2. When using the linearized motion and observation models the optimization problem from (2.23) becomes:

\[
\delta^* = \arg \min_\delta \left\{ \sum_{i=1}^{M} \| F x_{i-1} - G_{x_i} \bar{x}_{i-1} - a_i \|_\Lambda_i^2 + \sum_{k=1}^{K} \| H_{k} x_{i_k} + J_{k} l_{j_k} - c_k \|_\Sigma_k^2 \right\}
\]

(2.24)

\[ \delta \overset{\Delta}{=} \{ \bar{X}, \bar{L} \} \]

where \( a_i \overset{\Delta}{=} x_{i-1}^0 - f_i(x_{i-1}^0, u_i) \) is the motion prediction error and \( c_k \overset{\Delta}{=} z_k - h_k(x_{i_k}^0, l_{j_k}^0) \) is the observation residual. Note that \( G_{x_i} \) is not the same Jacobian as normally derived for the nonlinear motion, shown in section 4.2, but the equivalent for when the input signal is transformed into state variables. In this case \( G_{x_i} = -I_{d \times d} \), with \( d \) the dimension of \( x_i \).

The covariance matrices \( \Lambda_i, \Sigma_k \) can be dropped from this point by performing a variable change. With \( \Sigma^{-1/2} \) being the matrix square root of \( \Sigma^{-1} \) it is possible to rewrite the Mahalanobis norm as:

\[
\| e \|^2 \overset{\Delta}{=} e^T \Sigma^{-1} e = (\Sigma^{-T/2} e)^T (\Sigma^{-T/2} e) = \| \Sigma^{-T/2} e \|_2^2
\]

(2.25)
Multi-robot Information Fusion

This yields the possibility to eliminate the covariance from (2.24) by pre-multiplying the Jacobians and the prediction error with the corresponding \textit{transposed matrix square root of the inverse covariance}. From this point we assume that the Jacobians \( F_{x_i}^{-1}, G_{x_i} \) and the prediction error \( a_i \) are pre-multiplied by \( \Lambda^{-T/2} \). A similar operation holds for the observation Jacobians \( H_{k}^{i}, J_{k}^{i} \) and the residual \( c_k \) to eliminate the observation covariance \( \Sigma \). When this is done the Jacobian matrixes can be collected into an observation matrix \( A \) while the prediction error vectors \( a_i \) and \( c_k \) are put into a Right-Hand Side (RHS) vector \( b \). The following standard least squares problem is obtained:

\[
\delta^* = \arg\min_{\delta} \| A\delta - b \|_2^2
\]  

(2.26)

For more information about the background and calculation cost of SAM we refer the reader to [33].

\subsection{2.2.2 Adding Information}

Adding information to the SAM framework is easy and cost-effective. All types of information are added in essentially the same way, by extending the observation matrix with a new column and/or row. The difference is in-between what nodes, or states, the information is constraining. The observation matrix \( A \) quickly becomes large since it grows linearly with the number of observations and robot poses. On the other hand, if topological loops are avoided it will remain sparse over time, see figure 2.2, which will have a positive effect on the calculation resources needed for solving the least squares problem.

The size of \( A \) is directly related to the dimension of robot and landmark states. If dimension of robot states is denoted, \( d_x \), landmarks, \( d_l \) and observation \( d_z \) the size of \( A \) will be \((Nd_x + Kd_z) \times (Nd_x + Md_l)\). Also, \( A \) has a typical block structure and if \( M = 2 \), \( N = 3 \) and \( K = 4 \) the following observation matrix with the corresponding RHS is obtained:

\[
A = \begin{bmatrix}
G_{x_1} & F_{x_1} & 0 & 0 \\
F_{x_2} & G_{x_2} & 0 & 0 \\
F_{x_3} & G_{x_3} & 0 & 0 \\
H_1 & J_1 & 0 & 0 \\
H_2 & J_2 & 0 & 0 \\
H_3 & J_3 & 0 & 0 \\
H_4 & J_4 & 0 & 0 \\
\end{bmatrix}, \ b = \begin{bmatrix}
a_1 \\
a_2 \\
a_3 \\
c_1 \\
c_2 \\
c_3 \\
c_4 \\
\end{bmatrix}
\]  

(2.27)

where \( F_{x}, G_{x}, H \) and \( J \) are the motion and observation Jacobians pre-multiplied by the corresponding \textit{transposed matrix square root of the inverse covariance}. The same pre-multiplication holds for the RHS vector \( b \). It should be noted that it is not necessary to add a new robot pose node for each sampled pose, unless new sensor information is retrieved. In such cases one may just as well bundle predictions, as long as the linearization is eligible.
Figure 2.2: (Left) The sparsity pattern of the observation matrix $A$ with observation and motion Jacobians of one robot from an experiment conducted in [VI]. (Right Top) The information matrix $I = A^T A$ of the same observation matrix. (Right Bottom) The information matrix has been decomposed by cholesky decomposition to a triangular matrix to speed up calculations when solving the least squares problem in (2.26).
Figure 2.3: The two robots each have a local coordinate system \( \{G\} \). By using a set of rendezvous observations \( z_{j,m} \) between robot poses \( b_{i,p} \) can be estimated, representing the link between the two maps.

2.2.3 Collaborative Smoothing and Mapping

Collaborative SAM is a straightforward extension to SAM, introduced in [IV]. Extended work with some real experiments is found in [VI]. C-SAM is a method for alignment and joining of map information from two or more robots storing information using the SAM structure. The robots need to be working as a team, sharing trajectory information, for map alignment. If landmarks are shared the maps can be joined as well. The solution is based on re-using each individually optimized observation matrix together with information gathered during a rendezvous. No initial correspondence is assumed, although if an estimate of the transformation between the two reference frames is available it can be used as a prior. The rendezvous consists of relative pose observations, rendezvous observations, between robots together with the poses from where the observations were made, rendezvous poses. It is necessary to synchronize the estimated poses with the rendezvous observations. This can be accomplished in many different ways, but since the robots will be communicating information a simple method is to use the communication link to synchronize the clocks. The time synchronization makes it possible to associate an observation made by one robot with the pose of the other, using a time stamp. Although the robots are assumed to be time-synchronized there are likely to be errors in time synchronization. How this synchronization error affects the motion estimate is further discussed.
in section 4.3.3. Also, this does not eliminate possible spurious or false measurements. It is important to note that these observations do not necessarily have to be in both directions. On the other hand, they help to eliminate spurious measurements and give more information when performing map alignment, decreasing the possibilities of false association.

As mentioned earlier the transformation between robot maps can be available as a prior, otherwise the base node is estimated purely from the rendezvous observations. Two maps are said to be aligned when the base node optimization performed over the rendezvous observations has converged. Since the map alignment is represented by the base node it is easy to transform information in between the robot local sub-maps. When the transformation is available one can choose whether to integrate the new information or keep the information intact as a sub-map without losing any computational power.

This work is done in 2D/3DOF and therefore the base node is represented as a translation with a single orientation:

$$b^p_q = \begin{bmatrix} x^p_q \\ y^p_q \\ \phi^p_q \end{bmatrix}$$ (2.28)

The base node is evaluated from the perspective of one robot and the reverse relation can be calculated as:

$$b^q_p = -R^{-1}(\phi^p_q)b^p_q$$ (2.29)

The C-SAM least squares problem is solved in the same manner as for a single robot. Before any map information is shared it needs to be locally optimized by solving (2.26). $A^p$ and $A^q$ are the observation matrixes from each of the two robots to be aligned. Each robot $p, q$ has its own map represented in the local coordinate frames $\{G^p\}$ and $\{G^q\}$ respectively. Both robot maps are rearranged in a block matrix structure into a complete observation matrix $A'$. The C-SAM observation matrix then becomes:

$$A' = \begin{bmatrix} A^p & 0 & 0 \\ 0 & A^q & 0 \\ C^p & C^q & B^p_q \end{bmatrix}$$ (2.30)

where the last row $[C^p \ C^q \ B^p_q]$ consists of three block matrixes containing observation information from the rendezvous. This last row in the observation matrix is later referred to as the connector. A complete sparsity pattern of the C-SAM observation matrix can be seen in figure 2.4.

