Treemap: An $O(\log n)$ Algorithm for Simultaneous Localization and Mapping

Udo Frese

Bremen Institute of Safe Systems
ufrese@informatik.uni-bremen.de

Abstract. This paper presents a very efficient SLAM algorithm that works by hierarchically dividing the map into local regions and subregions. At each level of the hierarchy each region stores a matrix representing some of the landmarks contained in this region. For keeping the matrices small only those landmarks are represented being observable from outside the region.
A measurement is integrated into a local subregion using $O(k^2)$ computation time for k landmarks in a subregion. When the robot moves to a different subregion a global update is necessary requiring only $O(k^3 \log n)$ computation time for n overall landmarks.
The algorithm is evaluated for map quality, storage space and computation time using simulated and real experiments in an office environment.

1 Introduction

The problem of making a map from local observations is a very old one basically as old as maps themselves. While geodesy, the science of surveying in general dates back to 8000 B.C., it was the achievement of C.F. Gauss to first formalize the problem from the perspective of statistical estimation in his article *"Theoria combinationis observationum erroribus minimis obnoxiae"* [1](1821).

Simultaneous Localization and Mapping

In the much younger realm of robotics the corresponding problem is that of simultaneous localization and mapping (SLAM). It requires the robot to continuously build a map from sensor data while traveling through the environment. It has been under research since the mid 80ies gaining enormous popularity in recent years. A majority of approaches adhere to the Gaussian formalization. They estimate a vector of n features, e.g. landmarks or laserscan reference frames by minimizing a quadratic error function, i.e. by solving implicitly a linear equation system. With this well established methodology the main question is how to compute or approximate the estimate efficiently. To make this more explicit, there are three important requirements an ideal SLAM algorithm should fulfill that were first proposed by the author in [2] and further discussed in [3]:

(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 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 ($O(n)$).*

(R1) states that the map shall represent nearly all information contained in the measurements, thus binding the map to reality and limiting approximations. The other postulates (R2) and (R3) regard efficiency, requiring linear space and time consumption. The contribution[1] of this paper is a hierarchical SLAM algorithm that meets the above mentioned requirements. It works by dividing the map into regions and subregions. When integrating a measurement it needs $O(k^2)$ computation time for updating the estimate for a region with k landmarks, $O(k^3 \log n)$ when the robot moves to a different region and $O(kn)$ to compute an estimate for the whole map. There is an extension to the algorithm not covered in this paper that applies "nonlinear rotations" to individual regions to greatly reduce the linearization error caused by error in the robot orientation [4]. The algorithm is landmark (feature) based, requires known data association and assumes a "topologically suitable building" (§6).

Spatial Cognition

As formalized above the task can be described as computing global coordinates from local measurements. This corresponds to the distinction between egocentric and allocentric spatial memories reported in cognitive psychology [5]. It is a way of integrating spatial observations that is very suited for mobile robots, because the result is a single estimate that incorporates all information available, i.e. one concrete map that is (statistically) consistent with all observations. Together with the corresponding uncertainty information, generally provided as a covariance matrix, such an estimate is very useful. Any derived spatial quantity, like distances and angles can be directly computed from the estimate together with its uncertainty without any complex inference process. Especially most existing algorithms that use a map, like path planning, navigation and localization are designed with global coordinates. Even when the task involves human robot communication, for instance matching natural language descriptions ("after passing the entrance hall turn right") it appears to be promising to directly match qualitative predicates resulting from natural language processing ("right") with an estimated metrical map using empirical definitions of the predicates [6, 7].

[1] This article is based on research conducted during the authors Ph.D. studies at the German Aerospace Center (DLR) in Oberpfaffenhofen.

The paper is organized as follows. After a brief review of related work (§2) the algorithm is presented (§3...§8). It follows an investigation of map quality and computation time based on simulations (§9) and experiments on a real robot in an 60m × 45m office building (§10).

2 State of the Art

After the fundamental paper by Smith et al. [8] in 1988 most work on SLAM was based on the Extended Kalman Filter (EKF) that allows to treat SLAM theoretically thorough as an estimation problem. However, the problem of large computation time remained. The most time consuming part is to update the EKF's covariance matrix after each measurement, taking $O(n^2)$ time for n landmarks. This limited the use to small environments ($n \lessapprox 100$ landmarks).

Recently, interest in SLAM has increased drastically and several, more efficient algorithms have been developed. Many approaches exploit, that observations are *local* in the sense that from a single robot pose only few k landmarks are visible. In the following the more recent contributions will be briefly reviewed. A general overview is given by Thrun [9] and a discussion of the inherent structure of SLAM by Frese [3].

To the authors knowledge the first SLAM algorithm achieving computation time below $O(n^2)$ per measurement while maintaining a consistent estimate for the whole map was the relaxation algorithm by Duckett et al. [10, 11]. They employed an iterative equation solver called *relaxation* to the linear equation system appearing in maximum likelihood estimation. One iteration is applied 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. This was later improved by the Multilevel Relaxation (MLR) algorithm [12]. It optimizes the map at different levels of resolution similar to multigrid methods used for numerical solution of partial differential equations leading to $O(kn)$ computation time even when closing loops.

Montemerlo et al. [13] 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 (M particles) in which every particle represents a sampled robot trajectory plus a set of n Kalman filters estimating the position for each landmark. The number of particles M is a difficult tradeoff between computation time and quality, especially since it is not clear how M scales with the complexity of the environment. However, the algorithm can handle uncertain landmark identification, which is a unique advantage over the other algorithms discussed in this section.

Guivant and Nebot [14] 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 $O(kn^{3/2})$ with $O(n^{3/2})$ storage space needed.

Thrun et al. [15] 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. The algorithm exploits the observation that the information matrix is approximately sparse[2] 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. Thrun et al. use relaxation but updating only $O(k)$ landmarks after each measurement (using so-called amortization). In general this can negatively affect map quality, since in the numerical literature, relaxation is reputed to need $O(n^2)$ time for reducing the equation error by a constant factor [16].

Bosse et al. [17] avoid the computational problem of updating an estimate for 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.

Paskin [18] views the estimation problem as a Gaussian graphical model. He proposed the *Thin Junction Tree Filter (TJTF)* 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 $O(k^3n)$ time by passing marginalized distributions along the edges of the junction tree. This algorithm is closely related to the treemap algorithm proposed in this paper although both have been indepently developed from completely different perspectives. The correspondence is basically that both use a tree and pass marginalized distributions (TJTF) equivalent to Schur-complements (treemap) along edges.

In the next section the treemap algorithm proposed in this paper will be introduced. It can be used in the same way as CEKF providing an estimate for k landmarks of a local region but with only $O(k^3 \log n)$ computation time when changing the region instead of $O(kn^{3/2})$ for CEKF. Alternatively the algorithm can also compute a global estimate for all n landmarks with computation time $O(kn)$. As reported in the experiments, the prefactor in the $O(kn)$ computation is so small, that this can be done for almost "arbitrarily" large maps (12.37ms for $n = 11300$) being the main contribution from a practical perspective.

3 Basic Idea of the Algorithm

The basic idea of the treemap algorithm is to organize the map hierarchically by decomposing the information into small parts called *information blocks* (IBs) and distributing these IBs along the hierarchy. Then each update involves only a small part of the information. For verification, consider figure 1a with a building that is divided into two parts A and B. Now consider the following question:

[2] this property has later been proven by the author [4, 3].

