A Multigrid Approach for Accelerating Relaxation-based SLAM

Udo Frese[1] and Tom Duckett[2]

[1] DLR Oberpfaffenhofen
German Aerospace Center
Institute of Robotics und Mechatronics
D-82230 Wessling, Germany
Udo.Frese@dlr.de, www.robotic.dlr.de
[2] AASS Learning Systems Lab
Department of Technology
Örebro University
S-70182 Örebro, Sweden
tdt@tech.oru.se, www.aass.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 so-called multigrid methods used for solving partial differential equations. The approach overcomes the relatively slow convergence of previous relaxation methods because it optimizes the map at multiple levels of resolution. The resulting algorithm has an update time that is linear in the number of mapped features, even when closing very large loops, and offers advantages in handling non-linearities compared to previous approaches. Experimental comparisons with alternative algorithms using two well-known data sets are also presented.

1 Introduction

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 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 [8, 7]. 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.

A desirable property for any SLAM algorithm is that the computation time for updates should be linear in the number of features n stored in the map [5]. To achieve this objective, we have investigated so-called multigrid methods for solving partial differential equations [2], resulting in a new SLAM algorithm for solving the equation system of a single topological hypothesis called 'Multilevel Relaxation'. Relaxation is an iterative method for solving equations, which is equivalent to Gauss-Seidel iteration or Gibbs sampling at zero temperature. The new approach improves on the previously introduced algorithm [3], which was significantly slower when closing large loops, by carrying out the optimization process at multiple levels of resolution in the underlying map.

In the following sections, we derive the basic algorithm for single level relaxation (§2), followed by an overview of multigrid methods (§3) and the Multilevel Relaxation algorithm (§4). Due to lack of space the description is brief, a more extended discussion can be found in [4]. Results including experimental comparisons with alternative algorithms are presented in §5, and the conclusion in §6 discusses how to embed the new algorithm within a framework for tracking multiple topological hypotheses.

1.1 Related Work

Guivant and Nebot [6] introduced a so called Compressed Extende Kalman Filter (CEKF) for real-time mapping. By restricting the Kalman update to a subset of landmarks in a local area, updates can be performed at cost $O(1)$ and then transferred to the overall map in $O(n^2)$. With a further approximation, that can be reduced to $O(n)$, though the problem of closing large loops is not yet solved.

Montemerlo et al. [9] used a particle filter to track the pose of the robot, where each particle also includes a set of Kalman filters estimating the position for each landmark. This approach is able to represent and search between multiple hypotheses for the full map (i.e., robot pose plus all landmark positions), but the particle set must be large enough to contain a particle sufficiently close to the true pose of the robot at all times. The algorithm requires $O(M \log n)$ time for M particles, though it is not clear how the number of particles scales with the complexity of the environment.

Thrun et al. [12] applied extended information filters utilizing the sparsity of the information matrices in SLAM, as proposed by Frese and Hirzinger [5]. The equation solving is performed iteratively by relaxation. The authors propose to relax only $O(1)$ landmarks at each step, which would result in a constant time algorithm. However, in the numerical literature, relaxation is reputed to need $O(n^2)$ time for reducing the equation error by a constant factor [2, 10, §19.5]. For instance after observing n landmarks each $O(1)$ times, the algorithm will

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: $A^h x^h = b^h$

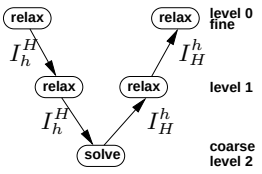


Fig. 1. General multigrid algorithm (V-cycle), and example with 3 levels.

have spent only $O(n)$ time on equation solving, so it is doubtful whether this approach will suffice in general.

2 Single Level Relaxation

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 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 certain time. 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 [10, §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 $\chi^2(x) = x^T Ax - 2x^T b$.
3. Find the minimum $\hat{x}$ of $\chi^2(x)$ by solving $Ax = b$.

The first two steps are the same used in most least square nonlinear 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.

