A Multilevel Relaxation Algorithm for Simultaneous Localisation and Mapping

Udo Frese
Bremen Institute of Safe Systems
BISS, TZI, FB3
University of Bremen
D-28334 Bremen, Germany
Email: ufrese@informatik.uni-bremen.de
Per Larsson
NamaTec AB
Box 421
SE-69127 Karlskoga, Sweden
Email: per.larsson@gmail.com
Tom Duckett
AASS, Dept. of Technology
Örebro University
SE-70182 Örebro, Sweden
Email: tom.duckett@tech.oru.se

***Abstract*— This paper addresses the problem of simultaneous localisation and mapping (SLAM) by a mobile robot. An incremental SLAM algorithm is introduced that is derived from multigrid methods used for solving partial differential equations. The approach improves on the performance of previous relaxation methods for robot mapping because it optimizes the map at multiple levels of resolution. The resulting algorithm has an update time that is linear in the number of estimated features for typical indoor environments, even when closing very large loops, and offers advantages in handling non-linearities compared to other SLAM algorithms. Experimental comparisons with alternative algorithms using two well-known data sets and mapping results on a real robot are also presented.**

***Index Terms*— mobile robot navigation, SLAM, metric-topological maps, Gauss-Seidel relaxation, Galerkin multigrid.**

I. Introduction

A. Motivation

To navigate in unknown environments, an autonomous robot requires the ability to build its own map while maintaining an estimate of its own position. The problem of simultaneous localisation and mapping (SLAM) has received much attention in recent years, and has been described as the autonomous answer to the question "where am I?" [1]. The SLAM problem is hard because the same sensor data must be used for both mapping and localisation. We can separate two major sources of uncertainty in solving this problem:

$(i.)$ the *continuous* uncertainty in the positions of the robot and observed environmental features, and
$(ii.)$ the *discrete* uncertainty in the identification and re-identification of environmental features (data association).

Any approach to the SLAM problem that considers both types of uncertainty must somehow search the space of possible *maps*, since alternative assignments in data association can produce very different maps.

Our approach belongs to a family of techniques where the environment is represented by a graph of spatial relations between reference frames that is obtained by scan matching [2], [3]. With this approach, it is natural to separate the topological (discrete) and geometric (continuous) elements of the representation, and to consider tracking the M most likely topological hypotheses as a practical solution to the SLAM problem. Alternative topological hypotheses generally correspond to decisions over whether or not to "close a loop", based on the uncertainty in the re-identification of previously mapped features. The key problem here is that to evaluate the likelihood of one single hypothesis, a large linear equation system has to be solved in order to infer the most likely geometric representation given a particular topology.

In the rest of this paper, we only consider how to solve the equation system for a single topological hypothesis, though our solution to this problem is motivated by the computational requirements for a full SLAM algorithm that considers both types of uncertainty. In the conclusion (Section VII), we discuss how to embed the new algorithm within a framework for tracking multiple topological hypotheses.

B. Requirements of an Ideal SLAM Algorithm

The difficulty in SLAM arises from the fact that sensor measurements are never exact but always subject to measurement noise. However, leaving aside the question of uncertain data association for now, there exists an optimal solution for

SLAM under reasonable assumptions: the measurement noise is assumed to be independent and drawn from a Gaussian distribution with known covariance. The map with the largest likelihood, the Maximum Likelihood Estimate (ML-estimate) is then the best estimate possible. The optimal map can be computed by solving the linear equation system obtained from the measurements. A standard least squares algorithm such as Levenberg-Marquardt [4] could be applied. However, this approach would not be practical for real-time operation in large environments: for n landmarks and p robot poses the computational cost of the Levenberg-Marquardt algorithm is $O((n+p)^3)$.

Another important issue in SLAM is non-linearity. The equations involved are non-linear, mainly in the robot orientation ϕ, which occurs as a $\begin{pmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{pmatrix}$ rotation matrix. Non-linearity is usually treated by linearization. Severe linearization errors result if the error in the robot orientation exceeds $\approx 15°$. Some algorithms can solve this problem by relinearizing, i.e., updating the linearization point and recomputing all measurement Jacobians once the estimate changes (see Section II).

In general three criteria are important to assess the performance of a SLAM algorithm, concerning respectively *map quality*, *storage space* and *computation time*. The following requirements for an ideal SLAM algorithm were identified by Frese and Hirzinger [5]:

R1. *Bounded uncertainty:* the uncertainty of any aspect of the map should not be much larger than the minimal uncertainty that could be theoretically derived from the measurements.

R2. *Linear storage space:* the storage space of a map covering a large area should be at most linear in the number of landmarks ($O(n)$).

R3. *Linear update cost:* incorporating a measurement into a map covering a large area should have a computational cost at most linear in the number of landmarks ($O(n)$).

In summary, requirement R1 states that the map should represent nearly all information contained in the measurements, thus binding the map to reality. R2 and R3 concern efficiency, requiring linear space and time consumption. There is a subtle tradeoff between R1 and R3 since in general approximations make processing faster but the resulting map less precise. R1 implies the ability to close loops, because any break in the mapped representation of free space would prevent the robot from navigating further. The most important requirement from a practical perspective is R3, limiting the amount of time spent on each new measurement. So our goal for the algorithm described here was to achieve linear computation time while being able to instantaneously close loops.

C. Accelerating Relaxation-based SLAM

One method for solving the linear equation system in SLAM is to apply Gauss-Seidel relaxation [4], an iterative procedure that is equivalent to Gibbs sampling at zero temperature [6], as first applied to the SLAM problem in [7]. The basic principle is to optimize the pose for each frame in turn, based on the local relations to the connected frames, i.e., "pick a node and move it to where its neighbours think it should be". One iteration consists of updating every frame once by this rule. The method is useful for real-time SLAM applications because the computation is distributed while the robot is moving, instead of resolving the equation system from scratch at each iteration. It has been applied and extended by a number of authors [8], [9], [10], [11], [12], [13], [14].

Relaxation performs well on requirement R1, enabling relinearization and converging to the non-linear maximum likelihood solution when required (see Section III), and also R2. With respect to R3, relaxation is computationally cheap, requiring $O(1)$ iterations at $O(n)$ cost for incremental map building in most situations [8]. However, the method is slow to converge when closing large loops (i.e., when returning to a previously visited location after traversing a large cycle), because the accumulated error must be back-propagated and corrected throughout the rest of the map. In general, Gauss-Seidel relaxation requires $O(n)$ iterations, i.e., $O(n^2)$ time to reduce the equation error by a constant factor [4, §19.5].

The contribution of this paper is a new SLAM algorithm called 'Multilevel Relaxation' that is an order of magnitude faster than standard relaxation at closing loops, enabling map building to be performed on-line in linear time for all situations. This algorithm is based on so-called multigrid methods for solving partial differential equations [15], and accelerates the convergence of Gauss-Seidel relaxation by optimising the map at multiple levels of resolution. In contrast to the previously published algorithms [7], [8], the new algorithm does not require any global orientation sensor, and is shown to converge to the *non-linear* maximum likelihood solution.

The rest of this paper is structured as follows. After a review of related work (Section II), we derive the basic algorithm for single level relaxation (Section III). This is followed by an overview of multigrid methods (Section IV) and the Multilevel Relaxation algorithm (Section V), then experimental results (Section VI) and conclusions.

II. Related Work

In the last five years interest in SLAM has increased significantly, resulting in a large number of algorithms that are more efficient than the traditionally used Extended Kalman Filter (EKF) [16]. We briefly review the more recent contributions, referring the reader to a general overview given by Thrun [17].

Least squares estimation and incremental least squares estimation in general lead to linear equation systems, which is an established and thoroughly studied area of numerical mathematics. So to improve upon these general methods, new approaches must exploit some special property of the SLAM problem in order to reduce computational costs.

Most approaches exploit the fact that the field of view of the robot's sensors is limited. Thus, at any point in the environments, only few landmarks in the vicinity of the robot are observable and can be involved in measurements. The number k of these landmarks influences the computation time of the algorithm. It depends on the sensor type and the density of landmarks, but does not grow when the map gets larger. So this number is small, e.g., $k \approx 10$ in practice, and is usually considered as constant $k = O(1)$ for theoretical analysis.

	(R1)			(R2)	(R3)		
	UDA	nonlinear	quality	memory	update	global update	loop
Maximum Likelihood		√	√	m	$(n+p)^3$		
EKF			√	n^2	n^2		
CEKF			√	$n^{\frac{3}{2}}$	k^2	$kn^{\frac{3}{2}}$	
Relaxation		√	√	kn	kn		kn^2
FastSLAM	√	√	?	Mn	$M \log n$		
SEIF			?	kn	k^2		
TJTF			√	k^2n	k^3	k^3n	
Treemap		√	√	kn	k^2	$k^3 \log n$ resp. kn	
Multilevel Relaxation		√	√	kn	kn		

Fig. 1. Performance of different SLAM algorithms with n landmarks or frames respectively, m measurements, p robot poses and k landmarks / frames local to the robot. FastSLAM is a particle filter approach (M particles). Compare the remarks in Section II concerning the quality of the estimates provided by fastSLAM and SEIF. UDA stands for 'Uncertain Data Association' meaning that the algorithm can handle landmarks with uncertain identity. The treemap algorithm needs only $O(k^3 \log n)$ when computing an estimate for a selected $O(k)$ number of features.

