Graphical SLAM for Outdoor Applications

John Folkesson
Department of Mechanical Engineering
Massachusetts Institute of Technology
Cambridge, MA 02139
johnfolk@mit.edu

Henrik I. Christensen
Center for Autonomous Systems,
Kungliga Tekniska Hgskolan
S-100 44 Stockholm, Sweden
hic@kth.se

Abstract

Application of SLAM outdoors is challenged by complexity, handling of non-linearities and flexible integration of a diverse set of features. A graphical approach to SLAM is introduced that enables flexible data-association. The method allows for handling of non-linearities. The method also enables easy introduction of global constraints. Computational issues can be addressed as a graph reduction problem. A complete framework for graphical based SLAM is presented. The framework is demonstrated for a number of outdoor experiments using an ATRV robot equipped with a SICK laser scanner and a CrossBow Inertial Unit. The experiments include handling of large outdoor environments with loop closing. The presented system operates at 5Hz on a 800 MHz computer.

1 Introduction

A fundamental competence needed in mobile systems is the ability to localize. Almost any task that an autonomous mobile system is designed to perform will require some form of localization. The localization relies on establishing a correspondence between a world model and measurements made by the robot's sensors. In cases where GPS or other similar navigation aid is not available, the use of environmental features or landmarks is a way of establishing the correspondence.

In many applications the availability of a pre-defined world model is not entirely realistic. Examples include moving into a crash site, or entering a forest for the first time. To enable operation under such conditions there is a need to perform localization and mapping concurrently, to ensure that the robot does not get lost. The problem of *S*imultaneous *L*ocalisation *a*nd *M*apping (SLAM) is by now a well-known problem in robotics (Bekey, 2005). Early research in particular considered SLAM for in-door applications (Chatila and Laumond, 1985a; Leonard et al., 1992; Lu and Milios, 1997b; Thrun et al., 2001; Wijk and Christensen, 2000), but it has also been extensively studied for out-door applications both on the ground (Guivant and Nebot, 2001a; Guivant et al., 2004; Folkesson and Christensen, 2003;

Montemerlo and Thrun, 2006), in the air (Jung and Lacroix, 2003; Kim and Sukkarieh, 2004) and underwater (Newman and and R. Rikovski, 2003).

Out-door environments will typically represent significantly larger areas where complexity considerations are more pressing. Given more feature variation and significantly higher complexity this poses and ideal environment to study the problem of SLAM systems for large scale environments.

Traditionally a number of approximations have been adopted to manage the overall process, however it is not clear that these systems have well posed convergence properties. The key challenges for design of effective SLAM systems are considered to be:

1. Convergence in the presence of non-linearities
2. Handling of global constraints such as loop closing
3. Bounded computational complexity

This paper is organized as follows. Initially the SLAM problem is presented in Section 2. The Graphical SLAM method is presented in Section 3. To address computational issues it is advantageous to simplify the underlying graph structure to capture models in a compact form, as presented in Section 4. For the implementation a number of considerations should be taken into account as presented in Section 5. The platform used for experimental evaluation is briefly presented in Section 6. Using this platform a significant number of experiments have been performed in urban and non-urban environments. Some of the urban experiments are presented in Section 7 to allow evaluation of the presented method. The presented method and some of the achieved results are discussed in Section 8. Finally, a conclusion and open issues are presented in Section 9.

2 Background on the SLAM Problem

The problem of localization and mapping is a relatively old problem that has been studied for centuries as part of surveying. In robotics the problem was introduced in the 1980s by a series of seminal papers by Smith, Self and Cheeseman (Smith et al., 1987), and also reported by Chatila and Laumond (Chatila and Laumond, 1985b). The two coupled estimation problems are the construction of a map of the environment, $\mathbf{X_f}$, and the determination of the path followed by the robot $\mathbf{X_r}$. The measurements for this estimation come from the motion sensors, $\mathbf{D}$, and a set of feature observations $\mathbf{F}$. Early work on SLAM assumed that the problem could be formulated as a recursive estimator using for example an Extended Kalman Filter (EKF), where the state vector is composed as:

$$s(t) = \begin{bmatrix} x_{r,t} & x_{f,1} & \dots & x_{f,i} & \dots & x_{f,N} \end{bmatrix}^T$$

where $x_{r,t}$ is the robot pose at time t and $x_{f,i}$'s are individual feature estimates. The EKF approach is quadratic complexity, $\mathcal{O}(N^2)$, which poses a challenge to scalability. In addition the linearization performed in each step of the Kalman filtering can result in divergence as presented in (Castellanos et al., 2004b; Julier and Uhlmann, 2001). The Kalman filter works on the principle of representing the state and measurements by their mean and covariance. This implies an approximation if they are not Gaussian.

A careful analysis of the covariance matrix reveals that the structure of the matrix is approximately block diagonal, which can be utilized to divide computations into (largely) independent block structures, corresponding to a sub-maps, as reported by Guivant and Nebot (Guivant and Nebot, 2001b), termed the Compressed-EKF (C-EKF). The EKF approach assumes that the underlying probability distribution is uni-modal and in addition it is assumed that linearization has a minimal impact on the estimation process. Another related problem for EKF estimators is the recursive nature of the process. Data-association are assumed to be correct at any time throughout the process. In theory multiple data-associations can be considered in a multi-hypothesis approach as presented by Bar-Shalom and Fortmann (Bar-Shalom and Fortmann, 1987) and Jensfelt and Kristensen (Jensfelt and Kristensen, 2001). However, the approach scales poorly in the presence of a poor signal-to-noise ratio.

Methods for division of the map into sub-maps, to limit complexity, has also been presented by Leonard et al (Bosse et al., 2004). Many authors have used frameworks based on sub-maps and local frames of reference in order to avoid problems of scale (Castellanos et al., 1999; Leonard and Newman, 2003; Tardós et al., 2002; Newman et al., 2003). Typically the local maps have some frame of reference tied to some features. The links between these maps then must be made globally consistent in some way. Many of the sub-map methods treat the features from different sub-maps as distinct from one another. This is easier but has the consequence that their is no consistent global feature location.

As the robot travels the connection between the robot's current pose and the features seen in the past gets weakened by the uncertainty in the robot's motion. This then causes the features observed currently to be more weakly coupled to those in the past than the other currently observed features. By removing the connections altogether a great simplification to the problem can be achieved. The Sparse Extended Information filter, SEIF, (Thrun et al., 2004) discards some information on the weaker connections between map features. The SEIF only has to keep the current robot pose in the filter. The older poses are marginalized out of the graph.

To address the problem on general probability distributions, as opposed to unimodal Gaussians, a sample based approach to mapping and localization can be used as presented by Thrun et al (Thrun et al., 1998). In its basic form the space of possible features and poses is sampled and updated over time. The advantage of a sample based approach is that it makes no assumption about the underlying probability distribution and it can consider multiple data associations. However, without careful consideration of the sample process the number of required samples can be excessive. Even here various approximations can be utilized for re-sampling (Thrun et al., 2000) and factorization (Montemerlo et al., 2002; Hähnel et al., 2003; Montemerlo and Thrun, 2003; Nieto et al., 2003; Stachniss et al., 2004) to achieve real-time performance.

An alternative approach to direct estimation is modelling of the problem by means of a Dynamic Bayesian Net (Jensen, 2001), as presented by Paskin (Paskin, 2003). A graphical approach is used in which the nodes are pose and feature estimates. The relations between these entities can be represented by arcs in the graph. Dependencies between different loops in the graph can be captured in so-called junction trees that provide an exact model for probability estimation. Paskin (Paskin, 2003) presented a model for optimization where approximations were used to generate thin-junction trees. This is needed as the general junction tree method has NP complexity. Computation of the conditional probabilities in a Bayesian Net is in general difficult and as such requires careful consideration. Recent work (Newman et al., 2006) has also adressed the linearization issues in closing of large loops.

We discuss some of the other graphical methods after we have explained our method so as to be able to compare. In this paper we build on ideas first presented in (Folkesson and Christensen, 2004).

From the above discussion is it clear that there is a need to carefully consider the design of SLAM methods to allow:

1. handling of non uni-modal probability distributions
2. explicit handling of non-linearities in the estimation
3. flexible handling of global constraints
4. any-time data-association.
5. real-time / bounded complexity operation.

In the following a method is presented that addresses these challenges.

3 Graphical SLAM

Graphs can be used to represent the underlying structure of the SLAM problem. This structure is the result of the way that the measurements are collected. The robot moves through the environment making relative measurements of features it passes. The path then gives us relations between all the measurements. By maintaining the path in the estimation one can avoid the problem of calculating correlations between distant features. These correlations arise when one tries to eliminate the intermediate robot poses from the SLAM problem.

One approach is to simply linearize all the measurements to obtain a sparse matrix representation. The sparse matrix has a graphical representation where the nodes are the poses of the robot and the features. The graph edges would then be the numerical matrix elements. To be more specific let us illustrate with a simple Gaussian SLAM estimation example. Let $\mathbf{y} = A\mathbf{x} - \mathbf{c}$ represent the measurements[1]. So each element of $\mathbf{y}$ is an independent normal measurement, (mean 0 and variance 1). These could be odometry or feature measurements. The $\mathbf{x}$ is the 'state' vector containing all the path and feature coordinates. It is $\mathbf{x}$ that is the objective of the estimation process. The A and $\mathbf{c}$ are our linearized measurement model. For an iterative algorithm, the linearization of the rows added in one iteration is normally done around the previous iteration's best estimate of $\mathbf{x}$. These rows are then appended to the A and $\mathbf{y}$.

It is well known, (Julier and Uhlmann, 2001),(Eustice et al., 2005),(Castellanos et al., 2004a), that as a result of doing the linearizations around the different points, one can get inconsistent estimates. Furthermore, if the best estimate of $\mathbf{x}$ changes significantly (on closing a loop in the robot path) then the linearizations themselves become poor approximations.