2.1 Derivation of the Linear Equation System

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

$$\chi^2(x) = \sum_{r \in \mathcal{R}} z^{rT} (C^r)^{-1} z^r, \quad \text{with } z^r = f(x_{a^r}, x_{b^r}) - \mu^r, \tag{1}$$

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

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. As usual, it is linearized at some linearization point $\breve{a}^r, \breve{b}^r$ corresponding to some estimate for the two frames. We use the most recent estimate for $\breve{b}^r$ and choose $\breve{a}^r$ so that $f(\breve{a}^r, \breve{b}^r) = \mu^r$. This means that the linearization points chosen for a measurement are consistent with the measurement itself. Compared to using the most recent estimate for $\breve{a}^r$ this produces a much smaller error when closing a loop.

$$f\left(\begin{pmatrix} a_x \\ a_y \\ a_\phi \end{pmatrix}, \begin{pmatrix} b_x \\ b_y \\ b_\phi \end{pmatrix}\right) \approx \mu + \underbrace{\begin{pmatrix} \cos\breve{b}_\phi & \sin\breve{b}_\phi & 0 \\ -\sin\breve{b}_\phi & \cos\breve{b}_\phi & 0 \\ 0 & 0 & 1 \end{pmatrix}}_{J_a}\left(a - \breve{a}\right) + \underbrace{\begin{pmatrix} -\cos\breve{b}_\phi & -\sin\breve{b}_\phi & \mu_y^r \\ \sin\breve{b}_\phi & -\cos\breve{b}_\phi & -\mu_x^r \\ 0 & 0 & -1 \end{pmatrix}}_{J_b}\left(b - \breve{b}\right)$$

Substituting the linearized function into (1) yields a quadratic approximation:

$$z^r \approx J_a^r\left(x_{a^r} - \breve{a}^r\right) + J_b^r\left(x_{b^r} - \breve{b}^r\right), \quad \chi^2(x) = \tag{2}$$

$$x^T \underbrace{\sum_{r\in\mathcal{R}} \begin{pmatrix} \cdots & \vdots & \cdots & \vdots & \cdots \\ \cdots & J_a^{rT}(C^r)^{-1}J_a^r & \cdots & J_a^{rT}(C^r)^{-1}J_b^r & \cdots \\ \cdots & \vdots & \cdots & \vdots & \cdots \\ \cdots & J_b^{rT}(C^r)^{-1}J_a^r & \cdots & J_b^{rT}(C^r)^{-1}J_b^r & \cdots \\ \cdots & \vdots & \cdots & \vdots & \cdots \end{pmatrix}}_{A} x - 2x^T \underbrace{\sum_{r\in\mathcal{R}} \begin{pmatrix} \vdots \\ J_a^{rT}(C^r)^{-1}\left(J_a^r\breve{a}^r + J_b^r\breve{b}^r\right) \\ \vdots \\ J_b^{rT}(C^r)^{-1}\left(J_a^r\breve{a}^r + J_b^r\breve{b}^r\right) \\ \vdots \end{pmatrix}}_{b}$$

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, which are normally only $O(1)$ for a given a. 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:

$$0 = \frac{\partial\left(\chi^2(x)\right)}{2\partial x} = \frac{\partial\left(x^TAx - 2x^Tb\right)}{2\partial x} = Ax - b \tag{3}$$

So with the definitions made above, the equation to be solved is $Ax = b$ for a sparse matrix A. The full posterior distribution is in principle given by $\exp(-\frac{1}{2}\chi^2(x))$, but this is of little practical value, since usually the posterior for some selected frames is desired. This requires computation of the covariance matrix A^{-1}, which appears to be impossible in less than $O(n^2)$ time, so in this paper we concentrate on computing the ML estimate $\hat{x}$.

2.2 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[3]:

$$x_i' = x_i + A_{ii}^{-1}(b_i - A_{i\bullet}x) \tag{4}$$

[3] $A_{i\bullet}$ denotes row i and $A_{\bullet i}$ denotes column i of A.