The rendezvous observations are added to the connector in the same way as regular observations are added to the observation matrix. However, since these observations span over the base node a new observation model is introduced. This observation model will add a new optimization constraint to the goal
function. The final goal function for C-SAM then becomes:

\[ \theta^* = \arg\min_{\theta} \left\{ \sum_{i=1}^{M} \left\| f_i(x_{i-1}^p, u_i^p) - x_i^p \right\|^2_{\Lambda_i} + \sum_{k=1}^{K} \left\| h_k(x_{k_i}^p, l_{k_i}) - z_{k_i}^j \right\|^2_{\Sigma_k} + \sum_{m=1}^{N} \left\| c_m(x_{im}^p, b_{qm}, x_{jm}^q) - z_{jm}^i \right\|^2_{\Sigma_m} \right\} \]  

(2.31)

where \( \theta \Delta \{ X^P, L^P, b_i^P, X^q \} \) are optimization variables and \( x_{jm}^q \) are pose estimates for the robot being observed during the rendezvous. The observation model \( c(.) \) is used for predicting the rendezvous observation. The observation model can be derived from geometric reasoning in figure 2.3.

\[ c_m(x_{im}^p, b_{qm}, x_{jm}^q) = R(\phi_{im})^{-1}(b_{qm} + R(\phi_{jm})x_{jm}^q - x_{im}^p) \]  

(2.32)

The observation residual for a single rendezvous observation, \( z_{jm}^i \), then becomes:

\[ n_m \Delta z_{jm}^i - c_m(x_{im}^p, b_{qm}, x_{jm}^q) \]  

(2.33)

The information from each rendezvous observation is stored in the connector, where each row is typically:

\[ C_m = \begin{bmatrix} C_{im}^p & \ldots & C_{jm}^q & \ldots & B_{qm}^p \end{bmatrix} \]  

(2.34)

\( C_{im}^p, C_{jm}^q \) and \( B_{qm}^p \) are the corresponding Jacobians of (2.32) pre-multiplied by the transposed matrix square root of the inverse observation covariance \( \Sigma_m \).

It should be noted that if computer resources become an issue it is not necessary to include the complete observation matrices from both robots, but only the rendezvous poses and rendezvous observations. This will on the other hand not allow complete map joining to be done but is most often used as an indication of to what extent further fusion is useful or not.

### 2.2.4 Base-node Estimate

This section describes a proposed method for finding an initial estimate of the base node, given a set of rendezvous observations. If bad or non-consistent observations happen to be inserted into the connector, major problems will occur. This is because the smoother uses all observations to solve the least squares problem, possibly causing the solution not to converge. If perfect association between the robots can be assumed this would not be an issue, but in a real scenario this is not a valid assumption. In [VI] some real experiments have been conducted and the data set includes many false rendezvous observations. Therefore, it is of great interest to eliminate false and spurious measurements at an early stage, preferably before any optimization is done. For this purpose a base node estimation was developed. This algorithm can be applied to the...
Figure 2.4: (Left) The observation matrix $A'$ for a C-SAM solution is illustrated by a sparsity pattern. It looks somewhat different than a regular $A$ matrix. The most obvious difference is the connector, located in the bottom part, connecting the block matrices from each robot. (Top right) The information matrix has lost some of its sparsity due to the connector, correlating poses and landmarks from both maps. (Bottom right) The information is shown decomposed using cholesky decomposition. Note the dense areas over the rendezvous.
Multi-robot Information Fusion

rendezvous observations online, directly when the measurement is made, and is therefore well suited for both sequential filtering methods like EKF or batch solutions like the C-SAM. The test is meant to eliminate inconsistent observations and estimate a base node based only on the consistent ones. This estimate can then be used as an a priori for further fusion using C-SAM or similar.

Each of the rendezvous observations is evaluated to find a corresponding base node. This is done by formulating a “mini” SAM problem. A local observation matrix \( O_m \) is created from the covariances of the rendezvous poses related to the rendezvous observation together with the connector related to observation \( m \).

\[
O_m = \begin{bmatrix}
-M_{i_m}^p & 0 & 0 \\
0 & -M_{j_m}^q & 0 \\
C_{i_m}^p & C_{j_m}^q & B_{j_m}^p \\
\end{bmatrix}
\]

(2.35)

where \( M_{i_m}^p \) is the matrix square root of the inverse sub-matrix covariance related to poses \( i \) while \( M_{j_m}^q \) is the corresponding covariance for poses \( j \). Observation Jacobians \( C_{i_m}^p \) and \( C_{j_m}^q \) are the same as in (2.34). The RHS vector \( o_m \) is constructed in the same way as in a regular SAM with the observation prediction error pre-multiplied by the transposed matrix square root of the inverse covariance. The motion prediction error from the local sub-maps from each robot is reused.

\[
o_m = \begin{bmatrix}
a_{i_m}^p \\
a_{j_m}^q \\
b_{j_m}^p \\
\end{bmatrix}
\]

(2.36)

Then the optimization problem is solved for a base node using the same least squares optimization as in a regular SAM:

\[
\delta^* = \arg\min_{\delta} \| O_m \delta - o_m \|_2^2
\]

\[
\delta = \{ \tilde{x}_{i_m}^p, \tilde{x}_{j_m}^q, \tilde{b}_{j_m}^p \}
\]

(2.37)

This yields a base-node estimate based on a single observation only. After the optimization is completed the resulting covariance matrix for each observation pair is retrieved by:

\[
P_m = (O_m^T O_m)^{-1}
\]

(2.38)

When all rendezvous observations have been evaluated the estimated base nodes will be spread, not point towards the same solution. Therefore, a simple consistency test is carried out where the median, \( \mu_{1/2}(b_q^p) \), of all base nodes is calculated and then tested against each of the solutions, see figure 2.5. This is accomplished by calculating the distance between each solution and the median, weighted with the covariance of the base node prediction.

\[
D_m = (\mu_{1/2}(b_q^p) - b_{j_m}^p)^T P_m^{-1} (b_{j_m}^p - \mu_{1/2}(b_q^p)) (\mu_{1/2}(b_q^p) - b_{j_m}^p)
\]

(2.39)

The solution is accepted when \( D_m \) is less than a given threshold \( \xi_2 \). This results in the observation \( m \) being consistent and therefore accepted to be included in
Framework

Figure 2.5: (a) The final set of base node solutions for the rendezvous measurements that passed the test and are accepted to be used in the connector. The joined average indicates the initial estimate of the base node to be used in the C-SAM optimization. (b)-(d) The same data set is projected onto the 2D plane, together with the $3\sigma$ covariance of the joined estimate.

the C-SAM observation matrix, equation (2.30). To retrieve a base-node estimate of all accepted rendezvous observations the joined estimate and covariance are calculated as:

$$
\mathbf{P}(b_q^p)^{-1} = \sum_{i=1}^{m_1} \mathbf{P}(b_q^p_i)^{-1}
$$

$$
b_q^p = \mathbf{P}(b_q^p)(\sum_{i=1}^{m_1} \mathbf{P}(b_q^p_i)^{-1} b_q^p_i)
$$

If simultaneous rendezvous observations are made between two robots, the uncertainty of the initial base node estimate can be reduced by joining the
two estimates, see figure 2.6. This is accomplished by using (2.29) to transform all estimates into the same robot frame and thereafter adding them up using (2.40).

The main reason for applying this method is to eliminate spurious rendezvous observations. It should be noted that the method is not an exact model, but a simple and conservative approximation to eliminate inconsistent base nodes. This helps us not to start a map alignment until we are certain it will be successful. Also, since the final estimator uses all information from the selected rendezvous observations to optimize the complete map the approximations made here do not influence the final results. The worst thing that can happen is that some valid rendezvous observations are discarded in the initial estimate. This is on the other hand not a great problem, since once the initial estimate is retrieved and a map alignment has been successfully conducted it is easy to start adding more rendezvous observations by simply checking the final consistency test against the base node estimate instead of the median as done initially.

The complexity of the trajectories will influence the result of this algorithm. More complex trajectories will benefit the results if they can be achieved without too much slipping, causing poor motion prediction. It is also strongly encouraged that the discarded rendezvous observations be retried against the estimated base node retrieved after an optimization is done. This is because there may possibly be candidates that may have been discarded on false premises and therefore are valid once the trajectories have been straightened up by the smoother.
Sensors are an essential part of a robotic system. The purpose of the sensors is to gather information about how the system behaves and how this corresponds to what is predicted using motion and observation models. Assume there are two types of sensors mounted on a robot: proprioceptive and exteroceptive. The first type is used to track movements of the robot using wheel encoders, gyros or similar. The information from these sensors is seen as input signals to the motion model described in chapter 4. This chapter will cover exteroceptive sensors used to make observations of the environment. It involves making observations of other group members as well as making relative measurements to landmarks in the surroundings.

3.1 Observation Model

All exteroceptive sensors discussed in this work make measurements in polar coordinates. For this reason the measurements need to be transformed into an observation in cartesian coordinates before being used in any of the frameworks presented in chapter 2. The work also applies to other sensor types as long as the observation is presented in the same way.

Figure 3.1 describes the variables involved in a polar measurement. By geometric reasoning it is possible to transform the measurements into an observation in cartesian coordinates as:

\[
\mathbf{z} = \begin{bmatrix} r \cos(\beta) \\ r \sin(\beta) \\ \alpha \end{bmatrix}
\]

(3.1)

The measured variables \( \beta, r \) and \( \alpha \) are assumed to include some measurement
error. These errors are collected in a vector $\gamma$

$$\gamma = \begin{bmatrix} \hat{r} & \hat{\beta} & \hat{\alpha} \end{bmatrix}^T$$  \hspace{0.5cm} (3.2)$$

The errors are expected to be zero mean and normally distributed with the corresponding variances $\sigma_r$, $\sigma_\beta$ and $\sigma_\alpha$ respectively.

$$E[\gamma] = 0$$

$$E[\gamma\gamma^T] = \Gamma$$  \hspace{0.5cm} (3.3)$$

Similar to equation (3.1) the covariance of the measurement needs to be transformed into observation coordinates. The observation covariance $\Sigma$ is calculated as:

$$\Sigma = \nabla_{r,\beta,\alpha} \Gamma \nabla_{r,\beta,\alpha}^T$$  \hspace{0.5cm} (3.4)$$

where $\nabla_{r,\beta,\alpha}$ is the Jacobian of (3.1) evaluated at $r = \hat{r}$, $\beta = \hat{\beta}$ and $\alpha = \hat{\alpha}$.