[1]The $\mathbf{y}$ is also the normalized innovation.

These issues will be addressed later but we ignore them for now and continue with our example. The sparseness of the measurement matrix A is the result of the fact that it models relative measurements taken from a robot pose to either another robot pose or a feature. Thus, each element in $\mathbf{y}$ depends on two nodes in the graph. The nodes being robot pose and feature states. The matrix elements of A then correspond to edges in the graph.

As A typically has more rows than columns, we must solve for the $\mathbf{x}$ that will minimize $\mathbf{y} = A\mathbf{x} - \mathbf{c}$. This can be done in batch type solutions by either Cholesky Factorization or QR Factorization (Dellaert, 2005). Cholesky forms the so called 'Information Matrix', $A^T A$, and then solves the equation $A^T A\mathbf{x} = A^T\mathbf{c}$. The QR method solves the QR factorization of A into an orthonormal transformation, Q^T of the $\mathbf{y}$ such that the transformed equation is $Q^T A\mathbf{x} = Rx = Q^T\mathbf{c}$. This can be solved by back substitution since R is zero below the 'diagonal'. A third alternative is to form the Information Matrix and do a QR decomposition on it. All these methods exploit the sparseness of A to solve the system relatively quickly. So far we are still discussing batch mode. The most important aspect of solving this system is the order of the rows in A. If an optimal order can be found then very large systems can be solved relatively quickly.

The desire to solve this system on-line as the data comes in, has lead to a number of approaches. The extended Kalman Filter EKF and Extended Information EIF can be viewed as solutions to the system where all the intermediate pose states have been eliminated from the problem. This would seem to be a reasonable thing to do as these states will not be needed to linearize the future measurements. The system then becomes smaller and larger areas can be mapped before computational complexity becomes an issue. For moderately fast processors the computation time is not the main issue. The problem is that such large maps simply cannot be estimated consistently due to the linearization errors. Thus the scaling problem has forced us to look at the linearization issues.

3.1 Graphical Models as MAP Estimators

We would like to present the reader with the justification for the graphical approach as sketched in the preceding paragraphs. In doing so we will gain further insight into the important data association problem as well as the generalization to non-Gaussian models. We can then formulate our graphical method as an extension of these ideas. First, we define some notation:

N_p: The number of robot poses less one.
$\mathbf{x}_{r,i}$: The coordinate vectors of the robot poses, i runs from 0 to N_p.
N_f: The number of features
$\mathbf{x}_{f,j}$: The coordinate vectors of the features, j runs from 1 to N_f.
N_m: The number of feature measurements.
$\mathbf{f}_k$: The feature measurement values, k runs from 1 to N_m.
$\mathbf{d}_i$: The dead-reckoning measurement values, i runs from 1 to N_p.
Λ_a: A particular association of the feature measurements $\{\mathbf{f}_i\}$ to features.
N_{fa}: The number of features associated to measurements by Λ_a

We can now state the problem as finding an explanation of the measurements. That is, values for $\{\mathbf{x}_{r,i}\}, \{\mathbf{x}_{f,j}\}$ and a Λ_a to explain the $\mathbf{f}_k$ and $\mathbf{d}_i$. The probability of an explanation can be rewritten as[2],

$$P(x_r, x_f, d, f, \Lambda_a) = P(d, f|x_r, x_f, \Lambda)P(x, \Lambda_a) \tag{1}$$

The second term on the right is the a priori probability of a particular state and data association. One must make some assumptions about this in order to move further. We chose to assume that it is independent of the robot and feature states, the x's. Thus, all states are equally probable before we start taking measurements. The dependence on the data association cannot be ignored as that would lead us to associate a new feature to every new measurement. By doing so one could find a feature state that exactly explains the feature measurement. What is needed is a mechanism to favor existing features even when they do not line up exactly with the measurements. Thus the prior must favor smaller N_{fa}. One simple prior that has this property is an exponential dependence:

$$P(x_r, x_f, \Lambda_a) \propto P(\Lambda_a) = P(N_{fa}) \propto e^{-\lambda N_{fa}} \tag{2}$$

The parameter λ gives us control over how much we favor simpler maps. It will give us a criteria for data association. We can now collect this into a statement about the probability of our explanation:

$$P(x_r, x_f, d, f, \Lambda_a) \propto P(d|x_r)P(f|x_r, x_f, \Lambda_a)e^{-\lambda N_{fa}} \tag{3}$$

Here we show explicitly that the dead-reckoning measurements, d, do not depend on the features. Further factorization into a product of terms depending on pairs of state vectors can be made[3]:

$$P(x_r, x_f, d, f, \Lambda_a) \propto \prod_{i=1}^{Np} P(\mathbf{d}_i|\mathbf{x}_{r,i}, \mathbf{x}_{r,i-1}) \prod_{j=1}^{Nm} P(f_j|\tilde{\mathbf{x}}_{r,j}, \mathbf{x}_{f,ja})e^{-\lambda N_{fa}} \tag{4}$$

Where we use, $\tilde{\mathbf{x}}_{r,j}$, to indicate the pose coordinates, (one of the $\mathbf{x}_{r,i}$), from which the j'th feature measurement was taken. Also, $\mathbf{x}_{f,ja}$ is the feature vector associated to the j'th measurement by data association a. Paskin in (Paskin, 2003) uses a product like this to create a Markov graph which can be used to make inferences on the states.

We chose to work with sums rather than the products shown above. Therefore, we take minus the log of the probability and work to minimize the negative of the log-likelihood function. Explicitly we form an energy as;

[2]We drop the subscripts and boldfacing to indicate the whole set of values.

[3]This assumes independence of the measurements wrt both time and feature.

$$E(x_r, x_f, d, f, \Lambda_a) = -\sum_{i=1}^{Np} \ln P(\mathbf{d}_i|\mathbf{x}_{r,i}, \mathbf{x}_{r,i-1}) - \sum_{j=1}^{Nm} \ln P(f_j|\tilde{\mathbf{x}}_{r,j}, \mathbf{x}_{f,ja}) + \lambda N_{fa} \tag{5}$$

As a practical rearrangement of terms we note:

$$\lambda N_{fa} = \lambda N_m - \sum_{j=1}^{N_{fa}} \lambda(n_{ja} - 1). \tag{6}$$

Here n_{ja} is the number of measurements associated to the j^{th} feature by association a. In this way we see the energy 'benefit', (decrease), of associating a new measurement with an existing feature is simply λ. This will have to be balanced against the 'cost', (increase), in energy that appears in the newly added term to the second sum in 5. So essentially, a new feature measurement will add:

$$-\ln P(f_j|\tilde{\mathbf{x}}_{r,j}, \mathbf{x}_{f,ja}) - \lambda \tag{7}$$

After adding this to the energy we need to minimize it wrt. the state variables. If the final resulting energy is less than before we added the measurement, we can accept the match. If not it is more likely that the measurement is of a new feature, rather than associated with feature j. In the case of the EKF where the entire system has been approximated by a Gaussian the test above is simply another way of using the Mahalanobis distance for gating of measurements.

Now to be rather more specific about the probabilities we assume an independent Gaussian odometry model. So that the incremental changes in the robot pose will be measured in the presence of Gaussian noise. The movement of the robot then gives rise to dead-reckoning energy terms[4]:

$$E_{di} = -\ln P(d_i|\mathbf{x}_{r,i}, \mathbf{x}_{r,i-1}) = \frac{1}{2}\xi_i k_i \xi_i \tag{8}$$

The ξ_i are the coordinates of the incremental transformation between pose $i-1$ and i less the measured value of these. These coordinates are non-linear functions of the robot pose coordinates.

$$\xi_i = T(\mathbf{x}_{r,i}|\mathbf{x}_{r,i-1}) - \mathbf{d}_i. \tag{9}$$

The matrix k_i is the inverse covariance of the dead-reckoning estimate. Despite having an independent Gaussian estimate for the noise, we still end up with a non-linearity. If the transformation function T was linear then the 'square root' of k_i would become the rows

[4]We have dropped the normalization terms that would be additive constants.

of the measurement matrix A mentioned in the introduction to this section. As it is, one needs to calculate Jacobians and multiply k_i by them to get the contribution of this to the information matrix (or $A^T A$).

We point out these other interpretations to help readers understand our energy model. We will work directly with the non-linear energy and not the measurement or information matrices. We will, however, linearize the energy around particular solutions to direct our search for lower energy solutions. This linearization will sometimes be identical to a partial information matrix, at that solution. The difference then is that we do not save the linearization but rather recalculate it around new states as needed. At other times, we will linearize permanently but in a different way so as to avoid the problems associated with a global frame.

Each feature energy term can also be written as a Gaussian of a non-linearly transformed relative feature coordinate. The assumption of having a Gaussian is not at all essential but it is specific. On the other hand we do require the energy function to be twice differentiable.

3.2 Graph Nodes

Our graph will differ from the standard graphs in that we will have two distinct types of nodes, state nodes and energy nodes. The edges will always connect a state node to an energy node. The state nodes will be either pose nodes or feature nodes. The pose nodes hold the values for $\mathbf{x}_{r,i}$ while the feature nodes hold $\mathbf{x}_{f,j}$.

The simpler energy nodes will have two edges and thus be connected to two state nodes. The energy nodes will then be able to calculate the energy term that results from a measurement between the two state nodes. Thus, some energy nodes will be specialize to calculated dead-reckoning energy E_{di}, (and its derivatives). These will be connected to two pose nodes. Other energy nodes will calculate feature energies. These will connect to a pose node and a feature node.

In this way we can calculate the energy of a given state by summing the energy of all the energy nodes in the graph. Typically we are only interested in changes to the energy so we need only calculate the energy for the energy nodes attached to state nodes that have changed.

