An $O(\log n)$ Algorithm for Simultaneous Localization and Mapping of Mobile Robots in Indoor Environments

Ein $O(\log n)$ Algorithmus zur simultanen Lokalisierung und Kartierung von mobilen Robotern in Innenräumen

Submitted to the

Technische Fakultät der
Universität Erlangen–Nürnberg

in partial fulfillment of the requirements for
the degree of

DOKTOR–INGENIEUR

from

Udo Frese

Erlangen — 2004

As dissertation accepted by the
Technische Fakultät der
Universität Erlangen–Nürnberg

Date of submission:	7th January 2004
Date of defense:	28th May 2004
Dean:	Prof. Dr. rer. nat. A. Winnacker
Reviewer:	Prof. Dr.-Ing. H. Niemann
	Prof. Dr.-Ing. G. Hirzinger

To Frauke

Abstract

This thesis addresses the Simultaneous Localization and Mapping (SLAM) problem, a key problem for any truly autonomous mobile robot. The task for the robot is to build a map of its environment and simultaneously determine its own position in the map while moving.

The problem is examined from an estimation-theoretic perspective. The focus is on the core estimation algorithm which provides an estimate for the map and robot pose from two sensor inputs: The first sensor is odometry, i.e. the observation of the robot's movement from the revolution of its wheels. The second is the observation of environment features, so called landmarks. The optimal solution based on maximum likelihood or least square estimation needs excessive computation time, i.e. $O((n+p)^3)$ for n landmarks and p robot poses. Popular approaches like Extended Kalman Filter (EKF) are more efficient but still need $O(n^2)$ computation time and suffer from linearization errors.

The first contribution of this thesis is an analysis of SLAM, in particular under the aspect of the inherent uncertainty structure of a map estimate. The key result can be phrased as "Certainty of relations despite uncertainty of positions". The discussion further analyzes the linearization error in SLAM and identifies the error in the robot's orientation as dominant source.

The second and main contribution is 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. On the level of finest subdivision, i.e. the lowest level, these matrices are naturally small because the regions are small and contain few landmarks only. On higher levels regions are large and contain many landmarks. For keeping the matrices stored at higher levels small only those landmarks are represented being observable from outside the region. This way it is ensured that even on high levels of hierarchy each matrix represents only few landmarks and computation is efficient.

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. Furthermore, the proposed hierarchy allows "nonlinear rotation" of the matrix stored at a certain region. Thereby linearization problems can be removed.

The algorithm is evaluated for map quality, storage space and computation time using simulation experiments and experiments with a real mobile robot in an office environment.

Kurzfassung

Diese Arbeit beschäftigt sich mit dem Problem simultaner Lokalisierung und Kartierung (Simultaneous Localization and Mapping, SLAM), einem Schlüsselproblem für jeden wirklich autonomen Roboter. In diesem Problem besteht die Aufgabe eines Roboters in Bewegung in der Darstellung einer Karte seiner Umgebung sowie in der gleichzeitigen Bestimmung der eigenen Position auf dieser Karte.

Das Problem wird aus einer schätztheoretischen Perspektive betrachtet. Dabei liegt der Schwerpunkt auf dem zentralen Algorithmus, der eine auf zwei Sensorgrößen basierende Schätzung für die Karte und die Roboterposition liefert: Die erste Sensorgröße ist die Odometrie, das heißt, die Bestimmung der Roboterbewegung über die Drehungen seiner Räder. Die zweite ist die Beobachtung von Umgebungsmerkmalen, so genannten Landmarken. Es gibt für dieses Problem eine optimale Lösung durch Maximum Likelihood bzw. quadratische Ausgleichsrechnung, die allerdings unmäßig hohe Rechenzeit, nämlich $O((n+p)^3)$ für n Landmarken und p Roboterpositionen benötigt. Gängige Ansätze, wie der Extended Kalman Filter (EKF), sind effizienter, brauchen aber immer noch $O(n^2)$ Rechenzeit und werden zudem durch Linearisierungsprobleme beeinträchtigt.

Der erste Beitrag in dieser Arbeit ist eine Diskussion von SLAM, speziell der inhärenten Struktur der Unsicherheit einer Kartenschätzung. Das Schlüsselresultat läßt sich als “Sicherheit von Beziehungen trotz Unsicherheit von Positionen” zusammenfassen. Weiterhin analysiert die Diskussion Linearisierungsfehler in SLAM und identifiziert den Fehler in der Roboterorientierung als dominante Ursache.

Im Hauptteil wird ein sehr effizienter SLAM Algorithmus erarbeitet, der die Karte hierarchisch in lokale Regionen und Unterregionen aufteilt. Auf jeder Ebene der Hierarchie speichert jede Region eine Matrix, die einige der in der Region enthaltenen Landmarken repräsentiert. Auf der untersten Ebene, das heißt der Ebene der feinsten Unterteilung, sind diese Matrizen automatisch klein, weil die Regionen klein sind und nur wenige Landmarken enthalten. Auf höheren Ebenen sind die Regionen groß und enthalten viele Landmarken. Um auch die in diesen Regionen gespeicherten Matrizen klein zu halten, werden nur die Landmarken repräsentiert, die von ausserhalb der Region beobachtbar sind. Dadurch ist sichergestellt, dass auch auf höheren Ebenen jede Matrix nur wenige Landmarken repräsentiert und Berechnungen effizient bleiben.

Eine Messung wird in eine lokale Unterregion integriert unter Verwendung von $O(k^2)$ Rechenzeit für k Landmarken in der Region. Wenn der Roboter eine neue Unterregion betritt, muss eine globale Aktualisierung mit $O(k^3 \log n)$ Rechenzeit durchgeführt werden. Weiterhin ermöglicht die vorgeschlagene Hierarchie die in einer Region gespeicherte Matrix "nichtlinear zu drehen". Damit werden Linearisierungprobleme vermieden.

Durch Simulationen und Experimente mit einem realen mobilen Roboter in einer normalen Büroumgebung erfolgt eine Auswertung des Algorthmus bezüglich der Kartenqualität, des Speicherplatzbedarfs und der Rechenzeit.

Acknowledgments

The research results presented in this thesis were conducted during my work at the Institute of Robotics and Mechatronics of the German Aerospace Center (DLR). It is my pleasure to thank Prof. Gerd Hirzinger for the opportunity to work in the exciting field of robotics, for generous support, inspiration and for the freedom to pursue the research topic I wished.

I am very grateful to Prof. Heinrich Niemann, the supervisor of this thesis, for the opportunity to discuss my research at the Chair for Pattern Recognition (LME) at the University of Erlangen-Nürnberg.

I am very much indebted to those who helped me along with advice, discussions, critical comments, and by making the robot work. Especially, I wish to thank Berthold Bäuml, Frauke Bokemeyer, Christoph Borst, Tom Duckett, Matthias Hähnle, Steffen Haidacher, Ulrich Hillenbrand, Martin Hörmann, Christian Ott, Gisela Scheffler, Norbert Sporer, and Michael Suppa.

I would like to thank my parents Ingrid and Jürgen Frese for constant support and encouragement. Finally, and most of all, I want to thank Frauke for patience and affection during the last two years.

Oberpfaffenhofen, December 2003 Udo Frese

Contents

Inhalt

List of Figures

List of Tables

List of Notations

Symbol	Reference	Definition
General		
n	§1.3, §2.14	Number of landmarks
m	§1.3, §2.14	Number of measurements
p	§1.3, §2.14	Number of robot poses
k	§1.3, §2.14	Number of landmarks local to the robot
Chapter 2		
$\widehat{\dots}$		Estimate of a random variable
$\widetilde{\dots}$		Error of the estimate of a random variable
$p_r = (p_x, p_y, p_\phi)$	(2.1), p. 50	Robot pose (x-position, y-position, orientation)
$v_r = (v_x, v_y, v_\phi)$	(2.1), p. 50	Robot velocity (x-translation, y-translation, rotation) in robot coordinates
$d_r = (d_x, d_y, d_\phi)$	(2.8), p. 51	Discrete step the robot has moved (x-translation, y-translation followed by rotation)
C_p	(2.6), p. 50	covariance of odometric position estimate $\hat{p}$
J_1, J_2	(2.11), p. 52	Jacobians of odometry (as a dynamic function)
J_3, J_4	(2.16), p. 52	Jacobians of odometry (as a measurement function)
J_5, J_6	(2.19), p. 53	Jacobians of landmark measurement
$l = (l_x, l_y)$	(2.18), p. 53	Landmark position
$m = (m_x, m_y)$	(2.18), p. 53	Landmark observation in robot coordinates
x	(2.21), p. 57	Map, i.e. vector of landmark positions (and robot pose)
f_i	(2.21), p. 57	i-th measurement function
y_i	(2.21), p. 57	i-th measurement
C_i	(2.21), p. 57	i-th measurement covariance

Symbol	Reference	Definition
$\chi^2(x)$	(2.22), p. 57	Information block, containing some information, for instance some measurements on the landmarks represented in x; minus the log-likelihood of x given these measurements; special IBs are marked by a subscript
$\hat{x}$	(2.23), p. 57	Map estimate
A, b	(2.25), p. 58	Second and first order part of a quadratic function $x^T Ax + x^T b$ used as an information block, A is a symmetric positive semidefinite matrix
J_{f_i}	(2.24), p. 58	Jacobian of i-th measurement function
C	(2.29), p. 60	Covariance (among other things part of EKF state), different covariances are distinguished by subscript
$\mathrm{ruc}(f)$	(2.96), p. 81	Relative uncertainty (compared to maximum likelihood) of aspect f of a map
Chapter 3		
$\left(\begin{smallmatrix} P & R^T \\ R & S \end{smallmatrix}\right)$	(3.4), p. 98	Decomposition of A as a 2×2 block matrix
$\left(\begin{smallmatrix} c \\ d \end{smallmatrix}\right)$	(3.4), p. 98	Decomposition of b as a 2 block vector
$\left(\begin{smallmatrix} y \\ z \end{smallmatrix}\right)$	(3.4), p. 98	Decomposition of x as a 2 block vector
I		Identity matrix of appropriate size
$\mathcal{L}(A), \mathcal{L}(b)$	§3.2	Set of landmarks represented at a matrix (A), vector (b) respectively
$\mathcal{E}$	§3.3	Set of landmarks to be eliminated
B, B'	Def. 6	Elimination matrix for A_{21}
$\left(\begin{smallmatrix} B_{11} & B_{12} & B_{13} \\ B_{21} & B_{22} & B_{23} \\ B_{31} & B_{32} & B_{33} \end{smallmatrix}\right)$	§3.7	Block decomposition of B
Rot_α	(3.40), p. 123	Rotation matrix rotating all landmarks by α
Trans_d	(3.40), p. 123	Vector translating all landmarks by d
$e := (x^0, w, e^0)$	(3.50), p. 126	Linearization point of an IB
$e(x)$	(3.52), p. 126	Linearization error at x
Chapter 4		
$\mathbf{n}$, $\mathbf{r}$, $\mathbf{s}$, $\mathbf{a}$	Fig. 4.1	Node in the tree map
$\mathbf{n}_\uparrow, \mathbf{n}_\swarrow, \mathbf{n}_\searrow$,	Fig. 4.1	Parent and children of node $\mathbf{n}$
$\mathbf{n}_{\downarrow\mathbf{r}}, \mathbf{n}_{\downarrow\overline{\mathbf{r}}}$	Fig. 4.1	Child of $\mathbf{n}$ that is or is not respectively ancestor of $\mathbf{r}$
$\mathbf{n}_{\mathrm{CIB}}$, $\mathbf{n}_{\mathrm{SIB}}$, $\mathbf{n}_{\mathrm{BIB}}$, $\mathbf{n}_{size}$, $\mathbf{n}_\alpha$, $\mathbf{n}_{\hat{x}}$	page 133	Components stored at node $\mathbf{n}$: CIB, SIB, BIB, size, rotation angle α, estimate $\hat{x}$
$\mathcal{L}(\mathbf{n})$	page 133	Landmarks represented at a node $\mathbf{n}$ in the tree map

Symbol	Reference	Definition
root	page 133	Root of the tree
l	page 133	Landmark
eN $[l]$	page 133	Elimination node of landmark l
$\mathcal{A}, \mathcal{B}, \mathcal{C}, \ldots$	page 133	Set of landmarks
$optHTPSteps$	Fig. 4.2	Number of tree map optimization steps performed each time after changing the actual BIB
$maxDistance$	§4.2	Maximal distance of landmarks in the same BIB
$maxAngle$	Fig. 4.9	Angle below which to stop iteration in nonlinear mode
$\mathrm{par}(\mathbf{n})$	(4.2), p. 145	Partitioning of node **n**
$\mathbf{n}_{size}$	(4.3), p. 145	Number of BIBs below node **n**
$\mathrm{bal}(\mathbf{n})$	(4.3), p. 145	Balancing of node **n**
$\mathrm{Bal}(\mathbf{n})$	(4.3), p. 145	Balancing constraint for node **n**: $\mathrm{bal}(\mathbf{n}) \in \mathrm{Bal}(\mathbf{n})$
lca	§4.5	Least common ancestor of two nodes
Acronyms		
ML	§2.6	Maximum Likelihood
LLS	§2.7	Linearized Least Square
EKF	§2.8	Extended Kalman Filter
SLAM	§1.1	Simultaneous Localization and Mapping
IB	§3.2	Information Block
BIB	§3.2	Basic Information Block
CIB	§3.2	Condensed Information Block
SIB	§3.2	Substitution Information Block
SPD	§3.2, App. A	Symmetric positive definite ($x^T A x > 0 \; \forall x \neq 0$)
SPSD	§3.2, App. A	Symmetric positive semi definite ($x^T A x \geq 0 \; \forall x$)
HTP	§4.4	Hierarchical tree partitioning

Chapter 1

Introduction

Throughout the history of robotics up to now there have been two driving forces: Autonomy and applications. The quest for autonomy is motivated by the old dream to build an artefact acting independently like a human being. A multitude of different robots and of different approaches for controlling them has been devised. Accordingly a lot of scientific progress has been achieved, but in general, robot autonomy is still more a long term vision than reality. Applications, on the other hand, are mostly motivated by the idea to create something useful on a short-term perspective. There are many commercially successful robot applications, especially in automotive industry. However, most of these robots perform preprogrammed, repetitive movements with little or no autonomy.

In recent years, there has been increasing commercial interest in new applications other than industrial production. In the emerging field of service robotics robots operate in a human environment and perform tasks like cleaning, fetching or carrying objects, surveillance, clearing, and general assistance to handicapped persons. A service robot can move freely, neither mounted at a specific place nor confined to a separated area. So the robot must constantly adapt its behavior to the environment as perceived by its sensors. Thus, any successful application in service robotics requires considerable autonomy for the robot.

In a typical scenario a service robot operates indoors like in an office, hospital or home environment. Basically, it is a small computer-controlled vehicle about the size of a wheelchair and may be equipped with a robot arm to manipulate objects. Such a robot is often called a mobile robot. By measuring the revolution of its wheels a mobile robot is able to observe its own movement. Typically an ultrasonic sensor or laser scanner is used to detect walls and obstacles. Both measure a distance based on the time an ultrasonic or light pulse respectively needs for traveling to the obstacle and back. For scanning distances along different directions, either several sensors are being used or the sensor is fastened at a rotating motor. Such a scan

basically provides a horizontal slice of the environment. Other widely used sensors are electronic cameras. Presumably, these are the most versatile sensors because they can be used to perceive almost everything relevant to a service robot including objects, places, people, and obstacles. On the other hand it is often very difficult to extract this information from camera images.

For a mobile robot the main challenge is navigation, i.e. moving from place to place. Following the founders of the research field J.J. Leonard and H.F. Durrant-Whyte, this task involves answering the questions "Where am I?", "Where am I going?", and "How do I get there?" [LDW92]. None of these questions can be answered without establishing a map before because to name a place or path some representation of the physical environment is needed. So the first question is "What is my map?". This is the topic of this thesis.

When operating in an indoor environment usually a floor plan is available and appears to be usable as a map. Of course, this remains unsatisfactory if autonomy is the goal. If a map has to be provided manually, a robot is not autonomous any more. But even from a practical perspective this approach is much more difficult than it seems at first sight. In fact, providing a precise map of a large building with sufficient details included to be useful is an expensive and inconvenient task. For instance, Thrun et al. [TBF98] report that building a map manually for a robotic museum-tour guide took one week, whereas building a map autonomously took about one hour.

So the most convincing approach, both from the perspective of autonomy and of applications, is to have the robot create its own map while moving through the building.