In many cases when implementing it is easier to rewrite $\Sigma$ as a function of the

---

**Figure 3.1:** A single range and bering measurement between one robot and another. The parameters are the same for measurement of an object or feature.
rotation matrix $R(\beta)$. If measurement errors are assumed to be uncorrelated
the observation covariance can be written as:

$$
\Sigma = R(\beta) \begin{bmatrix}
\sigma_r^2 & 0 & 0 \\
0 & r^2 \sigma_\beta^2 & 0 \\
0 & 0 & \sigma_\alpha^2
\end{bmatrix} R(\beta)^T
$$

(3.5)

where

$$
R(\beta) =
\begin{bmatrix}
\cos(\beta) & -\sin(\beta) & 0 \\
\sin(\beta) & \cos(\beta) & 0 \\
0 & 0 & 1
\end{bmatrix}
$$

(3.6)

The measurement errors are different for different types of sensors and are
presented separately for each type in section 3.3.

It is important to note that when the errors in $\gamma$ are transformed through
equation (3.3) they will no longer be strictly zero mean. The nonlinearities
will cause a bias in the observation and in some situations this can cause great
problems. For the sensor geometry studied here it is assumed to be an adequate
assumption to ignore this bias. The approximations made by describing an
observation in this way is not necessary for the estimation. However, to make
it easier to integrate different sensors into the frameworks this approach has
been chosen.

### 3.2 Observation Prediction

This section will derive a general observation model used as a guide for how to
derive innovation in sensor signals. Throughout this work an observation is seen
as the relative pose between a measuring robot and an observed landmark or
another robot. For simpler notation it is assumed that the landmark is currently
included in the state vector $x$. An observation $z$ is modeled as a function $h(.)$ between states in the state vector:

$$
z(k) = h(x(k)) + v(k)
$$

(3.7)

where $v(k)$ is an observation error with the following characteristics:

$$
E[v(k)] \approx 0
$$

$$
E[v(k)v^T(k)] = \Sigma(k)
$$

(3.8)

For the general case it is assumed that the observation is non-linear and a
Taylor expansion results in an adequately linearized model.

$$
h(x(k)) = h(\hat{x}(k|k-1)) + \nabla h_x \hat{x}(k|k-1) + H.O.T
$$

(3.9)

where $\nabla h_x$ is the Jacobian of $h(x(k))$ evaluated at $x(k) = \hat{x}(k|k-1)$. Since
$h(.)$ takes the state vector as argument, $\nabla h_x$ will be a vector containing the
partial derivatives from the object or robot being observed in that instant.

$$
\nabla h_x = \begin{bmatrix}
\cdots & \frac{\partial h(x)}{\partial x_j} & \cdots & \frac{\partial h(x)}{\partial x_i} & \cdots
\end{bmatrix}
$$

(3.10)
An observation $\hat{z}$ is predicted as:

$$\hat{z}(i|j) \triangleq E[z(i)|Z^j], \quad Z^j = \{z_1 \ldots z_j\} \quad (3.11)$$

where all the observations up to time $j$ are taken into account. If the Taylor series is truncated at the first order and all the observations up to time $k - 1$ are taken into account an observation prediction can be described as:

$$\hat{z}(k|k - 1) \approx E[h(\hat{x}(k|k - 1)) + \nabla h_x \hat{x}(k|k - 1) + v(k)|Z^{k-1}]$$

$$\quad = h(\hat{x}(k|k - 1)) \quad (3.12)$$

The observation error consists of two parts: the sensor error $v$ and observation model prediction error $\tilde{h}(\hat{x}(k|k - 1))$. This total error is usually referred to as the Innovation

$$\nu(k|k - 1) = z(k) - h(\hat{x}(k|k - 1)). \quad (3.13)$$

This error can be approximated by Taylor expansion. Note that the Taylor series is truncated at the second order.

$$\nu(k|k - 1) = z(k) - h(\hat{x}(k|k - 1))$$

$$\quad = h(\hat{x}(k|k - 1)) + \nabla h_x \hat{x}(k|k - 1) + v(k) + H.O.T$$

$$\quad - h(\hat{x}(k|k - 1))$$

$$\quad = \nabla h_x \hat{x}(k|k - 1) + v(k) + H.O.T \quad (3.14)$$

The Innovation is then used to calculate the total uncertainty of the observation, including both errors from the measurement and the errors in pose prediction of the robots that are transformed through the observation model.

$$S(k|k - 1) = E[\nu(k|k - 1)\nu^T(k|k - 1)]$$

$$\approx E[\nabla h_x \hat{x}(k|k - 1) + v(k)]^T (\nabla h_x \hat{x}(k|k - 1) + v(k))^T$$

$$\quad = \nabla h_x E[\hat{x}(k|k - 1)\hat{x}^T(k|k - 1)]\nabla h_x + E[v(k)v(k)^T]$$

$$\quad = \nabla h_x P(k|k - 1)\nabla h_x + \Sigma(k) \quad (3.15)$$

For clearer notation the substitution $H = \nabla h_x$ is made

$$S(k|k - 1) = HP(k|k - 1)H^T + \Sigma(k) \quad (3.16)$$

Since the observation prediction is always made within states of the state vector $x$ there may be cross-correlations hidden within the predicted covariance $P(k|k - 1)$ which need to be included in the calculations. It is also possible that multiple observations are made in the same instance; in such cases the observation will add constraints between the involved states, which will be seen in the innovation covariance, (3.16), which it is important to include in a possible map update.
3.2.1 Observation between Two Poses

Based on what is presented above we can derive the special observation function between two poses in the state vector. This is useful in many situations, especially when validating a new measurement, to see if the corresponding observation is consistent with the existing states or not. Assume $x_i$ is the pose from where the measurement is made and $x_j$ is the pose expected to be measured.

$$\dot{\hat{z}}_j(k|k-1) = h(\mathbf{\hat{x}}_i(k|k-1), \mathbf{\hat{x}}_j(k|k-1))$$

$$h(x_i, x_j) = R^{-1}(\phi)(x_j - x_i)$$

(3.17)

where

$$R(\phi) = \begin{bmatrix} \cos(\phi) & -\sin(\phi) & 0 \\ \sin(\phi) & \cos(\phi) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

(3.18)

The covariance also needs to transformed and it is especially important that the correlations between the states are included in the transformation. The covariance for the observation becomes:

$$P(\dot{\hat{z}}_j) = HP(\mathbf{\hat{x}})H$$

(3.19)

where $H$ is the Jacobian matrix from (3.10) and the necessary partial derivatives of (3.17) are derived as:

$$\frac{\partial h(x)}{\partial x_i} = -R(\phi_i)^{-1} \begin{bmatrix} 1 & 0 & -(y_j^G - y_i^G) \\ 0 & 1 & x_j^G - x_i^G \\ 0 & 0 & 1 \end{bmatrix}$$

$$\frac{\partial h(x)}{\partial x_j} = R(\phi_i)^{-1}$$

(3.20)

3.3 Characteristics for Common Sensors

Since all exteroceptive sensors discussed in this work make measurements in polar coordinates it is worth mentioning that there is a distortion when transforming from polar to cartesian coordinates. The linearized observation error is described by the ellipse spanned by the noise variables $\sigma_r$ and $\sigma_\beta$, see figure 3.2. If $\sigma_\beta$ is large a noticeable bias is seen in the transformation from measurement to observation. As described below the characteristics of the sensors used in this work indicate rather low $\sigma_\beta$ values. Ignoring this effect is therefore assumed to be adequate.

A comparison between vision and laser range scanners is presented in [72]. The performance of localization using a priori information with different sensors is studied. Not only precision and estimation error are studied but also robustness and the possibility to perform feature association. This is done for
Figure 3.2: All sensors discussed in this work make measurements in polar coordinates. The linearized observation error in cartesian coordinates is described by the ellipse spanned by the noise variables $\sigma_r$ and $\sigma_\beta$. If $\sigma_\beta$ is large a noticeable bias is seen in the transformation from measurement to observation.