The data association is made by connecting edges in the graph from the feature energy node to a feature state node. In practice, either we will accept the closest match or throw out the measurement. Therefore, testing a data association consists in creating the energy node, relaxing the state and then comparing the energy before and after. If the energy goes up we return the system to the original state and delete the energy node. As shown in Eq.(7), there is only one parameter that determines the matching. It is a limit on the amount of energy increase the match can create. The values used for λ are the same as we use for the mahanolobis test in the EKF. They range between 4 and 16 depending on the environment.

3.3 Method Overview

We start by laying out the basic operation of our SLAM method. We will explain all the terms in turn in the next few subsections. The arrival of new feature measurements will trigger a sequence of operations on the graph as shown in figure 1. The dead-reckoning estimate of the robot's motion is used to add a new pose node to the end of the graph. Then the graph is relaxed. At that point, the older sections of the graph have nodes combined into what we call star nodes. This reduction in the number of nodes reduces the complexity of the graph at the cost of some linearization approximations. Next the feature measurements are used to add links between the current pose node and the feature nodes. These links are then tested to see if they improve the overall solution. If not they are removed. This testing also involves relaxing the graph but is done more locally then the relaxation done at the start of each cycle.

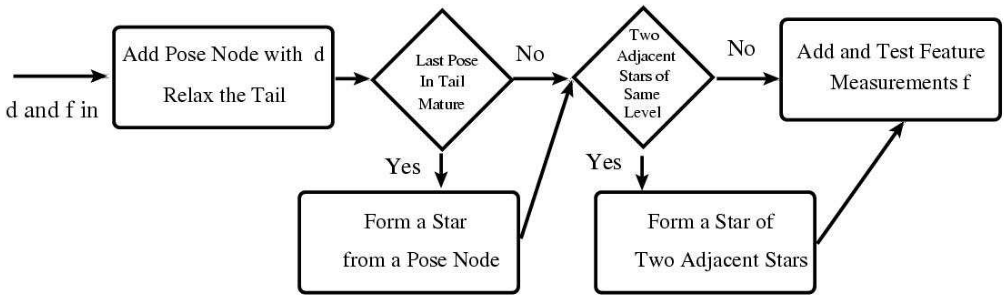
d and f in
Add Pose Node with d
Relax the Tail
Last Pose In Tail Mature
No
Yes
Form a Star from a Pose Node
Two Adjacent Stars of Same Level
No
Yes
Form a Star of Two Adjacent Stars
Add and Test Feature Measurements f

Figure 1: The basic sequence of operations performed on the graph at each iteration. The adding of a pose node corresponds to a predict step with an EKF. Adding the feature nodes corresponds to an update step. Stars are not always formed. The criteria is if the last pose node of the tail has features that are still being observed.

This flow chart diagram captures the sequence of operations performed on the graph upon the arrival of new data. The grapher holds only pointers to a few key nodes of the graph. All the operations are done by following the edges from those nodes. There is no persistent matrix or vector that corresponds to the state of the graph. These are formed as needed for sections of the graph.

The two types of nodes that form our graph are state nodes and energy nodes. A state node stores a state that it can provide to its attached energy nodes. The energy nodes use the state to calculate the energy and its derivatives. The state node can then sum up the contributions from each edge to calculate its partial derivatives and do a simple 'Gauss-Seidel' like relaxation, as explained in the next subsection. All graph operations could be done using these capabilities.

It is the inclusion of the actual measurement model into the energy node that allows it to do the calculation. This is in contrast to other graphical SLAM methods where these functions would simple return stored numbers that were the result of linearizations performed when the node was created. These numbers typically being stored in the edges themselves with no need for energy nodes at all.

3.4 Graph Updates

Finding the MAP solution is equivalent to finding the minimum energy. The minimization must be done over all states and all feasible graphs. Where feasible graphs are all the possible assignments of measurements to features. This task is made difficult by both the combinatoric explosion of the number of feasible graphs and the existence of local minima for each graph. There is no general solution to this problem.

In this section we will describe how one can drive the system to a lower energy state given an approximate solution. The approximate solution is the current best estimate available from the last update modified in some way. The modification could be adding or removing an energy node for a measurement. It might also involve adding state nodes. The effect of the modification will be felt locally and will decay exponentially as one moves away from the modified nodes. An important exception to this is when the modification changes the topology of the graph, (i.e. closes a loop). In that case the effect of the modification can be global.

Assuming we are in the local effects case, we can drive the graph to the local minimum closest to the current state by relaxation. There are a number of strategies that will all work well. They all rely on linearization of the measurements. This results in a quadratic expansion of the energy at each energy node. So for a dead-reckoning node we get:

$$E_{di} \approx \bar{E}_{di} + (\Delta\mathbf{x}_{r,i}, \Delta\mathbf{x}_{r,i-1})\, G_{di} + \frac{1}{2}(\Delta\mathbf{x}_{r,i}, \Delta\mathbf{x}_{r,i-1})\, H_{di}\, (\Delta\mathbf{x}_{r,i}, \Delta\mathbf{x}_{r,i-1})^T . \qquad (10)$$

Here H is the Hessian and G the gradient of the energy for this energy node at the current state. We can make a similar expansion for the feature measurements. By adding up the contributions from all nodes one can form the linear equation that must be solved to drive the state into the minimum of the approximate system[5]. This would be far too much calculation than is justified. One simple procedure is to sum only the terms associated with a single state node and modify that node to the solution of: $H_{ii}\Delta\mathbf{x}_{s,i} = -G_i$. Here i is the index of the state node (s indicates either a feature, f, or robot pose, r) and the Hessian and Gradient are the total for all energy nodes attached to this state node. In general the solution to this might not lower the energy due to the nonlinearity. If it does not we can reduce the magnitude of $\Delta\mathbf{x}_{\mathbf{s},\mathbf{i}}$ until it does or to zero if it is already at the minimum in this direction. After this node relaxation we can move to another state node and repeat. This amounts to a Gauss-Seidel type update, but with the full non-linear measurement functions recalculated at each iteration.

Such a procedure is guaranteed to not increase the energy. If we are away from a saddle point we will converge to the local minima. In the case of the saddle point we can detect the negative eigenvalues in H and apply a simple gradient decent.

The main difficulty with this sequential node processing approach is that it is very slow to move two strongly coupled nodes. When one of the nodes is treated as fixed the other

[5]The total of the Hessian over all energy nodes is the information matrix.

cannot move much. To overcome this we must instead solve a subsystem of several nodes simultaneously.

The question then is how large to make the subsystem and when to stop relaxing. We do not need to update all the nodes in the graph but may need to update the same node multiple times. We propose a procedure for deciding which nodes to chose as illustrated in figure 2.

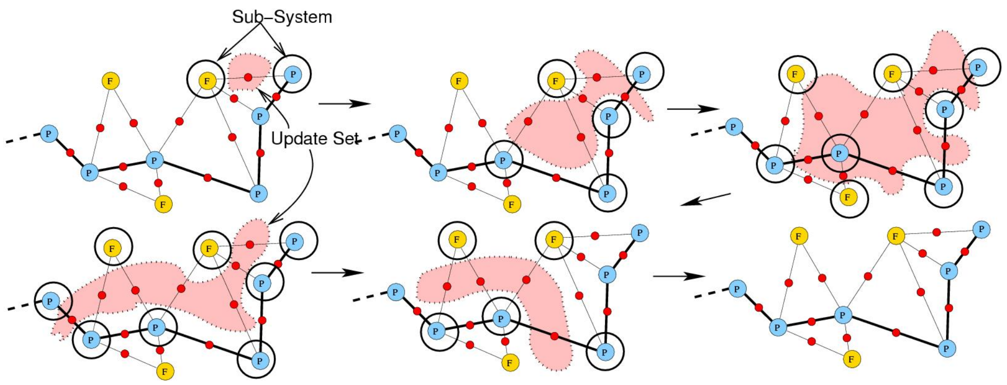

Figure 2: This figure illustrates the relaxation algorithm after adding an energy node. The energy nodes are the small dark (red) circles on the edges connecting two state nodes. Here the pose nodes are labeled with P (blue) and the feature nodes with F (yellow). The 'update set', shown shaded, is a set of energy nodes that have changed significantly. The 'sub-system', shown as the circled nodes, is the set of state nodes that are attached to the update set. We see the evolution of the two sets for six iterations of the algorithm.

```
1. Set the update set = the newly added energy node.
2. Form the sub-system of state nodes as those with an edge to the update set.
3. Relax the state nodes of the sub-system simultaneously.
4. Clear the update set
5. For all energy nodes attached to the sub-system;
     If the energy change was significant add to the update set.
6. If the update set is not empty goto 2.
   Else done.
```

As a test for a significant change in the energy of an energy node we have used a change greater than .01 in magnitude and 5% relative to the final energy of the node. The main advantages of this algorithm are its simplicity and its ability to spread the update quickly over the graph as is needed. One can solve the linearized subsystem exactly using QR or Cholesky factorization. QR has better numerical stability.

A second method of relaxing the graph is to grow a fixed number of state nodes from a seed node at the modified location in the graph. The way to grow the set can be via a simple breadth first search on the graph or to only add the pose nodes going back from the current position. The motivation for the latter approach is the simplicity of the resulting matrix. If only pose nodes along the path are included one has the optimal ordering in a block tri-diagonal matrix. This can be solved very quickly to adjust long sections of the path. The features can then be adjusted individually.

Using a breadth first search allows one to update both the features and poses together. This can be necessary especially when there are small loops in the path. It is generally harder to find the optimal ordering of the matrix in this case. More time is needed for the same number of nodes as compared to doing only pose nodes along the path. One can expect a nm^2 dependence where n is the total number of nodes and m is the 'average' amount of fill (discussed below) produced when solving.