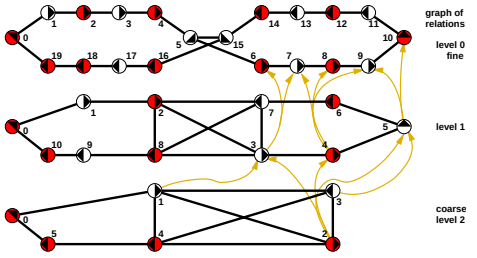


Fig. 2. Example for a three level hierarchy: For each coarser level every even numbered frame is selected and the odd ones are interpolated from their even predecessor and successor. Coarse frames (black) are represented on the next coarser level, fine frames (white) are interpolated. Lines show the 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 $6 \ldots 10$)

From the perspective of minimizing $x^T Ax - 2x^T b$, this means finding the minimum x_i if all other x_j remain unchanged. In a single iteration, (4) 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 $x^T Ax - 2x^T b$, so it will converge to the unique minimum $A^{-1}b$, thereby solving the equation. Since A is sparse, evaluating (4) takes $O(1)$ and a single iteration $O(n)$ time. For typical A, $O(n)$ iterations are needed to reduce the error by a constant factor [2, 10, §19.5]. However local or oscillating parts of the error are reduced much more effectively than smooth or global parts, so in practice, few $(1-3)$ iterations suffice, except when closing a large loop [3].

3 Multigrid Linear Equation Solvers

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 so-called multigrid methods in the 1970's [1, 2]. 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.

3.1 Geometric Multigrid

To realize this idea, a single iteration of relaxation is first performed at the finest level. The remaining residual $b^h - A^h x^h$ is then restricted to the next coarser level by a *restriction* operator I_h^H [4]. On the coarsest level, the residual equation is solved directly (e.g., by Cholesky decomposition [10, §2.9]). Then the solution

[4] We follow the literature on multigrid methods in distinguishing different levels by superscript ^{h}. For two levels ^{h} denotes the finer and ^{H} the coarser level.

x^H is interpolated to the next finer level by an *interpolation* operator I_H^h and used to update the solution x^h there. 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. 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. 1). It needs $O(n)$ time, since the size of the levels decreases exponentially. For suitable I_H^h, I_h^H and A^H it reduces the error by a constant factor [2].

3.2 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 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 $\chi^2(x)$):

$$g(x) = {x^h}^T A^h x^h - 2{x^h}^T b \tag{5}$$

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

$$0 = \frac{1}{2}\frac{\partial\left(g(I_H^h x^H)\right)}{\partial x^H} = \overbrace{{I_H^h}^T A^h I_H^h}^{A^H} x^H - \overbrace{{I_H^h}^T}^{I_h^H} b \tag{6}$$

So by using the Galerkin operator $I_h^H = {I_H^h}^T$, $A^H = {I_H^h}^T A^h I_H^h$, the coarse equation 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 $O(n)$ time per iteration.

4 Multilevel Relaxation

In this section we describe 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. 2). It uses a multilevel representation for equation (3) 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 tasks: (*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 $O(\log n)$, the third step takes $O(cn)$.

4.1 Data Structure

The algorithm maintains the graph of relations $\mathcal{R}$ in the usual way, with linked lists that allows efficient traversal of the set of edges incident upon a given node. Each relation r stores the corresponding Gaussian μ^r, C^r and linearization point $\breve{a}^r$, $\breve{b}^r$ (§2.1). 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 (4) in $O(1)$. 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.

4.2 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 $O(1)$ frames are involved, so the update is performed in $O(1)$ per level and $O(\log n)$ 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. Since each row contains only $O(1)$ nonzero blocks, updating only the affected blocks is little faster and more complicated. 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 (2) 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 (7) shows I_H^h:

$$I_H^h = \begin{pmatrix} I & & \\ E_1^- & E_1^+ & \\ & I & \\ & E_3^- & E_3^+ \\ & & I \\ & & & \ddots \end{pmatrix}, \qquad \begin{array}{l}(I_H^h)_{ij}\neq 0\\ \Leftrightarrow j\in C(i)\end{array} \tag{7}$$

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 (9). 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 $A^H = U^T U$.

4.3 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 of the rotation [4]. To reduce this problem, we use the following geometric formula for interpolating a fine frame b from coarse frames a and c:

$$b = a + \alpha(c-a) + \beta(c-a)^{\perp}, \quad \alpha \in [0 \dots 1],\ \beta \in [-1 \dots +1], \tag{8}$$

$$\begin{pmatrix} b_x \\ b_y \\ b_\phi \end{pmatrix} = E_b^- a + E_b^+ c, \quad E_b^- = \begin{pmatrix} 1-\alpha & \beta & 0 \\ -\beta & 1-\alpha & 0 \\ 0 & 0 & \frac{1}{2} \end{pmatrix}, \quad E_b^+ = \begin{pmatrix} \alpha & -\beta & 0 \\ \beta & \alpha & 0 \\ 0 & 0 & \frac{1}{2} \end{pmatrix} \tag{9}$$

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.