Each of the localization processes. It is concluded that both sensor types can attain comparable precision levels. However, the vision systems contain more complex matching problems and therefore require more sophisticated solutions to make the process robust. It is also concluded that a priori maps usually contain information better suited for laser sensors, such as walls, in contrast to vision which only detects corners and doorframes. When working with feature detection the resulting measurement uncertainty is heavily dependent on the feature being observed. This is relevant for all types of sensors and some general information about sensor-object characteristics for a laser range scanner can be found in [7]. In other work different sensors have been combined to improve perception. In [73] an extrinsic calibration of a camera and laser range finder is presented. The method mainly improves the camera calibration.
### 3.3.1 Laser Scanner

![Figure 3.3: Scanning geometry for a laser scanner. Note that there is a limited reach for the laser beams and therefore only three individual distance measurements are made on the object.](image)

The Laser Range Scanner (LRS) has become a popular sensor among robotics researchers during the last few years, mainly because they are very robust and do not require highly advanced algorithms to extract spatial data. In some literature this sensor is also referred to as time-of-flight laser. It is well suited for making distance measurements in a two dimensional plane. The measurements are made by letting a single laser ranger make measurements in an arc, rotating around the sensor center, see figure 3.3. These sensors are used for many purposes such as obstacle avoidance, map building, and detection of safety-critical regions around autonomous vehicles, to detect whether a path is free or not.

The measurement errors are characterized by

$$\sigma_\beta, \sigma_r \propto \begin{cases} \text{Const}, & r \in \tau_{\text{range}} \\ \infty, & r \not\in \tau_{\text{range}} \end{cases}$$

(3.21)

where infinity represents out of range. The characteristic of this sensor is that it makes high accuracy distance measurements with $\sigma_r$ in the range of centimeters.

The limited range in distance measurements gives a built-in windowing effect. For a typical robot laser the distance is limited to tens of meters. The distance measurement is also dependent on the surface reflectance for where the laser beam hits the target. If parts of the laser beam hit a target, this causes jump edges, resulting in faulty range measurements.

The sensor is designed so that the laser beam of the laser ranger sweeps the plane in an arc. The laser ranger then makes distance measurements at discrete
angles as the beam is being swept. The bearing information for an observation depends on how many measurements are made in one sweep, which gives the resolution of bearing measurements. Therefore, $\sigma_\beta$ is moderate and in the range of tenths of milliradians. There is also a slight effect of beam divergence which is in the range of a milliradian.

### 3.3.2 Camera

This is the classic mobile robot sensor, used since the very earliest experiments, [74]. It is a fairly cheap sensor that can be used for many purposes. It has become even cheaper in recent years since methods for calibration have been developed that do not require the high tolerance optics. In [75] a calibration procedure for short focal length off-the-shelf CCD cameras was developed. This enables low cost USB cameras to be used for robot perception at a reasonably low cost. This work is later extended in [76] where a four-step camera calibration procedure with implicit image correction is presented. The main issues with using cameras are the cost in computational resources to handle the data. Image recognition and processing is a separate field of research and has made great progress in recent years, especially due to the increased amount of computer power.

Compared to a laser scanner the camera has a high angular resolution and therefore makes bearing measurements with low variance. The resolution is easily calculated as the number of pixels divided by the field of view $\sigma_\beta$ is in the range of milliradians. The camera noise parameter can be described as:

$$\begin{align*}
\sigma_r & \propto \infty \\
\sigma_\beta & \propto \text{Const}
\end{align*}$$

(3.22)

where infinity represents no information retrieved, since there is basically no distance information from one single camera measurement, unless some image processing is added together with object recognition. It is also possible to extract distance information by moving the camera and triangulating the features detected in the images. This is usually referred to as structure from motion, [77, 78].

In the appended work, the camera is only used to make bearing measurements between objects and robots. If the “stochastic map” framework is used, a minimum of two measurements are needed for a filter update, since no distance information is available. In this work, this is not an issue since a centralized framework is used with a robot team making simultaneous rendezvous observations, [II].

### 3.3.3 Stereo Camera

Since a monocular camera extracts only bearing information and needs some type of displacement or advanced feature recognition to be able to estimate...
depth, a natural evolution is to use two cameras with a given displacement. This differs from structure by motion in the sense that the images can be captured at the same time and that it is possible to calibrate the offset between the sensors.

The stereo camera has historically been a popular sensor since it works in a similar way to human eyes. By image processing one tries to correlate images from the two cameras and then triangulate using the bearing to the object from the cameras and the calibrated offset between the cameras. This is usually referred to as stereo vision. This method still requires heavy image processing and also good models of how the cameras are mounted. The calibration is critical since this will have a great impact on the result.

In [79] stereo vision uncertainty is modeled and put into the context of robot motion planning. The uncertainty model for stereo vision not only covers quantization errors but also false feature matching. A strategy for resolving ambiguous matchings is also proposed (data association). The motion planning minimizes the expected minimum total cost for reaching the destination. This is based on the possibility of correct matching and optimal observation point.

From a robot team perspective it is possible to retrieve stereo vision by mounting a camera on each robot. This is not the same as structure by motion since observations within the team will help out when solving the association problem. This will not necessarily be better than regular stereo vision, due to a worse estimate of the offset between the cameras. On the other hand it will provide different perspectives which, if the association problem is solved, can be of much greater value than low uncertainty.

The measurement characteristics for a stereo camera can be described as:

\[
\sigma_r \propto r^2 \\
\sigma_\beta \propto \text{Const}
\]

where the camera pair actually manages to extract distance information from the two simultaneous images. The uncertainty in the range measurement, \(\sigma_r\), is very much dependent on the resolution of the camera, the distance between the two cameras and the amount of computer power applied in the image processing. For the bearing measurement, it is in the same range as a single camera; \(\sigma_\beta\) is in the range of milliradians.
Multi-robot Information Fusion
Motion Models

Figure 4.1: Motion models are used for predicting pose changes and the corresponding uncertainty of the robot, based on current information. The ellipse represents uncertainty for the estimated robot position with the most likely position at the center of the ellipsoid, in each time step.

A motion model is a mathematical description of the kinematics and/or dynamics of a mechanical system. This work only deals with kinematic models for mobile robot platforms in a 2D/3DOF environment. However, the princi-
Multi-robot Information Fusion

... for deriving such a model for systems of higher dimensions is in general the same. Since the kinematics discussed are non-linear the general description will handle this case although it applies equally to linear systems. Motion models can be divided into two basic categories: holonomic and non-holonomic. Mobile platforms are inherently non-holonomic. The differences between these are further discussed in section 4.1. Throughout this work all motion models are presented directly in discrete form since in a computer based system all measurements are sampled discretely.

The motion model is mainly used for predicting pose change of the robot, see figure 4.1. It is also used for characterizing motion errors. This requires constant measurements to be made of proprioceptive sensors which are used as input signals to the motion model. Due to the non-holonomic nature of mobile robots it is not the pose itself that generates the uncertainty, but the path the platform has taken to reach the pose. The errors accumulate over time, as the platform moves, resulting in an uncertainty of the predicted pose. There are two major sources causing these errors. Firstly there is slippage in contact between wheels and surface. This error is difficult to model and much related to the surface itself. Another source of error is model imperfections. These are caused by approximations, incorrect model parametrization or errors in driving variables.

In section 4.2 a general representation of a moving robot is derived. In the following section, 4.3, the same model is presented in a 2D/3DOF environment together with three different error propagation models, used in the appended papers. Some comparisons are also made between the models and directions are given as to how to use them successfully.

4.1 Holonomic and Non-holonomic Systems

The dynamics of mechanical systems with ideal holonomic constraints was largely completed as of Lagrange’s monumental Mécanique analytique [80]. Systems with non-holonomic constraints were first described and named by Hertz [81]. The distinction between a holonomic and a non-holonomic system is whether the system output signals are historically dependent on the input signals. If the output signals are dependent on the order in which a set of input signals are applied the system is non-holonomic.

Holonomic kinematics can be expressed in terms of algebraic equations which constrain the internal joint angles, of an Industrial Robot, to the absolute position/orientation of the Tool Center Point (TCP). Non-holonomic kinematics are only expressible with differential relationships. This distinction has several implications for the implementation of a control system as well as the error propagation. In figure 4.2 a holonomic and a non-holonomic system are shown. Both systems have the same input signals, angle of the joints and angle of the wheels, respectively. As a holonomic system returns to the original internal (joint) configuration it also returns to the original system position
Motion Models

Figure 4.2: In system (a) the position of the coordinate system is unambiguous from the two joint angles, hence a holonomic system. The two discs connected with a rod (b) roll over a surface and the position is ambiguous, dependent on the disk angles. There are multiple solutions of the position for the same disc angles. This results in a non-holonomic system.

(coordinates). With a non-holonomic system, a return to the original internal (wheel) configuration does not guarantee a return to the original system position (coordinates). A holonomic system is characterized by the states not being historically dependent on control input, but only on the current input.

4.2 General Representation

The motion of a mobile platform can be modeled as a stochastic process. This general representation is meant to describe how to predict a pose change and the resulting uncertainty of the prediction for a mobile robot. Early work on modeling of stochastic processes is found in [82]. Other work that is more closely related to what is presented here can be found in [12] and [56].