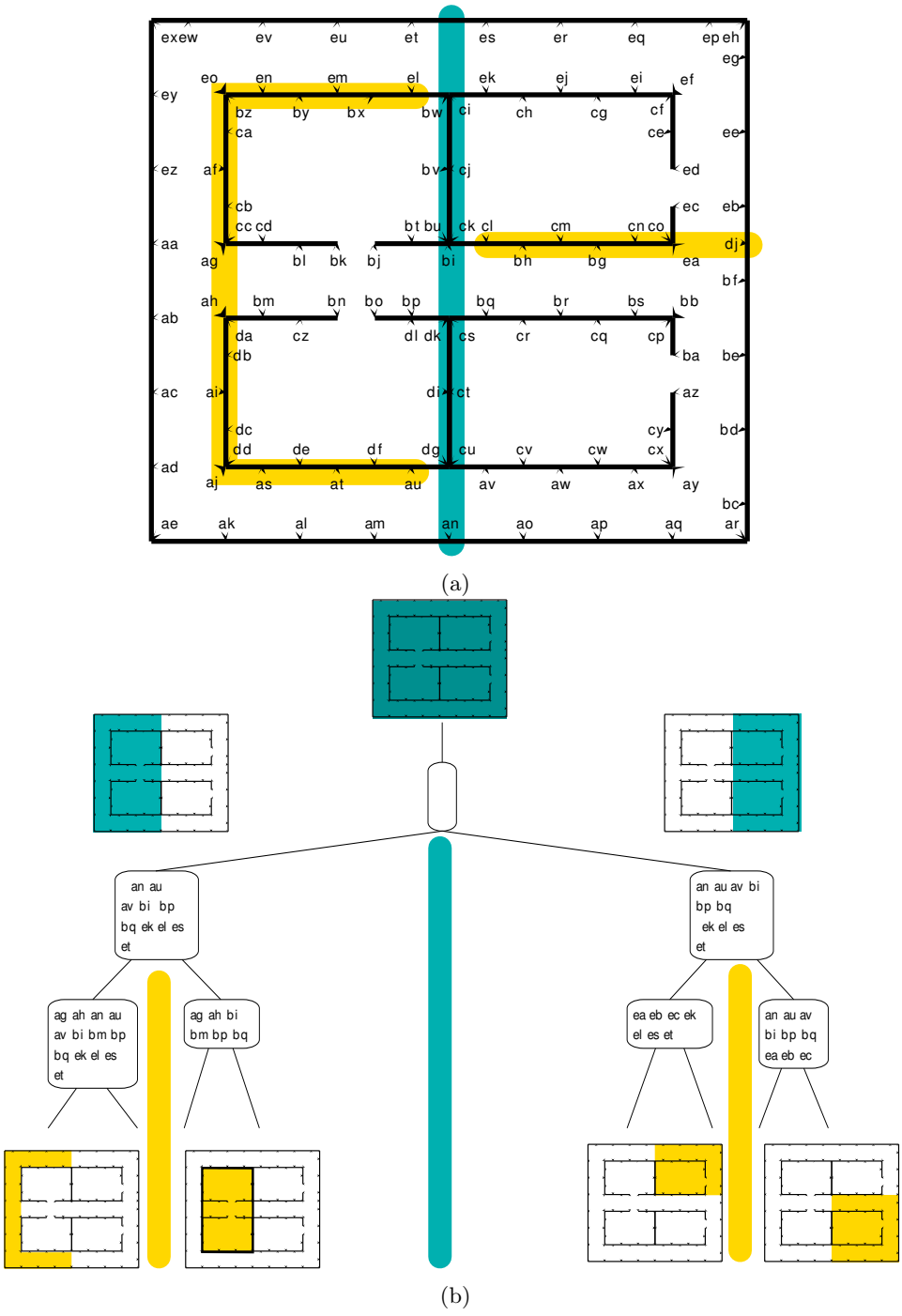


Fig. 1. First two levels of a hierarchically decomposed building (a) and respective tree representation (b). The first level is indicated by bold dark-gray lines, the second level by bold light-gray lines. The region corresponding to a node is shown next to the node.

If the robot is in part A, what is the information needed about B?

Some landmarks of B are observable from A and thus may be involved in measurements while the robot is in A. For integrating these measurements, the algorithm must have all information about these landmarks explicitly available. It is important that this information comprises more than just the measurements that directly involve those landmarks. Rather all measurements in B can indirectly yield information about the landmarks observable from A. So the information needed about B is the whole integrated information of all measurements made while the robot is in B on landmarks observable from A. In the following this information is said to be *condensed*, since it comprises everything from the measurements made in B that is needed outside of B.

The idea can be applied recursively by dividing the building into a hierarchy of regions (Fig. 1a). The recursion stops when the size of a region is comparable to the robot's field of view. The condensed information for the different regions can be computed by recursion. For a specific region condensed information for the two subregions is integrated. After that, all landmarks not being observable from outside the region are removed from representation. This process is called *elimination* of landmarks. This is how the information is decomposed into two parts: Part 1 contains information about eliminated landmarks and is stored at the region and not considered further. Part 2 contains information about the landmarks observable from outside and is passed to the next region above. This part contains every information about the region that is necessary when the robot is outside of the region.

At each moment the robot position corresponds to a particular region on the lowest level of hierarchy called the *actual* region into which new landmark observations can be integrated. When the robot is moving the actual region changes from time to time and a global update has to be performed. The key advantage of the hierarchical decomposition is that therefor only the condensed information of the actual region and all regions above need to be updated.

In a similar way an estimate for the local landmarks can be computed. The final integrated information about a landmark is stored in the region where the landmark has been eliminated. So the information about landmarks of the actual region can be collected by traversing the hierarchy down to the actual region.

4 Treemap Data Structure

This section introduces the *treemap* data structure used by the algorithm. At first, it will be assumed that the robot's observations are landmark – landmark measurements. Under this assumption the algorithm is exact up to linearization. In §7 the algorithm will be extended to integrate also landmark – robot and robot – robot (odometry) measurements with a small approximation when changing regions. Both linearization and odometry approximation are performed when storing a measurement in the treemap. The actual computation of the least

square estimate from the stored information is performed exactly without further approximations. Thereby the algorithm computes a consistent estimate that is statistically compatible with all measurements following requirement (R1).

Data Structure

The hierarchy is realized by a binary tree. Each node corresponds to a region and stores information about the landmarks of this region in so called information blocks (IBs). These IBs are quadratic error functions that describe the negative log-likelihood for a vector of landmark positions given the information represented by the IB. Internally they are represented by a small matrix (the information matrix) and a vector. It is said that an information block, a matrix or a vector respectively *represents* a landmark, if it contains information about it. This means that a row / column of the matrix or an entry of the vector corresponds to the landmark.

The regions corresponding to nodes are not defined geometrically, but rather as a set of landmarks being close to each other. At each moment there is one leaf called the *actual leaf* that corresponds to the region where the robot is currently located. All leaves hold a *Basic Information Block* (BIB). New measurements are integrated into the BIB of the actual leaf called the *actual BIB*. Thus, integration of all BIBs constitutes the complete information contained in the treemap. The information is recursively integrated and decomposed along the tree as described in the previous section: Each node holds a *Condensed Information Block* (CIB) for the information about landmarks observable from outside the region. The node is said to *represent* these landmarks, since the nodes CIB contains all information about this region needed from outside the region. Furthermore, each node holds a *Substitution Information Block*[3] (SIB) containing the information about eliminated landmarks, needed when the robot is inside the region.

Definition 1 (Node) *A node represents those landmarks that are represented both in BIBs inside and in BIBs outside the subtree below this node. It stores a* Condensed Information Block *(CIB) containing the integrated information of all BIBs below this node on the landmarks represented at this node. It further stores a* Substitution Information Block *(SIB) that contains the information from the childrens' CIBs (leaf's BIB resp.) that is not contained in the nodes CIB.*

According to this definition, a landmark is represented from each leaf where the BIB represents the landmark up to the least common ancestor of all those leaves. The least common ancestor is called *elimination node* of the landmark, since it is that node the landmark is eliminated from the CIB and finally stored into a SIB. The different elimination nodes are maintained in an array.