When marginializing out a node from the graph we create links between all the nodes that were connected to the eliminated node. Some of these may have had direct links before, thus no new fill is produced. The nodes that were not connected before the elimination but were connected after constitute the fill produced by the elimination. The complexity of solving the system depends on the order of elimination. We use a greedy approach by starting from the end of the graph eliminating at each step the node that creates the least fill. This is very close to optimal if the robot is exploring new areas the whole time.

4 Graph Simplifications

We have presented the general graphical approach to SLAM and shown how one can drive the system to better solutions starting from a nearby state. The main difficulty that remains is how to solve when the solution state is far from the current state. In this situation the simple downhill relaxation will prove hopeless as the system finds a local minimum far from the global one. The problem is twofold. First the linearization of all the energies in the global frame will tend to hold the nodes fixed in that frame rather than fixed relative to their neighbors as the true system should be. Second the shear number of dimensions is too large for an effective search.

Our solution is based on a simplification of the graph by forming what we call star nodes. These then lead to an approximation to the true system that can be solved using local coordinates instead of global ones. Figure 3 shows the process of graph reduction by star nodes.

1. We start by selecting a single pose node in an older section of the graph.
2. We then linearize the energy nodes attached to this node and collect all the Hessian, (and Gradient), terms into matrices. These matrices have rows/columns for all the attached state nodes for these energy nodes.
3. We now temporarily move the state nodes to the minimum energy state for just these energy nodes, (detached from the rest of the graph).
4. We then repeat step 2 until there is no more movement.
5. We now have no gradient term as the state nodes are at the minimum.
6. We marginalize out the original pose node that has no other edges.

After moving the system to the minimum energy and marginalizing out the original pose node, the energy for the star node is the Taylor expansion around $\bar{\mathbf{x}}_{\mathbf{s}}$[6]:

[6]We have used $\mathbf{x}_s$ to indicate the coordinates of the all state nodes attached to the the star. This vector is in turn split into a base pose node subscript b and the remaining nodes subscript p.

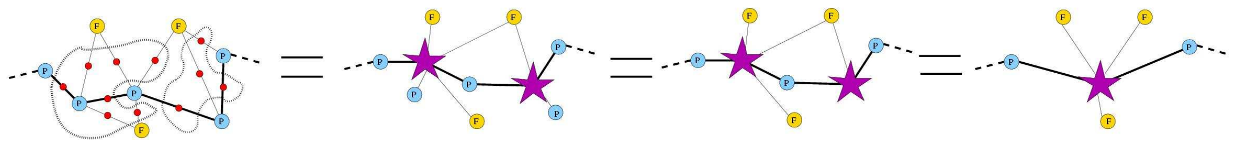

Figure 3: This is the process of star formation. We start by collecting the energy nodes attached to a pose node. These are linearized and the attached state nodes moved to the minimum energy of this sub-graph. The isolated pose nodes are marginalized out. The process can then be repeated using the star nodes as the energy nodes for a higher level reduction.

$$E = E(\bar{\mathbf{x}}_{\mathbf{s}}) + \begin{pmatrix} \delta\mathbf{x}_b^T & \delta\mathbf{x}_p^T \end{pmatrix} H_{ss}(\bar{\mathbf{x}}_{\mathbf{s}}) \cdot \begin{pmatrix} \delta\mathbf{x}_p \\ \delta\mathbf{x}_p \end{pmatrix} \tag{11}$$

The gradient term has vanished as we are at the minimum energy of this subgraph. We will make a coordinate transformation to the relative frame of one of the remaining pose nodes, which we call the base node. We can transform the Hessian matrix by the Jacobian of this transformation. Consider the transformation to the base frame at the current state $\bar{\mathbf{x}}_s$ and its linearization in terms of Jacobian matrices:

$$\bar{\mathbf{x}}_q = T(\bar{\mathbf{x}}_p|\bar{\mathbf{x}}_b). \qquad \delta\mathbf{x}_q = J_{qb}\delta\mathbf{x}_b + J_{qp}\delta\mathbf{x}_p. \tag{12}$$

Using this then allows us to write:

$$I = \begin{pmatrix} I & 0 \\ -J_{qp}^{-1}J_{qb} & J_{qp}^{-1} \end{pmatrix} \begin{pmatrix} I & 0 \\ J_{qb} & J_{qp} \end{pmatrix} \quad \rightarrow \quad \begin{pmatrix} \delta\mathbf{x}_b \\ \delta\mathbf{x}_q \end{pmatrix} = \begin{pmatrix} I & 0 \\ J_{qb} & J_{qp} \end{pmatrix} \cdot \begin{pmatrix} \delta\mathbf{x}_b \\ \delta\mathbf{x}_p \end{pmatrix} \tag{13}$$

We define $Q = J_{qp}^{-1}$ and $\widetilde{Q} = (J_{qb}, J_{qp})$. We can use this along with the invariance of the original subsystem to coordinate transformations to show[7]:

$$H_{ss}(\bar{\mathbf{x}}_s) = \widetilde{Q}(\bar{\mathbf{x}}_s)^T\{Q(\bar{\mathbf{x}}_s)^T H_{rr}(\bar{\mathbf{x}}_s)Q(\bar{\mathbf{x}}_s)\}\widetilde{Q}(\bar{\mathbf{x}}_s). \tag{14}$$

Now defining the Hessian in the base frame as:

$$H_{qq}(\bar{\mathbf{x}}_s) = Q(\bar{\mathbf{x}}_s)^T H_{ss}(\bar{\mathbf{x}}_s)Q(\bar{\mathbf{x}}_s). \quad \rightarrow \quad H_{ss}(\bar{\mathbf{x}}_s) = \widetilde{Q}(\bar{\mathbf{x}}_s)^T H_{qq}(\bar{\mathbf{x}}_s)\widetilde{Q}(\bar{\mathbf{x}}_s). \tag{15}$$

The H_{qq} matrix is the Hessian at the linearization point without the rows and columns of the base node rotated to the base frame. The Hessian might not have full rank as it includes

[7]When we write out the energy expansion in terms of $\delta\mathbf{x}_b$ and $\delta\mathbf{x}_q$, the terms that multiply $\delta\mathbf{x}_b$ have to vanish by the invariance to the base frame. That is to say, if we move the base node keeping the relative positions of all the other nodes to the base unchanged, $(\delta\mathbf{x}_q = 0)$,the energy will not change.

only part of the measurements for those feature nodes attached to this star. We want to have an explicitly stable representation so we do a singular value decomposition of H_{qq}.

$$H_{qq} = \sum_{k=1}^{m} \mathbf{U}_k \mu_k \mathbf{U}_k^T \tag{16}$$

Here k runs over the m positive eigenvalues μ_k and $\mathbf{U}_k$ are the eigenvectors. This lower dimensional matrix, H_{qq}, is thus forced to be positive definite. It is the projection of the original H_{ss} into the directions that actually contain information. So, for example, there is no information in the direction that corresponds to translation of all the states in the global frame. One can call this the observable subspace of the star node. Define the projection into the eigenspace:

$$\bar{u}_k = \mathbf{U}_k^T \bar{\mathbf{x}}_q, \quad u_k = \mathbf{U}_k^T \mathbf{x}_q, \quad \Delta u_k = u_k - \bar{u}_k \tag{17}$$

We can now write the energy for the star node in a compact invariant and stable form,

$$E = E(\mathbf{u}) + \sum_{k=1}^{m} \frac{\mu_k}{2} (\Delta u_k)^2 \tag{18}$$

We can also calculate the derivatives of the energy around any new state. So for example the gradient at $\mathbf{x}_s$ becomes

$$G_s(\mathbf{x}_s) = \sum_{k=1}^{m} \Delta u_k(\mathbf{x}_s) \mu_k U_k^T \widetilde{Q}(\mathbf{x}_s). \tag{19}$$

The Hessian is similar. The eigenvalues and eigenvectors are saved along with $\bar{u}_k$. These can then be used to find the energy for any new state. Notice that the linearization here will be exactly correct if the subgraph is later rigidly rotated and translated due to global constraints being applied. It is only relative shifts between the states nodes of the star that will result in linearization errors. These shifts will be small if the states are tightly constrained by the star node.

The stars are not formed near the current pose node. This part of the graph is being updated and will change as new information is added. It is then possible to wait to initialize features until having observed them for some time. When the feature is finally initialized one can add all the observations going back along the graph. Thus there is no hurry to use the measurements. After forming the star nodes however such retrospective adjustments are no longer possible. That is one reason to wait to form stars until the robot has left that part of the environment.

We maintain a length of exact energy nodes with dense pose nodes behind the current pose node, along the robot path. We call this the tail. The tail contains those pose nodes attached to currently observed features. Beyond the tail we form stars. We start forming stars from all the energy nodes connected to a selected pose node. We then skip a pose node and repeat. Then by choosing the pose node between the two new stars we can form a larger star as in figure 3. This process continues until we feel the star should not become larger. In practice this was when we had eliminated 64 pose nodes. When combining two stars we only combine stars of the same level, that is to say stars that have eliminated the same number of pose nodes.

There are two reasons for limiting the size of the stars. First, as mentioned, we want the star's states to be tightly coupled to avoid large linearization errors. Second we want to be able to easily solve the system. Too large a star might make this calculation difficult. Thus, we leave some pose nodes in the graph and do not reduce the graph completely. The system remains sparse. This sparseness is exact, in contrast to Sparse Extended Information Filters where spareness is tied to a loss of information (Lui and Thrun, 2003; Thrun et al., 2004; Eustice et al., 2005).