1.1 Simultaneous Localization and Mapping

SLAM[1], i.e. the Simultaneous Localization and Mapping problem, aims at a fully autonomous answer to the question "Where am I?" by providing an autonomously built map. While moving through an environment the robot is demanded to derive a map from its perceptions and simultaneously determine its own position in this map. From the late eighties this problem has been explored. First contributions were made by Chatila [CL85], Smith, Self and Cheeseman [SSC88] and Durrant-Whyte [DW88]. The name *simultaneous localization and mapping* was later introduced by Durrant-Whyte [DWRN95].

Figure 1.1 illustrates the basic idea. Even though the studies conducted in this thesis are generally valid for a wide range of scenarios, let us consider the setting of the experiments to be reported later in this thesis: A mobile robot moves through an office building. It is driven by wheels and observes its own movements by integrating the wheels' revolution (odometry).

[1] Also called Concurrent Mapping and Localization (CML) by some other authors

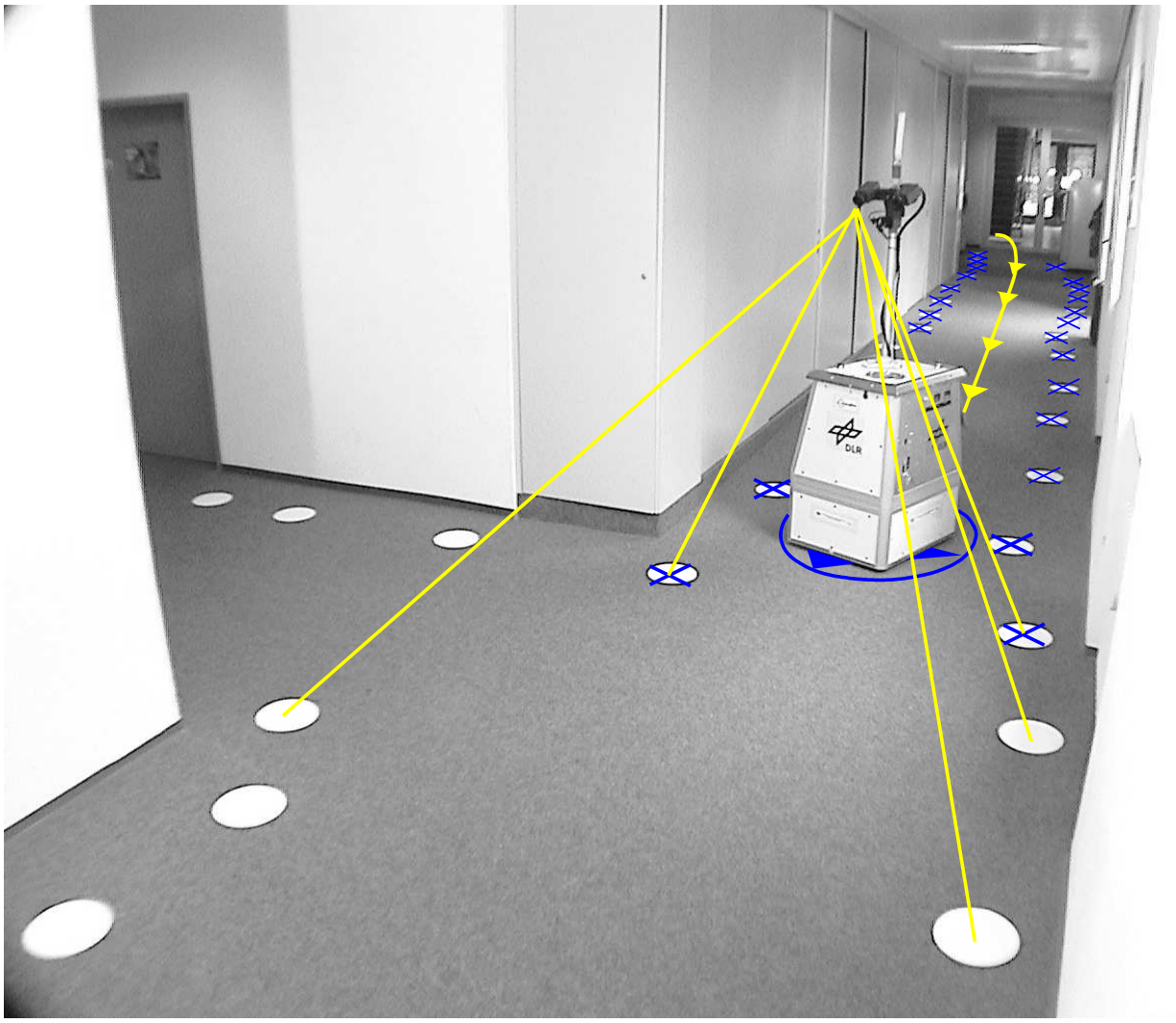


Figure 1.1: SLAM: A mobile robot moves through an unknown building, observing its own movement (yellow/light curve with arrows) and landmarks (yellow/light lines). From these data a map of the building is being computed as seen so far (blue/dark crosses). The robot's own position (blue/dark circle with triangle) in the map is being determined. In the experiments for this thesis white circular discs are used as artificial landmarks. In a real application they would be replaced by natural landmarks like corners, walls, doors, etc.

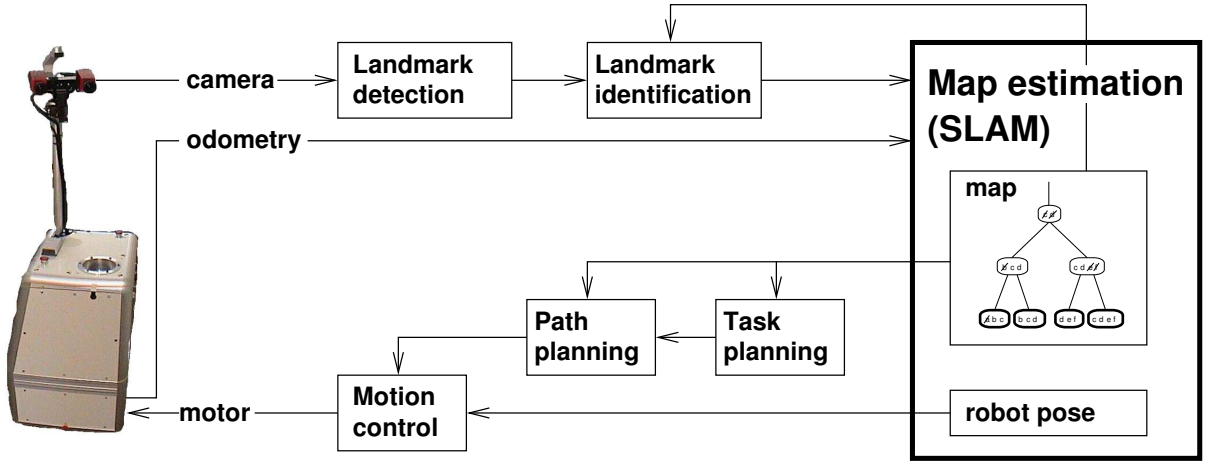


Figure 1.2: SLAM based navigation system: Landmarks are detected and identified by comparing them with the landmarks already represented in the map. The identified landmark observations and odometry measurements are passed to the map estimation algorithm estimating robot pose and map. Task planning, path planning and motion control use these information to drive the robot. The map estimation algorithm is the key component of such a system. The proposed algorithm is based on hierarchical decomposition of the map represented by a tree.

An electronic camera is used to observe distinguished features in the environment, so called landmarks. By processing a camera image landmarks are detected determining their positions in relation to the robot. When observed a second time a landmark is recognized as being the same. This process is called identification and performed by comparing the relative position of the observed landmarks with the landmarks in the map created so far. Based upon these data the robot computes a map of the landmarks in the building and determines its own position in this map. In a landmark-based approach, a map essentially is a set of positions of landmarks.

This map can be used, in turn, for task and path planning, i.e. to provide an answer to the other two questions "Where am I going?" and "How do I get there?". Therefore the map must be extended to represent free space and paths. This may easily be achieved by a graph of waypoints: While moving through a previously unknown area, every once in a while the robot's position is being added as a waypoint and being linked with adjacent waypoints in the graph. While navigating later the robot heads from waypoint to waypoint. Like the landmarks the waypoints are mathematically treated in the same way, even though they will not be observed again. Instead, the waypoints' position relative to nearby landmarks is known from the map. So when the robot observes these landmarks, it implicitly knows the waypoints' position, too.

All these operations are performed continously while the robot is moving such that the robot has a map of the explored environment available at any time. Figure 1.2 shows the flow of data in the SLAM system.

If odometry and landmark observations were exact, SLAM would be a simple problem to solve: The robot pose would be directly obtained from odometry and the landmark positions could be straightforwardly computed by composing the known robot pose with the observations from the landmark sensor.

The difficulty in SLAM arises from the fact that measurements are never exact but always subject to measurement noise. Thus, the map must be estimated from uncertain data, consequently it is uncertain too. To give an example: for the robot used in the experiments for this thesis, the landmark sensor has a measurement uncertainty of ≈ 0.05m and odometry of ≈ 1.6m after 20m of traveling. In robotics it is very common to be faced with uncertain measurements. These uncertainties, especially the uncertainty of the robot pose derived by odometry, are made particularly difficult by SLAM because they accumulate while the robot is moving. Thus, uncertainties can reach arbitrarily high values. This challenge has been verbalized in the foreword of the well known textbook "Autonomous Robot Vehicles" [CW90]:

> *"The key scientific and technological issue in robotics is that of coping with uncertainty [...] Mobile robots operating in large, unstructured domains must be able to cope with significant uncertainty in the position and identity of objects. In fact, the uncertainty is such that one of the most challenging activities for a mobile robot is simply going from A to B."*
>
> Tomás Lozano-Pérez

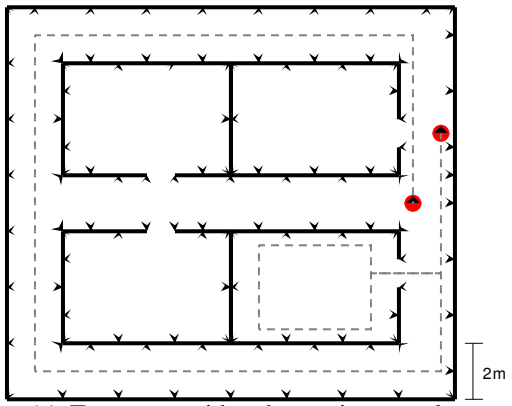
It is important to realize that it is not possible to treat the measurements as if they were exact and simply accept the resulting error in the map. Instead, the measurements must be evaluated statistically by combining them in a way that takes their uncertainty into account. Figure 1.3 shows, what remarkable differences this makes. The goal certainly is to achieve a map that is as precise as possible despite uncertain measurements. This task is much more difficult than computing an exact map from exact measurements. It is performed by the map estimation algorithm estimating the map and robot pose by statistically evaluating odometry and identified landmark observations. It is the core part of any SLAM system ("*the SLAM algorithm*").

When considering a complete SLAM system (Fig. 1.2), most of its parts heavily depend on the specific environment, robot, and sensor used. In contrast, the map estimation algorithm works very much the same regardless of these factors. Indeed, there are SLAM implementations in different environments including indoor, outdoor, underwater, and airborne as well as different robots and sensors including ultrasonic, laser scanner, and camera. A brief overview will be given in §1.3. All these implementations basically solve the same estimation problem by their core algorithm. Therefore, the core estimation algorithm is the main focus of most work on SLAM, including this thesis.

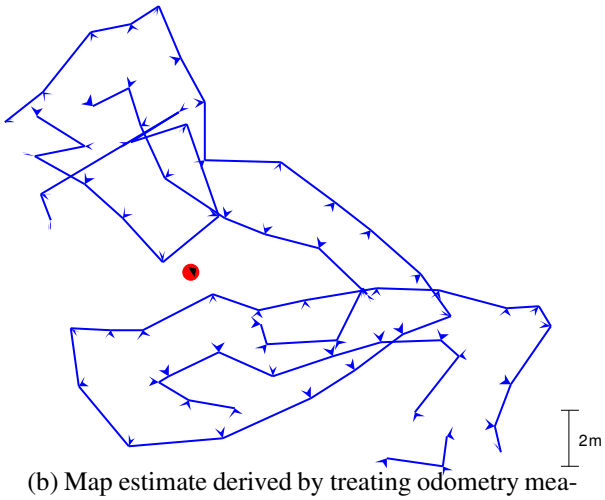
Basically, there is an optimal solution for SLAM under reasonable assumptions: The measurement noise is assumed to be independently Gaussian distributed with known covariance. This allows to evaluate the likelihood of a particular map given the observations made by the robot so far. The map with the largest likelihood, the *Maximum Likelihood Estimate* (ML-estimate) is the best estimate possible [PTVF92, §15.1][2].

Thanks to the Gaussian error distribution, the optimal map can be computed by least squares estimation by solving a large linear equation system. A standard least squares estimation algorithm like Levenberg-Marquardt [PTVF92, §15.5] can perform this computation. This approach is well established and has been theoretically and experimentally verified [SSC88, LM97, DNDW+99]. Historically, the use of least squares estimation for the pur-

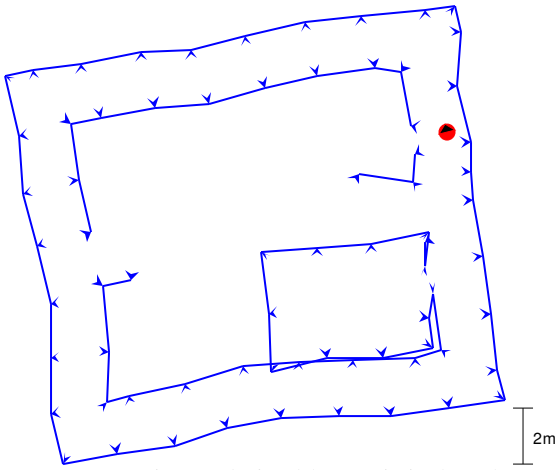
[2]Best possible means that it has a maximum probability of causing the observed measurements. Furthermore, it is the map with the highest conditional probability given the measurements, if the true map is assumed to be drawn from a uniform a-priori distribution.



(a) True map with robot trajectory along the outer corridor. Landmarks are depicted by "ϒ" symbols. The robot is shown at start and end of the trajectory.



(b) Map estimate derived by treating odometry measurements as if they were exact.



(c) Map estimate derived by statistical evaluation.

Figure 1.3: Statistical evaluation of measurements: The map estimate based on treating odometry as if it was exact is nearly useless. The map estimate based on a statistical evaluation still has errors, especially in the overall map orientation but is much better. All maps shown in §1 – §5 are based on simulations with comparatively large artificial measurement noise.

pose of building maps dates back to the invention of this method by C.F. Gauss himself ("Theoria combinationis observationum erroribus minimis obnoxiae", Göttingen, 1821 [Gau21]).

When performing SLAM a growing map must be provided while the robot is moving. This requires computing an up to date estimate after each measurement. Each time a large system of equations must be solved. For n landmarks and p robot poses (places from which the robot observed landmarks) the computational cost of the Levenberg-Marquardt algorithm is $O((n + p)^3)$. To give a figure, a medium-sized office building[3] mapped in the experiments reported in this thesis covers an area of $60\text{m} \times 45\text{m}$ and contains 29 rooms, $n = 725$ landmarks, $p = 3297$ robot poses, and $m = 29142$ measurements. It can be seen that the Levenberg-Marquardt algorithm is far too slow. Many researchers, the author of this thesis included, have been motivated by this challenge and devised more efficient algorithms for performing the map estimation. Most of these approaches compute an approximation to the ML estimate to improve efficiency. In this case, apart from computation time, the quality of the provided estimate needs to be considered. In general three criteria are important to assess the performance of a SLAM algorithm:

(I) Quality of the map estimate. Specifically, the uncertainty of the map estimate in comparison to the corresponding uncertainty of the optimal maximum likelihood estimate

(II) Storage space used by the map

(III) Computation time for updating the estimate after integrating a measurement

The goal of the thesis is to follow these criteria in two steps: First, the requirements for an algorithm to be postulated from a theoretical perspective will be analyzed and second, an algorithm fulfilling these requirements as far as possible will be devised.

1.2 Structure of SLAM Uncertainty

SLAM is different from many other estimation problems in that there is a very specific structure in the uncertainty of the estimate. This uncertainty and, hence, its structure is not caused by a particular SLAM algorithm but is inherent to the general type of sensors used[4]. The landmark sensor measures the landmark position relative to the robot and odometry measures the robot movement relative to itself. Both cannot observe absolute position or orientation like a compass or the Global Positioning System (GPS), but measure some local relation to the robot. Figure 1.4

[3] Part of the DLR, Institute of Robotics and Mechatronics' building, Oberpfaffenhofen, Germany

[4] This may be observed in the uncertainty of the optimal maximum likelihood estimate. Since any other estimator yields a larger uncertainty, this uncertainty is inherent to the problem.