Guivant and Nebot [18], [19] developed a modification of the EKF called *Compressed EKF* (*CEKF*) that allows the accumulation of measurements in a local region with k landmarks at cost $O(k^2)$ independent from the overall map size n. When the robot leaves this region, the accumulated result must be propagated to the full EKF (*global update*) at cost $O(kn^2)$. An approximate global update can be performed more efficiently in $O(kn^{\frac{3}{2}})$ with $O(n^{\frac{3}{2}})$ storage space needed.

Duckett et al. [7], [8] employed an iterative equation solver called *relaxation* to the linear equation system appearing in maximum likelihood estimation. They apply one iteration after each measurement with computation time $O(kn)$ and $O(kn)$ storage space. After closing a loop, more iterations are necessary leading to $O(kn^2)$ computation time in the worst case. In addition, their algorithm was based on the assumption of a global orientation sensor to ensure linearity. Both of these problems are solved in this paper by the Multilevel Relaxation algorithm, so that computation time can be reduced to $O(kn)$ even when closing large loops.

Montemerlo et al. [20] derived an algorithm called *FastSLAM* from the observation that the landmark estimates are conditionally independent given the robot pose. Basically, the algorithm is a particle filter [21] in which every particle represents a sampled robot trajectory plus a set of n Kalman filters estimating the position for each landmark as a Gaussian distribution. These distributions are independent, so n small covariance matrices are needed instead of one large matrix. The computation time for integrating a measurement is $O(M \log n)$ for M particles with $O(Mn)$ storage space. This algorithm can cope with uncertain landmark identification, which is a unique advantage over the other algorithms discussed in this section. However, the efficiency of this algorithm depends crucially on M being not too large, and it is not clear how this number scales with the complexity of the environment. A very large number of particles would be needed to close a loop, because a particle filter integrates measurements by choosing a subset of particles that is compatible with the measurements from the set of already existing particles (resampling), neither modifying the robot trajectory represented by the particles chosen, nor back-propagating the error along the loop. So the particle set must be large enough to contain a particle sufficiently close to the true pose of the robot at all times, otherwise map quality will be degraded.

Thrun et al. [22] presented a "constant time" algorithm called the *Sparse Extended Information Filter (SEIF)*, which uses an information matrix instead of a covariance matrix to represent uncertainty, as proposed by Frese and Hirzinger [5]. The algorithm exploits the fact that the information matrix is approximately sparse requiring $O(kn)$ storage space. The information matrix representation allows integration of a new measurement in $O(k^2)$ computation time, but to produce a map estimate a system of n linear equations must be solved. The equation solving is performed iteratively by relaxation. As in the approach of Duckett et al., this could be done by applying one iteration of relaxation with $O(kn)$ computation time. However, Thrun et al. propose not to relax all n landmarks, but only $O(k)$ of them (using so-called amortization), thereby formally obtaining an $O(k^2)$ algorithm. In the numerical literature, relaxation is reputed to need $O(n^2)$ time for reducing the equation error by a constant factor [15], [4]. For example, after observing n landmarks each $O(1)$ times, the SEIF algorithm will have spent only $O(n)$ time on equation solving, so it is unclear whether this approach will suffice in general.

Bosse et al. [23] avoid the computational problem of updating an estimate for $O(n)$ landmarks in their *Atlas* framework by dividing the map into submaps. There is no global coordinate system, rather each submap performs estimation in its own local frame. If information about landmarks in different submaps is required, uncertain spatial relations along a sequence of overlapping submaps are combined. However, a global reference frame usually simplifies implementation of algorithms using the map, and there are other techniques realizing submaps on a linear algebra level [24], [25], so an overall least square estimate could still be computed efficiently.

Paskin [26] views the estimation problem as a Gaussian graphical model, which is closely related to the graph of relations used in this paper. He proposed the *Thin Junction Tree Filter (TJTF)*. It is based on the observation that if a set of node separates the graph into two parts, then these parts are conditionally independent given estimates for the separating nodes. The algorithm maintains a junction tree ($O(k^2n)$ space), where every edge corresponds to such a

separation. Estimation is performed in $O(k^3 n)$ time by passing marginalized distributions along the edges of the junction tree. To keep the dimension of these distributions and thus computation limited, the tree is simplified by performing an approximation (similar to sparsification) on the represented distributions. From the perspective of "linear equation solving" this approach is complementary to the approach proposed in this paper. In the Gaussian case the junction tree algorithm is a direct, i.e., exact equation solver based on Schur-complements (corresponding to the marginalized distributions) and approximation is performed on the equation level. In contrast Multilevel Relaxation is an iterative, i.e., approximate equation solver but does not approximate the equations themselves.

Frese [24] recently proposed the *treemap* algorithm that, although independently developed, is even more closely related to TJTF. It divides the environment into a parts-whole-hierarchy represented as a binary tree. In order to integrate a measurement, all nodes from a single leaf up to the root need to be updated by passing Schur-complements along the edges, in very similar fashion to a junction tree. In contrast to TJTF, the algorithm uses a balanced tree with a designated root, so this operation needs only $O(k^3 \log n)$ time. In order to compute an estimate, information is propagated from the root down to the leafs. This can be performed more efficiently per node leading to an overall computation time of $O(kn)$. If only estimates for $O(k)$ selected features are needed, as with CEKF, computation time is even $O(k^3 \log n)$. The algorithm further handles linearization problems, but is far more difficult to implement compared to Multilevel Relaxation.

Fig. 1 gives a summary of the performance of the SLAM algorithms discussed in this section, according to the criteria R1-R3 described in Section I-B.

III. Single Level Relaxation

This section derives the basic algorithm for relaxation on a map with a single level, which provides the basis for the multi-level algorithm described in Section V.

The input to the algorithm is a set $\mathcal{R}$ of $m = |\mathcal{R}|$ relations on n planar frames. Each relation $r \in \mathcal{R}$ describes the likelihood distribution of frame a^r relative to frame b^r. It is modelled as a Gaussian distribution with mean μ^r and covariance C^r. The output is the maximum likelihood (ML) estimation vector $\hat{x}$ for the poses of all the frames.

In the context of SLAM, each frame corresponds to the robot pose at a particular time. Note that in order to achieve linear storage space and computation time, the number of frames should be bounded to the total area covered, as in [8], i.e., the map should not grow if the robot moves repeatedly through the same area. This issue is discussed further in Section V-E.

Each relation corresponds to a measurement of the relative pose between two frames, either by odometry for consecutive frames or as the result of matching the laser scans (or other sensor readings) taken at the respective robot poses. As usual, the mean μ^r of such a relation is the actual measurement and the covariance C^r is taken from a suitable model of the measurement uncertainty.

The algorithm proceeds in three steps [4, §15]:

1) Linearize the measurement functions.
2) Compute a quadratic error function $\chi^2(x)$ and represent it by a matrix A and a vector b as $\chi^2(x) = x^T A x - 2x^T b$.
3) Find the minimum $\hat{x}$ of $\chi^2(x)$ by solving $Ax = b$.

The first two steps are used in most least square non-linear model fitting algorithms. Specific to relaxation is the way of solving $Ax = b$. It is performed by going through all block rows A_i and solving $(Ax)_i = b_i$ for x_i. This process is repeated until convergence.