4.1 Global Constraints

We now show how the star nodes can be used to efficiently solve the global constraints on the graph. As we have stated, the global constraints need to be solved using relative coordinates, global frame coordinates will not work. We notice that the star nodes already have an explicit expansion of the Hessian in terms of the relative coordinates $\mathbf{x}_q$. The only problem being the existence of the many shared features between adjacent star nodes. The relative coordinates of these are different in the two base frames. We get around this by observing that had we simply not considered these features as being the same in the different stars we would not have any problem. We therefore temporarily make this approximation, which creates a system of star nodes that are only coupled through their common pose node. Notice that we do not remove the large constraint of the feature within a single star. We do remove a usually much smaller amount of constraint that the feature gives between two stars.

This system will then be trivial to solve as we now will show. We start by removing the tail from the graph. Thus the only energy nodes we have are star nodes. Afterwards, we can re-attach the tail keeping its relative position to the attachment point the same.

The loops in the graph around the pose path give rise to stars that have three or more pose nodes[8]. These then give constraints which are easily found by traversing the graph along pose star edges. Each pose node that is not a base node for some star node is a marker for a loop constraint. In addition, we may have explicitly just discovered a new constraint not yet built into the graph. All these constraints then give rise to an equation that can be written:

$$\sum_{i \in stars} C_i \delta \mathbf{x}_{iq} = \mathbf{c}. \tag{20}$$

Here we have cut the tail off the graph completely and only have star nodes. The C_i are

[8] Whenever a larger loop in the path is discovered we form a star node out of two star nodes, one from each end of the loop.

matrices that have no columns for the feature parts of the stars and a row for each pose dimension and each loop. These are initially calculated around the current state. Our goal is to find a change to the current pose states $\delta\mathbf{x}_{iq}$ that will satisfy this equation without raising the energy too much. We naturally introduce a Lagrange multiplier, γ.

$$\sum_{i \in stars} \delta\mathbf{x}_{iq}^T H_{iqq} \delta\mathbf{x}_{iq} + \gamma(\sum_{i \in stars} C_i \delta\mathbf{x}_{iq} - \mathbf{c}). \tag{21}$$

$$\sum_{i \in stars} \begin{pmatrix} H_{iqq} & C_i^T \\ C_i & 0 \end{pmatrix} \begin{pmatrix} \delta\mathbf{x}_{iq} \\ \gamma \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{c} \end{pmatrix}. \tag{22}$$

$$\delta\mathbf{x}_{iq} = -(H_{iqq}^{-1} C_i^T)\gamma. \tag{23}$$

$$\gamma = -\left(\sum_{i \in stars} C_i H_{iqq}^{-1} C_i^T \right)^{-1} \mathbf{c}. \tag{24}$$

Starting with this last equation we note that the C do not multiply the feature parts of H^{-1}. Therefore we need not bother calculating these rows to get γ. Furthermore we have the eigenvector decomposition of H_{iqq} stored in the star nodes so that this last equation is very easy to compute. We also notice that although this does formally give us the change to the star feature in the equation for $\delta\mathbf{x}_{iq}$ we will get different feature solutions for each star the feature is attached to. We therefore set the features separately. The result is we can ignore the existence of the features completely here in solving for the pose node states.

We must repeat this procedure a number of times to correct for the linearization errors in the constraint equations. Once the pose states are known we go back and relax all the features.

5 Implementation Considerations

Up to now we have discussed feature measurements in a rather abstract way. We have assumed that the features in the map have some representation and that the representation can be transformed into different frames of reference. We also assumed that the measurements will then give us some information on the features relative to the robot at the time of the observation.

In our experiments outdoors, we have used SICK laser scanners and wall features. The feature detection consists of a search for lines in the range scans. These lines then give us at least two numerical measurements, the angle and perpendicular distance to the wall. In some cases one can also detect the endpoint of the wall. This happens when the robot has a clear view around the corner of the wall. In other cases one has information on where the wall is but not where it is not. This then leads to partial observation of the wall.

It is important that we represent the walls in a way that allows for partial observations and explicit frames of reference. We also need to be able to estimate the pitch and roll of the robot as it moves or we will not be able to make accurate maps outdoors.

To extract lines from the SICK scans we have used a Hough Transform on the scan data. One can use any line extraction but the Hough Transform has the the advantage of finding all the lines deterministicly. If it is properly optimized for the application it is at least as fast as another line extraction (Folkesson, 2005).

One further detail of line detection with the SICK scanner is the finite beam width of the laser. Modeling the beam as having 0 width will lead to bias when observing a wall from a robot traveling along the wall keeping a close distance to the wall and looking ahead, the typical situation. Unfortunately, the effective beam width of the reflected laser light from the wall depends on many factors, such as distance and reflectivity. The reflectivity of the wall cannot be well predicted. At any rate, we found that for the LMS291 a constant .02 radian beam width gave reasonably good results in these environments. Better models might be a good idea though.

6 Experimental Platforms

The approach presented above has been evaluated in a number of out-door settings. All of the experiments have been performed with an iRobot All Terrain Robot Vehicle (ATRV-2). The robot has no differential and direct control of all 4 wheels, which implies that the resulting steering is skid-based. The skid steering introduces significant drift in the odometric feedback. In addition the vehicle has inflated wheels which in general result in a significant component of systematic drift in the odometry.

To compensate for the odometric drift an on-board inertial measurement unit, CrossBow DMU-6x, is used for generation of independent estimation of ego-motion. A combined odometry/INS estimation of ego-motion is feed to the SLAM system.

As already mentioned in the previous section, for estimation of environmental features only the on-board SICK LMS 291 laser scanner is used. The laser scanner is mounted to have a scan surface that is parallel to the horizontal plane.

7 Evaluation

We will present here results in two outdoor settings using our ATRV robot and the SICK scanner to detect walls of the buildings. We have kept the tuning parameters of error models, feature detection, matching and initialization constant for all our experiments.

7.1 Experiment 1 - 100 meter Square with One Loop

The first setting was around an approximately square building measuring about 70 meters to each side. The area around the building comprises some smaller buildings, parked cars and fences. These have some usable wall features as well as some things that look like walls but should not be used, ie. hedges and fences. For this building we have a hand-made map to compare to..

In the top left figure of Fig. 4 we show the results for the Extended Kalman Filter on this data. The robot has explored this building by circling it counter-clockwise starting from 3 o'clock. One can see the typical problem of accumulation of errors which become quite large by the time the robot completes the loop. The total number of SICK laser scans (at 5 Hz) is 8,035 (181 segments). This EKF map took only 25 sec to compute off-line.

In the top right figure of Fig. 4 we show the results of the full Graphical SLAM solution on this same data. The results are noticeably better due to the more flexible data association provided by the graphical method. We show here all the 8,035 pose nodes of the graph. This map took 159 seconds to calculate off-line. This is slower than the EKF map but much less than the 1,607 seconds it took to collect the data.

Figure 4, second row shows the graph simplification and loop closing results. This map took 130 seconds which includes the time for detection and calculation of the loop closing. One can see that after closing the loop we have a consistent map which aligns fairly well with the hand made map.

7.2 Experiment 2 - 100 meter Square with Three Loops

We have a second, more challenging, dataset from this site. The robot was driven by hand around three loops passing at times through the interior of the building. Fig. 5 top left shows the EKF map which took 85 seconds (8,749 scans, 361 segments). It is essentially a useless map as the large error on completing the first loop lead to a false matching inside the corridor.

Next in Figure 5 we show the result of detecting and closing the three loops using the graphical SLAM method. This took 486 seconds but lead to a consistent and usable map.

7.3 Experiment 3 - 170 x 260 meters with Three Loops

The final experiment is on the KTH campus. Here we drove the robot by hand for about 1 hour (14,215 scans of 361 segments). There were a number of times that the robot was driving blind as the SICK scanner had to reboot due to direct sunlight shining into it. The final map has 312 walls and 3 loops were closed. This is summarized in Figures 6, 7 and 8. The map took 1,122 seconds, (18m42s) to make which included detection of the loops. We have not discussed our detection of the loops as that is a separate algorithm (Folkesson, 2005). It was, however, part of the experiments. This automatic detection was much more time intensive here as there were many possible combinations of wall matches to check. This check was done every 100 iterations and included all nearby features that were far apart along the edges of the graph. The EKF solution on this data was again inconsistent but only took 313 seconds, without any special loop detection running.

Let us emphasize here that the parameters were not changed between these experiments. Had we tuned the parmeters for each dataset slightly better results might have been achieved but we would not have gotten real validation. We have been working with various implementations of SLAM and graphical SLAM for some time. The values that worked were not any real surprise.

The final map, Fig. 9 compares well with a photograph of the campus from above. We can also notice that the final robot position, Fig. 8, is indeed correct as we see it prepared to re-enter the door it started from. The larger error in the lower right corner was due to direct sunlight dazzling the SICK while it turned south. This happens at several points along the path but here it is combined with significant odometry error.