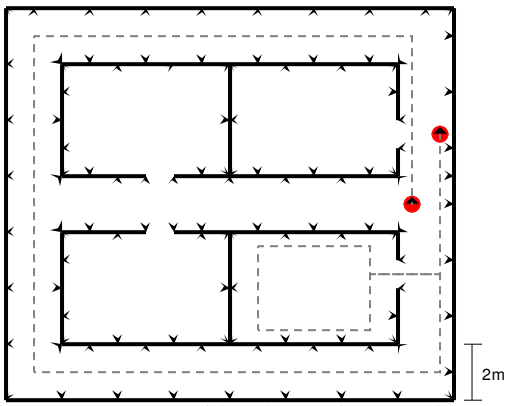
shows the consequences. For instance, the pose of a room at the end of a long corridor relative to its other end is derived from a long chain of local measurements and is, thus, quite uncertain. For a longer chain the uncertainty will be larger, so the error is accumulating and becoming larger and larger while the map grows. But although the room's pose can become arbitrarily uncertain, only its shape is rather precisely known. The reason is that the perceived shape of the room is only affected by the measurement error occurring while the room is mapped, not by the entire accumulated error. This structure is very important both for understanding SLAM and for devising an appropriate algorithm. To say it in a nutshell:

Certainty of Relations despite Uncertainty of Positions.

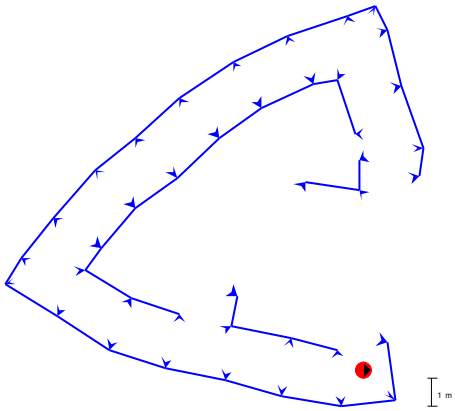
One should realize however, that the whole notion of an absolute position although technically helpful is somewhat artificial. All measurements are invariant under translation and rotation of the whole environment including the robot. Thus, to have well defined absolute positions, the first robot pose must be arbitrarily set as the origin and orientation of a global coordinate system. So actually, absolute position means position relative to the first robot pose. In general, distant relations are more uncertain than local relations. In particular, absolute positions corresponding to relations to the first robot pose, are the most uncertain relations in general.

There is a remarkable theorem by Newman [New99, DNC$^+$01] further highlighting this structure. When considering the situation while a robot is moving through the same area over and over again the map will get more and more precise. The limit of moving through the same area infinitely often will make all relations between landmarks exact. The only remaining uncertainty is the relation between the landmarks and the first robot pose, i.e. the overall pose of the map in absolute coordinates. This limiting situation is the extreme case of the general uncertainty structure: All relations are exact, i.e. perfectly certain, whereas all absolute landmark positions are still uncertain because the overall pose of the map in absolute coordinates is uncertain.

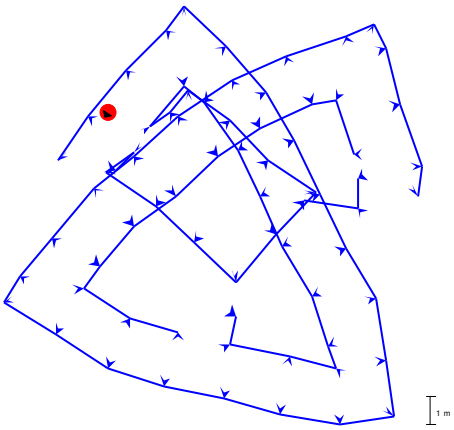
The implications of this structure are most prominent in the case of a long circular corridor that forms a closed loop (Fig. 1.4a). At the end of the loop the robot returns to its start. This situation becomes apparent to the robot when it observes and identifies an already known landmark. This measurement actually closes the loop, by providing the information to the SLAM algorithm that the robot has returned to the start of the loop. Due to accumulated error there will be a large gap in the map estimate between start and end (Fig. 1.4c). This is strongly inconsistent with the measurement closing the loop. When integrating this measurement the map estimate must change significantly. While the robot has moved along the loop it has observed adjacent landmarks from the same robot pose or two nearby poses. Thus the relative location of landmarks is precisely known and the loop cannot be closed by introducing a gap somewhere else in the loop,



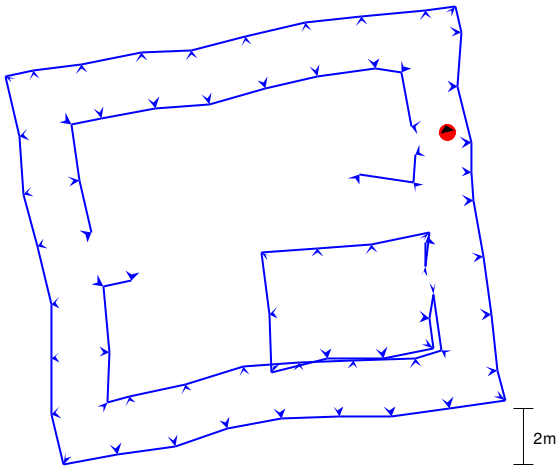
(a) True map with robot trajectory along the outer corridor.



(b) Errors accumulate while the robot is moving.



(c) The room's pose is uncertain while its shape is certain.



(d) Closing the loop requires deforming the whole map to back-propagate the error.

Figure 1.4: Structure of SLAM uncertainty.

because this would be inconsistent with the measurements made there. Hence, to make the map consistent with all measurements, it must be smoothly deformed matching start and end of the loop (Fig. 1.4d).

This process is sometimes called "back-propagating the error along the loop". When performing maximum likelihood estimation all this happens automatically since the estimator is derived from a consistent statistical model. Any algorithm unable of back-propagating violates criterion (I) because it provides an estimate being highly inconsistent with one of the measurements. Thus, for any SLAM algorithm, especially for one computing an approximate estimate, closing the loop is the hardest test case: To perform the back-propagation, the underlying data structure of the map must represent information on the uncertainty of the landmarks, specifically that the relative position of adjacent landmarks along the loop is precisely known despite of the uncertainty of their absolute position.

Many algorithms are based on an Extended Kalman Filter (EKF) [Gel74, SSC88] and represent uncertainty by a covariance matrix. For these approaches the critical information is represented in the strong correlation between nearby landmarks and weak correlation between distant landmarks. Historically, it is interesting to note that after the use of a covariance-based representation, first proposed by Smith et al. [SSC88], it took several years until the "importance of correlations" was fully realized [HBBC95, CTS97]. For example, a formerly popular representation was to store just a 2×2 covariance matrix for each individual landmark position instead of a large $2n \times 2n$ matrix describing correlations between each pair of landmarks. In the former data structure the information on tight coupling of adjacent landmarks despite the uncertainty in their absolute positions was left out. So when closing the loop the estimates for the landmarks along the loop could not be appropriately adjusted and a gap remained in the loop.

To summarize, closing the loop is the hardest test case for any SLAM algorithm: Since the estimate changes significantly after integrating a single measurement, errors introduced by the algorithm are far more obvious than in maps without loop. For this reason, the experiments presented in §5 and §6 all contain large loops.

Another important structural issue in SLAM is nonlinearity: The equations involved are nonlinear, mainly in robot orientation ϕ, which occurs as a $\left(\begin{smallmatrix}\cos\phi & -\sin\phi \\ \sin\phi & \cos\phi\end{smallmatrix}\right)$ rotation matrix. Nonlinearity is commonly treated by linearization. Severe linearization errors result, if the error in robot orientation exceeds $\approx 15°$. Some algorithms can solve this problem by relinearizing, i.e. updating the linearization point and recomputing all Jacobians once the estimate changes. For all EKF based approaches this is impossible, so they are limited to scenarios with small orientation error. The data structure used in this thesis allows to apply "nonlinear rotations" to parts of the map. Any error in the robot orientation having affected linearization can be efficiently corrected

without recomputing all Jacobians.

1.3 State of the Art Overview

In this section a brief overview over the current state of the art is given. The different SLAM algorithms established in literature are compiled and their properties regarding the criteria mentioned above are specified. A more detailed analysis is deferred to §2.14 after discussion of the SLAM problem.

Algorithms

The evolution of SLAM algorithms can be broadly divided into three phases:

In the first phase from the mid-eighties to the early nineties, the mathematical formulation of SLAM was still an open question and the special uncertainty structure discussed above was not yet fully recognized.

First approaches to build a map were based on so called *evidence grids* introduced by Moravec and Elfes [ME85, Elf89] in 1985. They divide the map into a regular grid with square cells of fixed size (typically $\approx$ 5cm). Each cell stores a real number $[0 \dots 1]$ representing the evidence of this cell containing an obstacle. The evidence is accumulated from different measurements involving this cell. Evidence grids are well suited to integrate the noisy low resolution information provided by ultrasonic sensors. However, they cannot represent robot pose uncertainty and thus are unable to perform SLAM.

Other authors followed a feature based approach as proposed by Brooks [Bro85]. They represent the map as composed from distinguished objects instead of using an unstructured representation as evidence grids. In order to represent uncertainty they maintain a graph of uncertain relations between the objects [CL85, CS86, DW88, Fau89]. These approaches can incorporate uncertainty in the robot pose and led to an estimation theoretic formulation of SLAM.

The second phase of SLAM development was initiated by the influential paper of Smith, Self and Cheeseman [SSC88] who first formulated SLAM mathematically thoroughly as an estimation problem. They realized that landmark estimates are highly correlated because of the accumulated error in the robot pose and proposed to represent all landmark positions and the robot pose in a common state vector in combination with a full covariance matrix of them. This representation is called *stochastic map* and basically is an Extended Kalman Filter (EKF) [Gel74]. It has been widely used and extended by several authors [Tar92, CTS97, CMNT99, HBBC95, New99].

However, the main problem of large computation time remained. The most time consuming part of the computation is to update the covariance matrix, taking $O(n^2)$ time for n landmarks. This limited the use of SLAM to small environments ($n \lessapprox 100$ landmarks).

Recently, interest in SLAM has increased drastically and several, more efficient algorithms have been developed. In contrast to the EKF based approaches, most of these algorithms are efficient enough to be used in medium sized environments ($n \approx 500$ landmarks). Some very fast approaches can even be used for large environments ($n \gtrapprox 10000$ landmarks), but for these algorithms there are some limitations regarding the quality of the estimated map in certain situations.

Most approaches exploit that the field of view of the involved 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 and the density of landmarks but does not grow when the map gets larger. So it is small, practically $k \approx 10$ and theoretically mostly considered constant $k = O(1)$.

Guivant and Nebot [GN01, GN02] 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. [DMS00, DMS02] employ 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. This problem was recently solved by Frese and Duckett [FD03] by a method called *Multilevel Relaxation*. They employ a multilevel approach similar to the multi grid methods used in the numerical solution of partial differential equations. So computation time could be reduced to $O(kn)$ even when closing large loops.

Montemerlo et al. [MTKW02] 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 [DdFG01] in which every particle represents a sampled robot trajectory and associated Gaussian distributions of the different landmark positions. These distributions are independent, so n small covariance matrices instead of one large matrix is needed. The computation time for integrating a measurement is $O(M \log n)$ for M particles with $O(Mn)$ storage space needed. This algorithm can cope with uncertain landmark identification – which

is a unique advantage. The efficiency crucially depends on M being not too large. On the other hand a large number of particles is needed to close a loop: 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 at least one particle of the set must already close that loop by chance, and either many particles are needed or there will be gaps in a loop closed already.

Thrun et al. [TKGDW02] presented a constant time algorithm called Sparse Extended Information Filter (SEIF). They followed a similar idea, also proposed by Frese and Hirzinger [FH01] and use a so called information matrix instead of a covariance matrix to represent uncertainty. The algorithm exploits that the information matrix is approximately sparse requiring $O(kn)$ storage space. A proof for sparsity also being central to the algorithm presented in this thesis, will be given in §2.11. The information matrix representation allows to integrate a measurement in $O(k^2)$ computation time, but to give an estimate a system of n linear equations must be solved. Similar to the approach of Duckett et al. this could be done by applying one iteration of relaxation with $O(kn)$ computation time. Thrun et al. propose not to relax all n landmarks, but only $O(k)$, thereby formally obtaining an $O(k^2)$ algorithm. However, it is not clear, whether this will suffice in general because in numerical literature relaxation is reputed needing $O(n)$ iterations with $O(kn^2)$ computation time for solving an equation [Bri99, PTVF92, §19.5].

To the author's knowledge the four algorithms CEKF, relaxation, fastSLAM, and SEIF discussed above are the most efficient algorithms available today.

Systems

SLAM, when reduced to the core map estimation algorithm, is a well formalized and truly general problem. This can be concluded from the wide diversity of SLAM implementations involving different environments, robots, sensors and landmarks that have been successfully deployed in the last years. In all SLAM systems discussed below least squares and maximum likelihood estimation are applied as core algorithm.

The most common type of application are wheeled robots in office environments: Castellanos and Tardos [CTS97] use a laser scanner to detect walls and edges of walls. Durrant-Whyte [DWRN95] detects edges from ultrasonic data. Neira and Tardos [NT01] as well as Castellanos [CT00] detect edges as vertical lines using a camera and computer vision. Duckett et al. [DMS00] as well as Gutmann and Konolige [GK99] do not extract explicit landmarks but use raw laser scans instead, deriving the spatial relation between different scans by a scan-matching

algorithm. Duckett and Nehmzow [DN00] recognize places by classifying ultrasonic sensor response patterns. Gutmann et al. [GFS03] use color landmarks extracted from images by a camera fastened at a quadruped and a humanoid robot.

Outdoor applications usually use quite specific approaches that depend on the environment and terrain under consideration: Guivant and Nebot [GN02] use a car equipped with a laser scanner as a robot and move through a park detecting trees as landmarks. Folkesson and Christensen [FC03] autonomously map a military urban warfare training facility with an outdoor mobile robot using walls as landmarks. Montemerlo et al. [MTKW02] conducted a conceptual study for an autonomous robot to be deployed on Mars, detecting rocks on a basically planar surface with a laser scanner. Feder and Leonard [Fed99] as well as Williams et al. [WDDW02] use an autonomous underwater vehicle (AUV) and an underwater sonar scanner to detect buoys as landmarks. Kin and Sukkarieh [KS03] implemented SLAM on an unmanned airplane using white plastic sheets as landmarks detected by a camera.

Thrun et al. realized impressive applications with algorithms based on Expectation Maximization (EM): A robotic museum tour guide [BCF$^+$99, TBF98], deployed in the "Deutsches Museum zu Bonn" and "Carnegie Museum of Natural History", conducted several thousand tours. An outdoor wheeled mobile robot equipped with laser scanner autonomously explored a part of an abandoned mine [THF$^+$03] and a small helicopter performed autonomous 3D terrain mapping [TDH03].

1.4 Thesis Contribution

Analysis

The first contribution of this thesis is a thorough analysis of the SLAM problem (§2) as such. The structure of the inherent uncertainty of an estimated map is characterized as briefly sketched in §1.2. Formally, the central topic is the information matrix of all landmarks. Its structure corresponds to the uncertainty structure of an estimated map. The algorithm presented in this thesis and the SEIF algorithm proposed by Thrun et al. [TKGDW02] approximate this matrix by a sparse matrix. In the discussion a formal proof is given that the information matrix of all landmarks is indeed approximately sparse, i.e. most entries are very small.

Furthermore, the linearization error incurring in SLAM will be analyzed by discussing its sources and structure. The duality between information and covariance based representations will be explained and related to the three "textbook" methods for solving SLAM, i.e. maximum likelihood, least squares and EKF.

Finally, a set of three requirements which an ideal SLAM algorithm should satisfy will be proposed and established. These requirements concern map quality, storage space and computation time. They were proposed prior to the development of the algorithm [FH01]. The discussion includes a mathematical formalization of the term "Quality of the Map Estimate" introduced in §1.2.