A. Derivation of the Linear Equation System

Maximizing likelihood is equivalent to minimizing negative log likelihood or χ^2 error energy:

$$\chi^2(x) = \sum_{r \in \mathcal{R}} z^{r\,T} (C^r)^{-1} z^r, \tag{1}$$

with

$$z^r = f(x_{a^r}, x_{b^r}) - \mu^r, \tag{2}$$

where x_{a^r} and x_{b^r} denote the 3 rows of vector x, corresponding to the x, y and ϕ coordinates of frame a and b, and where

$$f\left(\begin{pmatrix} a_x \\ a_y \\ a_\phi \end{pmatrix}, \begin{pmatrix} b_x \\ b_y \\ b_\phi \end{pmatrix}\right) = \begin{pmatrix} (a_x - b_x)\cos b_\phi + (a_y - b_y)\sin b_\phi \\ -(a_x - b_x)\sin b_\phi + (a_y - b_y)\cos b_\phi \\ a_\phi - b_\phi \end{pmatrix}.$$

The measurement function f maps the two poses of the two frames a^r and b^r to the relative pose of a^r with respect to b^r. Essentially f rotates the difference vector $a - b$ into b's coordinate frame. As usual, it is linearized at some linearization point $\breve{a}^r, \breve{b}^r$ corresponding to some estimate for the two frames (explained in Section V-D). The linearized measurement function is derived by evaluating the Jacobians J_a and J_b of $f(a, b)$ with respect to a and b:

$$f\left(\begin{pmatrix} a_x \\ a_y \\ a_\phi \end{pmatrix}, \begin{pmatrix} b_x \\ b_y \\ b_\phi \end{pmatrix}\right) \approx f^r_{\text{lin}}\left(\begin{pmatrix} a_x \\ a_y \\ a_\phi \end{pmatrix}, \begin{pmatrix} b_x \\ b_y \\ b_\phi \end{pmatrix}\right) \tag{3}$$

$$= f(\breve{a}^r, \breve{b}^r) + \underbrace{\begin{pmatrix} \cos\breve{b}^r_\phi & \sin\breve{b}^r_\phi & 0 \\ -\sin\breve{b}^r_\phi & \cos\breve{b}^r_\phi & 0 \\ 0 & 0 & 1 \end{pmatrix}}_{J_a}\left(a - \breve{a}^r\right) + \tag{4}$$

$$\underbrace{\begin{pmatrix} -\cos\breve{b}^r_\phi & -\sin\breve{b}^r_\phi & -(a_x - b_x)\sin\breve{b}^r_\phi + (a_y - b_y)\cos\breve{b}^r_\phi \\ \sin\breve{b}^r_\phi & -\cos\breve{b}^r_\phi & -(a_x - b_x)\cos\breve{b}^r_\phi - (a_y - b_y)\sin\breve{b}^r_\phi \\ 0 & 0 & -1 \end{pmatrix}}_{J_b}\left(b - \breve{b}^r\right).$$

Setting $a = x_{a^r}$ and $b = x_{b^r}$ and substituting into (2) yields

$$z^r \approx f(\breve{a}^r, \breve{b}^r) + J^r_a\left(x_{a^r} - \breve{a}^r\right) + J^r_b\left(x_{b^r} - \breve{b}^r\right) - \mu^r \tag{5}$$

$$= J^r_a x^r_a + J^r_b x^r_b - \underbrace{\left(J^r_a \breve{a}^r + J^r_b \breve{b}^r - f(\breve{a}^r, \breve{b}^r) + \mu^r\right)}_{y^r}, \tag{6}$$

where y^r contains all terms independent of x. It follows that

$$\chi^2(x) \approx \sum_{r \in \mathcal{R}} \begin{pmatrix} x_{a^r}{}^T J^{r\,T}_a (C^r)^{-1} J^r_a x_{a^r} \\ + \; x_{a^r}{}^T J^{r\,T}_a (C^r)^{-1} J^r_b x_{b^r} \\ + \; x_{b^r}{}^T J^{r\,T}_b (C^r)^{-1} J^r_a x_{a^r} \\ + \; x_{b^r}{}^T J^{r\,T}_b (C^r)^{-1} J^r_b x_{b^r} \\ -2\, x_{a^r}{}^T J^{r\,T}_a (C^r)^{-1} y^r \\ -2\, x_{b^r}{}^T J^{r\,T}_b (C^r)^{-1} y^r \\ + \; y^{r\,T} (C^r)^{-1} y^r \end{pmatrix}. \tag{7}$$

The terms involve x either quadratically (lines 1–4) or linearly (lines 5–6). They can be sorted by rows of x (either a^r or b^r) and grouped into matrix A and vector b as

$$\chi^2(x) \approx x^T \underbrace{\sum_{r\in\mathcal{R}} \begin{pmatrix} \cdots & \vdots & \cdots & \vdots & \cdots \\ \cdots & J_a^{r\,T}(C^r)^{-1}J_a^r & \cdots & J_a^{r\,T}(C^r)^{-1}J_b^r & \cdots \\ \cdots & \vdots & \cdots & \vdots & \cdots \\ \cdots & J_b^{r\,T}(C^r)^{-1}J_a^r & \cdots & J_b^{r\,T}(C^r)^{-1}J_b^r & \cdots \\ \cdots & \vdots & \cdots & \vdots & \cdots \end{pmatrix}}_{A} x \\ -2x^T \underbrace{\sum_{r\in\mathcal{R}} \begin{pmatrix} \vdots \\ J_a^{r\,T}(C^r)^{-1}y^r \\ \vdots \\ J_b^{r\,T}(C^r)^{-1}y^r \\ \vdots \end{pmatrix}}_{b} + \sum_{r\in\mathcal{R}} y^{r\,T}(C^r)^{-1}y^r. \tag{8}$$

Each relation r contributes to block-rows a^r and b^r of b and the intersection of these rows and columns in A. Since χ^2 is invariant under movement of the whole map, A is singular. To make it positive definite, a relation between frame 0 and a global frame is added ($J_b = 0$).

The matrix A is called the information matrix, and is the inverse of the estimation covariance matrix. A block $A_{ab} \neq 0$ appears only between frames a, b with a common relation. Due to limited sensor range there are only few frames with a relation to a given frame. The number of these frames k has a similar influence to the number of local landmarks in the CEKF [19] and SEIF [22] algorithms, since it depends on the sensor and environment but does not grow with map size, i.e., $k = O(1)$. This sparsity is essential for the efficiency of relaxation. The ML estimate $\hat{x}$ minimizes $\chi^2(x)$ or equivalently makes the gradient equal to 0:

$$0 = \frac{\partial\left(\chi^2(x)\right)}{\partial x} = \frac{\partial\left(x^TAx - 2x^Tb\right)}{\partial x} = 2(Ax - b). \tag{9}$$

So with the definitions made above, the equation to be solved is $Ax = b$ for a sparse matrix A.

B. Iterative Solution by Relaxation

The basic idea of relaxation is to solve the equation system $Ax = b$ one (block-) row at a time. Relaxation of (block-) variable x_i consists of solving (block-) row i of the equation for x_i considering all other x_j as fixed:

$$x_i' = x_i + A_{ii}^{-1}(b_i - A_{i\bullet}x), \tag{10}$$

where $A_{i\bullet}$ denotes (block-) row i and $A_{\bullet i}$ denotes (block-) column i of A. From the perspective of minimizing $x^TAx - 2x^Tb$, this means finding the minimum x_i if all other x_j remain unchanged. Intuitively this corresponds to the basic principle of relaxation: "pick a node and move it to where its neighbours think it should be". In a single iteration, (10) is used to update all x_i. After x_i is updated, the new value is used in the update of all following x_j, $j > i$ (Gauss-Seidel relaxation).

Every iteration reduces $x^TAx - 2x^Tb$, so x will converge to the unique minimum $A^{-1}b$, thereby solving the equation. Since A is sparse, evaluating (10) takes $O(k)$ and a single iteration $O(kn)$ time. For typical A, $O(n)$ iterations are needed to reduce the error by a constant factor [15], [4, §19.5]. However, local or oscillating parts of the error are reduced much more effectively than smooth or global parts, so in practice, few (1 to 3) iterations suffice, except when closing a large loop [8].

IV. Multigrid Linear Equation Solvers

This section gives an overview of the background theory on multigrid methods used for solving linear equation systems.

Historically, relaxation has been widely used for the numerical solution of partial differential equations (PDE). These continuous equations appear, for instance, in the simulation of heat flow, fluid dynamics or structural mechanics. As an example, the solution to a heat flow problem is a function $\mathbb{R}^3 \to \mathbb{R}$ assigning a temperature to each point in 3-D space. Numerically they are solved by discretizing the function onto a grid of sampling points. Thereby the PDE is converted into an ordinary sparse linear equation system. It is often solved using relaxation. The problem with this approach is that *oscillating* parts of the error are reduced efficiently, but it takes much longer to reduce the remaining *smooth* error.

A breakthrough was the development of multigrid methods in the 1970's [27], [15], more or less replacing successive overrelaxation and conjugate gradients as prevalent methods for solving sparse linear equations.

The idea is to discretize the PDE at different levels of resolution. Relaxation on a fine level (high resolution) effectively smooths the error. Then relaxation on a coarser level is used to reduce that error, which on the lower resolution is again more oscillatory.

A. Geometric Multigrid

In the geometrical context underlying most PDEs, a hierarchy of coarser levels is easily constructed by discretizing the PDE onto grids with increasing grid spacing, i.e., onto fewer sampling points. We follow the literature on multigrid methods in distinguishing different levels by superscript ^{h}. For the transition between two levels ^{h} denotes the finer and ^{H} the coarser level.

A single iteration of relaxation is first performed at the finest level. The remaining residual $b^h - A^hx^h$ is then restricted to the next coarser level by a *restriction* operator I_h^H, also known as the fine-to-coarse operator. This process is repeated until the coarsest level is reached. At the coarsest level, the residual equation is solved directly (e.g., by Cholesky decomposition [4, §2.9]). Then the solution x^H is interpolated to the next finer level by an *interpolation* operator I_H^h, also known as the coarse-to-fine operator, and used to update the solution x^h there. This process is repeated until the finest level is reached.