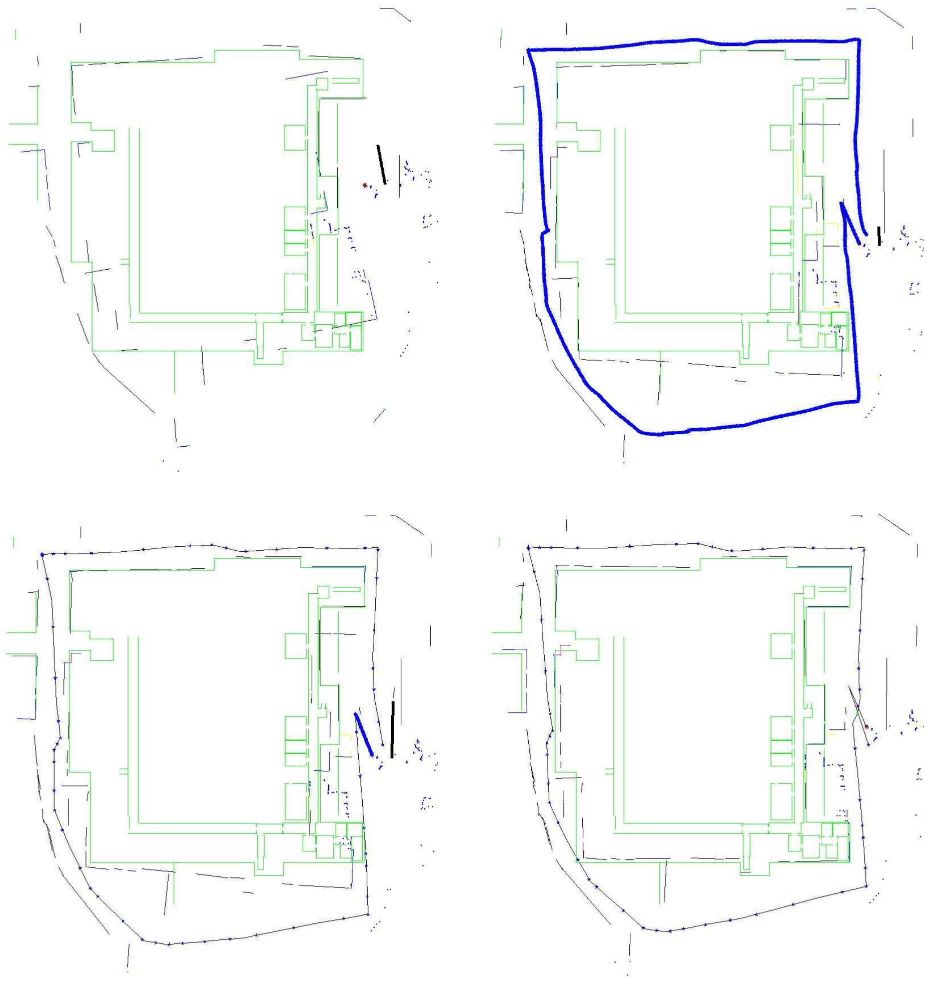

Figure 4: Here we show the first experiment with the EKF solution on top left and the full Graph solution without star node simplification on lines are the hand made map of the building which can be compared to the darker, (black) lines of the SLAM map. The pose nodes of the graph are also shown in the right and below figures. In the second row we show the results of using the star nodes to simplify the graph and close the loop. We see the graph just as the loop is detected on the left and just after the loop closing is performed right.

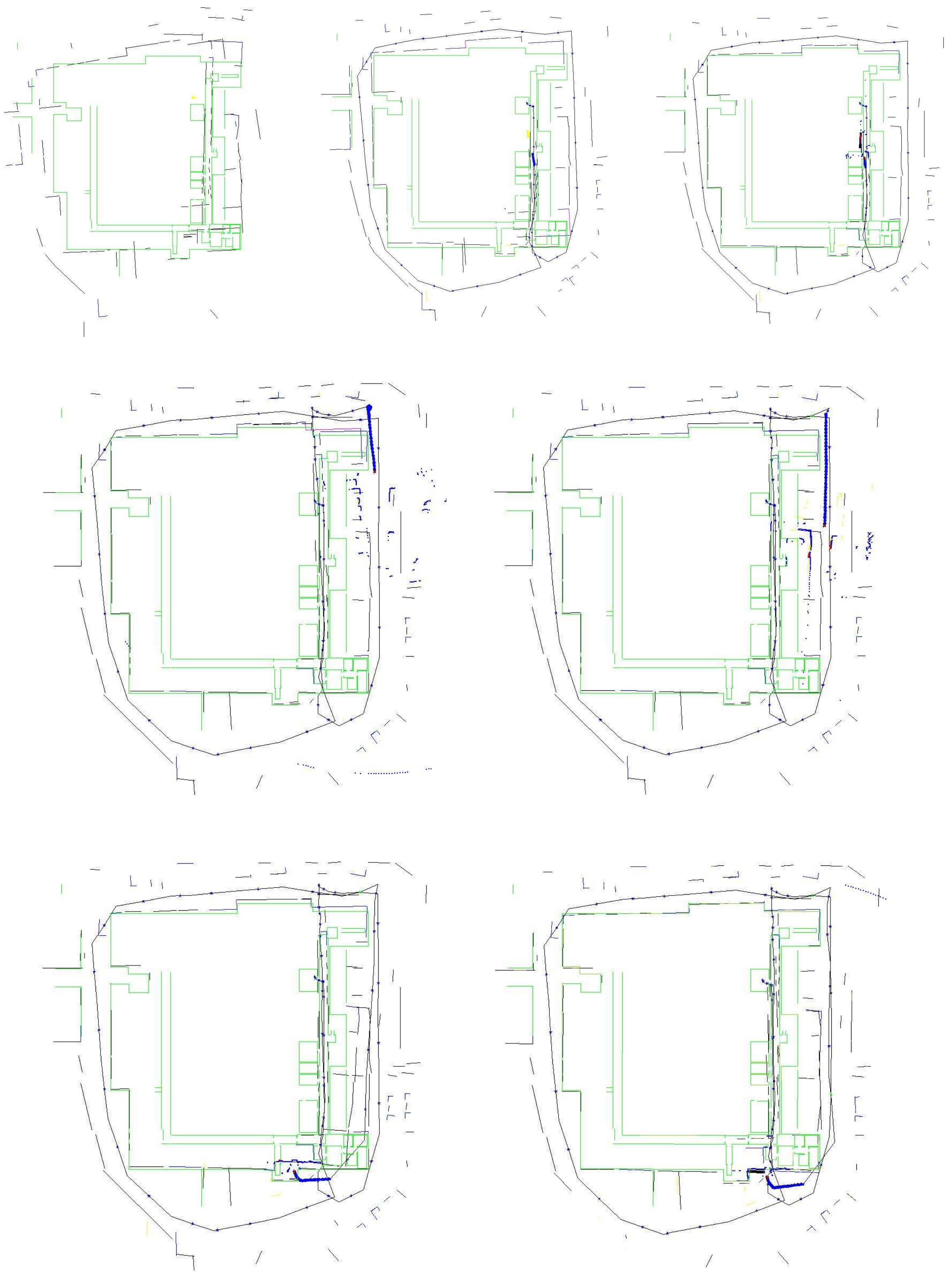

Figure 5: Here we show the second data using the EKF solution (left). The features were matched incorrectly on closing the first loop leading to a hopelessly inconsistent map. The Graphical Slam solution is shown in the center and right. We see the graph just as the loop is detected in the center top and just after the loop closing is performed, left top. The second and third loop closings are shown in the second and third rows of figures. Notice how the map is converging to the hand made map.

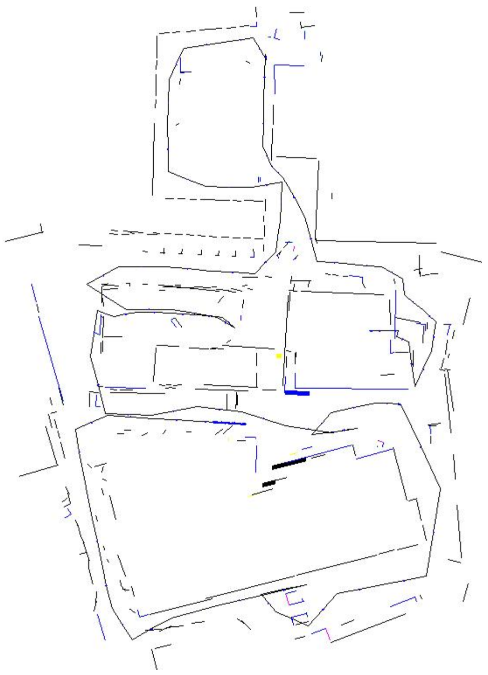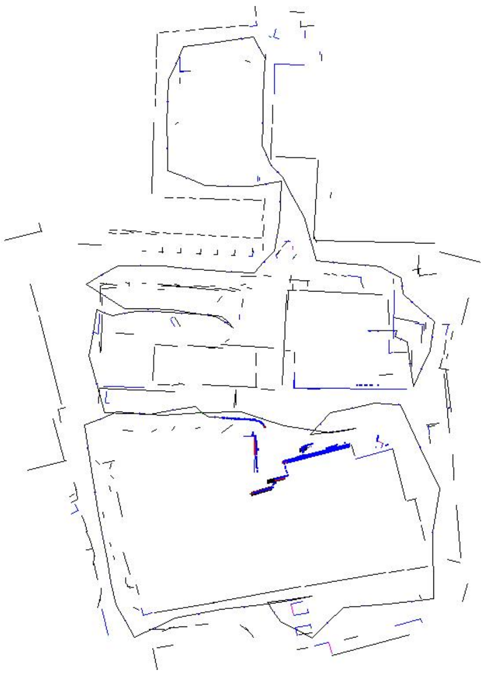

Figure 6: Here we show the third data set closing the first loop. We see the graph just as the loop is detected at the left and just after the loop closing is performed to the right.