Figure 2 shows the role of the different IBs (BIB, CIB, SIB) and how a nodes CIB and SIB are computed recursively from the children's BIB resp. CIB. For the moment, the symbols **(+)** and **(S)** can be viewed as black box operations

[3] the name is explained in §5

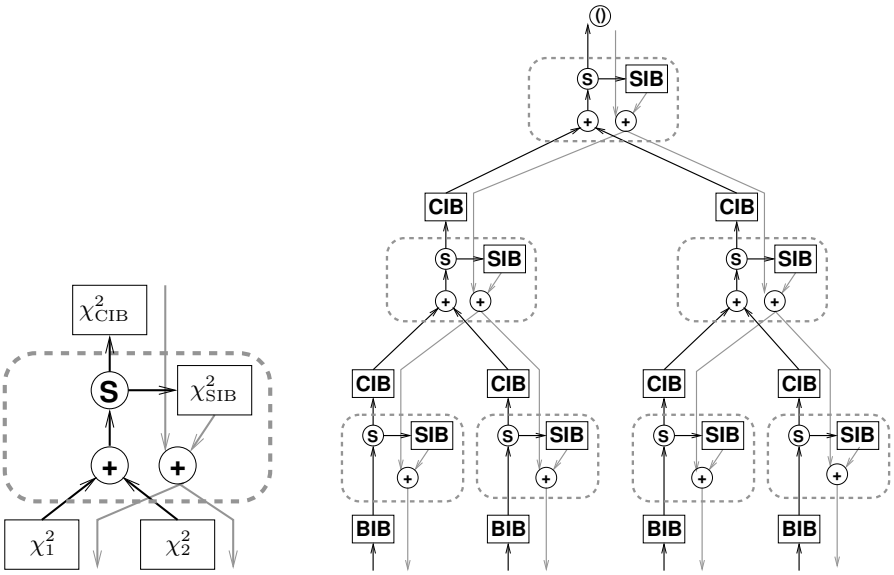


Fig. 2. Integration and decomposition of information in a single node (oval) and a three level tree: Two IBs χ^2_1 and χ^2_2 from the nodes children are integrated (+) and then decomposed (**S**) into a CIB χ^2_{CIB} and a SIB χ^2_{SIB}. The CIB is passed to the parent and the SIB stored at the node (black arrows). Later, an estimate for the landmarks represented at the node is combined (+) with the SIB, resulting in an estimate for the landmarks represented at the children nodes (gray arrows).

integrating and decomposing information. A detailed explanation will follow. Altogether the intention of this approach is to eliminate landmarks as early as possible, so all CIBs and SIBs represent only few landmarks and all involved matrices are small and efficient to handle.

Integration of a measurement

It is currently assumed that all observations are measurements of relative landmark positions (see §7). As long as the observed landmarks are represented in the actual BIB, the measurement can be integrated there and the local estimate can be updated by EKF equations. Such an update does not use the treemap at all and its computation time $O(k^2)$ is independent from the size of the map.

When a landmark is observed that is not represented in the actual BIB, a new BIB must be made the actual one and a global update is required. Since the actual BIB has changed all CIBs and SIBs of ancestor nodes are invalid and must be updated. However most CIBs and SIBs remain unaffected, so computation is highly efficient. After that an estimate for the landmarks represented in the new actual BIB has to be computed. This is done proceeding from the root down to the actual BIB. At each node an estimate for landmarks represented at the

childrens' nodes is computed by combining an estimate for the node's landmarks with the nodes' SIB. In order to compute an estimate for all landmarks the tree is traversed recursively.

Representation of IBs

The purpose of the algorithm is to compute a maximum likelihood estimate for the map. This is equivalent to finding the minimum of the *negative* log-*likelihood* given the statistical information known from the measurements. Since Gaussian noise is assumed, this is a quadratic error function $\chi^2_{\text{all}}(x)$. Each information block also represents a quadratic error function $\chi^2_{\text{IB}}(x)$ referring to the conditional likelihood of landmark position vector x given the information represented by the IB. $\chi^2_{\text{IB}}(x)$ is the negative logarithm of this likelihood and stored using a constant γ, a vector b and a so called information matrix A being symmetric positive semidefinite (SPSD), as

$$\chi^2_{\text{IB}}(x) := x^T A x + x^T b + \gamma = \sum_{i,j} A_{ij}\, x_i x_j + \sum_i b_i\, x_i + \gamma. \tag{1}$$

This is the usual representation of a quadratic function. Each row / column of A and each entry of b corresponds to a landmark's x- or y-coordinate or the robot's x-, y-coordinate or orientation ϕ.

5 Elimination of Landmarks by Schur-Complement

This section presents how to use a mathematical technique called Schur - complement to compute a node's CIB and SIB from the CIB of both children. The first step is to integrate the CIB from both children by simply adding ((+) in figure 2). The second step is to eliminate some landmarks by decomposing the result into two parts (**S**). The first part does not depend on eliminated landmarks any more (CIB). The second part is a maximum likelihood substitution of eliminated landmarks by the remaining ones with a known uncertainty (SIB). The structure of the SIB as a substitution with uncertainty is the reason for the second part of the decomposition being called substitution information block.[4]

This operation is a redistribution of information, since the integrated information of both input CIBs is equal to the integrated information of the resulting CIB and SIB. Figure 2 illustrates the underlying data flow. In the following, the formulas for the integration and decomposition are given ([4] for a derivation).

Lemma 1 *Let $\chi^2_1(x)$ and $\chi^2_2(x)$ be two stochastically independent information blocks. Then the integrated information is*

$$\chi^2(x) = \chi^2_1(x) + \chi^2_2(x). \tag{2}$$

[4] From an abstract statistical perspective, this is just decomposing $P(x) = P\left(\begin{smallmatrix} y \\ z \end{smallmatrix}\right)$ as $P(z)P(y|z)$, i.e. as the product of a marginalized distribution of z and a conditional distribution of y with parameter z.

If both IBs represent different sets of landmarks, the matrices and vectors have to be permuted and extended, so the same columns / rows correspond to the same landmark. For ease of notation it is assumed that A is decomposed into 2×2 blocks such that block row / column 1 corresponds to landmarks to be eliminated and stored in the SIB:

$$\chi^2(x) = x^T A x + x^T b + \gamma \tag{3}$$

$$= \chi^2 \begin{pmatrix} y \\ z \end{pmatrix} = \begin{pmatrix} y \\ z \end{pmatrix}^T \begin{pmatrix} P & R^T \\ R & S \end{pmatrix} \begin{pmatrix} y \\ z \end{pmatrix} + \begin{pmatrix} y \\ z \end{pmatrix}^T \begin{pmatrix} c \\ d \end{pmatrix} + \gamma. \tag{4}$$

The following lemma provides the formulas for decomposing χ^2 into χ^2_{CIB} and χ^2_{SIB} performing the elimination.