The propagation of the residual from fine to coarse and then of the solution back from coarse to fine is called a V-cycle (Fig. 2). Since the size of the levels decreases exponentially, the computation time needed is $O(kn)$, being asymptotically the same as the time needed for the finest level (see section V-F). For a suitable choice of I_H^h, I_h^H and A^H, one V-cycle reduces the error by a constant factor [15].

B. Galerkin Multigrid

For PDEs, A^H can be naturally derived as the discretization onto a smaller set of sampling points. I_H^h and I_h^H are usually chosen as linear interpolation and weighted averaging respectively. If no "natural" choice for A^H and I_h^H is available, the

FOR Level h from fine to coarse
Relax equation on level h: $A^h x^h = b^h$
Restrict residual to next level H: $b^H = I_h^H (b^h - A^h x^h)$
Solve equation on coarsest level $A^H x^H = b^H$
FOR Level H from coarse to fine
Interpolate solution to next level h: $x^h = x^h + I_H^h x^H$
Relax equation on level h: $A^h x^h = b^h$

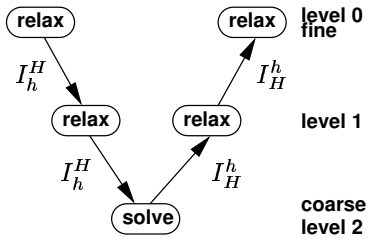


Fig. 2. General multigrid algorithm (V-cycle), and example with 3 levels.

Galerkin operator defines them purely algebraically for a given interpolator I_H^h. It is derived from the equivalent minimization problem (which on the finest level is just the original problem of minimizing $\chi^2(x)$):

$$g(x) = {x^h}^T A^h x^h - 2{x^h}^T b. \tag{11}$$

Since the coarse x^H corresponds to the fine $I_H^h x^H$, the coarse equation must minimize $g(I_H^h x^H)$:

$$0 = \frac{\partial \left(g(I_H^h x^H) \right)}{\partial x^H} \tag{12}$$

$$= \frac{\partial \left({x^H}^T {I_H^h}^T A^h I_H^h x^H - 2{x^H}^T {I_H^h}^T b \right)}{\partial x^H} \tag{13}$$

$$= 2 \big(\overbrace{{I_H^h}^T A^h I_H^h}^{A^H} x^H - \overbrace{{I_H^h}^T}^{I_h^H} b \big) \tag{14}$$

So by using the Galerkin operator $I_h^H = {I_H^h}^T$, $A^H = {I_H^h}^T A^h I_H^h$, $b^H = {I_H^h}^T b$ the coarse equation $A^H x^H = b^H$ minimizes $g(x)$ over the range of the interpolator. Relaxation on any level thereby reduces $g(x)$, ensuring convergence to the unique solution for any I_H^h. For fast convergence, however, the choice of I_H^h is still crucial.

Another point to consider is that I_H^h has to be local in some sense, otherwise coarser matrices will become increasingly dense, taking more than $O(kn)$ time per iteration.

C. Algebraic Multigrid

There exist so-called algebraic multigrid approaches that define the interpolator in a purely algebraic form without any reference to an underlying geometry or PDE [28]. In principle these approaches appear advantageous for a problem with an irregular geometry such as SLAM. We implemented a variant of "direct" interpolation [28, §4.2]. It interpolates a frame i so that the result satisfies $(Ax)_i = b_i$ given all coarse frames x_j and given linear interpolates for the fine frames $x_j, j \neq i$. However, for our data, this approach led to unacceptably dense matrices (see Section VI), so we did not use it. Instead, we applied a problem specific interpolator that is described in Section V-C.

V. Multilevel Relaxation

This section describes the Multilevel Relaxation algorithm proposed in this paper.

Unlike many PDEs, in SLAM the problem is not discretized onto a regular grid, so the question is how to define the hierarchy of coarser levels. The algorithm exploits the fact that the frames form a sequence, namely the robot's trajectory, so selecting every second frame is a suitable way of generating a coarser level (Fig. 3). It uses a multilevel representation for equation (9) with a sparse matrix A. On this hierarchy it implements a Galerkin based V-cycle. The algorithm is incremental, updating $\hat{x}$ for each new frame. Such an update involves three steps: $(i.)$ Extend A^h, b^h on all levels necessary to represent the new frame. $(ii.)$ Update A^h, b^h and I_H^h based on the new relations. $(iii.)$ Apply c V-cycles to update the ML estimate $\hat{x}$. The first two steps involve only few entries of A^h and b^h and take $O(k^2 \log n)$ time, the third step takes $O(kcn)$ time.

A. Data Structure

The algorithm maintains the graph of relations $\mathcal{R}$, with linked lists that allows efficient traversal of the set of edges incident upon a given node. Each relation r stores the corresponding Gaussian distribution μ^r, C^r and linearization point $\breve{a}^r$, $\breve{b}^r$ (Section III-A).

For the multilevel hierarchy, each level h contains the sparse equation matrix A^h, vector b^h, overall solution $\hat{x}^h$, residual solution x^h and the sparse interpolation matrix I_H^h. A^h is stored as a set of 3×3 blocks $\{(i, j, A_{ij}) | A_{ij} \neq 0\}$. Blocks of a given row are linked for efficient traversal. This allows computation of $(Ax)_i = A_{i\bullet} x$ and relaxation by equation (10) in $O(k)$. The interpolator I_H^h is stored in an array, since each row contains at most two blocks. We define $C(f)$, the set of coarse frames from which f is interpolated, and $F(f)$, the fine frame corresponding to f, as

$$C(f) = \begin{cases} \{\frac{f}{2}\} & f \text{ even} \\ \{\frac{f+1}{2}\} & f \text{ odd} \wedge \text{last} \\ \{\frac{f-1}{2}, \frac{f+1}{2}\} & \text{otherwise} \end{cases}, \tag{15}$$

$$C(\mathcal{F}) = \bigcup_{f \in \mathcal{F}} C(f), \quad F(f) = \begin{cases} 2f - 1 & \text{last} \\ 2f & \text{else} \end{cases} \tag{16}$$

B. Update

When new measurements arrive, a new frame is introduced into A^h, b^h and I_H^h, new relations are added and the equation is updated on each level. Only $O(k)$ frames are involved, so the update is performed in $O(k^2)$ per level and $O(k^2 \log n)$ total.

Our approach is to always recompute a complete row of A^h, b^h, and I_H^h, keeping track of the changed rows from fine to coarse level (Fig. 4). Some time could be saved if only the part of a row that actually changed were recomputed. In each row there are only $O(k)$ entries anyway, so this approach would

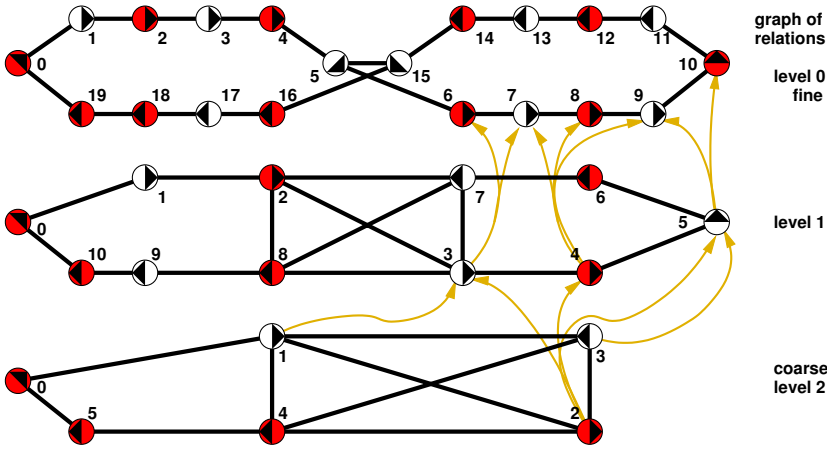


Fig. 3. Example for a three level hierarchy: For each coarser level every even numbered frame is selected and the odd ones are interpolated from their even predecessor and successor. Coarse frames (black) are represented on the next coarser level, fine frames (white) are interpolated. Lines show the sparsity pattern of A^0, A^1 and A^2. The first level is identical to the graph of relations $\mathcal{R}$. Arrows show the interpolator I_1^0 and I_2^1 (only for frames $6 \dots 10$)

Let $\mathcal{F}^h$ be the set of all frames involved in a new relation $r \in \mathcal{R}'$
Extend $\hat{x}^h, x^h$ with initial estimates for new frames
Define linearization points $\breve{a}^r$, $\breve{b}^r$ for new relations $r \in \mathcal{R}'$ (Section V-D)
Recompute rows $\mathcal{F}^h$ of A^h, b^h by (8) for finest level h
FOR Level h from fine to coarse
Recompute rows $\mathcal{F}^h$ of I_H^h by (22)
Let $\mathcal{F}^H = C(\mathcal{F}^h)$ be the set of affected coarse frames
Copy $\hat{x}_f^H$ from corresponding fine frame $\hat{x}_{F(f)}^h$ for all $f \in \mathcal{F}^H$
Recompute rows $\mathcal{F}^H$ of $A^H = {I_H^h}^T A^h I_H^h$
Compute Cholesky decomposition $A^H = U^T U$ for coarsest level H
Perform c V-cycles (Fig. 2).