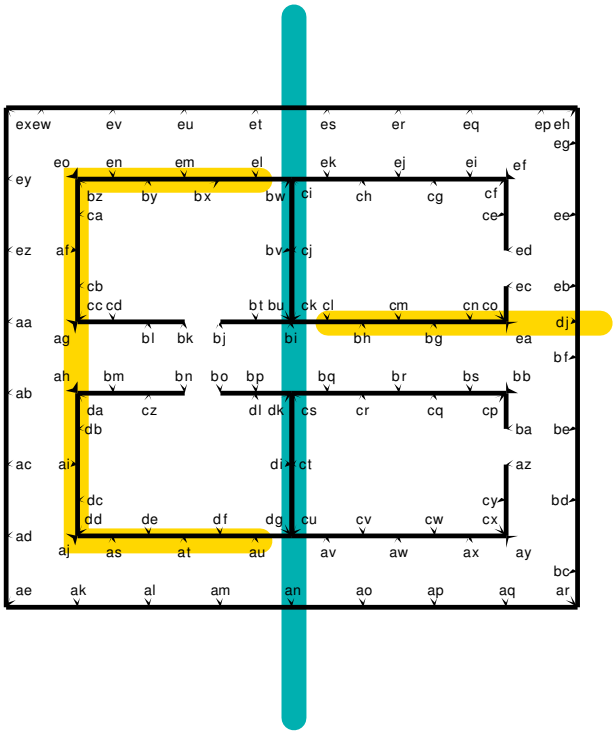
Algorithm

The second and main contribution of this thesis is a novel, highly efficient SLAM algorithm (§3, §4). It works by dividing the map hierarchically into regions of k landmarks. Integration of a measurement takes $O(k^2)$ time within a region and $O(k^3 \log n)$ when the robot changes to a different region. This is less than linear in the number of landmarks and much more efficient than all SLAM algorithms currently known. The basic outline is as follows:

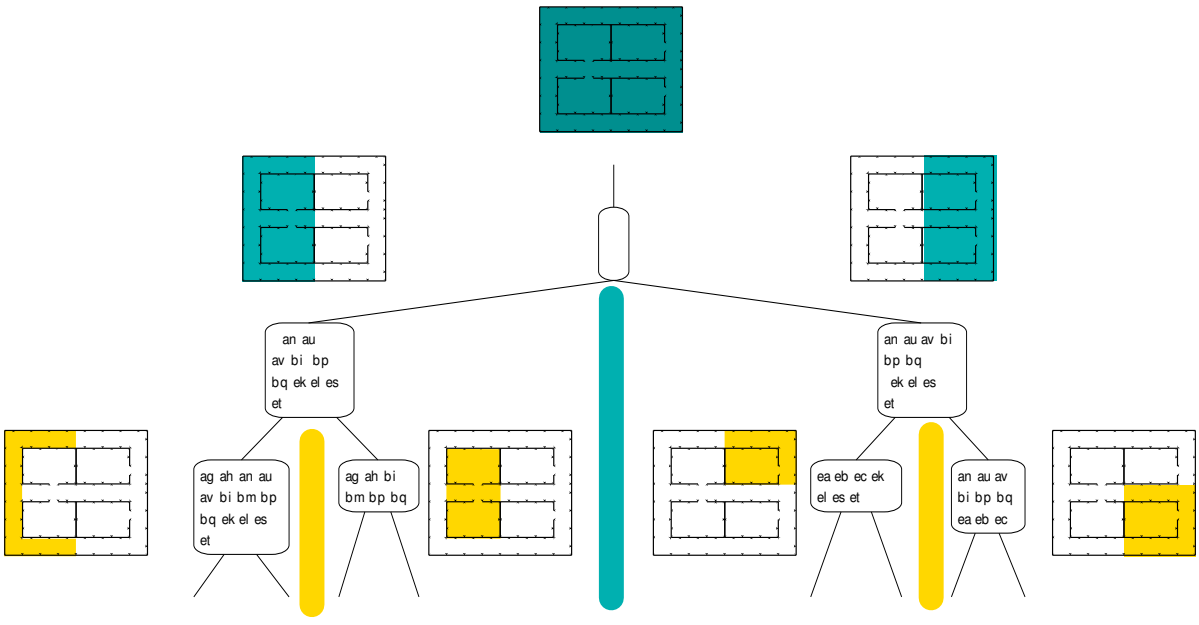
The algorithm uses so called information matrices as uncertainty representation. Instead of using a large matrix representing all landmarks, it decomposes the matrix into many small matrices each representing only a few landmarks. The decomposition is designed in such a way that only a few small matrices need to be updated in order to integrate a new measurement, which can be done extremely efficiently.

To derive the decomposition, the map is dynamically divided into a hierarchy of regions and subregions. Each region at each level of the hierarchy stores an information matrix representing some of the landmarks contained in this region. On the level of finest subdivision these matrices are naturally small because the corresponding regions are small and contain only few landmarks. On higher levels regions are large and contain many landmarks. In order to keep the information matrices stored at these regions small, only those landmarks are represented that may also be observed from outside the region. For a large region most landmarks lie not close to the region's border and only few can be observed from the outside. This is particularly applicable and accurate for indoor environments. Often the border of a region is a door or corridor and, thus, very small. This way it is ensured that even on high levels of the hierarchy each matrix represents only few landmarks and computation is efficient.

A new measurement is integrated into the region at the robot's current location. The estimate for the landmarks in this region is updated using the EKF equations in $O(k^2)$ time. When the robot enters a new region a global update is necessary. The key point for the overall efficiency is that only the region containing the robot and the region in the hierarchy above need to be updated. All other regions remain untouched. Thus, a global update takes only $O(k^3 \log n)$ computation time. The update integrates the whole information provided by the measurement



(a)



(b)

Figure 1.5: 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.

including effects on distant landmarks, for instance when closing a loop. However, it does not update the complete map estimate because this would need $O(kn)$ computation time. Instead, the algorithm exploits that only an estimate for the landmarks of the local region needs to be computed. This is the only way to achieve an update time sublinear in n because changing the estimate for all landmarks takes $O(n)$. Nevertheless updating the whole map is still feasible in practice, because the prefactor involved in $O(kn)$ is extremely small as will be shown in the experiments.

Figure 1.5 shows an example for such a hierarchy and its representation using a tree.

A further advantage of hierarchical decomposition is the solution of linearization problems caused by nonlinearities in the robot's orientation. Hierarchical decomposition allows to change the linearization point that has been used to integrate measurements in a specific region by applying a "nonlinear rotation" to the information matrix stored at this region. That is why the algorithm computes a solution very close to maximum likelihood even when facing large orientation errors.

1.5 Thesis Overview

This thesis is organized as follows:

§2 Chapter 2 discusses the uncertainty structure of map estimates

§3 Chapter 3 presents the linear algebra part of the proposed algorithm which manipulates, decomposes and integrates small parts of information using information matrices

§4 Chapter 4 describes the bookkeeping part of the algorithm building and maintaining the hierarchy

§5 Chapter 5 evaluates the algorithm with respect to map quality, storage space and computation time with simulation experiments

§6 Chapter 6 presents experiments with a real robot using computer vision based landmark detection

§7 Finally chapter 7 summarizes the thesis' main contributions and suggests potential direction for future research

Chapter 2

Uncertainty Structure of Map Estimates

This chapter provides a brief discussion of the structure of the SLAM problem. The analysis is not strictly formal but based both on thought experiments and mathematical derivation. It provides the insights that have been used in devising the algorithm presented in this thesis:

Simultaneous Localization and Mapping (SLAM) is the problem of estimating a map and the robot's pose within this map from landmark and odometry observations. As the measurements are subject to errors, so is the estimated map. The nature of these measurements is that they are relative to the robot and unable to notice the absolute position of the robot and the environment. Consequently, regardless of the estimator used, relative aspects of the map estimate will be comparatively certain, whereas absolute aspects will be comparatively uncertain.

The first section (§2.1) formally introduces the measurement equations used in this thesis.

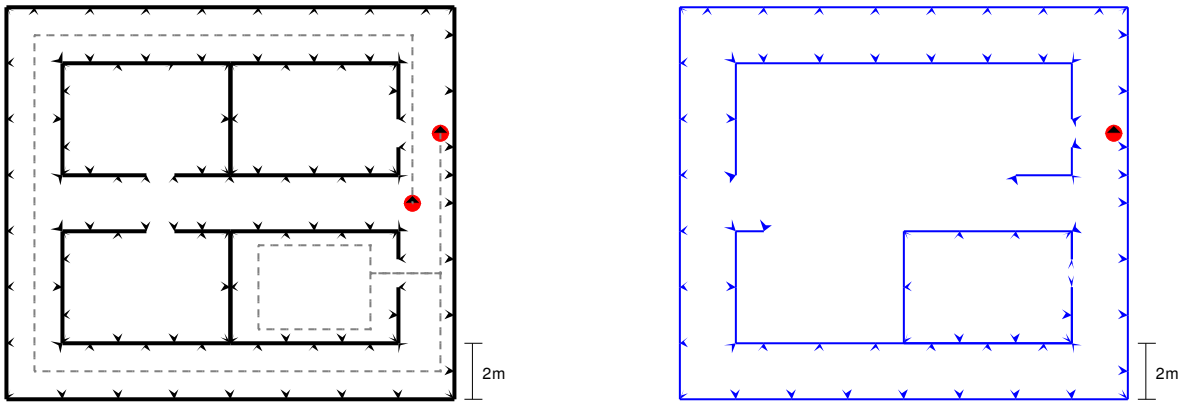
The main part of this chapter (§2.2 . . . §2.5 and §2.10. . . §2.12) discusses the structure of this inherent uncertainty. The analysis is substantiated by a proof of the approximate sparsity of SLAM information matrices:

> *The off-diagonal entries corresponding to two landmarks decay exponentially with the distance traveled between observation of the first and second landmark.*

This result has strong implications both for the algorithm presented in this thesis and for understanding the structure of SLAM uncertainty. The key result can be summarized as

> *Certainty of Relations despite Uncertainty of Positions.*

The second part of the chapter (§2.6 . . . §2.9) compares three common estimation techniques: *Maximum Likelihood* (ML), *Linear Least Squares* (LLS) and *Extended Kalman Filter* (EKF). Under the assumption of Gaussian measurement noise all three are related and for linear measurement equations they are even identical. SLAM is a nonlinear problem, so the latter two suffer



(a) (b)

Figure 2.1: Building used as an example throughout the thesis (a) "true" building and robot trajectory (b) exact map without measurement noise.

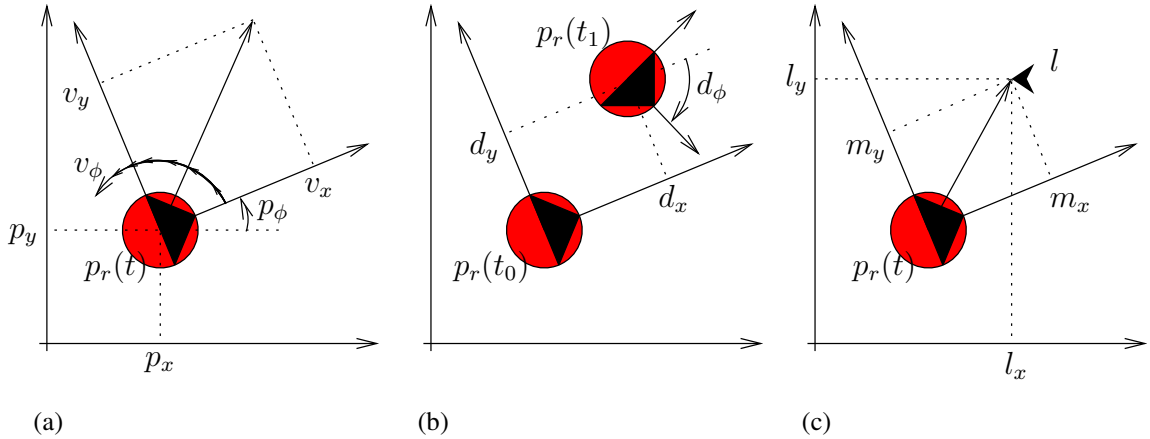
from linearization error. In §2.9 the structure of the linearization error is analyzed identifying the *orientation nonlinearity* as primary source of error.

Section §2.13 takes an intuitive perspective on the problem and proposes three requirements an ideal SLAM solution should fulfill. Section §2.14 discusses the current state of the art under this perspective. Finally the last section (§2.15) summarizes the analysis of the uncertainty structure and sketches connections to the algorithm. Throughout this thesis, the building shown in figure 2.1 will be used as an example.

2.1 Measurement Equations

In this section the variables describing the robot and landmarks and the measurement equations for odometry and landmark observations are defined:

When moving through a planar or nearly planar environment, the state of the mobile robot is described by three variables two for the robot position and one for the robot orientation (Fig. 2.2a). Similarly, a landmark is described by two variables for its position (Fig. 2.2c). For the purpose of this discussion a map consequently is a state vector of stacked landmark positions and robot poses. Depending on the context the vector may represent just the most recent, none or all robot poses. Since SLAM is an estimation theoretic problem the uncertainty of measurements and map estimates is important. Being described as a covariance matrix or alternatively as a so called information matrix, both are symmetric positive definite matrices, in which each row and column corresponds to one variable of the state vector.



(a) (b) (c)

Figure 2.2: Coordinates describing the robot state $p_r = (p_x, p_y, p_\phi)^T$ and measurements: (a) Continuous odometry measurement: velocity in robot coordinates $v_r = (v_x, v_y, v_\phi)^T$ (b) Discrete odometry measurement: relative movement $d_r = (d_x, d_y, d_\phi)^T$ from $p(t_0)$ to $p(t_1)$ (c) Landmark measurement: relative position $m = (m_x, m_y)^T$ of landmark at $l = (l_x, l_y)^T$

In general, the uncertainty for an estimate is derived from an a-priori model for the measurement and measurement uncertainty. The measurement is defined by a measurement function that maps the system state, i.e. map and robot pose to the measurement. Its Jacobian determines how much the measurement uncertainty as defined by a covariance matrix contributes to the uncertainty of the estimate. Thus in the following formulas for measurement functions, Jacobians and covariances of odometry and the landmark sensor will be derived:

In order to make the discussion independent from specific sensors it is assumed that the landmark sensor provides a landmark position and odometry provides the robot's velocity in robot coordinates with corresponding covariance matrix. This approach is feasible for most landmark sensors (laser scanner, stereo vision, mono vision with preprocessing) and considering odometry for all robots using wheels, tracks [CW90] or legs [Eur02] for locomotion. Odometry measurements are continuous occurring at every point of time. Nevertheless, the result is passed to the SLAM algorithm in discrete steps. This requires integrating the continuous measurements for some time and deriving a discrete measurement of the path traveled.

Continuous Odometry Equations

The discussion in this thesis is restricted to the planar environment case. The robot state is represented by 3 variables as $p_r = (p_x, p_y, p_\phi)^T$. The system provides an estimate for the robot's

velocity $\hat{v}_r = (\hat{v}_x, \hat{v}_y, \hat{v}_\phi)^T$ in robot coordinates. The measurement noise is assumed to be Gaussian and white with known covariance C_v obtained from a calibrated model of the mobile robot. By rotating v_r by an angle of p_ϕ, v_r is transformed into world coordinates and the kinematic equation of the system is derived. If v is replaced by $\hat{v}$ and p_r by $\hat{p}_r$ the odometric estimate for the robot position can be computed as

$$\dot{p}_r = \begin{pmatrix} \dot{p_x} \\ \dot{p_y} \\ \dot{p_\phi} \end{pmatrix} = \begin{pmatrix} \cos p_\phi & -\sin p_\phi & 0 \\ \sin p_\phi & \cos p_\phi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} v_x \\ v_y \\ v_\phi \end{pmatrix}, \quad \dot{\hat{p}}_r = \begin{pmatrix} \cos \hat{p}_\phi & -\sin \hat{p}_\phi & 0 \\ \sin \hat{p}_\phi & \cos \hat{p}_\phi & 0 \\ 0 & 0 & 1 \end{pmatrix} \hat{v}. \tag{2.1}$$

To derive a differential equation for the estimation error $\tilde{p}_r = \hat{p}_r - p_r$, the rotation by p_ϕ is factored out resulting in a rotation by $\hat{p}_\phi - p_\phi = \tilde{p}_\phi$ in front of the velocity estimate $\hat{v}_r$:

$$\dot{\tilde{p}}_r = \dot{\hat{p}}_r - \dot{p}_r = \begin{pmatrix} \cos p_\phi & -\sin p_\phi & 0 \\ \sin p_\phi & \cos p_\phi & 0 \\ 0 & 0 & 1 \end{pmatrix} \left[\begin{pmatrix} \cos \tilde{p}_\phi & -\sin \tilde{p}_\phi & 0 \\ \sin \tilde{p}_\phi & \cos \tilde{p}_\phi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \hat{v}_x \\ \hat{v}_y \\ \hat{v}_\phi \end{pmatrix} - \begin{pmatrix} v_x \\ v_y \\ v_\phi \end{pmatrix} \right]$$