Assume that the kinematic motion of a robot is described by a non-linear, discrete-time state transition function with some additional noise

\[ x(k) = f(x(k - 1), u(k)) + w(k) \]  

(4.1)

where \( x(k) \) is the robot state vector at time \( k \). \( u(k) \) is a vector containing driving parameters obtained from the proprioceptive sensors, such as rotational and driving speeds. As mentioned earlier, there will always be some errors in motion. One major source are errors in the driving parameters which can
be modeled as a stochastic process. The input signal error \( \tilde{u}(k) \) is therefore described as:

\[
\tilde{u}(k) = u(k) - \hat{u}(k|k)
\]

\[
E[\tilde{u}(k)] = 0
\]

\[
Q_u = E[\tilde{u}(k)\tilde{u}^T(k)]
\]  
(4.2)

and the corresponding system noise \( w \) as:

\[
E[w(k)] = 0
\]

\[
Q_w = E[w(k)w^T(k)]
\]  
(4.3)

If this principle is applied to the prediction of the state vector the state estimate error at time \( i \) is described as \( \tilde{x}(i|j) \) with the corresponding covariance \( P(i|j) \), given all observations up to time \( j \).

\[
\tilde{x}(i|j) = x(i) - \hat{x}(i|j)
\]

\[
P(i|j) = E[\tilde{x}(i|j)\tilde{x}^T(i|j)|Z^j]
\]

\[
E[\tilde{x}(i|j)] = 0
\]  
(4.4)

If equation (4.1) is linear and the noise is gaussian the prediction will be equal to the conditional mean of the input signals, with the corresponding covariance. Since only non-linear models are considered it will be an approximation. Taylor series is an accepted method for making such approximations of non-linear functions. An expansion of (4.1) as a Taylor series results in an approximative linear version of the transition function.

\[
x(k) = f(\hat{x}(k-1), \hat{u}(k)) + \nabla f_x \hat{x}(k-1) + \nabla f_u \hat{u}(k) + O(\hat{x}^2(k-1), \hat{u}^2(k)) + w(k)
\]

\[
(4.5)
\]

The Jacobian \( \nabla f_x \) and \( \nabla f_u \) is evaluated at \( x(k-1) = \hat{x}(k-1|k-1) \) and \( u(k) = \hat{u}(k) \). If the Taylor series is truncated at first order, reusing the expected state at time \( k-1 \) yields the following state prediction.

\[
\hat{x}(k|k-1) = E[x(k)|\hat{x}(k-1|k-1), \hat{u}(k)]
\]

\[
\approx E[f(\hat{x}(k-1|k-1)), \hat{u}(k)) + \nabla f_x (x(k-1) - \hat{x}(k-1|k-1)) + \nabla f_u (u(k) - \hat{u}(k)) + w(k)|\hat{x}(k-1|k-1)]
\]

\[
= f(\hat{x}(k-1|k-1)), \hat{u}(k))
\]

\[
(4.6)
\]

When inserting this into (4.4) the corresponding approximated prediction error \( \tilde{x}(k|k-1) \) becomes

\[
\tilde{x}(k|k-1) = x(k) - \hat{x}(k|k-1)
\]

\[
= f(\hat{x}(k-1), \hat{u}(k)) + \nabla f_x \hat{x}(k-1) + \nabla f_u \hat{u}(k) + O(\hat{x}^2(k-1), \hat{u}^2(k)) + w(k)
\]

\[
+ w(k) - f(\hat{x}(k-1), \hat{u}(k))
\]

\[
\approx \nabla f_x \hat{x}(k-1) + \nabla f_u \hat{u}(k) + w(k)
\]

\[
(4.7)
\]
Assuming the prediction $\hat{x}(k|k-1)$ in (4.6) is equal to the conditional mean, the covariance can be approximated as the square of the conditional mean error $\tilde{x}(k|k-1)$. This is done by inserting (4.7) into (4.4), resulting in:

$$P(k|k-1) \approx E[\tilde{x}(k|k-1) \tilde{x}^T(k|k-1)|\hat{x}(k-1|k-1)]$$

For clarity, the final error prediction function (4.8) is rewritten to incorporate (4.2) and (4.3) together with the substitution $F_x = \nabla f_x$ and $G_u = \nabla f_u$ resulting in the following two equations:

$$\dot{\hat{x}}(k|k-1) = f(\hat{x}(k-1|k-1), u(k))$$

$$P(k|k-1) = F_x P(k-1|k-1) F_x^T + \Lambda(k)$$

This way of presenting the motion and corresponding uncertainty is applicable to almost all kinematic models since it has the ability to introduce noise from both external as well as internal driving variables. For each different motion model the transition function and two corresponding Jacobians need to be derived. In case of a non-linear transition function these need to be recalculated for each time step, otherwise no recalculation is necessary.

### 4.3 Time Discrete Constant Turn Model

From the general presentation of mobile robot motion derived in section 4.2 there are two parts describing the motion in equation (4.9). The first is the estimated conditional mean and the second the corresponding uncertainty, or covariance. In most implementations the conditional mean is modeled with a non-linear, sample time dependent, approximation of the true kinematic motion, such as [15, 44] for example. The approximation is based on a small rotational change between two samples. This way of modeling is only adequate in the case of high sampling rate, relative to the rotational speed. Since the approximation error grows with the distance traveled it will eventually cause a biased prediction. The error propagation is also often simplified by only adding correlated process noise directly to the robot states and thereby ignoring the correlation effect of errors in input signals.

In this work the robot motion is modeled with a kinematically exact non-linear model. The model error is independent of sample time and is therefore
not sensitive to sampling frequencies. The basic model with error propagation is derived in section 4.3.1 and is used in the first three appended papers [I–III]. An extension of the error propagation is derived in section 4.3.2 where lateral slipping error is taken into account, further used in [IV]. Similar work is presented in [VIII] where the case of \( \Omega \approx 0 \) is derived. This work, on the other hand, applies for all \( \Omega \). In the final two appended papers [V, VI] uncertainty in sampling time was encountered. Since the model in section 4.3.1 is not dependent on sampling time it was well suited for extension to include these errors. This is derived in section 4.3.3 where the impact the sample time errors have on the error propagation of the motion is also shown.

### 4.3.1 Basic Model

![Figure 4.3: A mobile robot platform moving according to the discrete constant turn model described in this section. From this figure an exact discrete model is derived.](image)

A typical non-holonomic motion platform is shown in figure 4.3. If the front wheel is turned and the platform moves forward, all states will change resulting in a new pose based on the turning angle and traveled distance. The motion of
the robot platform can be described by the following state and input vectors.

\[
x(k) = [x(k), y(k), \phi(k)]^T \\
u(k) = [v(k), \omega(k)]^T
\]

(4.10)

Note that the kinematic model presented is not controllable in a lateral direction. The pose change \(U(k)\), in robot local coordinates, will be dependent on the speed \(v\) and rotational speed \(\omega\). For a given sample interval \(T\) the velocities will correspond to a traveled distance \(D\) and rotated angle \(\Omega\), as illustrated in figure 4.3, resulting in:

\[
U(k) = \begin{bmatrix}
D(k) \\
\Omega(k)
\end{bmatrix}
\]

(4.11)

For the Discrete Constant Turn model described here the transition function in (4.1) is according to:

\[
f(x(k - 1), u(k)) = x(k - 1) + R(\phi(k - 1))U(k)
\]

(4.12)

where

\[
R(\phi) = \begin{bmatrix}
\cos(\phi) & -\sin(\phi) & 0 \\
\sin(\phi) & \cos(\phi) & 0 \\
0 & 0 & 1
\end{bmatrix}
\]

(4.13)

Instead of approximating a discrete model from a continuous equation, it is possible to extract the pose change using geometric reasoning in figure 4.3. The pose change of the platform during sample time \(T\) becomes:

\[
U(k) = \begin{bmatrix}
\frac{v(k)}{\omega(k)} \sin(T\omega(k)) \\
\frac{v(k)}{\omega(k)} (1 - \cos(T\omega(k))) \\
T\omega(k)
\end{bmatrix}
\]

(4.14)

For clearer equations the linear system error \(w(k)\) is discarded. All errors are therefore caused by noise in input signals propagated through the transition function. The input noise is described by the same principle as used in the general representation (4.2):

\[
\tilde{v}(k) = v(k) - \hat{v}(k) \\
\tilde{\omega}(k) = \omega(k) - \hat{\omega}(k)
\]

(4.15)

where \(\tilde{v}(k)\) and \(\tilde{\omega}(k)\) are errors in the driving parameters and modeled as zero mean, uncorrelated, white sequences with variances \(\sigma^2_v\) and \(\sigma^2_\omega\) respectively, resulting in the following covariance matrix:

\[
Q_u = \begin{bmatrix}
\sigma^2_v & 0 \\
0 & \sigma^2_\omega
\end{bmatrix}
\]

(4.16)
Based on the assumptions made in section 4.2 the state prediction and error propagation is derived from (4.12) as:

$$
\hat{x}(k|k-1) = f(\hat{x}(k-1|k-1), \hat{u}(k))
$$

$$
P(k|k-1) = F_x P(k-1|k-1) F_x^T + G_u Q_u(k) G_u^T
$$

(4.17)

where $F$ and $G$ are the Jacobians of (4.12) evaluated in the same way as in section 4.2 as:

$$
F_x = \begin{bmatrix}
1 & 0 & \frac{\dot{\phi}(k)(\cos(\hat{\phi}(k-1) + T \hat{\omega}(k)) - \cos(\hat{\phi}(k-1) - 1))}{\hat{\omega}(k)} \\
0 & 1 & \frac{\dot{\phi}(k)(\sin(\hat{\phi}(k-1) + T \hat{\omega}(k)) - \sin(\hat{\phi}(k-1) - 1))}{\hat{\omega}(k)} \\
0 & 0 & 1
\end{bmatrix}
$$

(4.18)

It is important to note that $F_x$ and $G_u$ are only valid when $\hat{\omega}(k) \neq 0$. If $\hat{\omega}(k) = 0$ it is necessary to re-evaluate $F_x$ and $G_u$ for $\lim_{\omega \to 0}$ to obtain numerical stability.