Fig. 4. Update of multigrid hierarchy after adding new relations $\mathcal{R}'$.

be more complicated and only slightly faster. For a new frame A^h, b^h and I_h^H are extended as necessary.

Let $\mathcal{F}^h$ be the set of frames adjacent to a new relation. From the sparsity pattern in (8) it can be seen that only A_{ij}^h, b_i^h for $i, j \in \mathcal{F}^h$ changes, so we recompute rows $\mathcal{F}^h$. For each $f \in \mathcal{F}^h$ the blocks generated by all relations incident to f are added to $A_{f\bullet}^h$ and b_f^h.

Next the interpolator I_H^h is updated. From the Galerkin principle we are free to choose the interpolator, so we update rows $\mathcal{F}^h$, to limit the resulting change in A^H. Equation (17) shows the structure of I_H^h:

$$I_H^h = \begin{pmatrix} I & & & \\ E_1^- & E_1^+ & & \\ & I & & \\ & E_3^- & E_3^+ & \\ & & I & \\ & & & \ddots \end{pmatrix}, \qquad \begin{matrix} (I_H^h)_{ij} \neq 0 \\ \Leftrightarrow j \in C(i) \end{matrix} \tag{17}$$

Row $(I_H^h)_{f\bullet}$ has blocks at columns $C(f)$, so updating rows $\mathcal{F}$ changes columns $C(\mathcal{F})$. For a coarse f, row $(I_H^h)_{f\bullet}$ is a single identity block I. For a fine f, it is two blocks E_f^+ and E_f^- defined by equation (22). The last step is to update $A^H = {I_H^h}^T A^h I_H^h$, resulting in a change to columns $C(\mathcal{F})$ of I_H^h and $\mathcal{F}$ of A^h. A change in A_{ij}^h changes rows $C(i)$ of A^H, so that rows $C(\mathcal{F})$ are recomputed. When a level H has less than $n_{\min}$ frames (32 in our experiments), the equation on that level is solved directly using a Cholesky decomposition $A^H = U^T U$.

C. Interpolation

There are several difficulties in devising a good interpolator in the context of SLAM: $(i.)$ The interpolator must be based on the matrix A^h or estimate $\hat{x}^h$, not on the set of relations, since the latter is only available at the finest level. $(ii.)$ It must be rotation invariant, since otherwise it creates apparent orientation information in the coarse equations, since for some orientations the interpolation fits better than for others. Since orientation is usually very uncertain [5], this effect distorts the coarse solution. $(iii.)$ This may even happen for rotation invariant interpolators due to linearization. To see this, we consider the following relation and substitute into (5):

$$\mu = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, C = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \breve{a} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \breve{b} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \tag{18}$$

$$\chi^2 \left(\begin{smallmatrix} a \\ b \end{smallmatrix} \right) = \begin{matrix} (a_x - b_x - 1)^2 + (a_y - b_y - b_\phi)^2 \\ +(a_\phi - b_\phi)^2. \end{matrix} \tag{19}$$

If, for instance, the rotation invariant interpolator $b = a$ is chosen, χ^2 simplifies to $1 + b_\phi^2$, representing apparent absolute orientation information. To reduce this problem, we use the following geometric formula for interpolating a fine frame b

from coarse frames a and c:

$$b = a + \alpha(c-a) + \beta(c-a)^{\perp}, \tag{20}$$

$$\alpha \in [0 \dots 1],\ \beta \in [-1 \dots +1], \tag{21}$$

$$\begin{pmatrix} b_x \\ b_y \\ b_\phi \end{pmatrix} = E_b^- a + E_b^+ c, \tag{22}$$

$$E_b^- = \begin{pmatrix} 1-\alpha & \beta & 0 \\ -\beta & 1-\alpha & 0 \\ 0 & 0 & \frac{1}{2} \end{pmatrix}, \quad E_b^+ = \begin{pmatrix} \alpha & -\beta & 0 \\ \beta & \alpha & 0 \\ 0 & 0 & \frac{1}{2} \end{pmatrix}. \tag{23}$$

It defines the vector $b-a$ as a linear combination of $c-a$ and the orthogonal vector $(c-a)^{\perp}$. Therefore it is rotation invariant. The constants α and β are chosen so that $E_b^- \hat{a} + E_b^+ \hat{c} = \hat{b}$, but clipped to avoid extreme cases. Thereby the position of b relative to a and c closely matches the position used for linearization, and the above mentioned problems are reduced.

D. Non-linearity and Convergence

To obtain a consistent estimate incrementally, a single V-cycle for each new frame appears to suffice (see Section VI), even when closing a loop. We update the linearization point $\breve{a}^r$, $\breve{b}^r$ of a portion of the relations afterwards (5% in our experiments), so that the map can converge to the *non-linear* ML estimate while the robot continues moving. This is a great advantage over EKF-based implementations, which do not allow changing of the linearization point after integration and can thus be subject to severe linearization errors [5].

For new relations we do not linearize at the most recent estimate, but instead choose $\breve{b}^r$ to be consistent with the measurement, making $f(\breve{a}^r, \breve{b}^r) = \mu^r$. Usually the measurement will be slightly more erroneous than the prior estimate but this is no problem since the linearization point will be replaced by the most recent estimate at the next relinearization. On the contrary, when closing a loop the prior estimate is perturbed by accumulated error. So the measurement itself is far closer to the true value and thus a much better initial linearization point.

For an immediate ML estimate $\hat{x}_{\text{ML}} = \arg\min_x \chi^2(x)$, iteration with a termination criterion is performed (Fig. 5). The idea is to stop when the equation error $\hat{x} - \hat{x}_{\text{ML}}$ is much smaller than the expected estimation error $\hat{x}_{\text{ML}} - x_{\text{true}}$. We estimate $\hat{\chi}^2_{\min} \approx \min_x \chi^2(x)$, by assuming exponential convergence. The convergence factor α is computed from the initial and last χ^2 values. The factor γ is a heuristic for the case that χ^2_0 is already very close to the minimum.

It is well known that the expected minimum $E\big(\min_x \chi^2(x)\big)$ is $3(m-n+1)$ and the expected $E\big(\chi^2(x_{\text{true}})\big)$ value is $3m$ [4, §15.1]. So $\frac{n}{m-n+1}\hat{\chi}^2_{\min}$ is a rough estimate for $\chi^2(x_{\text{true}}) - \min_x \chi^2(x)$. When $\chi^2(\hat{x}) < (1+\gamma)\hat{\chi}^2_{\min}$ with $\gamma = 0.1\frac{n}{m-n+1}$ the linearization points $\breve{a}^r$, $\breve{b}^r$ are updated, usually leading to further reduction of $\chi^2(\hat{x})$. If this happens three times in a row, iteration is stopped.

E. Linear Storage Space

Similar to the approach of Lu and Milios [2], our current implementation uses one frame for each robot pose (sampled every 0.5m). This makes both representation size and computation time grow as $O(kp)$ rather than $O(kn)$. Thus requirements (R2) and (R3) are only met if mapping is terminated immediately after the robot has passed through the whole environment a single time ($p = O(n)$), as in the experiments presented in Section VI.

Requirements (R2) and (R3) could be fully met by storing one frame for each *place* instead of each robot pose. This could be achieved, for example, using the technique for incremental map building employed by Duckett et al. [8], where new frames were added to the map at 1 meter intervals. A better alternative would be to replace older frames with new ones, so that the map could be updated continually throughout the normal operation of the robot (so-called *lifelong learning* [29]). Note, however, that the Multilevel Relaxation algorithm itself does not specify whether frames correspond to places, landmarks or robot poses, so the algorithmic complexity would be the same regardless of the underlying representation.

F. Linear Computation Time

By selecting every second frame, the coarsification scheme guarantees that the overall number of frames in all levels is $O(n)$. For a V-cycle to be executed in $O(kn)$ time, the overall number of non-zero matrix blocks must be limited to $O(kn)$, i.e., the number of frames connected to a single frame must not grow too fast in coarser levels. In the following we will argue without a formal proof that this holds for typical buildings.