Lemma 2 (Schur Complement) *Let $\chi^2\left(\begin{smallmatrix} y \\ z \end{smallmatrix}\right)$ be an information block as in (4) with P being symmetric positive definite (SPD). Then $\chi^2(x)$ can be uniquely decomposed into an information block $\chi^2_{CIB}(z)$ on z and an information block $\chi^2_{SIB}(Hz + h - y)$ on $Hz + h - y$, with $\chi^2_{SIB}(0) = 0$:*

$$\chi^2_{SIB}(w) = w^T P w, \quad H = -P^{-1} R^T, \quad h = -P^{-1} c/2. \tag{5}$$

An estimate can be easily computed from the SIBs: Since the root node represents no landmark, start with an empty estimate $\hat{x} = ()$, with covariance $C = ()$. Proceed down and use the estimate for a node's landmarks and the SIB stored there to derive an estimate for the landmarks represented at the node's children applying lemma 3:

Lemma 3 *Let $\chi^2(x)$ be decomposed as in lemma 2 and let $\hat{z}$ be an estimate with covariance C. Then the optimal estimate for y is*

$$\hat{y} = H\hat{z} + h, \text{ with } \operatorname{cov} \begin{pmatrix} y \\ z \end{pmatrix} = \begin{pmatrix} HCH^T + P^{-1} & HC \\ CH^T & C \end{pmatrix}. \tag{6}$$

With lemma 1, 2 and 3 the necessary tools for using a treemap are available (Fig. 2). Lemma 1 and 2 are used from the leaves up to the root (black arrows) and lemma 3 from the root down to the leaves (gray arrows). When a global update is performed only the way from the old actual BIB up to the root and down again to the new actual BIB has to be computed. Even when an estimate for all landmarks is desired, computation is extremely efficient, since lemma 3 (without covariance) requires just a small matrix-vector multiplication.

6 Assumptions on Topologically Suitable Buildings

The time needed for the computation discussed above depends on the size of the matrices involved, which is determined by the number of landmarks represented at the node's children. So for the algorithm to be efficient it is crucial that each node represents only a few landmarks. Thus, the tree must hierarchically

divide the building in a way that each node, i.e. each region, contains only a few landmarks observable from outside the region. Achieving this goal requires some sophisticated optimization of the tree, since it is not a simple bookkeeping task. As experiments and the following considerations confirm, this is possible for typical buildings, which will be called *"topologically suitable"*.

Typical buildings allow such a hierarchical partitioning because they are hierarchical themselves, consisting of floors, corridors and rooms. Different floors are only connected through a few staircases, different corridors through a few crossings and different rooms most often only through a single door and the adjacent parts of the corridor. Thus, on the different levels of hierarchy natural regions are: rooms, part of a corridor including adjacent rooms, one or several adjacent corridors and one or several consecutive floors (Fig. 3).

To allow a thorough theoretical analysis of the algorithm it is formally assumed that the building is topologically suitable:

Definition 2 (Topologically suitable building) *Let the building be decomposed into a hierarchy of regions according to definition 1. Let k ("number of local landmarks") be the maximum number of landmarks represented in a BIB. Then the building is said to be topologically suitable if the following holds:*

1. *For each node only $O(k)$ landmarks exist that are represented both in BIBs inside and in BIBs outside the subtree of this node.*
2. *Each BIB shares landmarks only with $O(1)$ other BIBs.*

The parameter k is small, since the robot can only observe a few landmarks simultaneously because its field of view is limited both by walls and sensor range. In particular, k does not increase when the map gets larger ($n \to \infty$). Although by this argument $k = O(1)$, the asymptotical expressions in this paper explicitly show the influence of k and do not formally assume k to be constant.

A counter-example for a not topologically suitable building is a large open storeroom with many boxes, where the robot can navigate arbitrarily not confined to designated paths. A region corresponding to one half of the hall will have a whole border line with the region corresponding to the other half and thus violate condition 1. For cross-country navigation, the same problem appears, when the robot builds an area-wide map covering every detail. However, in most cases the goal is to explore a large area rather than mapping a small area in detail. Thus, the robot will use passable paths once it has found them. So again, each region will be connected to the remaining map only with a few of these paths and definition 2 is fulfilled.

Condition 1 is powerful. The fact that buildings have such a loosely connected topology is a key property distinguishing SLAM from other estimation problems.

Computational Efficiency

By condition 2 there are $O(\frac{n}{k})$ nodes in the tree each storing matrices of dimension $O(k \times k)$ (condition 1). Thus, the storage requirement of the treemap is $O(k^2 \cdot \frac{n}{k}) = O(nk)$ meeting requirement (R2).

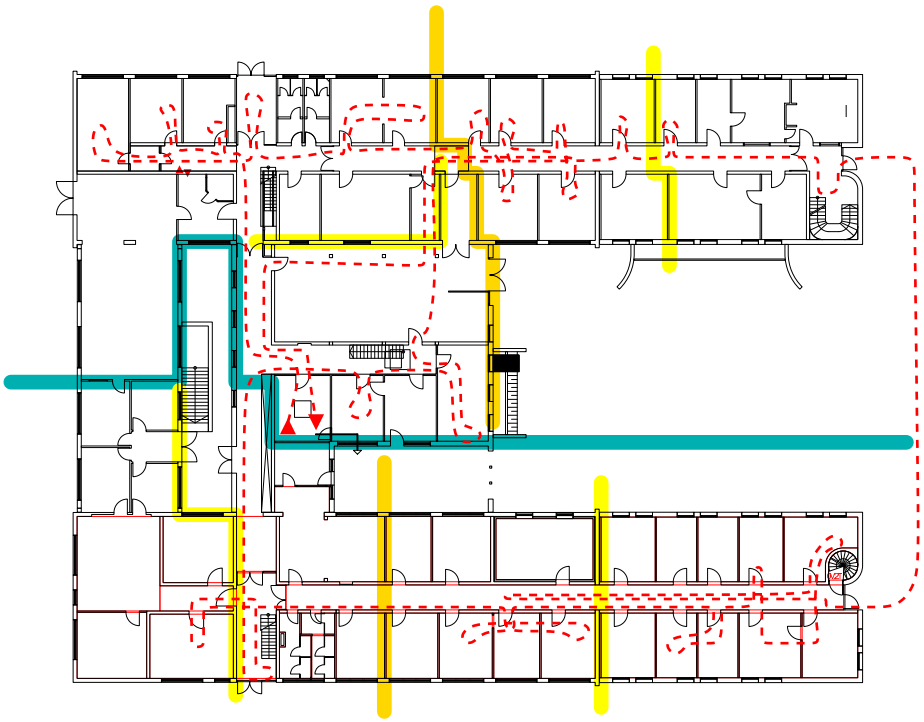


Fig. 3. DLR Institute of Robotics and Mechatronics – A typical topologically suitable building with the first three level of a suitable hierarchical partitioning. The building has been mapped in the experiments reported in §10, with the dashed line sketching the robots trajectory. Start and finish are indicated by small triangles.

Computation time depends: When a measurement involves only landmarks represented in the actual BIB it can be integrated into this BIB and the estimate can be updated using EKF equations. Similar to CEKF this needs $O(k^2)$ computation time, independent from n. Otherwise, a different BIB is made actual one and a global update has to be performed. The update basically requires recomputing the CIB and SIB from the old actual BIB up to the root and compiling an estimate from the root down to the new actual BIB ($O(k^3)$ per node). There are $O(\log n)$ nodes to be updated, so the overall time is $O(k^3 \log n)$. Under some circumstances, more nodes are involved and additional computation is necessary for bookkeeping but still with the same asymptotical complexity.

In order to compute an estimate not only for local but for all landmarks, lemma 3 must be applied recursively from the root down to all BIBs taking $O(kn)$. It will turn out in the experiments in §9 that the prefactor involved is extremely small. So while from a theoretical perspective the possibility to perform updates in sublinear time is most appealing, practically the algorithm allows computing an estimate for all landmarks in extremely large maps.