4.3.2 With Lateral Slip

In many situations, especially outdoors, unexpected sideways motion of the robot may occur. This can be caused by slippage in tilting terrain, skidding when cornering or other external factors. In all the motion models described in this work the kinematics do not allow for lateral motion. Also, the basic model from section 4.3.1 has a high correlation between rotation and sideways motion and is not capable of separating the source causing these types of error. This can be compensated for by increasing the noise parameters for the rotation uncertainty. On the other hand, this would make the estimate conservative for the orientation estimate instead. Therefore, it will not handle these situations in an appropriate manner.

To illustrate what happens with errors due to the integrated process noise a simulation scenario has been created to show the effect of different errors and how these would affect the error predictions. In figure 4.4(a) a simulated trajectory is shown for an example where unexpected lateral motion is introduced. In figure 4.4(c) it is clear that the lateral motion has caused the basic model to be overconfident about the pose estimate. The best thing to do is to introduce a separate noise parameter allowing for lateral slip and thereby decouple rotation and lateral motion errors. This, however, requires additional sensors measuring the rotational speed, such as a rate gyro, to decouple sources for sideways motion caused by rotation or lateral slip.
Figure 4.4: A robot is simulated to travel along a trajectory. Some lateral speed is inserted into the model. (a) The predicted trajectory of the robot. (b) The lateral velocity inserted into the model. (c) The simulated resulting poses are shown together with two error propagation models: one considers the lateral error while the other discards it. It is clear that if the error is discarded the estimate will be over confident about the robot position.
Multi-robot Information Fusion

To include lateral errors in the motion model of the same kinematics as used in section 4.3.1, the state prediction in (4.17) does not have to be changed. Only the error propagation is updated by adding an input state of lateral motion $v_{\text{lat}}$ to the input vector $u_{\text{lat}}(k)$ in (4.10). This will result in the following input vector:

$$u_{\text{lat}}(k) = [v(k), v_{\text{lat}}(k), \omega(k)]^T$$  \hspace{1cm} (4.19)

where $U_{\text{lat}}$ is the corresponding pose change. In a similar way as before it is possible to derive a kinematically correct model,

$$U_{\text{lat}}(k) = \begin{bmatrix}
v(k)
\frac{v(k)}{\omega(k)} \sin(\omega(k)T) - \frac{v_{\text{lat}}(k)}{\omega(k)} (1 - \cos(\omega(k)T))
\frac{v(k)}{\omega(k)} (1 - \cos(\omega(k)T)) + \frac{v_{\text{lat}}(k)}{\omega(k)} \sin(\omega(k)T)
\end{bmatrix} T\omega(k)$$  \hspace{1cm} (4.20)

now including the lateral motion $v_{\text{lat}}$. However, the kinematics are still constrained and therefore $v_{\text{lat}}(k) = 0 \forall k$. Because the kinematics are the same, only the Jacobian $G_{u_{\text{lat}}}$ needs to be updated.

$$G_{u_{\text{lat}}}(k) = R(\hat{\phi}(k-1)) \begin{bmatrix}
\sin(T\hat{\omega}(k)) & \cos(T\hat{\omega}(k)) - 1 & \hat{v}_x(k)(T\hat{\omega}(k) - \sin(T\hat{\omega}(k)))
\frac{\omega(k)}{\omega(k)} & \frac{\omega(k)}{\omega(k)} & \hat{v}_y(k)(1 + \cos(T\hat{\omega}(k))) + \omega(k)T \sin(T\hat{\omega}(k))
\end{bmatrix} T_{\hat{\omega}(k)}$$  \hspace{1cm} (4.21)

The corresponding input noise covariance also needs to be updated since a new variable is added

$$Q_{u_{\text{lat}}}(k) = \begin{bmatrix}
\sigma_v^2 & 0 & 0
0 & \sigma_{v_{\text{lat}}}^2 & 0
0 & 0 & \sigma_{\omega}^2
\end{bmatrix}$$  \hspace{1cm} (4.22)

The complete covariance update for the error propagation of the motion with lateral error then becomes

$$P(k|k) = F_{\text{x}}(k-1)P(k|k-1)F_{\text{x}}^T(k-1) + G_{u_{\text{lat}}}(k)Q_{u_{\text{lat}}}(k)G_{u_{\text{lat}}}^T(k)$$  \hspace{1cm} (4.23)

It can be difficult to tune the lateral noise parameter $\sigma_{v_{\text{lat}}}$. In practice it is not necessarily a fixed parameter but is likely to change during motion. It can often be set rather low and when innovation increases drastically, without any clear explanation, it can be “kicked high” to recover from the high innovation and manage to do data association anyway.

4.3.3 With Timing Error

In a sampled system there is always a possibility of differences between expected and actual sample times. Depending on the system architecture this jitter can
be deterministic or random. This section presents a method of incorporating these types of timing errors into the error propagation of the kinematic motion model from section 4.3.1.

If data packages are transmitted deterministically in a periodic stream, the receiving timing error can be dejittered by applying a filter, causing phase shift. In an off-line scenario this is not a problem although in an on-line situation the phase shift can cause problems when used in a feedback loop. In a master-slave architecture, data packets are often being “pulled” by the master. The error will then also include timing uncertainty in data transmission, especially in wireless communication. This can be a major problem since the transmission time is not necessarily the same for all transmissions. With event based operating systems there is no guarantee that a packet is given a “correct” time stamp. Therefore, the accuracy of the time stamp given to a certain transmission is also a factor that can play a significant role in this case. If this is not considered in the error propagation the estimate will become overconfident.

Two different situations for usage of the motion model are discussed below. The first considers general motion prediction based on new data from proprioceptive sensors. These sensors may have problems with timing jitter, as discussed earlier. If it is not handled correctly it can have a dramatic effect on the result because the error is additive. In many situations this type of error can be incorporated in the noise of the driving parameters of the basic model from section 4.3.1. However, since the motion prediction is used together with exteroceptive sensors, as discussed earlier, there are other sources of timing error that also need to be considered. Therefore, it is of interest to decouple the driving noise and other sources of timing errors. A simulation is conducted to illustrate what happens to the uncertainty when this type of error is not considered. It is clear that the uncertainty is heavily overconfident when the timing jitter is ignored, as shown in figure 4.5.

Another situation when the timing error comes into play is synchronization with exteroceptive sensors. These sensors normally require a pose prediction to calculate innovation or handle the association problem. This synchronization will not be perfect and this error can therefore be modeled as an uncertainty in sample time of the predicted pose for where the observation is made. Some experimental data has been collected to illustrate this problem. The setup uses a Hokuyo URG-04LX laser scanner [84] connected to a PC sending data using the built-in USB interface. The computer is set up running Linux with the robot and sensor server Player. The data sheet states that the sensor scans at a rate of 10Hz. The true scanning speed is output from the device and captured with an oscilloscope. The result is plotted in figure 4.6. From the plot one can see that the true sampling speed is much like what is stated in the data sheet. The variance is very low, < 1ms. When looking at the corresponding time stamp of the captured data packages made by the computer, the variance is much higher, see figure 4.7.

Since sample time is a parameter in the standard motion model from sec-
Multi-robot Information Fusion

4.3.1, it is possible to change it to a driving variable instead and thereby introduce an uncertainty in this variable for the error propagation.

\[ u_T(k) = [v(k), \omega(k), T]^T \]  

(4.24)

When introducing a timing error the distance traveled by the robot for each time sample will be ambiguous. Therefore, the timing uncertainty will be added to the error propagation of the pose change during the sample, which requires an updated Jacobian, \( G_{u_T} \):

\[
G_{u_T}(k) = \mathbf{R}(\hat{\phi}(k-1)) \begin{bmatrix}
\frac{\sin(T\omega)}{1 - \cos(T\omega)} & \frac{v(T\omega \cos(T\omega) - \sin(T\omega))}{v(\cos(T\omega) + T\omega \sin(T\omega) - 1)} & v \cos(T\omega) \\
0 & \frac{T\omega}{v(\cos(T\omega) + T\omega \sin(T\omega) - 1)} & v \sin(T\omega) \\
\end{bmatrix}
\]  

(4.25)

The corresponding input noise covariance also needs to be updated since a new variable is added:

\[
Q_{u_T}(k) = \begin{bmatrix}
\sigma_v^2 & 0 & 0 \\
0 & \sigma_\omega^2 & 0 \\
0 & 0 & \sigma_T^2 \\
\end{bmatrix}
\]  

(4.26)

The complete covariance update for the error propagation of the motion with lateral error then becomes:

\[
P(k|k) = F_x(k-1)P(k|k-1)F_x^T(k-1) + G_{u_T}(k)Q_{u_T}(k)G_{u_T}^T(k) \]  

(4.27)

As mentioned earlier, this noise parameter is dependent on the situation in which it is being used. It is likely that this parameter is going to be different for different sensors and part of the system. In some cases the timing error is summed up by different sources. It is suggested that some long term sampling be done using an accurate clock as reference to find appropriate noise parameters.
Figure 4.5: The error propagation is heavily effected by jitter in sampling time. A robot is simulated to travel along the trajectory shown in a). The motion is sampled each 0.3s with a standard deviation of 0.02s, shown in b) for all simulations. The final poses for all simulations are plotted in c). It is clear that when this small timing error is ignored the uncertainty is overconfident.
Figure 4.6: The sync signal from a URG-04LX laser scanner is captured on an oscilloscope. Gray areas illustrate over 30 seconds of accumulated pulses. As one can see, the frequency is stable at 10 Hz, as specified in the datasheet. The variance is also very low, <1ms, which indicates very stable sampling times.