Due to sensor range the distance between two frames connected on the finest level is bounded. Thus the maximal distance d between frames connected on level h is proportional to 2^h. Now assume that the number of fine level frames within this range is proportional to $d^\nu = (2^h)^\nu$ for some ν. Then on level h the average number of frames connected to a single frame is $(2^h)^\nu \cdot 2^{-h}$, the overall number of connections is $kn \cdot 2^{-h} \cdot (2^h)^\nu \cdot 2^{-h}$ and the number of connections on all levels is

$$\sum_{h=0}^{\log n} kn \cdot 2^{-h} \cdot (2^h)^\nu \cdot 2^{-h} = kn \cdot \sum_{h=0}^{\log n} \left(2^{\nu-2}\right)^h. \tag{24}$$

The result is $O(kn)$ for any $\nu < 2$ and $O(kn \log n)$ for $\nu = 2$. (It should be noted that this effect is caused by the interpolator and is not inherent to the general multilevel approach.) This is the worst case possible, since the robot is moving in 2-dimensions, so the area within a certain distance can be at most proportional to the square of that distance. This happens, for instance, when the robot performs a snake-like movement on an open plane. The other extreme is a "1-dimensional" corridor, where $\nu = 1$. In the experiments discussed later (Fig. 6), the worst value encountered was $\nu \approx 1.3$ between level 0 and level 1 for the Freiburg data. From this evidence and the observation that typical buildings are something inbetween a corridor and an open plane, we argue that normally $\nu < 2$ and consequently a V-cycle needs $O(kn)$ computation time.

VI. Results

A. Off-line Experiments

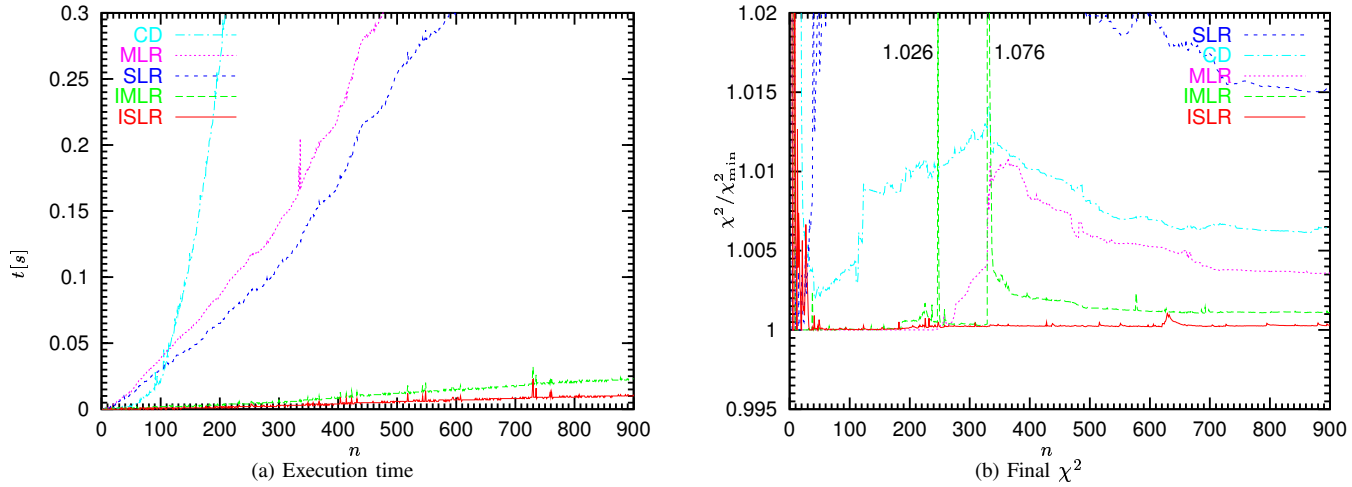
The performance of the proposed algorithm was first evaluated on two well known datasets, one from the University

$\chi_0^2 = \chi^2(\hat{x}); \quad ctr = 0; \quad i = 0$		
WHILE $ctr < 3$		
	Update $\hat{x}$ by 3 V-cycles, $\quad \chi_i^2 = \chi^2(\hat{x}); \quad i = i + 1$	
	$\gamma = 0.1\frac{n}{m-n+1} \quad \alpha = \sqrt[i]{\chi_i^2/(\gamma\chi_0^2)} \quad \hat{\chi}^2_{\min} = \chi^2_{i-1} - \frac{\chi^2_{i-1}-\chi^2_i}{1-\alpha}$	
	IF	$\chi_i^2 \geq \chi^2_{i-1} \vee \left(\chi_i^2 < (1+\gamma)\hat{\chi}^2_{\min}\right)$
	THEN	Update $\breve{a}^r$, $\breve{b}^r$ for all $r \in \mathcal{R}$; $\quad ctr = ctr + 1$
	ELSE	$ctr = 0$

Fig. 5. Computation of the approximate ML estimate $\hat{x}$.

	Freiburg			Wean Hall		
	iter.	time	χ^2	iter.	time	χ^2
Initial Estimate (IE)			16395061			1126227
Cholesky Decomposition (CD)		19.951 s	428397		1.268 s	6113
Single level relaxation (SLR)	12	0.437 s	431995	630	0.786 s	7122
Multilevel relaxation (MLR)	12	0.586 s	427178	12	0.059 s	5992
One MLR iteration (1-MLR)	1	0.023 s	501273	1	0.003 s	40375
Exact Minimum			425639			5986
Incremental MLR (iMLR)	1	avg. 14.4 ms	426104	1	avg. 1.6 ms	6178
Incremental SLR (iSLR)	1	avg. 8.6 ms	425759	1	avg. 0.7 ms	91772
n, m	906	8081		346	932	
Blocks $\neq 0$ in A^0, A^1, A^2	15770	9824	4154	2054	848	414

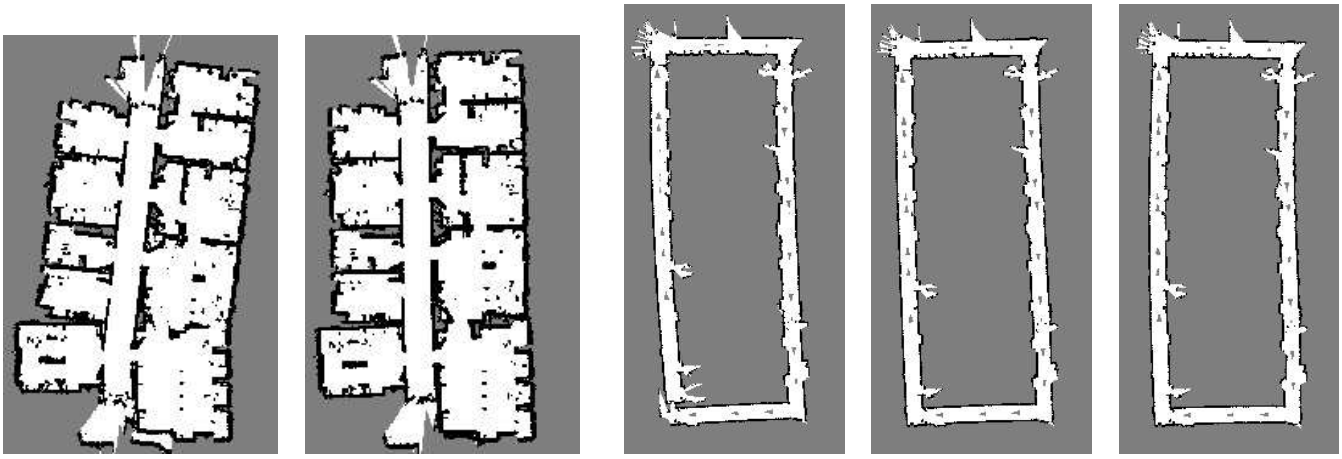
Fig. 6. Performance on Freiburg / Wean Hall data: CD, SLR, MLR, 1-MLR all compute a batch estimate for the whole data set. iMLR and iSLR incrementally process each new frame (the average time per frame is given). The exact minimum was computed by iterating MLR to numerical convergence of the equation $Ax = b$.



(a) Execution time (b) Final χ^2

Fig. 7. Performance on Freiburg data plotted over number of frames. Algorithm names are sorted from high to low values. (b) is scaled to show χ^2 values up to 2% above the minimum, which all correspond to excellent estimates.

of Freiburg [3] and a single loop taken from the Carnegie Mellon Wean Hall [30]. They were first processed by the software package *ScanStudio* [31], which performs the scan-matching. The resulting graph of relations was passed to our implementation, which computes the χ^2 function and uses either 'Cholesky decomposition' (CD), 'Single level relaxation' (SLR), 'Multilevel relaxation' (MLR) or 'One MLR iteration' (1-MLR) for minimization. Cholesky decomposition is a direct $O(p^3)$ equation solver [4, §2.9], which is included as a baseline for comparison of the relaxation algorithms. All four algorithms start with an initial estimate (IE) based on the first relation involving a frame. The last two methods 'Incremental Multilevel Relaxation' (iMLR) and 'Incremental Single Level Relaxation' (iSLR) apply a single MLR and SLR iteration for each new frame. Thereby they incrementally maintain a map estimate as our algorithm would actually be used on a mobile robot. All experiments were conducted on a Pentium IV, 1.7 GHz using LINUX/gcc 2.95.3 (Fig. 6, 7). For visualisation purposes, the results are presented as occupancy gridmaps [32], where white cells indicate areas that are believed to be 'empty', black cells indicate 'occupied' areas, and grey cells indicate 'unknown' areas (see Fig. 8).