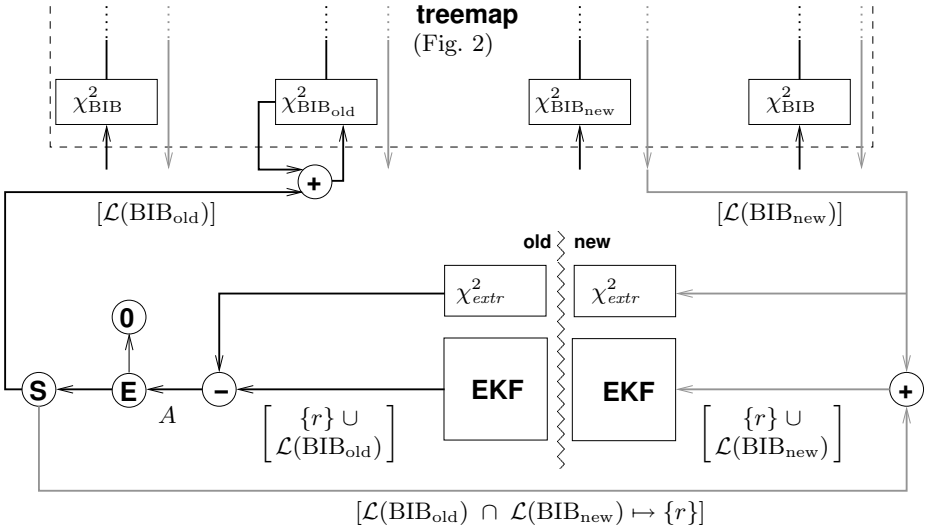


Fig. 4. Data flow between EKF and treemap when changing the actual BIB from $\mathrm{BIB}_{\mathrm{old}}$ to $\mathrm{BIB}_{\mathrm{new}}$. Represented landmarks are shown in brackets (Tab. 1). Black arrows depict information matrices, gray arrows covariance matrices.

7 Integration of Odometry Measurements

Up to now the observations have been assumed to consist of landmark – landmark measurements, i.e. information about the relative locations of a group of landmarks. Lu & Milios [19] established a well known approach utilizing this kind of information, where laserscan reference frames are treated as "landmarks" and the relative pose of two scans is "measured" by scan matching. Indeed the treemap algorithm could be readily used to solve the linear equation system derived by Lu & Milios reducing computation time from $O(n^3)$ to $O(kn)$ or to $O(k^3 \log n)$ for an incremental local estimate.

Most often measurements are landmark – robot measurements, i.e. information about the relative location of a landmark with respect to the robot. Another source of information is odometry, i.e. robot – robot measurements providing information of the current robot pose relative to a previous robot pose. These information could be processed as well by the algorithm described so far if all robot poses were explicitly represented as random variables to be estimated, like Lu & Milios did. However violates (R2) because it leads to map size growing even if moving through an area already mapped.

In the following it will be discussed how to avoid represantation of old robot poses by using an EKF as preprocessing stage.

Landmark – Robot Measurements

First assume that odometry can be neglected, i.e. the robot's motion is evident from the landmark observations alone: Each robot pose is considered as a separate random variable that can be eliminated since it will not appear in any further measurement: The landmark measurements made at a certain robot pose are integrated into an IB representing the robot pose and all involved landmarks as random variables. Then the robot pose is eliminated from the IB using Schur complement (lemma 2). The resulting IB does not represent the robot pose any more and can be integrated into the actual BIB just the same way like pure landmark – landmark measurements.

The precondition of this approach is that at least two common landmarks are being observed from successive robot poses. If this condition is met, odometry can often be neglected [20]. Theoretically, this is even appealing, since the assumption of stastistical independence between successive odometry measurements is hardly true in reality. Although this is not a theoretically optimal approach, it will presumably be a good choice in practice and considerably simpler than the more general approach described below.

Robot-Robot Measurements (Odometry)

When odometric measurements have to be integrated, it is necessary to represent the robot pose as a random variable. Thus old robot poses have to be eliminated later to prevent the map size from growing. This leads to new couplings introduced between all landmarks observed from an eliminated robot pose and in the end between all pairs of landmarks.

To avoid this dilemma a conservative approximation is performed. All coupling coefficients are eliminated except those with landmarks represented in the actual BIB. This means to deliberately discard the information contained in the eliminated coupling coefficients to make the representation less complex. It has been proven [3], that the occurring information matrices are approximately sparse. This theorem ensures, that couplings decay exponentially with distance traveled and not too much information is discarded by the elimination.

The measurements are integrated by an EKF as a preprocessing stage (Fig. 4). It represents the robot pose and all landmarks of the actual BIB and can directly integrate odometry and landmark observations. The information about the robot pose is exclusively contained in the EKF and not transferred into the treemap. When a global update becomes necessary all coupling coefficients between the robot pose and landmarks not represented in the new actual BIB are eliminated:

First, the EKF state is converted into an information block χ^2. Then, the information χ^2_{extr} is subtracted (-). This is the information obtained from the tree map the last time the EKF was initialized and must not be integrated a second time. The resulting difference A is the information gained from measurements since then. Next, the couplings between robot pose and landmarks not

Table 1. Random variables corresponding to different block rows / columns of A. $A_{21} = A_{12}^T$ is to be eliminated. $\mathcal{L}(\text{BIB})$ denotes the landmarks represented in BIB.

Blockrow	Notation	Random variables
1	$[\{r\}]$	Robot pose
2	$[\mathcal{L}(\text{BIB}_{\text{old}}) - \mathcal{L}(\text{BIB}_{\text{new}})]$	Landmarks represented in the old but not in the new actual BIB
3	$[\mathcal{L}(\text{BIB}_{\text{old}}) \cap \mathcal{L}(\text{BIB}_{\text{new}})]$	Landmarks represented in both the old and new actual BIB

represented in BIB_{new} are eliminated **(E)** by subtracting a SPSD matrix cancelling the necessary coefficients from A. This means, a part of the information is deliberately discarded **(0)** to give the remaining information a simpler structure. After this the robot pose is eliminated from the IB by Schur complement **(S)** and the resulting CIB is added to BIB_{old} **(+)** replacing it in the treemap. The corresponding SIB defines the robot pose as a function of landmarks which are both in BIB_{old} and BIB_{new} (due to **(E)**). After the estimate for BIB_{new} has been generated by updating the treemap, the SIB can be integrated **(+)**. The result is the estimate for the landmarks of the new actual BIB and the robot pose. Together with the corresponding covariance matrix the estimate is used as a new EKF state.

Stepwise Optimal Elimination of Off-Diagonal Entries

In this section the procedure **(E)** is derived. It eliminates some coupling entries in an information matrix A by subtracting a so called elimination matrix B. The key idea is to make B small so as little information as possible is discarded. This task is similar to the *sparsification* procedure used by Thrun et al. [15] in their Sparse Extended Information Filter (SEIF) algorithm. Their approach optimally approximates the original distribution in the sense of Kullback-Leibler (KL) divergence. In contrast to the approach taken here this is not conservative reporting some aspects of the map to be more precise than they actually are.

The problem is reduced from a k-D problem to k 1-D problems by eliminating different coupling entries columnwise, where the following theorem gives an optimal solution for eliminating a single column.

Theorem 1 (Elimination matrix) *Let A be a 3×3 block SPD matrix being decomposed as $A = \begin{pmatrix} \psi & r^T & w^T \\ r & S & W^T \\ w & W & X \end{pmatrix}$ with 1-dimensional first block row / column. Then the best elimination matrix for A_{21} is xx^T with x defined as*

$$x = A \begin{pmatrix} \gamma \\ \delta S^{-1} r \\ 0 \end{pmatrix}, \text{with} \tag{7}$$

$$\alpha = r^T S^{-1} r, \quad \beta = (\psi - \alpha)^{-1},$$
$$\lambda = \sqrt[4]{\psi (r^T S^{-1} r)},\ \gamma = \beta(\lambda - \lambda^{-1}\alpha),\ \delta = \beta(-\lambda + \psi\lambda^{-1}). \tag{8}$$

The result is optimal with respect to (R1) since it minimizes the *worst factor* by which the covariance of *any* aspect of the map is increased. For lack of space the reader is referred to [4] for a mathematical discussion.