Figure 4.7: a) Time stamps made by a PC for some sampled data from a URG-04LX laser scanner. In the histogram b) it is clear that the variance is greater than 5ms.
The main contribution in this thesis is the results from the appended papers presented in section 5.1. The work covers two major areas. First, multi-robot deployment uncertainty is discussed and dealt with from a Bayesian filtering perspective. Secondly, a smoothing framework for multi-robot teams is presented in theory and verified by experiments.

For the multi-robot deployment problem, teams of two or three robots are evaluated. Different formations and trajectory strategies are evaluated to find how these affect the uncertainty propagation. It is found that the formation should be chosen according to the task to be completed. It is also found that different formations can be used as a tool to balance out the uncertainty over the robot states.

The smoothing framework is shown to be robust to the association problem and an online algorithm is presented for estimating the relative reference frames of two-robots. This algorithm is used together with the smoothing framework presented in section 2.2.3 to show how spatial information can be joined within a robot team. The major benefits of using a smoother for sharing information within a mobile robot team are also discussed. With the presented smoothing framework it is shown how mobile robots can effectively cooperate and share information among themselves.

Simulated data shows that the choice of error propagation model can be crucial for the consistency of motion prediction, especially if it is overconfident about the estimate. This work presents some results regarding motion prediction models for traditional constant turn tricycle kinematics. A general error propagation model for lateral slip is derived in 4.3.2, which is an extension from work found in [83]. In section 4.3.3 a method of handling motion prediction uncertainty caused by inconsistent sample time for both exteroceptive and proprioceptive sensors is derived. Some experimental data is also shown to describe the benefit of the proposed method.
5.1 Results

A simulation experiment shows that it is possible to affect the resulting uncertainty of observations conducted with two mobile robots. It is shown that the placement of sensors among the robot team members has an impact on the resulting uncertainty. Different trajectories can also be used to balance uncertainty among the spatial variables, which can be useful if certain variables are more critical than others. Given a task to minimize uncertainty in distance between two mobile robots, where one is moving away and the other is stationary, the best results will be obtained if the camera is mounted on the moving robot, in contrast to the stationary one. It can drastically reduce orientation uncertainty, which is helpful when trying to solve the data association problem. This is shown for measurements made with a camera-like sensor in [I]. If the camera is placed on the stationary robot the resolution will have to be much higher to get the same results.

In [II] it is shown that if a robot team, consisting of three units, travels in different formations there will be certain formations that keep the orientational uncertainty low while others will keep the position uncertainty low. The uncertainty propagation for one robot in a team of three robots is evaluated. A total of five formations are evaluated for different uncertainty propagation models as a function of final uncertainty of the leading robot. It is shown that depending on the relation between driving noise parameters different formations perform better than others. It is worth mentioning that the triangular shaped formation does reasonably well for all noise parameters, but is never the best formation.

In [III] it is shown that there is a possibility to affect the balance of the uncertainty in a robot team by controlling the formation while deploying in open terrain. It is also shown that by considering characteristics from motion models and sensors while choosing the formation of a team, the total uncertainty of the team can be reduced. It is also shown that the uncertainty increase for the relative orientation within the team can almost be ignored, if the sensor and motion model are chosen correctly. Simulation experiments of two formations with three robots are conducted. The experiments show that the performance of the formations in an uncertainty perspective is heavily dependent on the distance traveled between sensor updates. The simulation results clearly show that there is a breakpoint where the one formation benefits over the other.

In [IV] the Collaborative Smoothing and Mapping (C-SAM) framework is presented. This is a straightforward framework for aligning and joining mapped landmarks and robot trajectories in a multi-robot system. Since the method not only joins the maps but also recovers the trajectory of the robots, it is well suited for exploration or mission control. In these situations it is of great value to recover trajectory information and reuse trajectories from other robots to verify a possible path in the newly discovered region. It is also shown how association between duplicate features is accomplished and how this is used by the smoother to straighten up the map and recover uncertainty.
Contributions

Paper [V] describes how the C-SAM framework can be integrated into an experimental platform for field exploration. A solution for cooperate traversability is presented. This work presents preliminary results of sensor data fusion where it is found that a 2-point smoother is not enough to robustly perform a traversability map of an outdoor scene. The solution presented uses the traversibility map to indicate where a vehicle can drive. The C-SAM framework is used for local navigation and the smoothed trajectory of the vehicle is used to show what path has been taken.

Some real experiments with the C-SAM algorithm are successfully implemented in [VI]. It is shown to be both efficient and robust when joining sub-maps. Real experiments with a cooperating robot team consisting of two robots is presented. An addition is made to the C-SAM framework for solving the association problem when initiating the base node. The experiments present results where map joining is successful with close to 50% faulty rendezvous observations.
Discussion and Future Outlook

During the research presented in this thesis a number of interesting observations have been made. In this section some of these observations will be brought up and discussed in terms of the results, how to interpret them, and what can be extended into further research.

Throughout this work non-linear motion and sensor models are used for both predictions and estimations. When doing this instead of using linearized models questions regarding the extra computational cost will arise. Whether the calculation cost is worth it or not depends on the system and what one wants to accomplish. There is less reason to use a nonlinear motion model if one does not rely on the nonlinear characteristics of the motion. The emphasis of this work have been to use the motion kinematics as a base in the reasoning for constructing robot teams. The non-linear characteristics of motion are essential for some of the presented results and crucial to the success of the experiments.

In simulation there is no problem of changing kinematic characteristics, such as noise parameters or kinematic constraints. In a real environment this is not as easy since it would require rebuilding the robots or changing the surrounding, such as the surface. Therefore, a test environment with different types of sensors and motion platforms would be of great value, to find out how the formation deployment results apply in a real environment. Another issue with the simulations is the lack of discontinuous slip, especially when turning the robot. These errors will likely have an impact on the result since in a real scenario a turning motion of a robot is likely to cause other errors than what is discussed in this work.

In the simulations from the initial work in appended paper [I] the association problem is assumed to be solved and the work is based on known features. The features consist of 2-meter wall segments and the uncertainties of the feature-
Multi-robot Information Fusion

observation model are known. This can be a major concern in a real scenario where not only the association problem needs to be solved but all features need to be mapped onto a specific feature model. For an indoor environment where mostly man-made objects are available for navigation, it may be possible to describe features with a few simple feature models such as straight walls, cylinders and door openings. In an outdoor environment it becomes much more complicated. If an incorrect observation model is assigned to an object there may be overconfident observations which in the long run will cause the estimate to diverge. It is also not clear how these results can be used in a totally unknown environment.

More work can also be done in the area of feature modeling and how different object-sensor combinations will impact the balance of uncertainty in a robot team. In combination with finding optimal formation changes during the deployment, considering multiple observations, the non linearities in object-sensor characteristics can be used in a more efficient way and thereby simplify the association problem. This would enable the possibility to optimize what features to use to best perform a given task.

When looking at the formation deployment problem discussed in appended papers [II, III] one obvious comment is that the formations are assumed to be identical at the start and when the measurements are made, no formation distortion. The reason for this assumption is that the results will otherwise be dependent on the error profile for a specific simulation. In a real scenario this is not possible since the different robots will have different errors relative to the conditional mean when measurement is made. On the other hand, these are intended to be guidelines for how different robot formations can affect the uncertainty for the group. Since the camera sensor is not distance dependent the group can be scaled in size and the distances between robots can change without any impact on the results. It is also a fact that the actual formation distortion is a function of the distance between the robots and the motion error parameters. If the robots are far apart a larger motion error can be accepted without having a great impact on the formation and thereby the formation distortion can be ignored.

As presented in paper [III] formation changes during a deployment will result in lower uncertainty than keeping the same formation at all times. For the constant turn model used in this work there are great correlations in angular uncertainty and lateral error. How much is naturally dependent on robot motion kinematics together with noise and sensor characteristics. Intuitively, it is of interest to maximize sensor information gain in states where the robots lose the most information when in motion. How the different states are affected depends on the traveled distance; however, it is at all times of great value to keep the uncertainty in orientation low when a robot team travels long distances over open or unguided terrain, since it will eventually have great impact on the position error. The suggestion is to make experiments with a real robot team to find the noise parameters to describe the uncertainty propagation of the system.
From that point it is possible to redo the simulations to find the breakpoints for that particular system. Together with sampling rates of rendezvous observations this can be used as guidelines for what formation to choose for the system of interest.