Linearizing $\sin \tilde{p}_\phi$ and $\cos \tilde{p}_\phi$ at $\tilde{p}_\phi \to 0$ and defining $\tilde{v} = \hat{v} - v$ yields

$$\dot{\tilde{p}}_r \approx \begin{pmatrix} \cos p_\phi & -\sin p_\phi & 0 \\ \sin p_\phi & \cos p_\phi & 0 \\ 0 & 0 & 1 \end{pmatrix} \left[\begin{pmatrix} 1 & -\tilde{p}_\phi & 0 \\ \tilde{p}_\phi & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \hat{v}_x \\ \hat{v}_y \\ \hat{v}_\phi \end{pmatrix} - \begin{pmatrix} v_x \\ v_y \\ v_\phi \end{pmatrix} \right] \tag{2.2}$$

$$= \begin{pmatrix} \cos p_\phi & -\sin p_\phi & 0 \\ \sin p_\phi & \cos p_\phi & 0 \\ 0 & 0 & 1 \end{pmatrix} \left[\begin{pmatrix} 0 & 0 & -\hat{v}_y \\ 0 & 0 & \hat{v}_x \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \tilde{p}_x \\ \tilde{p}_y \\ \tilde{p}_\phi \end{pmatrix} + \begin{pmatrix} \tilde{v}_x \\ \tilde{v}_y \\ \tilde{v}_\phi \end{pmatrix} \right]. \tag{2.3}$$

To the first order $\hat{v}_x \tilde{p}_\phi \approx v_x \tilde{p}_\phi$ and $\hat{v}_y \tilde{p}_\phi \approx v_y \tilde{p}_\phi$, so

$$\dot{\tilde{p}}_r \approx \begin{pmatrix} \cos p_\phi & -\sin p_\phi & 0 \\ \sin p_\phi & \cos p_\phi & 0 \\ 0 & 0 & 1 \end{pmatrix} \left[\begin{pmatrix} 0 & 0 & -v_y \\ 0 & 0 & v_x \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \tilde{p}_x \\ \tilde{p}_y \\ \tilde{p}_\phi \end{pmatrix} + \begin{pmatrix} \tilde{v}_x \\ \tilde{v}_y \\ \tilde{v}_\phi \end{pmatrix} \right] \tag{2.4}$$

$$= \underbrace{\begin{pmatrix} 0 & 0 & -\cos p_\phi v_y - \sin p_\phi v_x \\ 0 & 0 & -\sin p_\phi v_y + \cos p_\phi v_x \\ 0 & 0 & 0 \end{pmatrix}}_{A(p_\phi, v_x, v_y) :=} \begin{pmatrix} \tilde{p}_x \\ \tilde{p}_y \\ \tilde{p}_\phi \end{pmatrix} + \underbrace{\begin{pmatrix} \cos p_\phi & -\sin p_\phi & 0 \\ \sin p_\phi & \cos p_\phi & 0 \\ 0 & 0 & 1 \end{pmatrix}}_{B(p_\phi) :=} \begin{pmatrix} \tilde{v}_x \\ \tilde{v}_y \\ \tilde{v}_\phi \end{pmatrix}. \tag{2.5}$$

From this differential equation another differential equation for the covariance of the position estimate C_p can be derived [Hon90, §7.3]:

$$\dot{C}_p = B(p_\phi)\, C_v\, B(p_\phi)^T + A(p_\phi, v_x, v_y)\, C_p + C_p\, A(p_\phi, v_x, v_y)^T \tag{2.6}$$

Since $A(p_\phi, v_x, v_y)$ and $B(p_\phi)$ depend on true orientation and true velocity, they cannot be used for actual computation. Replacing by the corresponding estimates $A(\hat{p}_\phi, \hat{v}_x, \hat{v}_y)$ and $B(\hat{p}_\phi)$ yields a good estimator for the covariance

$$\dot{\hat{C}}_p = B(\hat{p}_\phi)\, C_v\, B(\hat{p}_\phi)^T + A(\hat{p}_\phi, \hat{v}_x, \hat{v}_y)\, \hat{C}_p + \hat{C}_p\, A(\hat{p}_\phi, \hat{v}_x, \hat{v}_y)^T. \tag{2.7}$$

The error made thereby is part of the general linearization error and discussed in §2.9. A non-linearized analysis of the odometry error can be found in [Min88]. Equation (2.1) and (2.7) are computed[1] in the control loop of the mobile robot exploiting the high sensor rate of motor encoders (typically 100 - 2000Hz).

The models used for the simulated and real experiments are described in §5.1 and §6.1.

Discrete Odometry Equations

Even with good algorithms SLAM involves a lot of computation and cannot be implemented in the robot's controller loop. Anyway, this is not necessary. Landmark sensors and recognition systems typically run at a low rate of 2-25Hz. Between two landmark observations a SLAM algorithm simply updates the robot's position estimate according to odometry. Only after the next landmark observation is available a more complex update is necessary. Thus, the odometry data between two landmark observations at time t_0 and t_1 is accumulated. It is then passed to the SLAM algorithm in form of a discrete step $d_r = (d_x, d_y, d_\phi)^T$ with corresponding covariance C_d. The step corresponds to a translation of $(d_x, d_y)^T$ in robot coordinates followed by a rotation of d_ϕ (Fig. 2.2b). The robot controller continously integrates (2.1) providing $\hat{p}_r(t)$ and (2.7) providing $\hat{C}_p(t)$. So the step of the path traveled from time t_0 to time t_1 must be computed from the values $\hat{p}_r(t_0), \hat{C}_p(t_0)$ and $\hat{p}_r(t_1), C_p(t_1)$ of the odometry at these two points by solving

$$\hat{p}_r(t_1) = f_1(\hat{p}_r(t_0), d), \tag{2.8}$$

with f_1 mapping old robot pose $\hat{p}_r(t_0)$ and step d to new robot pose $\hat{p}_r(t_1)$:

$$s := \sin \hat{p}_\phi(t_0), \quad c := \cos \hat{p}_\phi(t_0) \tag{2.9}$$

$$f_1\left(\begin{pmatrix}\hat{p}_x(t_0)\\ \hat{p}_y(t_0)\\ \hat{p}_\phi(t_0)\end{pmatrix}, \begin{pmatrix}d_x\\ d_y\\ d_\phi\end{pmatrix}\right) = \begin{pmatrix}\hat{p}_x(t_0)\\ \hat{p}_y(t_0)\\ \hat{p}_\phi(t_0)\end{pmatrix} + \begin{pmatrix}c & -s & 0\\ s & c & 0\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}d_x\\ d_y\\ d_\phi\end{pmatrix}. \tag{2.10}$$

[1] Numerically integrating (2.7) for a time step Δt by Euler integration may lead to non-positive definite results later in (2.13). It is advisable to integrate a small discrete step $\Delta t\, v$ using (2.8) and (2.12) instead.

The Jacobians of f_1 with respect to $\hat{p}_r(t_0)$ and d_r are

$$J_1 := \frac{\partial \hat{p}_r(t_1)}{\partial \hat{p}_r(t_0)} = \begin{pmatrix} 1 & 0 & -sd_x - cd_y \\ 0 & 1 & cd_x - sd_y \\ 0 & 0 & 1 \end{pmatrix}, \quad J_2 := \frac{\partial \hat{p}_r(t_1)}{\partial d_r} = \begin{pmatrix} c & -s & 0 \\ s & c & 0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{2.11}$$

The corresponding equation for the covariance C_d of $\hat{d}$ follows from the white noise assumption for $\hat{v}$. Both $\hat{p}(t_0)$ and $\hat{d}$ are independent and the covariance for $\hat{p}(t_1)$ can be expressed as

$$C_p(t_1) = J_1 C_p(t_0) J_1^T + J_2 C_d J_2^T. \tag{2.12}$$

Using $J_2^{-1} = J_2^T$ (2.8) is solved for $\hat{d}$ and (2.12) for C_d yielding the step

$$\hat{d} = J_2^T \left(\hat{p}_r(t_1) - \hat{p}_r(t_0)\right) \tag{2.13}$$

$$C_d = J_2^T \left(C_p(t_1) - J_1 C_p(t_0) J_1^T\right) J_2 \tag{2.14}$$

passed to the SLAM algorithm. Equation (2.10) expresses odometry as a dynamic equation which maps the old state $p_r(t_0)$ and the measurement d_r to the new state $p_r(t_1)$. The corresponding measurement equation which maps old and new state to measurement is

$$\begin{pmatrix} d_x \\ d_y \\ d_\phi \end{pmatrix} := f_2 \left(\begin{pmatrix} \hat{p}_x(t_0) \\ \hat{p}_y(t_0) \\ \hat{p}_\phi(t_0) \end{pmatrix}, \begin{pmatrix} \hat{p}_x(t_1) \\ \hat{p}_y(t_1) \\ \hat{p}_\phi(t_1) \end{pmatrix} \right) = \begin{pmatrix} c & s & 0 \\ -s & c & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \hat{p}_x(t_1) - \hat{p}_x(t_0) \\ \hat{p}_y(t_1) - \hat{p}_y(t_0) \\ \hat{p}_\phi(t_1) - \hat{p}_\phi(t_0) \end{pmatrix} \tag{2.15}$$

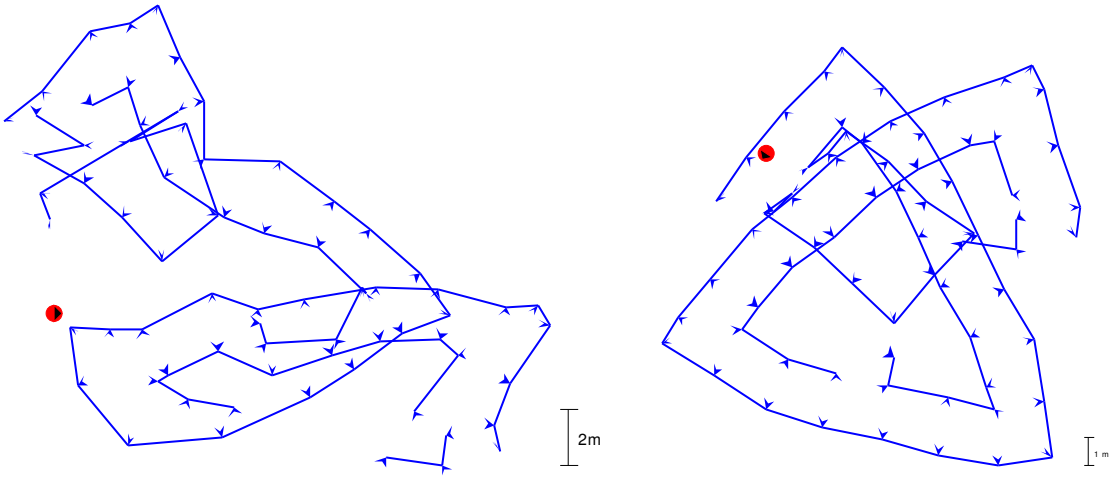
$$J_3 := \frac{\partial d_r}{\partial \hat{p}_r(t_0)} = \begin{pmatrix} -c & -s & 0 \\ s & -c & 0 \\ 0 & 0 & -1 \end{pmatrix}, \quad J_4 := \frac{\partial d_r}{\partial \hat{p}_r(t_1)} = \begin{pmatrix} c & s & 0 \\ -s & c & 0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{2.16}$$

Landmark observation

A huge amount of different landmarks and landmark sensors have been proposed in literature [CW90]. Some common examples are

- *Artifical landmarks:* ultrasonic beacons, radio transmitters, infrared transmitters, laser reflectors, visual markers of specific color or pattern, inductive loops in the ground
- *Natural landmarks:* corners, walls, vertical lines, visual corners, doors, ceiling grates, trees
- *Landmark sensors:* ultrasonic transducers, laser range scanners, cameras

In this thesis the discussion will be restricted to point landmarks in the plane that can be described by two coordinates $(l_x, l_y)^T$ and sensors that allow measuring the location of the landmark relative to the robot $(m_x, m_y)^T$ (Fig. 2.2c). This is feasible for most of the examples mentioned



(a) (b)

Figure 2.3: Error accumulation for map in figure 2.1 (a) without (b) with landmark observations. All maps in §1 – §5 are based on simulations where the measurements are perturbed by comparatively large artificial noise and bias (See §5 for a detailed specification).

---

above. Similar to odometry a measurement covariance C_m must be provided. It is further assumed, that the landmarks can be identified (see discussion in §2.5). The measurement equation and Jacobians are

$$c := \cos p_\phi, \quad s := \sin p_\phi \tag{2.17}$$

$$\begin{pmatrix} m_x \\ m_y \end{pmatrix} := f_3 \left(\begin{pmatrix} l_x \\ l_y \end{pmatrix}, \begin{pmatrix} p_x \\ p_y \\ p_\phi \end{pmatrix} \right) = \begin{pmatrix} c & s \\ -s & c \end{pmatrix} \begin{pmatrix} l_x - p_x \\ l_y - p_y \end{pmatrix} \tag{2.18}$$

$$J_5 := \frac{\partial m}{\partial p_r} = \begin{pmatrix} -c & -s & -s(l_x - p_x) + c(l_y - p_y) \\ s & -c & -c(l_x - p_x) - s(l_y - p_y) \end{pmatrix} = \begin{pmatrix} -c & -s & m_y \\ s & -c & -m_x \end{pmatrix} \tag{2.19}$$

$$J_6 := \frac{\partial m}{\partial l} = \begin{pmatrix} c & s \\ -s & c \end{pmatrix}. \tag{2.20}$$

2.2 Error Accumulation

Consider the robot moving through a known environment, i.e. by using an a-priori map or in a region already mapped with SLAM, then the uncertainty of the robot's pose remains bounded, as each observation of two landmarks reduces the uncertainty basically down to the landmark's

uncertainty plus the uncertainty of the observation.

However, if the robot moves through an unknown region the uncertainty of its pose in absolute coordinates will get arbitrarily large because the odometric error accumulates over time (Fig. 2.3a). The uncertainty can be reduced by fusing odometry with several measurements of a new landmark as the landmark passes by (Fig. 2.3b). For most sensors this effects much better results than just using odometry [TBF98]. Nevertheless, estimating the robot's position after traveling a long distance is still subject to accumulated error: Due to the limited sensor range the position is derived from a chain of several relations between successive landmarks.

For outdoor applications the problem can be facilitated by using a compass [LF99, DMS02], which is known not to work properly in buildings containing large amounts of steel.

The fact that errors may accumulate to arbitrarily high values distinguishes SLAM from many other estimation problems and gives rise to the problems discussed in §2.3 and §2.9.

2.3 Representation of Relativity

The author believes that the dominant aspect of SLAM is the need to model

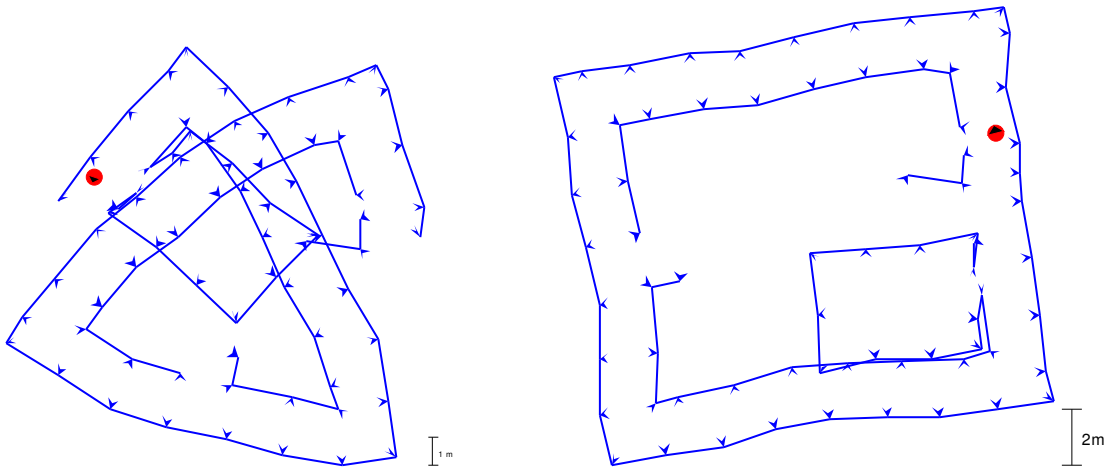
Certainty of Relations despite Uncertainty of Positions

This may be called *ability* to *represent relativity*. In the example scenario for instance, the pose of the room will be quite uncertain, while its shape will be highly certain.

If the robot moves through an unknown region and observes a sequence of landmarks, the uncertainty of relative positions of the landmarks only depends on the measurement errors of the landmarks by the robot and on the odometric error between those measurements. So the most precisely known relations are those concerning the relative location of adjacent landmarks.