Each measurement is affected by the elimination operation only once, namely the next time when the actual BIB changed. So the elimination procedure preserves topological information, i.e. when measurements report two landmarks to be close to each other this information will be included in the BIB although less precisely. Since propagation of information through the tree is extact, a loop will be closed in the estimate immediately after integrating the corresponding measurement. This indicates although does not proof that the algorithm complies with (R1) and will be further investigated with simulation experiments in §9 reporting the actual increase of error encountered.

8 Maintenance of the Hierarchy

Up to now the linear algebra part of the algorithm has been described. It provides the subalgorithms for manipulating IBs and in the end for computing an estimate from the measurements. The bookkeeping part of the algorithm takes care to update CIBs and SIBs as necessary using the subalgorithms decribed before. It further optimizes the tree, so that it is balanced and hierarchically partitions the set of BIBs in a way that at any level of hierarchy a partition shares only a few landmarks with BIBs not belonging to the partition. Thus the node corresponding to the partition represents only a few landmarks, and computation at this node is efficient. This is problem is in theory NP-complete, with many established heuristic approaches existing[21]. The algorithm incrementally optimizes the tree by moving a single subtree to a different location whenever a global update is performed[4].

There exists a nonlinear extension to the algorithm that corrects the linearization error resulting from large error in the robot orientation by applying "Nonlinear Rotations" to individual IBs before integrating them. The extension is omitted here due to lack of space refering the reader to [4] for an extensive discussion and experimental results handling up to 140° orientation error.

9 Simulation Experiments

This section presents the simulation experiments conducted to verify the algorithm with respect to the requirements (R1)-(R3). For this purpose, a simulation approach is advantageous because ground truth is available and it allows to repeat the same experiment with identical measurements but new independent measurement noise.

All experiments have been conducted on an Intel Xeon, 2.67 GHz with $2.5\%, 2^\circ$ noise for the landmark sensor, $0.01\sqrt{\mathrm{m}}$ noise for the odometry sensor (proportionally to square root of distance traveled) and a robot radius of 0.3m.

The algorithm's parameters are $optHTPSteps = 5$ steps of tree optimization per global update and $maxDistance = 5$m as maximum diameter of a region.

Clearly space (R2) and time (R3) consumption are straightforward to measure but how should one assess map quality with respect to requirement (R1)?

Assessment of Map Quality

With known ground truth the estimation error can readily be computed. But while it is a good measure for the overall system performance, it doesn't tell anything about the algorithm. An error, for example of 1m, could either be caused by large sensor noise despite an optimal algorithm or it could be caused by crude approximations in the algorithm despite precise sensor measurements. To assess the performance of the algorithm with respect to requirement (R1) the error must be compared to the "*minimal uncertainty that could be theoretically derived from the measurements*" as evident from the optimal nonlinear Maximum Likelihood estimate. So if, for instance the ML estimate has an error of 0.5m it can be concluded, that the algorithm has increased the error by 100%. This number i.e. the relative error indicates the prize to pay for using the algorithm instead of ML estimation and characterizes the algorithm's map quality with respect to (R1). Another point to consider when interpreting absolute error specifications is that the absolute error is accumulating and thus depends on the map size.

To summarize: When the focus is on the core estimation algorithm not on the overall system, relative not absolute error is the quantity to be considered.

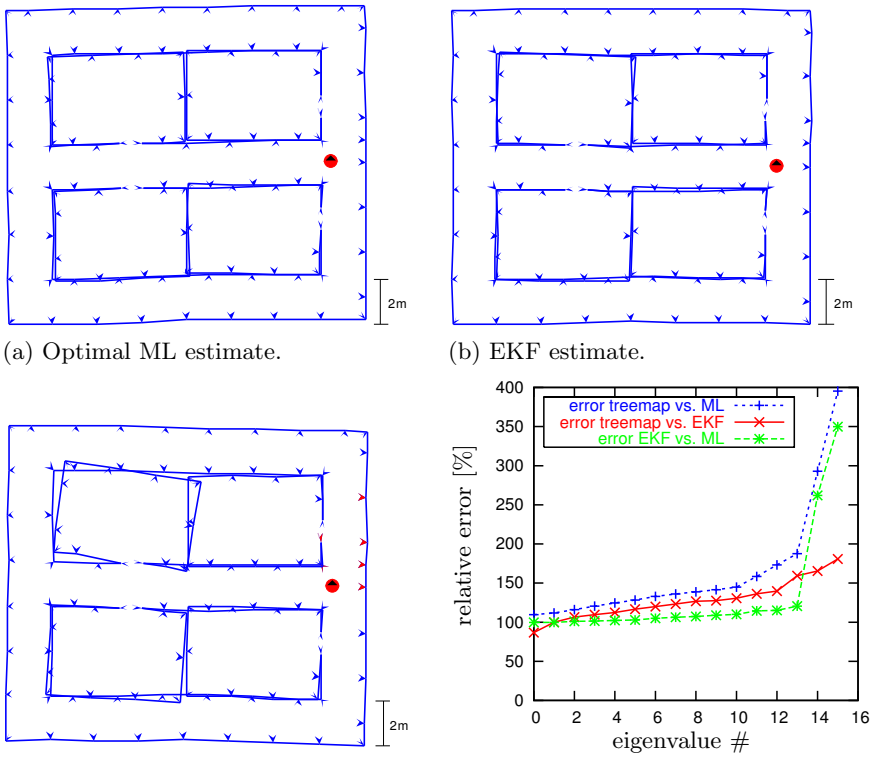
It is well known [3] that relative aspects of a map e.g. the distance between two landmarks have much less uncertainty than absolute landmark positions. Since the uncertainty of absolute landmark positions is often several meters navigation would be impossible otherwise. Thus it is essential, not only to look at the relative error of different landmarks but at the relative error of *any aspect* of the map as required by (R1). It has been derived [3] that this can be done by computing a generalized eigenvalue spectrum

$$Cv = \lambda\, C_{\mathrm{ML}} v \tag{9}$$

of the covariance of the algorithm's estimate C relative to the covariance of the maximum likelihood estimate C_{ML}. The generalized eigenvalue λ corresponding to an eigenvector v gives the squared relative error in the two estimates for the aspect corresponding to the eigenvector v. These eigenvalues characterizes the relative error encountered in different aspects of the map just the same way as ordinary eigenvalues characterize the absolute error in different aspects.

Small Map Experiment

The small map simulation experiment allows statistical evaluation of the estimation error and comparison with EKF and ML (Fig. 5). At first sight all three basically appear of same quality (except for the left upper room in the treemap



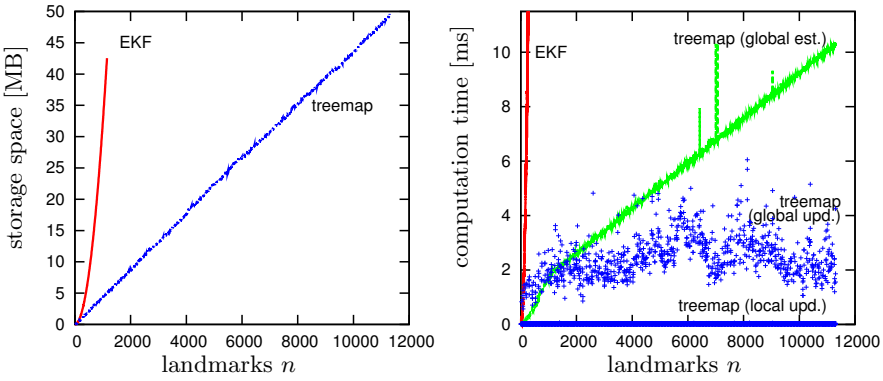
(a) Optimal ML estimate. (b) EKF estimate.

(c) Treemap estimate. (d) Relative error spectrum.