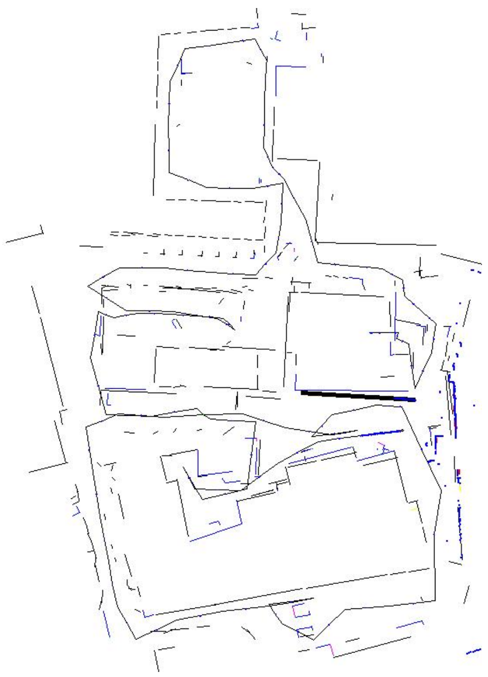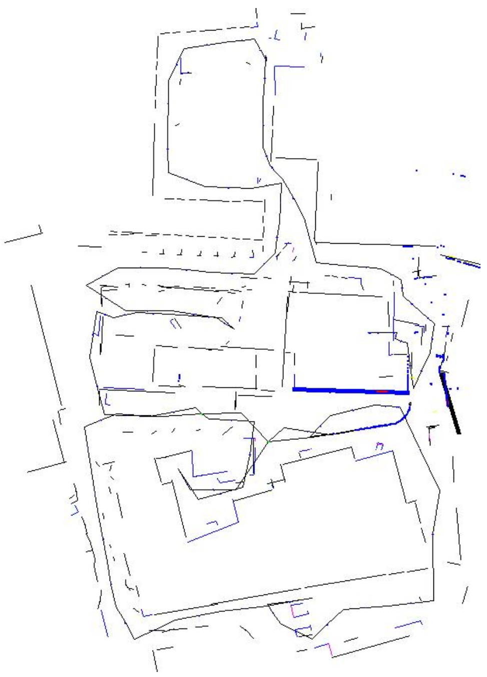

Figure 7: Here we show the third data set closing the second loop. We see the graph just as the loop is detected, left and just after the loop closing is performed, right.

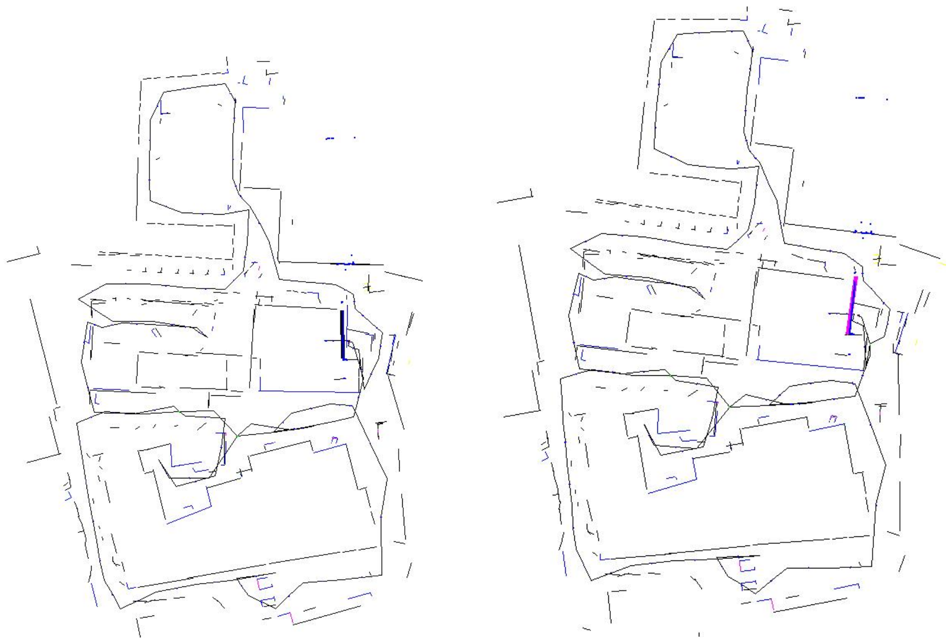

Figure 8: Here we show the third data set closing the third loop. We see the graph just as the loop is detected st the left and just after the loop closing is performed, right. Notice that the robot (in red) is correctly positioned outside the door to the lab, which it came though at the start.

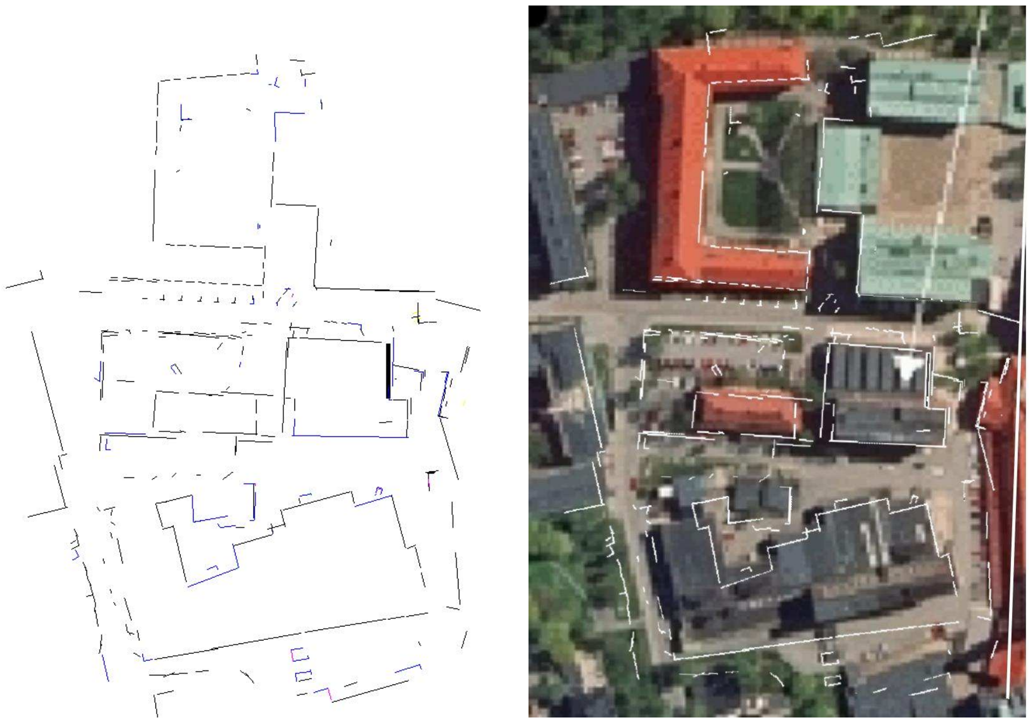

Figure 9: Here we compare our graphical SLAM solution to an aerial photo of the same area.

8 Discussion

It is always hard to quantify the quality of a SLAM solution. For example some of the EKF generated maps here suffered catastrophic failures at specific points. Had these somehow been avoided the maps would have looked much better. Typically a number of small errors can have very large effect due to the leverage of long robot trajectories. One needs to look at both local consistency and global consistency to get a true picture of the performance. Locally EKF is quite good, so long as no data association errors are made. The EKF maps cannot be scaled up to very large sizes mostly because of the linearisation in a global frame of reference.

Our graphical approach has the important advantage of being able to reuse the original measurements or the summarized local representation of these in star nodes. This then makes it theoretically scalable to very large maps. We have shown that it can make maps of data that we could not map with the EKF.

Having the original measurements available for a longer time also allows the delayed inclusion of data into the map. One can add or remove the data right up until it is included in some star node. This more flexible use of data is an advantage of our method. There are other ways to achieve this such as delayed filtering or using a filter with a short history of recent poses. However this flexibility is achieved, it is essential for a robust SLAM algorithm. It explains much of the improvement of our maps over the EKF maps before loop closure.

Others investigations of graphical representations to the SLAM problem include (Duckett et al., 2000; Frese and Duckett, 2003). These graphs represent a significantly simplified model of the underlying probabilities as compared to what is presented here. The measurement models are linearized in the same way as in an EKF. No effort is made to minimize the effects of this linearization, though the use of local frames of reference for instance. The relaxation method is also different than what we present here. Our method is rather more adaptive to the amount of tension in each edge of the graph.

Early graphs used for loop closing are found in (Gutmann and Konolige, 1999; Lu and Milios, 1997a). These methods were a source of inspiration to this work. Those graphs were built up from scan matching while we use a feature based map.

The issue of computational complexity has received much attention in SLAM. The EKF has a quadratic complexity with the number of features in the map. Our method has a complexity which is harder to be precise about. The individual updates can be constant time. This is because they normally involve only local relaxation of the graph. In our experiments we relaxed the graph quite hard resulting in computation times significantly longer than EKF. We could have chosen to relax less and get average updates about as fast as EKF. The maps do suffer but in a gradual way, not catastrophic. The global updates on closing a loop do of course depend on the size of the graph. The loop closing calculations are all formally linear in the size of the graph. One might be inclined to claim linear complexity for the whole process but we will not. The reason is that as the topology of the graph becomes more complex, (many loops), it takes longer to relax the graph satisfactorily. This observation is not easily quantified but prevents us from making the linear complexity claim.

So although it takes a linear time to relax the graph, (using for instance a single relaxation on each node or breaking the graph up into overlapping subsystems and relaxing each of these exactly), the number of times one must do this relax seems to depend on the topology and thus the size of the graph.

We should also say that although our method works well it is not infallible. Just as with all SLAM algorithms changes to the tuning of the many parameters in the models may cause dramatic differences. Even the timing of the loop closures can make the maps change dramatically. The fact that we do not re-tune the parameters between the different data sets is important in this context.

9 Conclusion

SLAM is by now considered a mature problem in robotics. The basic formulation of the problem is well-known and the objective for a system is well-characterized. That does not imply that this is an entirely solved problem. As mentioned in the introduction some of the challenges are convergence in the presence of non-linearities, introduction of global constraints, management of computational complexity, and flexible handling of data-association.

In this paper a graphical approach to SLAM was presented which addresses these issues. Using the graph formulation it is easy to test an association. It merely adds a new arc to the graph which can be removed if it leads to a worse solution. The criteria for determining the quality of the solution is the energy. This energy is unambiguously given by the probabilities underlying the measurements.

Non-linearities are computed exactly with no approximation until enough measurements have been included to assure a good linearization point. The star node is the construction for the linearization. It linearizes the measurements in a local frame. The non-linear transformations of the base frame of the star continue to be handled exactly. It is only the much smaller local movements between nearby features that are linearized. The star node make it possible to sub-divide the graph into largely independent components which facilitates real-time operation.