The uncertainty of the absolute robot pose before observing the first landmark however increases the uncertainty of the absolute position of all landmarks, acting as an unknown rigid body transformation on the whole set of observed landmarks. As the absolute robot pose is subject to error accumulation, the common situation is that relations are quite certain, whereas absolute positions can be arbitrarily uncertain. In extremely large maps this effect can appear at different scales: The relative positions of some landmarks in a room are much more precisely known than the position of the room in the building, which, seen as a relative position with respect to other rooms in turn is much more precisely known than the absolute position of the building.

Thus, a SLAM system should be able to represent the certainty of relations between landmarks despite large uncertainty in the absolute position of the landmarks. In particular, a representation assigning a single uncertainty value to each landmark only is insufficient.



(a) (b)

Figure 2.4: Closing the loop: (a) before (b) after integration

It is a very important result for the theory of SLAM when repeatedly moving through the same environment the uncertainty of any relation converges to zero [New99, DNC+01, theorem 2.2]. This theorem clarifies the uncertainty structure in the limit being of theoretical interest, but in general this is probably neither practical nor necessary. Most applications can exclusively be based on relative information: When navigating, for instance, it is not necessary to compute the exact trajectory from start to finish in some global coordinate system. Path planning will rather result in a sequence of waypoints. The location of each waypoint will be known relative to the surrounding landmarks. So that the robot, knowing its own pose relative to those landmarks, will be able to navigate from one waypoint to the next. A path defined this way will even remain valid when the map changes significantly while the robot is moving.

So it is important to focus on the behavior of the SLAM system when moving through the building the first time. The structure of the uncertainty is still complex and a single measurement may have a significant effect on the estimate. This effect is most prominent and probably the most important test case in general when closing large loops.

2.4 Implications of Closing the Loop

Assume the robot moves along a closed loop and returns to the beginning of the loop but has not yet re-identified any landmark, so this is not known to the robot. Typically, the loop is not closed in the map due to the error accumulated along the loop (Fig. 2.4a).

At the beginning of the loop a landmark is re-identified and the corresponding measurement is integrated into the map causing the loop to get closed. To achieve this, the SLAM system has to ”deform” the whole loop to incorporate the information of a connection between both ends of the loop without introducing a break somewhere else (Fig. 2.4b).

This goal is sometimes referred to as the map being ”topologically consistent” or “globally consistent” [LM97, DMS02], meaning that two parts of the map are represented to be adjacent if and only if this was observed by the robot. Within a landmark based approach adjacency is not explicitly modeled. Topological consistency has to be interpreted in the sense that two landmarks are represented being near to each other (the distance being low with low uncertainty), if and only if this was observed by some measurement.

It has to be emphasized that correct treatment of uncertainty contained in the measurements will implicitly yield the necessary deformation. More specifically, the precisely known relative location of each landmark with respect to adjacent landmarks prevents any break in the loop: If there was a break the relative positions of the landmarks on both sides of the break were highly incorrect, thus being inconsistent with the measurements made in this vicinity. So the map estimate which is consistent with all measurements automatically deforms smoothly when closing large loops.

An important insight is that any representation for the uncertainty of a map estimate must be able to “represent relativity” in order to achieve this kind of behavior.

2.5 Landmark Identification

The algorithm presented in this thesis assumes that observed landmarks can be identified. This is a very difficult, but highly important problem, since the fact of closing a loop is only evident from the re-identification of some other landmarks. Moreover, when integrating a wrongly identified landmark often the whole map is ruined as a consequence because two different places are assumed to be the same.

There are some approaches taking advantage of a tight integration of mapping and identification [BFJ+99]. Some explicitly represent multi hypothesis map distributions arising from uncertain identification [DWMdB+01]. However, it is often a good idea to separate both, as the SLAM problem can be formulated rather independently of the sensors used, whereas landmark identification usually depends heavily on them.

A great advantage of having an uncertainty representation for the map estimate is that it can provide a measure of plausibility for an assumed identification. Under the assumption of Gaussian noise such a measure can be provided by comparing the Mahalanobis distance with the

χ^2 distribution. This allows comparing different possible identifications of a group of observed landmarks and choosing the most likely one. Except for unusually dense landmarks (like vertical lines) this approach provides a reliable identification [NT01] and is used in real experiments described in this thesis (§6.4). In the simulation experiments, the identification is assumed to be correct.

2.6 Maximum Likelihood Estimation

There is a thorough and straightforward approach for optimally solving SLAM if independent Gaussian measurement errors with a-priori known covariance are assumed and computation time is no issue. In this case the optimal solution is the maximum likelihood (ML) solution [PTVF92, §15.1]. It is based on the a-priori known probability distribution for the measurement given the map. After the measurement has been made, this distribution is interpreted as a likelihood distribution for the maps given the measurement. The map with the largest likelihood is the ML estimate. Of all maps it has the largest probability of causing the observed measurements and thus is optimal in this sense. It is also the map with the largest probability if the true map is assumed to be drawn from a uniform a-priori distribution. By definition of Gaussian errors the likelihood for a map given a single measurement y_i is

$$p_i(x) \propto e^{-\frac{1}{2}q_i(x)}, \quad q_i(x) := (y_i - f_i(x))^T C_i^{-1}(y_i - f_i(x)). \tag{2.21}$$

Assume that the data under consideration consist of n landmarks, p robot poses and m measurements, the landmark positions and different robot poses form the parameter vector x having size $2n + 3p$ in the equation. The vector y_i is the i-th measurement, C_i its covariance and $f_i(x)$ is the corresponding measurement equation, i.e. the value the measurement should have if the landmark and robot poses were x.

The likelihood of x given all measurements is the product of the individual likelihoods, due to stochastical independence:

$$p(x) \propto \prod_{i=1}^{m} e^{-\frac{1}{2}q_i(x)} = e^{-\frac{1}{2}\sum_{i=1}^{m} q_i(x)} = e^{-\frac{1}{2}\chi^2(x)}, \quad \text{with} \quad \chi^2(x) := \sum_{i=1}^{m} q_i(x). \tag{2.22}$$

The function $\chi^2(x)$ is the negative log-likelihood of map x given the measurements and measures how good a map x explains the measurements made. Its name originates from the fact that its minimum follows a so called "chi-square" distribution. ML estimation means finding the maximum of $p(x)$ which is the minimum of $\chi^2(x)$:

$$\hat{x}_{\text{ML}} = \arg\max_x p(x) = \arg\min_x \chi^2(x). \tag{2.23}$$

In order to find the numerical minimum, a least squares nonlinear model fitting algorithm like Levenberg-Marquardt can be employed [PTVF92, §15.5], by iteratively linearizing the measurement equations. The linearized equations yield a quadratic approximation to χ^2, the minimum of which can be found by solving a large linear equations system. This approach requires to represent all old robot poses in the equations system, which consequently has $O(n+p)$ equations and variables. Solving such a system needs $O((n+p)^3)$ computation time and has to be performed in each iteration of the Levenberg-Marquardt algorithm. So this approach is not a practical solution for SLAM. Its invaluable benefit, however, lies in the fact that it can provide a reference for discussion and for comparison with efficient approaches.

2.7 Linear Least Squares

If the measurement functions f_i are linearized at some point x_i^0 with Jacobian J_{f_i}, the χ^2 function becomes quadratic:

$$f_i^{lin}(x) = f_i(x_i^0) + J_{f_i}(x_i^0)(x - x_i^0) \tag{2.24}$$

$$\begin{aligned}
\chi^2_{lin} &= \sum_{i=1}^{m} \left(y_i - f_i^{lin}(x)\right)^T C_i^{-1} \left(y_i - f_i^{lin}(x)\right) \\
&= \sum_{i=1}^{m} \left(y_i - f_i(x_i^0) - J_{f_i}(x_i^0)(x - x_i^0)\right)^T C_i^{-1} \left(y_i - f_i(x_i^0) - J_{f_i}(x_i^0)(x - x_i^0)\right) \\
&= \sum_{i=1}^{m} \left(y_i^{lin} - J_{f_i}(x_i^0)x\right)^T C_i^{-1} \left(y_i^{lin} - J_{f_i}(x_i^0)x\right), \\
&\qquad \text{with} \quad y_i^{lin} := y_i - f_i(x_i^0) + J_{f_i}(x_i^0)x_i^0 \\
&= x^T \underbrace{\left(\sum_{i=1}^{m} J_{f_i}(x_i^0)^T C_i^{-1} J_{f_i}(x_i^0)\right)}_{A} x + x^T \underbrace{\left(2\sum_{i=1}^{m} J_{f_i}(x_i^0)^T C_i^{-1} y_i^{lin}\right)}_{b} + \text{const}\,.
\end{aligned} \tag{2.25}$$

Such a quadratic function can always be represented as $x^T Ax + x^T b + \text{const}$, with a symmetric positive definite (SPD) matrix A and a vector b. The linearized least squares (LLS) estimate $\hat{x}_{\text{LLS}}$ being the same as the linearized maximum likelihood estimate, can be computed by

$$\hat{x}_{\text{LLS}} = \arg\min_x \chi^2_{lin}(x) \tag{2.26}$$

$$0 = \frac{\partial \chi^2_{lin}}{\partial x}(\hat{x}_{\text{LLS}}) = 2A\hat{x}_{\text{LLS}} + b \quad \Longrightarrow \quad \hat{x}_{\text{LLS}} = A^{-1}b/2. \tag{2.27}$$

The matrix A is called information matrix. Its inverse A^{-1} is the error covariance of the estimate $\hat{x}_{\mathrm{LLS}}$ [PTVF92, §15.6]. High entries in A correspond to precisely known relations. This matrix is sparse and has an important structure that is discussed in §2.11.

The quality of a linearized least squares estimate depends on the points of linearization chosen: If all measurements are *always* linearized at the latest estimate and the whole process is iterated until convergence, the final estimate is the ML estimate. Actually this is the way nonlinear least squares algorithms like Levenberg - Marquardt work. However this approach involves re-evaluating all Jacobians and thus storing all measurements.

Another approach is to linearize each measurement once and forever at the estimate in the moment of measurement. This way there is no need to re-evaluate the Jacobians and the measurements can be accumulated in matrix A and vector b. Each measurement involves only 5 or 6 variables in the case of landmark observation or odometry respectively. So J_{f_i} is sparse and the accumulation can be performed in $O(1)$. Nevertheless, to provide an estimate the equation system $A\hat{x} = b/2$ has to be solved, which takes $O((n+p)^3)$ or $O((n+p)^2)$ exploiting sparsity. With this approach, the estimate is subject to linearization error depending on the error in the different estimates used for linearization. The error magnitude depends on the robot and environment. Since especially the robot orientation error accumulates, it may easily exceed 45° in practical settings, for instance rendering all linearization of sine and cosine useless. The effect of the linearization error is discussed in §2.9 and illustrated in figure 2.5a (page 62).

It is important to note that the information matrix A represents all landmark positions and all robot poses. This means that there are two rows / columns for each landmark and three for each robot pose. A landmark observation can be integrated without extending the matrix. However, for each odometry measurement three new rows / columns corresponding to the new robot pose have to be added. Thus, the size of the representation still depends on the number of robot poses and is still growing even when moving through an already mapped area. This problem can be avoided by removing old robot poses from the representation via Schur complement (§3.3). The resulting information matrix P' of all landmarks and the actual robot pose is not exactly sparse any more. In §2.11 a proof for its approximate sparsity will be given. It can be replaced by conservative sparse approximation, so the matrix has only $O(n)$ entries and equation solving can be performed in $O(n^2)$. The algorithm proposed in this thesis takes this approach but hierarchically subdivides P', so that incremental equation solving can be performed even in $O(\log n)$ computation time.

2.8 Extended Kalman Filter

The Extended Kalman Filter (EKF) [Gel74] is the tool most often applied to SLAM [SSC88, Tar92, HBBC95, CMNT99] using the same measurement equations as for ML estimation. The EKF integrates all measurements into a covariance matrix C_{EKF} of the landmark positions and the actual robot pose without any measurements to be stored afterwards. The estimate $\hat{x}_{\text{EKF}}$ provided is the same as for linearized least squares, so EKF suffers the same linearization problems as will be discussed in §2.9.

Since the EKF maintains a covariance matrix instead of an information matrix, elimination of old robot poses is no problem but can simply be done by removing corresponding rows and columns. Actually, most implementations do not even generate rows and columns for old robot poses, but simply replace the old robot pose with the new one whenever integrating an odometry measurement. The updated estimate $\hat{x}'_{\text{EKF}}$ with covariance C'_{EKF} after an odometry measurement y with covariance C_y is given by

$$\hat{x}_{\text{EKF}} = \begin{pmatrix} \hat{x}_l \\ \hat{x}_p \end{pmatrix}, \quad C_{\text{EKF}} = \begin{pmatrix} C_{ll} & C_{lp} \\ C_{pl} & C_{pp} \end{pmatrix} \tag{2.28}$$

$$\hat{x}' = \begin{pmatrix} \hat{x}_l \\ f_1(\hat{x}_p, y) \end{pmatrix} \tag{2.29}$$

$$C' = \begin{pmatrix} I & 0 \\ 0 & J_1 \end{pmatrix}^T C \begin{pmatrix} I & 0 \\ 0 & J_1 \end{pmatrix} + \begin{pmatrix} 0 \\ J_2 \end{pmatrix}^T C_y \begin{pmatrix} 0 \\ J_2 \end{pmatrix} = \begin{pmatrix} C_{ll} & C_{lp}J_1 \\ J_1^T C_{pl} & J_1^T C_{pp} J_1 + J_2^T C_y J_2 \end{pmatrix}. \tag{2.30}$$

Here $\hat{x}_l$ is the first part of the state vector corresponding to the landmarks and $\hat{x}_p$ the part corresponding to the robot pose. The covariance matrix C_{EKF} is block decomposed accordingly. The function f_1 defined in (2.10) is the EKF *dynamic equation* that maps the old robot pose and the odometry measurement to the new robot pose. The Jacobians J_1 with respect to x_p and J_2 with respect to y are given in (2.11).

The key problem is to update the covariance matrix C_{EKF} after a landmark observation. From (2.25) it can be seen that a single term $J^T C_y J$ is added to the information matrix A resulting in

$$J := \begin{pmatrix} & \overset{\text{landmark}}{\downarrow} & & & \overset{\text{robot}}{\downarrow} \\ 0, \ldots, 0, & J_6 & , 0, \ldots, 0, & J_5 \end{pmatrix} \tag{2.31}$$

$$A' = A + J^T C_y^{-1} J \tag{2.32}$$

$$b' = b + 2J^T C_y^{-1}(y - f_3(\hat{x_{\text{EKF}}}) + J\hat{x_{\text{EKF}}}), \tag{2.33}$$

with f_3 measurement function, J its Jacobian, y the measurement and C_y the measurement covariance. The function f_3 is given by (2.18), its Jacobians J_5 by (2.19) and J_6 by (2.20). Since

the measurement is two dimensional, the change in the information matrix A has rank 2 and the resulting change in $C = A^{-1}$ and $\hat{x} = A^{-1}b/2$ can be efficiently computed via the Woodbury formula (appendix A.2). The result is the well known EKF update equation [SSC88, Gel74]

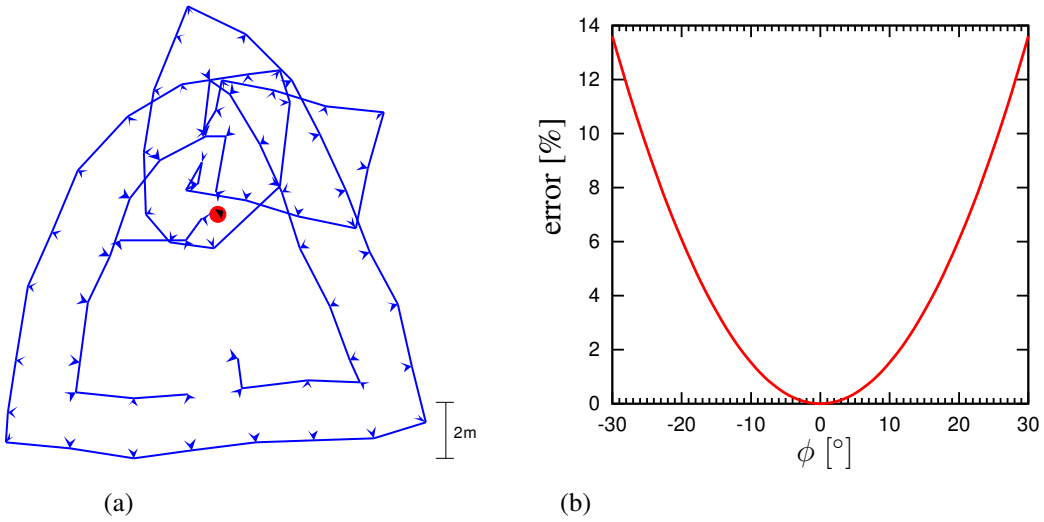
$$\begin{aligned}
C' &= A'^{-1} = \left(C^{-1} + J^T C_y^{-1} J\right)^{-1} && (2.34)\\
&\overset{Woodbury}{=} C - CJ^T \left(JCJ^T + C_y\right)^{-1} JC && (2.35)\\
\hat{x}' &= A'^{-1}b'/2 = C'b'/2 && (2.36)\\
&= \left(C^{-1} + J^T C_y^{-1} J\right)^{-1} \left(C^{-1}\hat{x} + J^T C_y^{-1}\left(y - f_3(\hat{x}) + J\hat{x}\right)\right) && (2.37)\\
&= \left(C^{-1} + J^T C_y^{-1} J\right)^{-1} \left((C^{-1} + J^T C_y^{-1} J)\hat{x} + J^T C_y^{-1}\left(y - f_3(\hat{x})\right)\right) && (2.38)\\
&= \hat{x} + \left(C^{-1} + J^T C_y^{-1} J\right)^{-1} J^T C_y^{-1}\left(y - f_3(\hat{x})\right) && (2.39)\\
&\overset{Woodbury}{=} \hat{x} + CJ^T \left(JCJ^T + C_y\right)^{-1}\left(y - f_3(\hat{x})\right). && (2.40)
\end{aligned}$$

For the SLAM problem, the update is moderately efficient due to the sparsity of the measurement Jacobian J. The measurement only involves the robot pose and landmark position, so J is 0 except in 5 columns. Consequently, if there are n landmarks, evaluating JC or CJ^T takes $O(n)$ operations and computing the inverse of the so called innovation covariance $\left(JCJ^T + C_y\right)^{-1}$ takes $O(1)$ operations. The dominant operation for the EKF is the update of C taking $O(n^2)$ operations. Compared with $O((n+p)^3)$ for linearized least squares, EKF is much more efficient. But as computation time is still so large EKF can practically be used for small environments ($n \lessapprox 100$) only.