Fig. 5. Small map simulation experiment results.

estimate) and perfectly usable for navigation. Quantitative inspection however will still show a notable difference:

Figure 5d compares the relative error in the three estimates in *all aspects* of the map. The error covariances C for treemap, C_{EKF} for EKF and C_{ML} for ML are approximatively determined by Monte Carlo simulation with 1000 runs. To limit the number of runs necessary only eight selected landmarks are evaluated. The square root of the smallest eigenvalue is 110% (87% vs. EKF) and the largest 395% (181% vs. EKF). This means that the map estimate computed by treemap has an error 10% larger in the best aspect and 295% larger in the worst aspect than the ML estimate. The typical (median) relative error is 137% compared to ML with two outliers of 395% and 293% and typically (median) 125% compared to EKF. The outliers are also apparent in the plot comparing EKF to ML, so they are probably caused by linearization errors occurring in EKF and treemap. This is surprising since at visual inspection the EKF map is so good one would hardly suspect linearization problems.



(a) Storage space. (b) Computation time per measurement.

Fig. 6. Large scale simulation experiment: Storage space and computation time over number of landmarks n. Observe different computation time for a local update, a global update and for computing a global estimate.

Large Scale Map Experiment

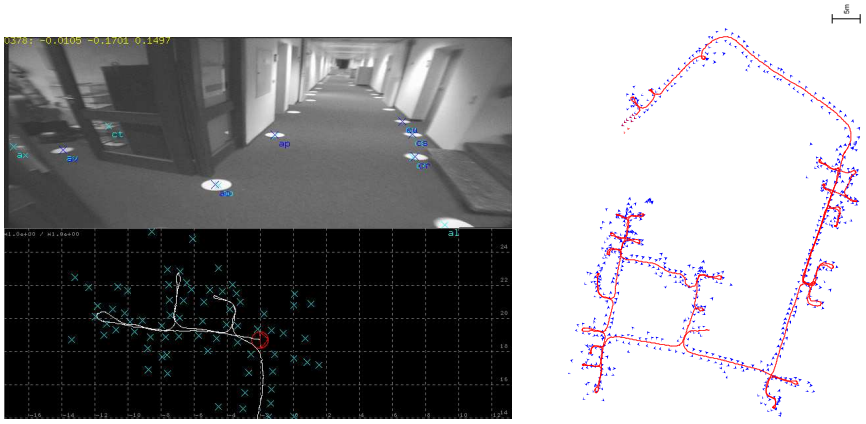
The second experiment uses an extremely large map consisting of 10×10 copies of the building used before (not shown for its size). The experiment encompasses $n = 11300$ landmarks, $m = 312020$ measurements and $p = 63974$ robot poses. The EKF experiment was aborted earlier due to large computation time.

In figure 6a storage space consumption is clearly shown to be linear for treemap ($O(kn)$) and super-linear ($O(n^2)$) for EKF. Overall computation time was 31.34s for treemap and 18.89 days (extrapolated $\sim n^3$) for EKF. Computation time per measurement is shown in figure 6b. Time for three different computations is given: Local updates (dots below < 0.5ms), global updates computing a local map (scattered dots above 0.5ms) and the additional cost for computing a global map are plotted w.r.t. n. The algorithm is extremely efficient updating an $n = 11300$ landmark map in 12.37ms. Average time is $1.21\mu\text{s}\cdot k^2$ for local update, $0.38\mu\text{s} \cdot k^3 \log n$ for global update and $0.15\mu\text{s} \cdot kn$ for a global map respectively. The latter is surely the most impressive result from a practical perspective.

10 Real World Experiments

The real world experiments reported in this section are used to demonstrate how to apply the treemap algorithm in practice by mapping the DLR Institute of Robotics and Mechatronics' building (Fig. 3). It is used as an example for a typical office building and indeed turns out to be "topologically suitable" as defined in §6. The algorithm is generating a balanced and well partitioned tree representation online and closes three large loops during mapping.

In the experiments a wheeled mobile robot was moved manually through the building. The robot is equipped with a camera system (field of view: $\pm 45°$) at



(a) Screen shot with live image and map estimate. (b) Before closing the loop.

Fig. 7. Real world experiments: Implementation mapping the DLR building.

a height of 1.55m. As maximum diameter of a region $maxDistance = 7$m was used. For the purpose of conducting this experiments circular artificial landmarks were set throughout the floor of the building (Fig. 7a) and visually detected by a combination of Hough-transform and a gray-level variance criterion.

Since the landmarks are identical, identification is based on their relative position employing two different strategies in parallel: Local identification is performed by simultaneously matching all observations from a single robot pose to the map taking into account both error in each landmark observation and error in the robot pose. For global identification considerable difficulties were encountered in detecting closure of a loop: Before closing the largest loop the accumulated robot pose error was 16.18m (Fig. 7b, 8) and the average distance between adjacent landmarks was $\approx$ 1m. With indistinguishable landmarks matching observations from a single image was not reliable enough.

Instead, the algorithm has been designed to match a map patch of radius 5m around the robot. When the map patch is recognized somewhere else in the map, the identity of all landmarks in the patch is changed accordingly and the loop is closed. It is a particular advantage of the treemap algorithm to be able to change the identity of landmarks already integrated into the map (referred to as *lazy data association* by Haehnel et al. [22]). Technical details of computer vision and landmark identification can be found in [4]. The final map contains 725 landmarks, 29142 measurements and 3297 robot poses (Fig.7b, 8). The results highlight the advantage of using SLAM because after closing the loop the map is much better and at visual inspection impressively good for such a large building. Figure 9 shows the internal tree representation used by the algorithm. On the average there are $k \approx 16.39$ landmarks represented in each BIB. The tree is balanced and well partitioned, i.e. no node represents too many landmarks. It can be concluded that the building is indeed topologically suitable in the sense discussed in §6. Computation time is extremely low (0.07ms per measurement)

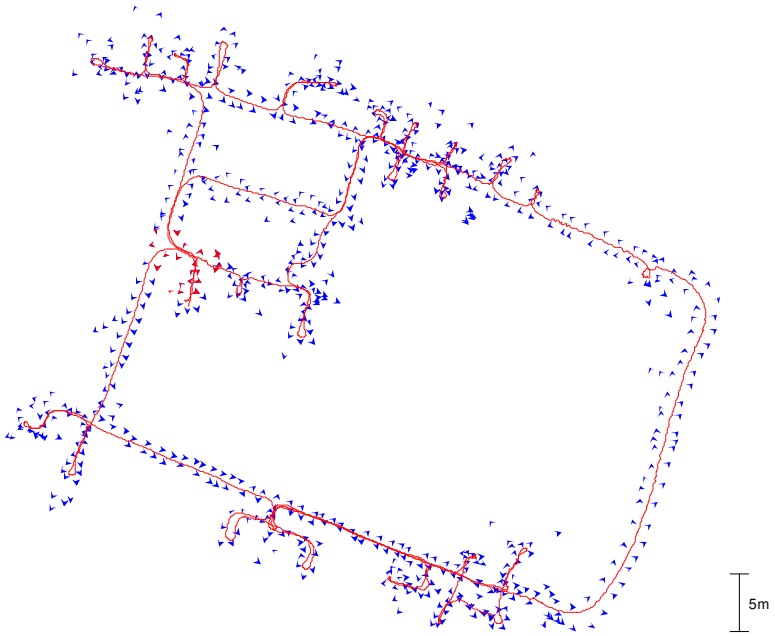


Fig. 8. Real world experiments: Final map estimate.

if, only a local update is performed as is the case most often. The average time is $0.77\mu\text{s} \cdot k^2$ for local update, $0.02\mu\text{s} \cdot k^3 \log n$ for global update and $0.04\mu\text{s} \cdot kn$ for a global map respectively (Fig. 10). Accumulated computation time is 2.95s for treemap and 601s (extrapolated $\sim n^3$) for EKF.