Global constraints are solved by making the approximation that the star nodes are 'independent' in the sense that any feature common to two stars is temporaily considered to be two separate features. The equations then become largely decoupled. This then allows the minimum energy state to be found quickly.

The presented framework for graphical SLAM has been presented but also demonstrated for a number of different environments where the performance in terms of mapping in comparison to a standard EKF was shown. The introduction and handling of global constraints was also demonstrated. The framework has generated results that clearly demonstrates it utility for mapping and estimation in rich environments. Future / current work includes use of the framework in sparse environments, integration of a richer variations in features, and integration of the SLAM framework with representations that are more topological by nature.

Acknowledgments

This research has been sponsored by the Swedish Foundation for Strategic Research through the Centre for Autonomous Systems.

References

Bar-Shalom, Y. and Fortmann, T. (1987). *Tracking and Data Association.* Academic Press, New York, NY.

Bekey, G. A. (2005). *Autonomous Robots.* Intelligent Robotics and Autonomous Agents. MIT Press, Cambridge, MA.

Bosse, M., Newman, P., Leonard, J., and Teller, S. (2004). Slam in large-scale cyclic environments using the atlas framework. *IJRR*, page 2004.

Castellanos, J. A., Neira, J., and Tardós, J. D. (2004a). Limits to the consistency of the EKF-based SLAM. In *Intelligent Autonomous Vehicles (IAV-2004), M. I. Ribeiro and J. Santos-Victor, Eds., IFAC/EURON, IFAC/Elsevier.*

Castellanos, M. Devy, J., and Tardós, J. D. (1999). Towards a topoligical representation of indood environments: A landmark-based approach. In *IEEE Conf.on Intelligent Robots and Systems IROS99.*

Castellanos, J. A., Neira, J., and Tardos, J. D. (2004b). Limits to the consistencyof the EKF-based SLAM. In Ribeiro, M. I. and Santos-Victor, J., editors, *Intelligent Autonomous Vehicles (IAV-2004)*, Lisboa, PT. IFAC/EURON, IFAC/Elsevier.

Chatila, R. and Laumond, J.-P. (1985a). Position referencing and consistent world modeling for mobile robots. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA'85)*, pages 138–145.

Chatila, R. and Laumond, J. P. (1985b). Position referencing and consistent world modeling for mobile robots. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA93)*, volume 1, pages 138–145.

Dellaert, F. (2005). Square root sam: Simultaneous location and mapping via square root information smoothing. In *Robotics: Science and Systems.*

Duckett, T., Marsland, S., and Shapiro, J. (2000). Learning globally consistent maps by relaxation. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA00).*

Eustice, R., Walter, M., and Leonard, J. (2005). Sparse extended information filters: Insights into sparsification. In *Proc. of the IEEE/RSJ International Conference on Intellegent Robots and Systems (IROS05).*

Folkesson, J. (2005). *Simultaineous Localization and Mapping with Robots.* PhD thesis, Computer Science and Communication, Royal Institute of Technology, SE-100 44 Stockholm, Sweden.

Folkesson, J. and Christensen, H. I. (2003). Outdoor exploration and SLAM using a compressed filter. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA03)*, volume 1, pages 419–427.

Folkesson, J. and Christensen, H. I. (2004). Robust SLAM. In *Proc. ofthe IFAC Symposium on Intelligent Autonomous Vehicles (IAV2004)*, volume 1.

Frese, U. and Duckett, T. (2003). A multigrid approach for accelerating relaxation-based SLAM. In *"Proc. of the IJCAI-03 Workshop on Reasoning with Uncertainty in Robotics (avail at http://www.aass.oru.se/Agora/RUR03/")*.

Guivant, J. E. and Nebot, E. (2001a). Optimization of the simultaneous localization and map-building algorithm for real-time implementation. *IEEE Transactions on Robotics and Automation,*, 17(3):242–257.

Guivant, J. E. and Nebot, E. M. (2001b). Optimization of the simultaneous localization and map-building algorithm for real-time implementation. *IEEE Transactions on Robotics and Automation*, 17(3):242–257.

Guivant, J. E., Nebot, E. M., Nieto, J., and Masson, F. (2004). Navigation and mapping in large unstructured environments. *IJRR*, 23(4).

Gutmann, J. and Konolige, K. (1999). Incremental mapping of large cyclic environments. In *Proc. of the 1999 IEEE International Symposium on Computational Intelligence in Robotics and Automation*, volume 1, pages 318–325.

Hähnel, D., Burgard, W., Fox, D., and Thrun, S. (2003). An efficient FastSLAM algorithm for generating maps of large-scale cyclic environments form raw laser range measurements. In *Proc. of the IEEE International Conference on Intelligent Robots and Systems*, volume 1, pages 206–211.

Jensen, F. V. (2001). *Bayesian Networks and Decision Graphs*. Springer Verlag, New York, NY.

Jensfelt, P. and Kristensen, S. (2001). Active global localisation for a mobile robot using multiple hypothesis tracking. *IEEE Transactions on Robotics and Automation*.

Julier, S. J. and Uhlmann, J. K. (2001). A counter examplee to the theory of simultaneous localization and map building. In *Proc. of the IEEE International Conference on Robotics and Automation*, pages 4238–4243.

Jung, I. K. and Lacroix, S. (2003). High resolution terrain mapping using low altitude aerial stereo imagery. In *ICCV*, Nice (F). IEEE.

Kim, J. and Sukkarieh, S. (2004). Autonomous airborne navigation in unknown terrain environments. *IEEE Transactions on Aeroscape and Electronic Systems*, 40:1031–1045.

Leonard, J. and Newman, P. (2003). Consistent, convergent and constant-time SLAM. In *Proc. of the 18th Joint Conference on Artificial Intelligence (IJCAI-03)*, San Francisco, CA. Morgan Kaufmann Publishers.

Leonard, J. J., Durrant-Whyte, H. F., and Cox, I. J. (1992). Dynamic map building for an autonomous mobile robot. *The International Journal of Robotics Research*, 11:286–298.

Lu, F. and Milios, E. (1997a). Globally consistant range scan alignment for environmental mapping. *Autonomous Robots*, 4:333–349.

Lu, F. and Milios, E. (1997b). Globally consistent range scan alignment for environmental mapping. *Autonomous Robots*, 4:333–349.

Lui, Y. and Thrun, S. (2003). Results for outdoor-SLAM using sparse extended information filters. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA03)*, volume 1, pages 1227–1233.

Montemerlo, M. and Thrun, S. (2003). Simultaneous localization and mapping with unknown data association using FastSLAM. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA03)*, volume 1, pages 1985–1991.

Montemerlo, M. and Thrun, S. (2006). Large-scale robotics 3-D mapping of urban structures. In Jr., M. H. A. and Khatib, O., editors, *Experimental Robotics IX*, volume 21 of *STAR*, pages 141–150, Heidelberg, DE. Springer Verlag.

Montemerlo, M., Thrun, S., Koller, D., and Wegbreit, B. (2002). FastSLAM: A factored solution to the simultaneous localization and mapping problem. In *Proc. of the National Conference on Artificial Intelligence (AAAI-02)*, Edmonton, Canada.

Newman, P. and and R. Rikovski, J. L. (2003). Towards constant-time slam on an autonomous underwater vehicle using synthetic aperture sonar. In *ISRR*, Sienna.

Newman, P., Cole, D., and Ho, K. (2006). Outdoor SLAM using visual appearance and laser ranging. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA06)*, volume 1, pages 1180–1188.

Newman, P., Leonard, J., and Rikoski, R. (2003). Towards constant-time slam on an autonomous underwater vehicle using synthetic aperture sonar. In *Proc. of the International Symposyum on Robotics Research (ISRR03)*.

Nieto, J., Guivant, J. E., and Nebot, E. (2003). Real time data association for FastSLAM. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA03)*.

Paskin, M. A. (2003). Thin junction tree filters for simultaneous localization and mapping. In Gottlob, G. and Walsh, T., editors, *Proc. of the 18th Joint Conference on Artificial Intelligence (IJCAI-03)*, pages 1157–1164, San Francisco, CA. Morgan Kaufmann Publishers.

Smith, R., Self, M., and Cheeseman, P. (1987). A stochastic map for uncertain spatial relationships. In *4th International Symposium on Robotics Research*.

Stachniss, C., Hähnel, D., and Burgard, W. (2004). Exploration with active loop-clossing for FastSLAM. In *Proc. of the IEEE International Conference on Intelligent Robots and Systems*, volume 1.

Tardós, J., Neira, J., Newman, P., and Leonard, J. (2002). Robust mapping and localization in indoor environments using sonar data. *International Journal of Robotics Research,*.

Thrun, S., Burgard, W., and Fox, D. (2000). A real-time algorithm for mobile robot mapping with applications to multi-robot and 3D mapping. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA'00)*, volume 1, pages 321–328, San Francisco, CA, USA.

Thrun, S., Fox., D., and Burgard, W. (1998). A probalistic approach to concurrent mapping and localization for mobile robots. *Autonomous Robots*, 5:253–271.

Thrun, S., Fox, D., Burgard, W., and Dellaert, F. (2001). Robust monte carlo localisation for mobile robots. *Artificial Intelligence*, 128(1–2):99–142.

Thrun, S., Liu, Y., Koller, D., Ng, A., Z.Ghahramani, and Durrant-Whyte, H. (2004). Simultaneous localization and mapping with sparse extended information filters. *International Journal of Robotics Research*, 23(8):690–717.

Wijk, O. and Christensen, H. (2000). Triangulation based fusion of sonar data with application in robot pose tracking. *IEEE Transaction on Robotics and Automation*, 16:740–752.