2.9 Linearization Error

There are two sources for a linearization error: The error of the robot's orientation estimate $\hat{p}_\phi$ (*orientation error*) and the error of the measurements d and m. This important fact can be seen when transforming the measurement Jacobians ($J_1 \ldots J_6$ from §2.1 equation (2.11), (2.16), (2.19), (2.20)) into robot coordinates. Therefore a rotation matrix R_2 or R_3 is multiplied left and / or right to the Jacobian if the functions result and / or argument is given in world coordinates:

$$s := \sin\hat{p}_\phi, \quad c := \cos\hat{p}_\phi, \quad R_2 := \begin{pmatrix} c & -s \\ s & c \end{pmatrix}, \quad R_3 := \begin{pmatrix} c & -s & 0 \\ s & c & 0 \\ 0 & 0 & 1 \end{pmatrix} \qquad (2.41)$$



(a) (b)

Figure 2.5: Linearization error: (a) closing the loop with EKF / LLS (b) linearization error of linearized rotation $\left|\binom{\cos\phi}{\sin\phi} - \binom{1}{\phi}\right|$ as a function of angular error ϕ

$$J_1 = R_3 \begin{pmatrix} 1 & 0 & -d_y \\ 0 & 1 & d_x \\ 0 & 0 & 1 \end{pmatrix} R_3^T, \quad J_2 = R_3 \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad J_3 = \begin{pmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{pmatrix} R_3^T, \tag{2.42}$$

$$J_4 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} R_3^T, \quad J_5 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} R_2^T, \quad J_6 = \begin{pmatrix} -1 & 0 & m_y \\ 0 & -1 & -m_x \end{pmatrix} R_3^T \tag{2.43}$$

The variables d_x, d_y, m_x, m_y involved in the transformed Jacobians are directly measurable. So their errors do not accumulate and are small enough to be neglected. Not so for the rotation matrices R_2, R_3 depending on p_ϕ. When moving through an unmapped area the orientation error accumulates. In practical settings $45°$ may easily be exceeded rendering all linearizations of sine and cosine useless.

The effect of processing the example scenario with EKF instead of using ML estimation is disastrous (Fig. 2.5a). Begin and end of the loop do not match and, even worse, the room although precisely known gets significantly larger than before. The reason for this is that EKF would have to move and rotate the room implicitly to make the map consistent. Instead, a rotation linearizing the angle at 0 is performed resulting in

$$\mathrm{Rot}(\phi) := \begin{pmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{pmatrix} \xrightarrow{\phi=0} \begin{pmatrix} 1 & -\phi \\ \phi & 1 \end{pmatrix} = \sqrt{1+\phi^2} \cdot \mathrm{Rot}(\arctan\phi). \tag{2.44}$$

The last equation can be seen by observing that the matrix is orthogonal and the first column vector $\left(\begin{smallmatrix}1\\ \phi\end{smallmatrix}\right)$ has a length of $\sqrt{1+\phi^2}$ and an angle of $\arctan\left(\frac{\phi}{1}\right) < \phi$. The consequence is that the room is larger than before and rotated by too small an angle. The linearization error made by linearized rotation of a unit vector by angle ϕ is shown in figure 2.5b. It grows quadratically with ϕ, so a large angular error results in a very large linearization error whereas a small angular error results in a negligible linearization error.

As a rule of thumb the linearization error is more than 3.4% when the angular error exceeds $15°$. This is comparable to a typical stochastical error making the linearized estimate inconsistent with the nonlinear χ^2 likelihood, for instance referring to the distance between two landmarks. On the other hand the linearization error is less than 0.38% and negligible if the angular error is below $5°$. In order to cope with nonlinearity it thus suffices to use linearization points with an error of $< 5°$. It is not necessary to be more precise.

Despite the well known magnetic disturbances in many buildings, if the robot is equipped with a compass, the orientation error can be bounded. The resulting linearization error can probably be neglected [DMS02]. If not the orientation error introduces severe artefacts into the linearized map estimate. These errors grow with growing map size.

The specific structure of the Jacobians exhibited in (2.42) allows to change the point of linearization of measurements already integrated into an information matrix by applying a rotation to that matrix. This fact is exploited by the algorithm presented in this thesis to handle linearization errors due to wrong robot orientation with utmost efficiency (§3.9).

2.10 Covariance vs. Information Matrices

Covariance and information matrices are complementary representations of uncertainty, since one is the inverse of the other. This duality extends to the operation of taking a submatrix, which is equivalent to applying Schur - complement in the inverse (Woodbury formula, appendix A.2):

$$\begin{array}{ccc} P - R^T S^{-1} R & \xleftrightarrow{\text{Inverse}} & P' \\ \text{Schur Complement}\uparrow & & \uparrow\text{Submatrix} \\ \left(\begin{smallmatrix} P & R^T \\ R & S \end{smallmatrix}\right) & \xleftrightarrow{\text{Inverse}} & \left(\begin{smallmatrix} P' & R'^T \\ R' & S' \end{smallmatrix}\right) \end{array} \tag{2.45}$$

This diagram certainly holds for any decomposition of A into 2×2 blocks $\left(\begin{smallmatrix} P & R^T \\ R & S \end{smallmatrix}\right)$. Of particular interest is the decomposition with rows and columns of the first block corresponding to landmarks (maybe including the current robot pose) and rows and columns of the second block

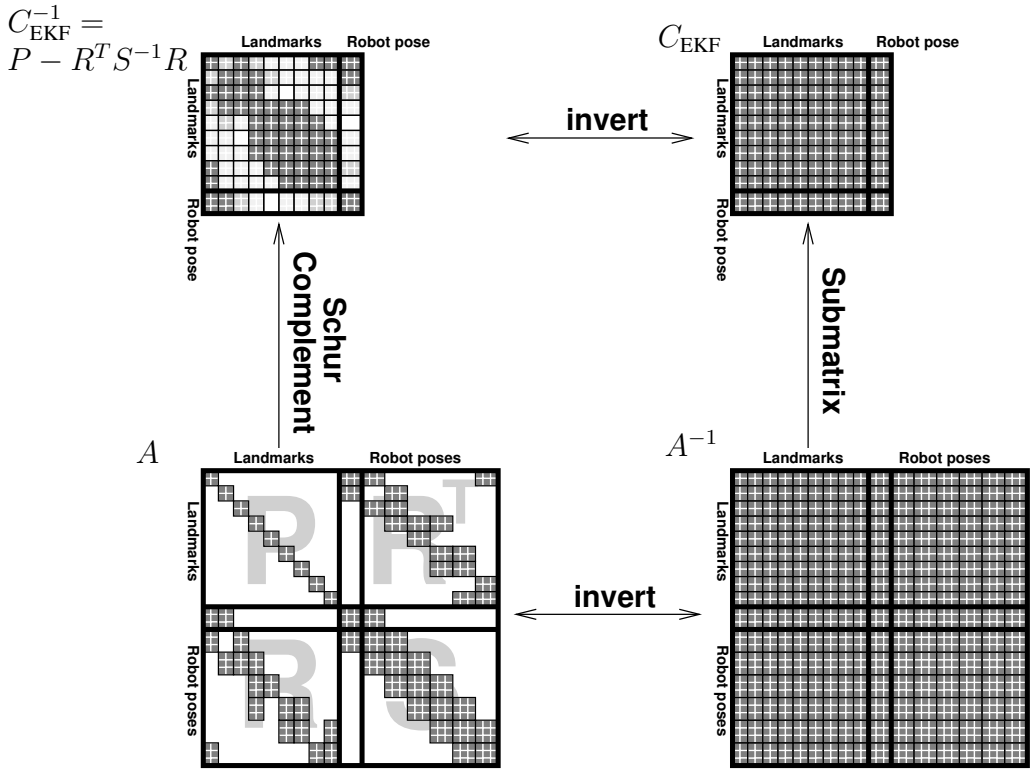


Figure 2.6: Relation between the least squares information matrix $A = \left(\begin{smallmatrix} P & R^T \\ R & S \end{smallmatrix}\right)$ (lower left) which represents all robot poses and the covariance matrix C_{EKF} (upper right) used by EKF only representing the actual robot pose. A^{-1} is the covariance matrix corresponding to A representing all robot poses (lower right). C is derived from A^{-1} as a submatrix. Accordingly C_{EKF}^{-1} (upper left) is the information matrix corresponding to C_{EKF} representing only the actual robot pose. It is derived from A via Schur complement.

So on the whole, taking a submatrix of a covariance matrix is equivalent to applying Schur complement to an information matrix.

corresponding to (old) robot poses. In this case $P' = \left(P - R^T S^{-1} R\right)^{-1}$ is the covariance matrix of all landmarks (and the current robot pose) as used by the EKF.

The Schur complement $P - R^T S^{-1} R$ equals the corresponding submatrix P minus a correction term $R^T S^{-1} R$. This term can be thought of as somehow "transferring" the effect of S into the realm of P via a mapping provided by the off-diagonal block R^T. The Schur complement plays a very important role in the algorithm proposed in this thesis when used for manipulating information in an information matrix representation.

Taking a submatrix of the information matrix or applying Schur - complement to the covariance matrix corresponds to random variables (landmark positions, robot poses) in the removed rows and columns being exactly known. Conversely taking a submatrix of the covariance matrix or applying Schur - complement to the information matrix corresponds to random variables in the removed rows and columns being unknown, i.e. all information about them is discarded.

The main difference between information and covariance matrix lies in the representation of indirect relations. Assume the robot is at pose P1 observing landmark L1 and moves to P2 observing L2. The measurements directly define relations P1-L1, P1-P2, P2-L2, indirectly constituting a relation L1-L2. The covariance matrix explicitly stores this relation in the off-diagonal entries corresponding to L1-L2, whereas the information matrix does not.

Thus the information matrix $A = \left(\begin{smallmatrix} P & R^T \\ R & S \end{smallmatrix}\right)$ used by LLS is sparse, having non-zero off-diagonal entries only for those pairs of random variables which are involved in a common measurement (Fig. 2.6). The inverse A^{-1} is the covariance matrix for the landmarks and *all* robot poses. A^{-1} represents all indirect relations explicitly and thus is not sparse. Removing the rows and columns corresponding to old robot poses yields the covariance matrix C of the EKF. Its inverse C^{-1} is the information matrix of all landmarks and the current robot pose. However, the inverse is not the corresponding submatrix of A, as eliminating all old robot poses from A requires computing their implicit effect on relations between the other random variables by Schur complement.

Although A is sparse the Schur complement $P - R^T S^{-1} R$ is dense, because S^{-1} is dense. What is turning out is that it is *approximately sparse* with an off-diagonal entry $(P - R^T S^{-1} R)_{l_1 l_2}$ corresponding to two landmarks l_1, l_2 decaying exponentially with the distance traveled between observation of l_1 and l_2. This important result will be shown in the next section:

2.11 Sparsity of SLAM Information Matrices

In this section the central result for the SLAM uncertainty structure is derived, saying that the information matrix appearing in SLAM is approximately sparse:

In the SLAM information matrix off-diagonal entries corresponding to two landmarks decay exponentially with the distance traveled between observation of first and second landmark.

This result is important both for computation and analysis. First, the approach of saving space and computation time by making the information matrix sparse is being confirmed. This approach has been proposed by the author of this thesis [FH01] and is the basis of the algorithm presented here. It is also utilized in the well known work of Thrun et al. on sparse extended information filters (SEIF) [TKG+02]. Second the result implies that the large scale uncertainty structure of a map estimate is generated by local uncertainties composed along the path the robot has been travelling. Thus, in contrast to the local uncertainty structure, it is rather simple and dominated by the map's geometry. This topic will be discussed in §2.12.

Proof Outline

First, the structure of information matrix A for all landmarks and *all* robot poses is being analyzed. It is a block matrix $A = \left(\begin{smallmatrix} P & R^T \\ R & S \end{smallmatrix}\right)$ with the first block row / column corresponding to the landmarks and the second corresponding to the different robot poses[2]. As discussed in the previous section, the diagonal blocks P and S are information matrices of two related subproblems: P is the information matrix of the mapping subproblem, describing the uncertainty of the landmarks, if the robot poses were known. Conversely, S is the information matrix of the localization subproblem, describing the uncertainty of the robot poses, if the landmarks were known. Both matrices are extremely sparse: P is block diagonal and S is block tridiagonal.

The matrix under investigation will be the information matrix of the landmarks only, i.e. without robot poses. It is $P - R^T S^{-1} R$ by Schur - complement. The role of R^T in this formula is to provide a mapping from robot poses to landmarks. It creates an off-diagonal entry between two landmarks, whose magnitude depends on the entry in S^{-1} corresponding to the two robot poses these landmarks have been observed from. S^{-1} is the covariance of all robot poses given the position of all landmarks. Hence the magnitude of an off-diagonal entry corresponding to two landmarks depends on the covariance the robot poses had if all landmark positions were known.

This covariance decays exponentially with the distance traveled. Intuitively the reason therefore is that in each localization step the pose estimate is replaced by a weighted sum of the old estimate and the measurements of observed landmarks. The covariance with a fixed old robot pose is reduced by a constant factor. Formally, this result is derived by bounding the eigenval-

[2]Observe the difference to figure 2.6, where the current robot pose is included in P.

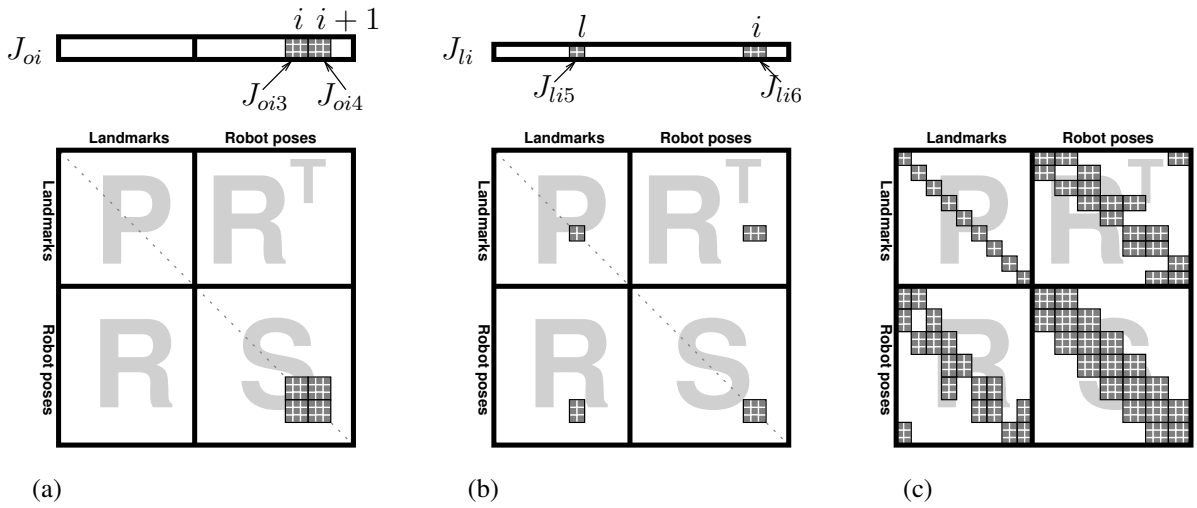


Figure 2.7: Sparsity pattern of $A = \left(\begin{smallmatrix} P & R^T \\ R & S \end{smallmatrix}\right)$: (a) Jacobian J_{oi} for an odometry measurement; Non-zero blocks generated in S thereby (b) Jacobian J_{li} for a landmark measurement; Non-zero blocks generated in P, R and S thereby (c) example for a complete matrix

ues of S (lemma 1 [Tee00]) and applying a theorem on the decay of off-diagonal entries in the inverse of band matrices (theorem 1 [DMS84]).