When performing a map joining based on robot trajectories there are a number of benefits over making raw feature to feature matching using landmark searching algorithms. The trajectory alignment can be conducted with a minimal set of data, since only the information related to the rendezvous is being considered. It is possible to align map information even though a rendezvous is conducted in unguided terrain. Also, since obstacle avoidance is expensive, in terms of time and energy, the trajectories from both robots can be reused and shared amongst the team members. From this perspective, cooperation is a tool for increasing efficiency when operating in partially discovered terrain. However, the obvious drawback is that a rendezvous must have been conducted for an alignment to take place.

In the PreeRunners project discussed in appended paper [V] the idea is to have a number of “satellite” vehicles for tracking. This includes finding possible paths for a main vehicle. Since Collaborative Smoothing and Mapping (C-SAM) explicitly extracts smoothed trajectories from the path of the robots it is of great interest to investigate how this information can be used for more efficient path planning. The difference between an estimated and a predicted trajectory of a vehicle can be used as meta information about the path quality which gives information about slippage. This information can be bundled with other sensor data such as acceleration, tilting or bouncing in uneven terrain. The information can then be reused by other team members to avoid these areas. This becomes especially interesting for heterogeneous robot teams since all robots will not necessarily have the same requirements for a clear path. Since information is also collected from different sensors, spread among the different team members, a broader perspective of the scene will be obtained. This yields a more solid base for concluding what is a possible path. One scenario would be for surveillance robots to meet and share information about visited areas. The information retrieved contains information about where the other robot was traveling and how successful the robot was when driving a path. By sharing this knowledge it is less likely that each robot will run into the same problems in the future.

In combination with larger robot teams the risk for false association in rendezvous will increase. Therefore, it is of further interest to investigate different methods for evaluating initial base-node estimates. In the work done so far a simple mean estimate is used for reference when selecting consistent base-node candidates. It may be more appropriate to use a density estimate of the base-node candidates as described in [85], or similar. In contrast to the mean estimate this method degrades the influence of spurious solutions, leaving the consistency of solutions a heavier weight. It would also be possible to give the final estimated base node a quality mark using the Probability Density Func-
tion (PDF), based on the rendezvous observations. A similar use of the PDF was presented in [86] where it was used to calculate the quality of a landmark instead. Also, more complex scenarios are of interest since the complexity of the trajectory during the rendezvous will have an impact on the resulting map alignment. In general, more complex trajectories during the rendezvous will increase the information gain, as long as the association is solved correctly. This is the drawback since more complex trajectories will increase the possibility of making a false association, due to slip, which will have a catastrophic impact on the results. It is worth mentioning that even though the algorithm for finding the base node presented in appended paper [VI] is presented together with the C-SAM framework, it can easily be used with other frameworks such as The Stochastic Map, presented in section 2.1.

With the presented smoothing framework it is shown how mobile robots can effectively cooperate and share information among themselves. It is found that when dealing with nonlinear kinematics and sensor models together with the C-SAM framework the initiation of the base node is important for the convergence speed of the optimization. In the long run this may also effect the result of the final map since more computational resources are necessary for handling all new information. In the worst case, some information must be discarded due to lack of computational resources. Even though the modeled kinematics in this work are presented in a nonlinear form it is questionable whether this is necessary in all situations. The smoothing approach used in $\sqrt{SAM}$ and C-SAM is not as sensitive to this and it is therefore questionable whether the extra calculation cost is worth it or not. However, when working with an Extended Kalman Filter based mapping framework, that is known to diverge due to nonlinearities, the nonlinear models is probably the way to go. Simulations conducted during the work have indicated drastic increases in calculation speed for solutions using linear kinematic models. Part of this is due to simpler implementations and the possibility to use predefined algebraic frameworks for doing calculations.

Since it is likely that robots exchanging map information will gain new information about areas recently explored, this will cause loops in the map resulting in dense sections of the observation matrix. So far it is not entirely clear what effect this will have on the required computational resources when performing the optimization. On the other hand, loop closing will tighten up the map and all features within the loop will be highly correlated. This gives a greater possibility for solving the association problem in the future and yields a much better mapping result.

A natural extension to the work presented in this thesis is to incorporate the knowledge about uncertainty propagation in different formations together with trajectory alignment. Similar ideas with single robot exploration were earlier presented in [87] and [88]. This would provide an efficient tool for reusing information retrieved from other team members. It can give a decision control mechanism more time and a priori information to base decisions on. However, in all simulated experiments communication limitations between robots have
Discussion and Future Outlook

not been considered at all. In a real scenario, at the time of writing, it is not reasonable to transfer a complete trajectory and traversability map as described in appended paper [V] between two robots in realtime. Therefore, it is important to investigate how to make the information distribution faster. This requires more advanced algorithms, such as channel filtering as described in [46].

For the formations studies to be useful in a real scenario it is necessary to have good knowledge of the error characteristics of each member of a team. Otherwise, the conclusions may be based on wrong terms. This raises the question of whether it is possible to create an automated error characteristics analysis for new motion platforms.
Summary of Papers

This chapter briefly summarizes the appended papers.

**Paper I**

On Sensor Fusion Between a Pair Of Heterogeneous Robots

This paper looks into the utility of fusing information within a team of two robots. The robots are used to explore an environment consisting of two short wall segments. The task for the robots is to measure the relative pose of the two walls using laser scanners. Both robots start near one of the walls and one robot needs to move closer to the other wall segment for the laser scanner to come into range. Measurements within the team, between the robots, are made using a camera. Two scenarios are investigated: one where the the camera is placed on the moving robot and the other where the camera is mounted on the stationary robot. It is shown that the results are much better if the camera is placed on the moving robot, rather than keeping it stationary.

**Paper II**

On utilizing geometric formations for minimizing uncertainty in 3 robot teams

In this paper the formation of a 3-robot team is studied, from the perspective of deployment uncertainty. Simulations show that the formation of a team has a great impact on the uncertainty increase of the team as it is deployed in unknown terrain. The sensor used in the simulation is of camera type and the motion model is a discrete constant turn model, described in section 4.3. Simulated results show that it is beneficial in some cases to change the formation
Multi-robot Information Fusion

of the team during deployment. It is shown that the formation has an impact on the balance of the covariance matrix. By choosing different formations it is possible to minimize the uncertainty in, for example, position at the expense of uncertainty in orientation.

Paper III

Effects on Uncertainty Utilizing Formation-Planning in Robot Teams

The third paper is based on the second paper. This work continues with the formation discussions begun in [II]. The same formations are used and further investigated with regard to what formation is better suited for what purpose. The question asked is what formation should be chosen if one is interested in low orientation uncertainty or position uncertainty, respectively. The results show that if one travels long distances with a 3-robot team equipped with a bearing only sensor there is reason to consider the formation. How the formation should be chosen is dependent on robot kinematics and sensor characteristics. An interesting result is that it seems to be possible to stabilize the orientation uncertainty at a fixed level and after that balance out all new uncertainty in position instead. This can be beneficial if one has a priori information about objects and knows what level of uncertainty one can accept to perform association between objects.

Paper IV

C-SAM: Multi-Robot SLAM using Square Root Information Smoothing

In this paper we have contributed an efficient and straightforward algorithm to align and join maps and trajectories in a multi-robot system, Collaborative Smoothing and Mapping (C-SAM). Since the algorithm not only joins maps but also recovers the trajectory of the robots it is well suited for mission control, where the trajectory information can be reused by robots to verify a possible path in the newly discovered region. It is also shown how association between duplicate features is accomplished. Two simulated scenarios are presented where the C-SAM algorithm is applied on two individually created maps. One basically joins two maps resulting in a large map while the other shows a scenario where sensor extension is carried out.
Paper V

Sensor Data Fusion for Terrain Exploration by Collaborating Unmanned Ground Vehicles

The goal for this work is to demonstrate the value of collaborative Unmanned Ground Vehicles (UGVs) in an outdoor setting. The setup consists of a main vehicle and a number of UGVs providing sensor platforms. The work covers a method for how to distribute traversability map information amongst the vehicles. The idea is that each UGV creates a traversability map of the area being explored. The information from each UGV is then fused to generate a complete terrain scene for all team members to utilize when deploying the terrain.

Paper VI

On using Trajectory Alignment for Multi-robot Map Joining

This paper presents a method for joining maps that are independently built by multiple robots. The key contribution of this work is an algorithm for solving the association problem when aligning trajectories from robot teams. This is then applied to multi-robot map alignment problem. The algorithm is also tested in two different experiments showing the usefulness and robustness of the algorithm. The stochastic framework used is C-SAM, presented in [IV]. This framework is especially appropriate since the trajectory for each robot is solved explicitly and is therefore well suited for trajectory alignment.
Bibliography


Bibliography


Multi-robot Information Fusion