11 Conclusion

The treemap SLAM algorithm proposed in this paper works by dividing the map into a hierarchy of regions represented as a binary tree. With this data structure, the computations necessary for integrating a measurement are limited essentially to updating a leaf of the tree and all its ancestors up to the root. From a theoretical perspective the main advantage is that a local map can be computed in $O(k^3 \log n)$ time. Practically, it is equally important that a global map can be computed in $O(kn)$ additional time allowing computation of a map with $n = 11300$ landmarks in 12.37ms on an Intel Xeon, 2.67 GHz.

With respect to the three proposed criteria the algorithm was verified theoretically, by simulation experiments, and by experiments with a real robot. A precondition is a typical, topologically suitable building as explained in §6.

From the author's perspective a drawback is the algorithms complexity necessary for performing bookkeeping in $O(k^3 \log n)$. Consequently a promising idea currently investigated is to simplify the algorithm for computing a global map in $O(kn)$ rather than a local in $O(k^3 \log n)$.

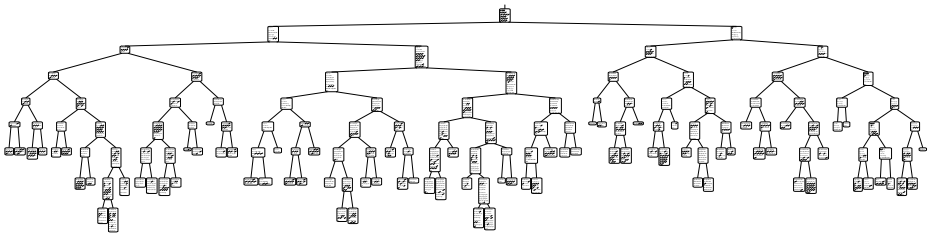
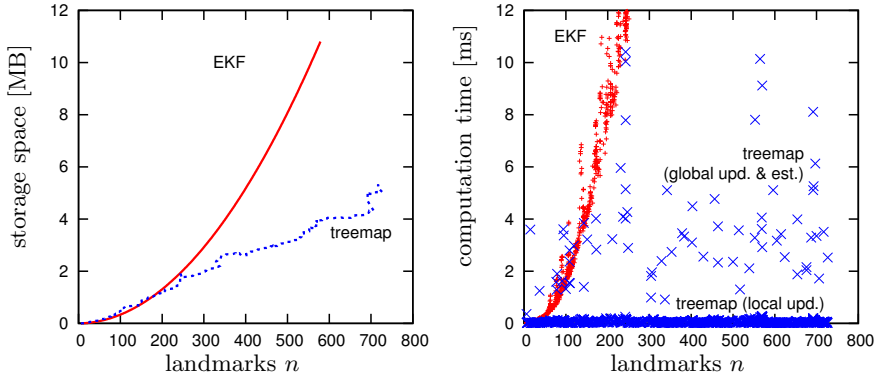


Fig. 9. Tree representation of the map. Size of the node ovals is proportional to number of represented landmarks.



(a) Storage space.

(b) Computation time per measurement. Observe local updates ≈ 0.07ms.

Fig. 10. Real experiment performance.

Apart from computation time, the most important challenge is landmark identification. Multi-Hypothesis tracking is generally seen as a promising idea to tackle situations where identification is difficult. With such an approach, efficiency of the core algorithm becomes even more crucial as it has to handle all hypotheses simultaneously, multiplying the computation time needed.

References

1. Gauss, C.: Theoria combinationis observationum erroribus minimis obnoxiae. Commentationes societatis regiae scientiarum Gottingensis recentiores **5** (1821) 6–93
2. Frese, U., Hirzinger, G.: Simultaneous localization and mapping - a discussion. In: Proceedings of the IJCAI Workshop on Reasoning with Uncertainty in Robotics, Seattle. (2001) 17 – 26
3. Frese, U.: A discussion of simultaneous localization and mapping. Autonomous Robots (2004) (submitted).
4. Frese, U.: An $O(\log n)$ Algorithm for Simulateneous Localization and Mapping of Mobile Robots in Indoor Environments. PhD thesis, University of Erlangen-Nürnberg (2004)

5. McNamara, T.: How are the locations of objects in the environment represented in memory? In Freksa, C., Brauer, W., Wender, C.H.K., eds.: Spatial cognition III: Routes and navigation, human memory and learning, spatial representation and spatial reasoning, Berlin: Springer-Verlag (2003) 174–191
6. Moratz, R., Tenbrink, T., Bateman, J., Fischer, K.: Spatial knowledge representation for human-robot interaction. In Freksa, C., Brauer, W., Habel, C., Wender, K., eds.: Spatial Cognition III. Springer (2003) 263 – 286
7. Skubic, M., Perzanowski, D., Blisard, S., Schultz, A., Adams, W., Bugajska, M., Brock, D.: Spatial language for human-robot dialogs. IEEE Transactions on System Man Cybernetics Part C **34** (2004) 154 – 167
8. Smith, R., Self, M., Cheeseman, P.: Estimating uncertain spatial relationships in robotics. In Cox, I., Wilfong, G., eds.: Autonomous Robot Vehicles. Springer Verlag, New York (1988) 167 – 193
9. Thrun, S.: Robotics mapping: A survey. Technical report, School of Computer Science, Carnegie Mellon University (2002)
10. Duckett, T., Marsland, S., Shapiro, J.: Learning globally consistent maps by relaxation. In: Proceedings of the IEEE International Conference on Robotics and Automation, San Francisco. (2000) 3841–3846
11. Duckett, T., Marsland, S., Shapiro, J.: Fast, on-line learning of globally consistent maps. Autonomous Robots **12** (2002) 287 – 300
12. Frese, U., Larsson, P., Duckett, T.: A multigrid algorithm for simultaneous localization and mapping. IEEE Transactions on Robotics (2004) (to appear).
13. Montemerlo, M., Thrun, S., Koller, D., Wegbreit, B.: FastSLAM: A factored solution to the simultaneous localization and mapping problem. In: Proceedings of the AAAI National Conference on Artificial Intelligence, Edmonton. (2002) 593–598
14. Guivant, J., Nebot, E.: Solving computational and memory requirements of feature based simultaneous localization and map building algorithms. Technical report, Australian Centre for Field Robotics, University of Sydney, Sydney (2002)
15. Thrun, S., Koller, D., Ghahramani, Z., Durrant-Whyte, H., A.Y., N.: Simultaneous mapping and localization with sparse extended information filters: Theory and initial results. In: Proceedings of the Fifth International Workshop on Algorithmic Foundations of Robotics, Nice. (2002)
16. Press, W., Teukolsky, S., Vetterling, W., Flannery, B.: Numerical Recipes, Second Edition. Cambridge University Press, Cambridge (1992)
17. Bosse, M., Newman, P., Leonard, J., Teller, S.: SLAM in large-scale cyclic environments using the Atlas framework. International Journal on Robotics Research (2003) (to appear).
18. Paskin, M.: Thin junction tree filters for simultaneous localization and mapping. In Gottlob, G., Walsh, T., eds.: Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI-03), San Francisco, CA (2003) 1157–1164
19. Lu, F., Milios, E.: Globally consistent range scan alignment for environment mapping. Autonomous Robots **4** (1997) 333 – 349
20. Gutmann, J., Konolige, K.: Incremental mapping of large cyclic environments. In: Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation, Monterey. (1999)
21. Fiduccia, C., Mattheyses, R.: A linear-time heuristic for improving network partitions. In: Proceedings of the 19th ACM/IEEE Design Automation Conference, Las Vegas. (1982) 175–181
22. Hähnel, D., Burgard, W., Wegbreit, B., Thrun, S.: Towards lazy data association in SLAM. In: In Proceedings of the 10th International Symposium of Robotics Research (ISRR'03). (2003)