(a) CD (b) iMLR (c) IE (d) MLR (e) iMLR

Fig. 8. Computed maps for Freiburg ($17m \times 26m$, a-b) and Wean Hall ($36m \times 74m$, c-e), see also Fig. 6.

For both datasets, MLR is much more efficient than CD and provides a better estimate. The latter point is true because CD solves the linearized problem, while all others perform non-linear minimization. It is worth noting that linearization effects can be seen in the χ^2 value despite the small orientation error. SLR needs the same number of iterations as MLR on the Freiburg data, thus being faster. But it is much slower on the Wean Hall data. The reason therefore lies in the difference between the two datasets (Fig. 8a, c). The Wean Hall data is a long loop with a large global error. MLR is more efficient in reducing this type of error than SLR, which needs many more iterations. The error in the Freiburg data is mainly local, so both MLR and SLR need the same number of iterations.

There is an inconsistency in the lower right two rooms of the Freiburg CD estimate (Fig. 8a), which is also visible for the MLR, SLR and 1-MLR estimates. The reason is that scans from the lower room and scans from the upper room overlap only slightly through the small doorway, so *ScanStudio* did not match any of them, and this inconsistency is not visible in the graph of relations.

iMLR and iSLR are much faster than CD, MLR and SLR if an incremental estimate is desired. For the Freiburg data, the estimates produced by both incremental algorithms are extremely good (Fig. 7, 8b) and better than CD and MLR most of the time, with the exception of two outliers occurring after integrating two inconsistent relations. iSLR performs extremely well here because the Freiburg data does not contain loops, which is the main situation where SLR requires more iterations.

The iSLR estimate is even better than the iMLR estimate. This is surprising, because iMLR performs the same amount of fine level relaxation as iSLR plus additional relaxation on coarse levels. We checked that the coarse level relaxation indeed reduced the linearized χ^2 error further. So it can be concluded that the phenomenom is related to linearization effects. On the other hand it should be kept in mind that both estimates are extremely good, with a χ^2 value less than 7% above the exact minimum.

For the Wean Hall data, the CD and MLR estimates are better than the iMLR estimate (Fig. 6, 8c-e), which is in turn much better than the iSLR estimate. This is because both perform only a single iteration after closing the loop. Here the advantage of iMLR can be seen, since it closes the loop consistently (Fig. 8e), which is not achieved by iSLR.

As claimed in Section V-F, the interpolator leads to sufficiently sparse matrices A^0, A^1, A^2, with each coarser level having 40% to 60% fewer non-zero blocks, equivalent to values of ν between 0.7 and 1.3. When using a variant of *direct interpolation* [28], A^1 has 29723 and A^2 has 25347 non-zero blocks, which is unacceptably dense.

The minimum $\chi^2_{\min}$ is much larger (Freiburg: $\times 20$, Wean Hall: $\times 3.4$) than the theoretically expected value $3(m-n+1)$. This shows that the scan matching covariance is overconfident [33] and stresses the importance of defining the termination criterion *relative* to $\hat{\chi}^2_{\min}$ in Section V-D.

B. Robotic Implementation

The Multilevel Relaxation algorithm has also been tested extensively as one component of a complete system for incremental mapping of unknown environments. The platform for these experiments was an Activmedia Peoplebot mobile robot equipped with a SICK LMS 200 laser scanner (Fig. 10). In this system, the robot is controlled remotely by a teleoperation interface via a client-server architecture. The server program running on the robot sends the sensor data and a visual image stream to the client PC (P4, 1200 Mhz), while the client sends the motor commands selected by the user back to the robot.

The largest map acquired by the robot is shown in Fig. 9, which comprises $n = 602$ frames (laser scans) and $m = 2108$ relations, in an area at the Technology building of Örebro University covering approximately 60×55 metres. The gridmap used for visualisation purposes was built off-line after mapping had been completed, but the Multilevel Relaxation algorithm was run online in real-time as the robot traversed the environment.

The Cox algorithm was used for real-time matching of laser scans [34] (using the standard implementation from

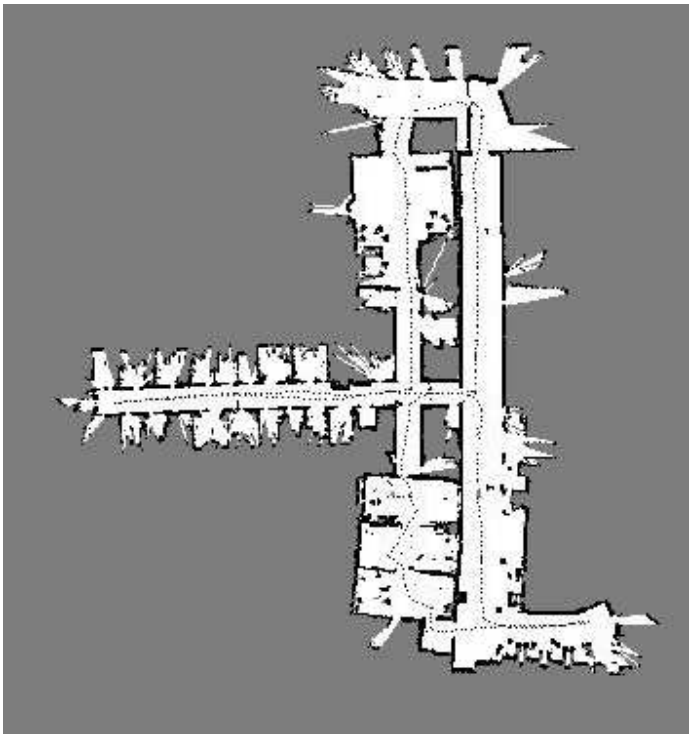


Fig. 9. Visualisation of the map acquired online at the Department of Technology, Örebro University, approximate size 60 × 55 metres.

Fig. 10. Activmedia Peoplebot mobile robot exploring part of the environment shown in Fig. 9

ScanStudio), due to its low computational cost. New laser scans were added to the map whenever the robot pose changed by a distance of at least 50 cm or an angle of at least $15°$. After adding a new scan, the client tried to match it with the previously recorded scan, in order to obtain an estimate of the displacement of the robot (odometry was only used here to provide an initial estimate of the relation for scan matching). If this match failed, then the client tried to match the current scan with the next previous scan. If both matches failed, then a new relation was taken directly from the recorded odometry data, corresponding to a so-called "weak link" in the approach of Lu and Milios [2]. For this map, only 3 such relations were obtained from odometry, while the rest were obtained by scan matching.

The radius used for checking of other possible scan matches was 1 m, i.e., whenever a new scan was added to the map, all other scans with a pose estimate less than 1 m from the current robot pose were matched to the new scan, and new relations (so-called "strong links" [2]) were added to the map for all successful matches.

Note that a more accurate map could possibly be achieved by using a more complex scan matching algorithm and a larger matching radius, but this could not be done in real-time with the currently used set-up.

VII. Conclusion

This paper introduced a new SLAM algorithm, Multilevel Relaxation, which is suitable for incremental, on-line use on a mobile robot in $O(n)$ time, including closing of large loops. This is possible because $(i.)$ the algorithm makes an iterative refinement to the existing solution at each step, rather than re-solving the equation system from scratch, and $(ii.)$ it exploits an important property of multigrid methods, namely that the residual error is geometrically smooth, i.e., it is distributed evenly over the whole map. In the case of closing a very large loop, as in the Wean Hall example presented, it can take several further iterations to converge to the maximum likelihood solution. However, the map is already geometrically *consistent* after a single iteration, that is, none of the measured relations are strongly violated in the estimated vector $\hat{x}$, and the map should be useful for navigation purposes. A further advantage of relaxation methods is that non-linearities can be handled by recomputing the linearization points as necessary. Remarkably, the result from a few iterations is already better than the exact solution of the linearized problem provided by Cholesky decomposition.

Relaxation is also potentially very useful for assisting in the task of data association (identification of observed environmental features), because relations between frames can be added and *removed* at any time (an advantage not yet exploited in this work). Furthermore, the minimum $\chi^2(\hat{x}) = \min_x \chi^2(x)$ provides a plausibility measure that allows assessment of different data association decisions, since the increase in $\chi^2(\hat{x})$ encountered after integrating a measurement equals the Mahalanobis distance of that measurement. These aspects make it possible to detect bad data association decisions and correct them retroactively (referred to as *lazy data association* by Hähnel et al. [35]). The approach is also useful for extracting covariance information for assessing map quality. To obtain column i of the covariance matrix A^{-1} equation $Ax = e_i$ must be solved, with e_i being the i-th unit vector. A single multigrid iteration should suffice, since the resulting covariance column provides a good initial guess for the columns corresponding to other landmarks nearby.