4.4 Nonlinearity and Convergence

To obtain a consistent estimate incrementally, a single V-cycle for each new frame appears to suffice (§5), 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 *nonlinear* 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 an immediate ML estimate $\hat{x}_{\mathrm{ML}} = \arg\min_x \chi^2(x)$, iteration with a termination criterion is performed. The idea is to stop when the equation error $\hat{x} - \hat{x}_{\mathrm{ML}}$ is much smaller than the estimation error $\hat{x}_{\mathrm{ML}} - x_{\mathrm{true}}$ (details in [4]).

5 Results

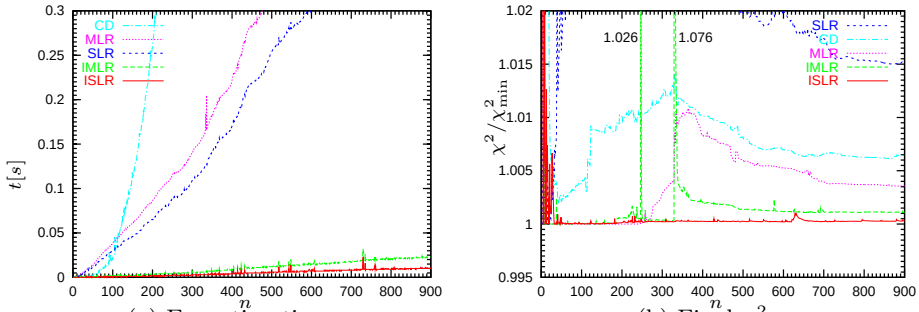
We have evaluated the performance of the proposed algorithm on two well known datasets, one from the University of Freiburg [7] and a single loop taken from the Carnegie Mellon Wean Hall [11]. They are processed by the software package *ScanStudio*[5] which performs the scan-matching. The resulting graph of relations is passed to our implementation, computes the χ^2 function and uses either 'Cholesky decomposition' (CD)[6], 'Single level relaxation' (SLR), 'Multilevel relaxation' (MLR) or 'One MLR iteration' (1MLR) for minimization. All four algorithms start with an initial estimate based on the first relation involving a frame. The last two methods 'Incremental Multilevel Relaxation' (IMLR) and

[5] We would like to thank Steffen Gutmann for the Freiburg data and the permission to utilize *ScanStudio*, and Sebastian Thrun for the Wean Hall data.

[6] a direct $O(n^3)$ equation solver [10, §2.9]

	Freiburg			Wean Hall		
	iter.	time	χ^2	iter.	time	χ^2
Initial Estimate			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 (1MLR)	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. 3. Performance on Freiburg / Wean Hall data: CD, SLR, MLR, 1MLR 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 $Ax = b$.

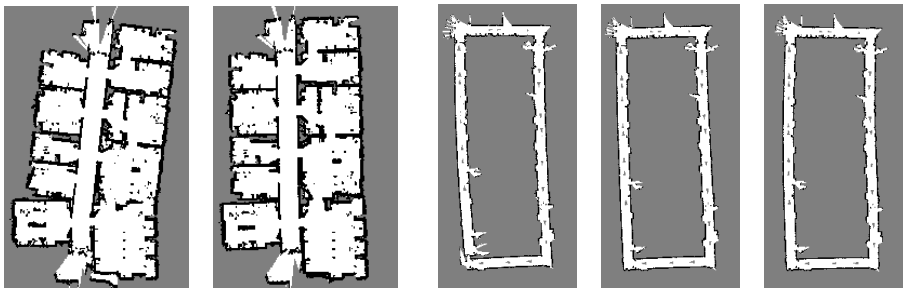


(a) Execution time (b) Final χ^2

Fig. 4. 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.

'Incremental Single Level Relaxation' (ISLR) apply a single MLR and SLR iteration for each new frame. Thereby they incrementally maintain an 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. 3, 4).

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 nonlinear minimization. It is worth noting that linearization effects can be seen in the χ^2 value despite the small orientation error. SLR is faster than MLR on the Freiburg data, but much slower on the Wean Hall data. The reason therefore lies in the difference between the two datasets (Fig. 5a, 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.



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

Fig. 5. Results for Freiburg ($17m \times 26m$, a-b) and Wean Hall ($36m \times 74m$, c-e).

There is an inconsistency in the lower right two rooms of the Freiburg CD estimate (Fig. 5a), which is also visible for the MLR, SLR and 1MLR 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, both estimates are extremely good (Fig. 4, 5b) and better than CD and MLR most of the time, with the exception of two outliers occurring after integrating two inconsistent relations. Surprisingly, the ISLR estimate is even better than the IMLR estimate, which has been found to be related to linearization effects. For the Wean Hall data, the CD and MLR estimates are initially better than the IMLR estimate (Fig. 4, 5d-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. 5e), which is not achieved by ISLR.

6 Conclusions and Future Work

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 if

necessary. Remarkably, the result from a few iterations is already better than the exact solution of the linearized problem provided by Cholesky decomposition.

Future work will include embedding the new algorithm in a framework for handling both the continuous and discrete uncertainty in the SLAM problem. This would be achieved by multi-hypothesis tracking in the space of possible maps, where one hypothesis corresponds to one possible topology. In this paper, we have only used the algorithm to solve the linear equation system for a single topological hypothesis (i.e., we assumed no data association errors), but it should also provide a good core engine for multi-hypothesis SLAM, e.g., by tracking the best M topological hypotheses, due to its ability to close loops efficiently. While loops occur relatively rarely in most indoor environments, alternative topological interpretations of the same sensor data within a multi-hypothesis framework will often correspond to decisions on whether or not to close a loop – this is why an efficient equation solver is highly desirable for solving the SLAM problem in its most general form.

References

1. A. Brandt. Multi-level adaptive solutions to boundary-value problems. *Math. Comp*, 31:333–390, 1977.
2. W. Briggs. A multigrid tutorial. Ninth Copper Mountain Conference On Multigrid Methods, 1999. (http://www.llnl.gov/CASC/people/henson/mgtut/ps/mgtut.pdf).
3. T. Duckett, S. Marsland, and J. Shapiro. Fast, on-line learning of globally consistent maps. *Autonomous Robots*, 12(3):287–300, 2002.
4. U. Frese and T. Duckett. A multigrid approach for accelerating relaxation-based SLAM. In *Proceedings of the IJCAI Workshop Reasoning with Uncertainty in Robotics, Acapulco*, pages 39–46, 2003.
5. U. Frese and G. Hirzinger. Simultaneous localization and mapping - a discussion. In *Proc. IJCAI Workshop on Reasoning with Uncertainty in Robotics, Seattle*, 2001.
6. J. Guivant and E. Nebot. Optimization of the simultaneous localization and map-building algorithm for real-time implementation. *IEEE Trans. Robotics and Automation*, 17(3):242–257, 2001.
7. J. Gutmann and K. Konolige. Incremental mapping of large cyclic environments. In *Proc. IEEE Int. Symposium on Computational Intelligence in Robotics and Automation - CIRA-99, Monterey, CA*, 1999.
8. F. Lu and E. Milios. Globally consistent range scan alignment for environment mapping. In *Autonomous Robots*, volume 4, pages 333 – 349, 1997.
9. M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM: A factored solution to the simultaneous localization and mapping problem. In *Proc. AAAI Nat. Conf. on Artificial Intelligence*, Edmonton, Canada, 2002.
10. W. Press, S. Teukolsky, W. Vetterling, and B. Flannery. *Numerical Recipes, Second Edition*. Cambridge University Press, 1992. ISBN 0-521-43108-5.
11. S. Thrun, W. Burgard, and D. Fox. A probabilistic approach to concurrent mapping and localization for mobile robot. *Machine Learning*, 31(5):1 – 25, 1998.
12. S. Thrun, D. Koller, Z. Ghahramani, H. Durrant-Whyte, and N. A.Y. Simultaneous mapping and localization with sparse extended information filters. In *Proc. of the Fifth Int. Workshop on Algorithmic Foundations of Robotics*, Nice, France, 2002.