Future work will include embedding the new algorithm in a framework for handling both the continuous and discrete uncertainty in SLAM. This would be achieved by multi-hypothesis tracking in the space of possible maps, where one hypothesis corresponds to one possible topology. Search among alternative topologies could be performed by global

optimization methods such as genetic algorithms [36]. The ability to add and remove relations at any time could then be used to optimize $\min_x \chi^2(x)$ over different topologies. This approach appears to be very promising for lifelong learning, since it would combine adaptive global optimization capability with the efficiency of Multilevel Relaxation in closing loops, i.e., in finding the best metric estimate for a given topology. While loops occur rarely in most indoor environments, alternative topological interpretations of the same sensor data usually correspond to decisions on whether or not to close loops – this is why an efficient equation solver is highly desirable for solving the SLAM problem in its more general form.

ACKNOWLEDGMENTS

Udo Frese contributed to these studies during his visit to Örebro University while he was a Ph.D. student at the German Aerospace Center (DLR), with additional funding from the European Community Marie Curie programme. The authors would like to thank Steffen Gutmann for the *ScanStudio* software and the Freiburg data, and Sebastian Thrun for the Wean Hall data. We also thank Ola Bengtsson for sharing his scan matching code, the anonymous reviewers, and Stephen Marsland for proofreading the penultimate draft.

REFERENCES

[1] J. Leonard and H. Durrant-Whyte, "Dynamic map building for an autonomous mobile robot," *The International Journal on Robotics Research*, vol. 11, no. 4, pp. 286 – 298, Aug. 1992.

[2] F. Lu and E. Milios, "Globally consistent range scan alignment for environment mapping," *Autonomous Robots*, vol. 4, pp. 333 – 349, 1997.

[3] J. Gutmann and K. Konolige, "Incremental mapping of large cyclic environments," in *Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation, Monterey*, 1999, pp. 318–325.

[4] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery, *Numerical Recipes, Second Edition*. Cambridge University Press, Cambridge, 1992.

[5] U. Frese and G. Hirzinger, "Simultaneous localization and mapping - a discussion," in *Proceedings of the IJCAI Workshop on Reasoning with Uncertainty in Robotics, Seattle*, Aug. 2001, pp. 17 – 26.

[6] F. Reif, *Fundamentals of Statistical and Thermal Physics*. New York: McGraw-Hill, 1982.

[7] T. Duckett, S. Marsland, and J. Shapiro, "Learning globally consistent maps by relaxation," in *Proceedings of the IEEE International Conference on Robotics and Automation, San Francisco*, 2000, pp. 3841–3846.

[8] T. Duckett, S. Marsland, and J. Shapiro, "Fast, on-line learning of globally consistent maps," *Autonomous Robots*, vol. 12, no. 3, pp. 287 – 300, 2002.

[9] A. Howard, M. Mataric, and G. Sukhatme, "Relaxation on a mesh: a formalism for generalized localization," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'01), Hawaii, USA*, 2001.

[10] D. Filliat and J. Meyer, "Global localization and topological map learning for robot navigation," in *From Animals to Animats 7. Proceedings of the Seventh International Conference on Simulation of Adaptive Behavior (SAB'02)*, 2002.

[11] S. Thrun, D. Koller, Z. Ghahramani, H. Durrant-Whyte, and N. A.Y., "Simultaneous mapping and localization with sparse extended information filters," in *Proceedings of the Fifth International Workshop on Algorithmic Foundations of Robotics, Nice*, 2002.

[12] K. Konolige, J. Gutmann, and B. Limketkai, "Distributed map-making," in *Proc. IJCAI Workshop on Reasoning with Uncertainty in Robotics (RUR 2003)*, Acapulco, Mexico, 2003, pp. 15–22.

[13] P. Beeson, M. MacMahon, J. Modayil, J. Provost, F. Savelli, and B. Kuipers, "Exploiting local perceptual models for topological mapping," in *Proc. IJCAI Workshop on Reasoning with Uncertainty in Robotics (RUR 2003)*, Acapulco, Mexico, 2003, pp. 15–22.

[14] P. Rybski, F. Zacharias, J. Lett, O. Masoud, M. Gini, and N. Papanikolopoulos, "Using visual features to build topological maps of indoor environments," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'2003)*, Taipei, Taiwan, 2003, pp. 850–855.

[15] W. Briggs, "A multigrid tutorial," Ninth Copper Mountain Conference On Multigrid Methods, Apr. 1999, (http://www.llnl.gov/CASC/people/henson/mgtut/ps/mgtut.pdf).

[16] R. Smith, M. Self, and P. Cheeseman, "Estimating uncertain spatial relationships in robotics," in *Autonomous Robot Vehicles*, I. Cox and G. Wilfong, Eds. Springer Verlag, New York, 1988, pp. 167 – 193.

[17] S. Thrun, "Robotics mapping: A survey," School of Computer Science, Carnegie Mellon University, Tech. Rep., Feb. 2002.

[18] J. Guivant and E. Nebot, "Optimization of the simultaneous localization and map-building algorithm for real-time implementation," *IEEE Transactions Robotics and Automation*, vol. 17, no. 3, pp. 242 – 257, 2001.

[19] J. Guivant and E. Nebot, "Solving computational and memory requirements of feature based simultaneous localization and map building algorithms," Australian Centre for Field Robotics, University of Sydney, Sydney, Tech. Rep., 2002.

[20] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "FastSLAM: A factored solution to the simultaneous localization and mapping problem," in *Proceedings of the AAAI National Conference on Artificial Intelligence, Edmonton*, 2002, pp. 593–598.

[21] A. Doucet, J. de Freitas, and N. Gordon, *Sequential Monte Carlo Methods in Practice*. Springer Verlag, New York, 2001.

[22] S. Thrun, D. Koller, Z. Ghahmarani, and H. Durrant-Whyte, "SLAM updates require constant time," School of Computer Science, Carnegie Mellon University, Pittsburgh, Tech. Rep., 2002.

[23] M. Bosse, P. Newman, J. Leonard, and S. Teller, "SLAM in large-scale cyclic environments using the Atlas framework," *International Journal on Robotics Research*, 2003, (to appear).

[24] U. Frese, "An $O(\log n)$ algorithm for simultaneous localization and mapping of mobile robots in indoor environments," Ph.D. dissertation, University of Erlangen-Nürnberg, 2004.

[25] M. Paskin, "Thin junction tree filters for simultaneous localization and mapping," University of California, Berkeley, Computer Science Division Technical Report CSD-02-1198, September 2002.

[26] ——, "Thin junction tree filters for simultaneous localization and mapping," in *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI-03)*, G. Gottlob and T. Walsh, Eds., San Francisco, CA, 2003, pp. 1157–1164.

[27] A. Brandt, "Multi-level adaptive solutions to boundary-value problems," *Math. Comp*, vol. 31, pp. 333 – 390, 1977.

[28] K. Stüben, "Algebraic multigrid (AMG): An introduction with applications," GMD - Forschungszentrum Informationstechnik, Tech. Rep. 70, Nov. 1999.

[29] T. Duckett, "Concurrent map building and self-localisation for mobile robot navigation," Ph.D. dissertation, Department of Computer Science, University of Manchester, 2001.

[30] S. Thrun, W. Burgard, and D. Fox, "A probabilistic approach to concurrent mapping and localization for mobile robot," *Machine Learning*, vol. 31, no. 5, pp. 1 – 25, 1998.

[31] J.-S. Gutmann, "Robuste navigation autonomer mobiler systeme," Ph.D. dissertation, University of Freiburg, 2000.

[32] H. Moravec and A. Elfes, "High resolution maps from wide angle sonar," in *Proceedings of the IEEE International Conference Robotics and Automation, St. Louis*, 1985, pp. 116 – 121.

[33] O. Bengtsson and A. Baerveldt, "Localization in changing environments - estimation of covariance matrix for the IDC algorithm," in *Proceedings of the International Conference on Intelligent Robots and Systems, Hawaii*, 2001, pp. 1931 – 1937.

[34] I. Cox, "Blanche - an experiment in guidance and navigation of an autonomous robot vehicle," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 2, pp. 193–204, 1991.

[35] D. Hähnel, W. Burgard, B. Wegbreit, and S. Thrun, "Towards lazy data association in SLAM," in *In Proceedings of the 10th International Symposium of Robotics Research (ISRR'03)*, 2003.

[36] T. Duckett, "A genetic algorithm for simultaneous localization and mapping," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'2003)*, Taipei, Taiwan, 2003, pp. 434–439.