The proof is based on a close inspection of different parts of A affected by different measurements, being formally defined in the following subsection. Keeping in mind the structure of A as shown in figure 2.7 and the summary of definitions in table 2.1 will be sufficient to understand the argument of the proof.

Sparsity Pattern of the Information Matrix A

As mentioned above, the information matrix for landmarks and robot poses can be decomposed as $A = \left(\begin{smallmatrix} P & R^T \\ R & S \end{smallmatrix}\right)$ with the first block row / column corresponding to landmarks and the second corresponding to robot poses. As each landmark is represented by 2 coordinates and each robot pose by 3, P consists of 2×2 - blocks for each landmark, S of 3×3 - blocks for each robot pose and R of 3×2 - blocks coupling landmarks and robot poses. So with n landmarks, m measurements and p robot poses, P is a $2n \times 2n$, R a $3p \times 2n$ and S a $3p \times 3p$ matrix. Throughout the whole section subscripts P_{ll}, R_{li} and S_{ij} refer to the block corresponding to landmark l and robot pose i or j respectively. The matrix A has a very specific sparsity pattern being analyzed in the following (Fig. 2.7):

Therefore denote by subscript oi the i-th odometry measurement measuring robot pose $i+1$

Symbol	Equation	Format	Definition
P_{ll}	(2.52)	2×2	Diagonal block of A corresponding to landmark l
P_{ll}^{i}	(2.50)	2×2	Contribution of the observation of landmark l from pose i to P_{ll}
R_{il}	(2.50)	3×2	Block of R corresponding to landmark l and robot pose i defined from the observation of landmark l from pose i
S_{ii}	(2.53)	3×3	Diagonal block of S corresponding to robot pose i
S_{ii}^{l}	(2.51)	3×3	Contribution of the observation of landmark l from pose i to S_{ii}
$S_{ii}^{\mathcal{L}}$	(2.53)	3×3	Contribution of all landmark observations from pose i to S_{ii}
$S_{ii}^{\mathcal{O}}$	(2.53)	3×3	Contribution of both odometry observations from pose i and $i-1$ to S_{ii}
$P'_{l_1 l_2}$	(2.57)	2×2	Block corresponding to landmark l_1 and l_2 of the information matrix P' of all landmarks without robot poses.
$\mathcal{L}_i$			Landmarks observed from robot pose i
$\mathcal{O}_l$			Robot poses from which landmark l has been observed
$d_{l_1 l_2}$	(2.56)		Number of movements between observation of landmarks l_1 and l_2

Table 2.1: Symbols used in the proof of theorem 2

relative to robot pose i with C_{oi} measurement covariance and J_{oi} measurement Jacobian. Further let $\mathcal{L}_i$ denote the set of landmark measurements taken from robot pose i and conversely $\mathcal{O}_l$ the set of robot poses from which landmark l has been observed. Clearly $l \in \mathcal{L}_i$ holds if and only if $i \in \mathcal{O}_l$. For a landmark $l \in \mathcal{L}_i$ let C_{li} be the covariance of the measurement of landmark l from robot pose i and J_{li} the measurement Jacobian. By (2.25), the information matrix A of all landmarks and all robot poses is the sum of $J_j^T C_j^{-1} J_j$ over all measurements with J_j Jacobian and C_j covariance. With the definitions made above, these measurements can be grouped by the robot pose and separated into odometry and landmark measurements as

$$A = \sum_{j=1}^{m} J_j^T C_j^{-1} J_j = \sum_{i=1}^{p} \left(J_{oi}^T C_{oi}^{-1} J_{oi} + \sum_{l \in \mathcal{L}_i} J_{li}^T C_{li}^{-1} J_{li} \right). \tag{2.46}$$

The Jacobian J_{oi} is sparse having a 3×3 nonzero block J_{oi3} at the columns corresponding to robot pose i and another block J_{oi4} at the columns corresponding to robot pose $i+1$. Similarly, J_{li} has a 2×3 nonzero block J_{li5} at the columns corresponding to robot pose i and a 2×2 block J_{li6} at the columns corresponding to landmark l (Fig. 2.7a, b). Expressions for $J_{oi3}, J_{oi4}, J_{li5}, J_{li6}$ are given in §2.1 by equations (2.16), (2.19), (2.20), but do not matter for discussion.

The structure of the Jacobians can be formally expressed using projection matrices. Let therefore I_i denote the block row of the identity matrix corresponding to robot pose i and I_l the block row corresponding to landmark l and express the Jacobians as

$$J_{oi} = J_{oi3} I_i + J_{oi4} I_{i+1}, \quad J_{li} = J_{li5} I_i + J_{li6} I_l \tag{2.47}$$

$$
\begin{aligned}
A =& \sum_{i=1}^{p} (J_{oi3} I_i + J_{oi4} I_{i+1})^T C_{oi}^{-1} (J_{oi3} I_i + J_{oi4} I_{i+1}) \\
&+ \sum_{i=1}^{p} \sum_{l \in \mathcal{L}_i} (J_{li5} I_i + J_{li6} I_l)^T C_{li}^{-1} (J_{li5} I_i + J_{li6} I_l)
\end{aligned} \tag{2.48}
$$

$$
\begin{aligned}
=& \sum_{i=1}^{p} I_i^T (J_{oi3}^T C_{oi}^{-1} J_{oi3}) I_i + I_{i+1}^T (J_{oi4}^T C_{oi}^{-1} J_{oi3}) I_i \\
&+ \sum_{i=1}^{p} I_i^T (J_{oi3}^T C_{oi}^{-1} J_{oi4}) I_{i+1} + I_{i+1}^T (J_{oi4}^T C_{oi}^{-1} J_{oi4}) I_{i+1} \\
&+ \sum_{i=1}^{p} \sum_{l \in \mathcal{L}_i} I_i^T (J_{li5}^T C_{li}^{-1} J_{li5}) I_i + I_l^T (J_{li6}^T C_{li}^{-1} J_{li5}) I_i \\
&+ \sum_{i=1}^{p} \sum_{l \in \mathcal{L}_i} I_i^T (J_{li5}^T C_{li}^{-1} J_{li6}) I_l + I_l^T (J_{li6}^T C_{li}^{-1} J_{li6}) I_l.
\end{aligned} \tag{2.49}
$$

An expression like $I_i^T(\dots)I_l$ places the 3×2 matrix in parentheses at the row and column corresponding to robot pose i and landmark l. The other combination $I_i^T(\dots)I_j$, $I_l^T(\dots)I_i$ and $I_l^T(\dots)I_l$ act similar. From (2.49) it can be seen that each odometry measurement generates four small blocks at the intersections of the rows and columns corresponding to two successive robot poses (Fig. 2.7a). Correspondingly, each landmark measurement is generating four small blocks at the intersections of the rows and columns corresponding to the robot pose and the landmark (Fig. 2.7b).

The terms in expression (2.49) can be separated into those that belong to P, R and S. From the result it can be seen, that P is block diagonal, S is block tridiagonal and R is sparse with a block being non-zero, if the landmark corresponding to that column has been observed from the robot pose corresponding to that row (Fig. 2.7c):

$$
P = \sum_{i=1,\, l \in \mathcal{L}_i}^{p} I_l^T \underbrace{(J_{li6}^T C_{li}^{-1} J_{li6})}_{P_{ll}^i :=} I_l, \qquad R = \sum_{i=1,\, l \in \mathcal{L}_i}^{p} I_i^T (J_{li5}^T C_{li}^{-1} J_{li6}) I_l, \tag{2.50}
$$

$$
S = \sum_{i=1}^{p} \begin{pmatrix} I_i^T (J_{oi3}^T C_{oi}^{-1} J_{oi3}) I_i + I_{i+1}^T (J_{oi4}^T C_{oi}^{-1} J_{oi3}) I_i \\ + I_i^T (J_{oi3}^T C_{oi}^{-1} J_{oi4}) I_{i+1} + I_{i+1}^T (J_{oi4}^T C_{oi}^{-1} J_{oi4}) I_{i+1} \end{pmatrix} + \sum_{i=1,\, l \in \mathcal{L}_i}^{p} I_i^T \underbrace{(J_{li5}^T C_{li}^{-1} J_{li5})}_{S_{ii}^l} I_i. \tag{2.51}
$$

In the following discussion the block diagonals of P and S will be of great importance, so an explicit formula for the diagonal block S_{ii} corresponding to robot pose i and P_{ll} corresponding to landmark l is derived. This is performed by grouping (2.50) and (2.51) by values of l and i

respectively. The result for P_{ll} is a sum over $\mathcal{O}_l$, the set of robot poses from which landmark l has been observed. Similarly, the result for S_{ii} is a term originating from odometry plus a sum over $\mathcal{L}_i$ the set of landmarks observed from robot pose i:

$$P_{ll} = I_l P I_l^T = \sum_{i \in \mathcal{O}_l} (J_{li6}^T C_{li}^{-1} J_{li6}) = \sum_{i \in \mathcal{O}_l} P_{ll}^i \tag{2.52}$$

$$S_{ii} = I_i S I_i^T = J_{oi3}^T C_{oi}^{-1} J_{oi3} + J_{o(i-1)2}^T C_{o(i-1)}^{-1} J_{o(i-1)2} + \sum_{l \in \mathcal{L}_i} J_{li5}^T C_{li}^{-1} J_{li5} \tag{2.53}$$

$$= \underbrace{J_{oi3}^T C_{oi}^{-1} J_{oi3} + J_{o(i-1)2}^T C_{o(i-1)}^{-1} J_{o(i-1)2}}_{S_{ii}^{\mathcal{O}} :=} + \underbrace{\sum_{l \in \mathcal{L}_i} S_{ii}^l}_{S_{ii}^{\mathcal{L}} :=} . \tag{2.54}$$

It can be observed that one part ($S_{ii}^{\mathcal{O}}$) of S_{ii} originates from odometry measurements and another part ($S_{ii}^{\mathcal{L}}$) originates from landmark observations. In the latter part matrices S_{ii}^l from all landmark observations made from that robot pose accumulate. Nevertheless, S_{ii} and $S_{ii}^{\mathcal{L}}$ are bounded, since the number of landmark observations from a certain robot pose depends on the sensor / landmark trait and will not grow when the map gets larger. As for the diagonal block P_{ll} corresponding to landmark l this is different. Here matrices from all observations of this landmark accumulate. Since the same landmark may be observed over and over again, P_{ll} will usually grow linear with time. Each block R_{il} of R is affected only by a single measurement of landmark l from robot pose i. So it is bounded and 0, if the landmark has not been observed from there. Table 2.1 gives an overview of the different parts defined above.

Schur Complement

As discussed above the information matrix A is sparse. However, its dimension depends on the number of robot poses. Thus, it grows even when moving through an area visited before and cannot be used for representation in a SLAM algorithm. So the robot poses must be eliminated via Schur complement (see §3.3 lemma 8 for a proof). The resulting information matrix for the landmarks alone is

$$P' := P - R^T S^{-1} R \tag{2.55}$$

and the inverse of the corresponding covariance matrix maintained by EKF. It is not sparse, since S^{-1} is dense. The purpose of this section is to prove that it is approximately sparse and in particular, that an entry $P'_{l_1 l_2}$ decays exponentially with the distance $d_{l_1 l_2}$ between observation of landmarks l_1 and l_2. Formally the distance is defined as

$$d_{l_1 l_2} := \min\big\{ |i - j| \;\big|\; i \in \mathcal{O}_{l_1}, j \in \mathcal{O}_{l_2} \big\}, \tag{2.56}$$

the number of robot movements between observation of l_1 and l_2. Let $l_1 \neq l_2$ and consider

$$P'_{l_1 l_2} = -\sum_{i,j=1}^{p} R^T_{il_1} (S^{-1})_{ij} R_{jl_2} = -\sum_{i \in \mathcal{O}_{l_1}} \sum_{j \in \mathcal{O}_{l_2}} R^T_{il_1} (S^{-1})_{ij} R_{jl_2}. \tag{2.57}$$

This equation is of high importance, because since R_{il_1} and R_{jl_2} are bounded, asymptotically $P'_{l_1 l_2}$ behaves like block $(S^{-1})_{ij}$ of S^{-1} corresponding to the robot poses, when l_1 and l_2 have been observed. Block $(S^{-1})_{ij}$ decays exponentially with the distance $|i-j|$ to the diagonal as will be derived later. By definition $|i-j| \geq d_{l_1 l_2}$ for all involved $i \in \mathcal{O}_{l_1}$ and $j \in \mathcal{O}_{l_2}$, so in the end $P'_{l_1 l_2}$ can be shown to decay exponentially with the distance $d_{l_1 l_2}$.

Exponential Decay of Off-Diagonal Entries

This subsection proves that an off-diagonal entry $(S^{-1})_{ij}$ decays exponentially with the distance $|i-j|$ to the diagonal. The result is then used to derive that $P'_{l_1 l_2}$ decays exponentially with the distance $d_{l_1 l_2}$ between observation of l_1 and l_2. Thereby the approximate sparsity of the SLAM information matrix P' is proven. The rate of decay depends on the ratio between $S^{\mathcal{O}}_{ii}$ and $S^{\mathcal{L}}_{ii}$, i.e. between the information gained from odometry and landmark observations:

Definition 1 (Characteristic Parameter). *For a sequence of odometry and landmark observations the characteristic parameters are:*

$$\omega := \max\left\{\omega | S^{\mathcal{L}}_{ii} \geq \omega S^{\mathcal{O}}_{ii} \; \forall i\right\} \quad \eta := \max_{i,l} \|P^i_{ll}\| \quad \rho := \min_{i,\, l \in \mathcal{L}_i} \|P^i_{ll} - R^T_{il} (S^{\mathcal{L}}_{ii})^{-1} R_{il}\| \tag{2.58}$$

Intuitively, this definition means: a) In each robot pose the information gained from landmarks is at least ω times the information transported from the last pose by odometry. b) The information gained from a single landmark measurement assumed the robot pose was known is at most η. c) The information gained about a landmark from all observations from a certain unknown robot pose assumed that all other landmarks were known is a least ρ.

Nevertheless, b) and c) need some explanation: P^i_{ll} is a submatrix of the information matrix for a single landmark observation of l from pose i (Fig. 2.7b). Thus, as discussed before, it represents the information known about landmark l from that measurement if all other random variables, in this case the robot pose, were known. Similarly, $\begin{pmatrix} P^i_{ll} & R_{il} \\ R^T_{il} & S^{\mathcal{L}}_{ii} \end{pmatrix}$ is a submatrix of the information matrix of all landmark observations from pose i. Therefore it represents the information there were known about landmark l and robot pose i if all other random variables, in this case the other landmarks, were known. Thus, the Schur complement $P^i_{ll} - R^T_{il}(S^{\mathcal{L}}_{ii})^{-1} R_{il}$ represents the same information without information about the robot pose. So in the end the term describes the information, if all other landmarks were known but the robot pose was unknown. The parameter ρ gives a lower bound on this information.

From the explanation above may be seen that[3]

$$P_{ll}^{i} \geq R_{il}^{T}(S_{ii}^{\mathcal{L}})^{-1}R_{il}. \tag{2.59}$$

All three parameters ω, η, ρ depend on the sensor / landmark / environment characteristic and do not change when the map size grows. So they may be considered as being constant

$$\omega = O(1), \qquad \eta = O(1), \qquad \rho = O(1). \tag{2.60}$$

The central argument of the overall proof uses a theorem by Demko, Moss and Smith [DMS84, theorem 2.4] that provides an exponentially decaying bound for the entries of the inverse of a symmetric positive definite (SPD) band matrix S. The bounds refer to a single entry of S denoted by $S_{\#ij} \in \mathbb{R}$ to avoid confusion with the 3×3 matrix block $S_{ij} \in \mathbb{R}^{3\times 3}$ corresponding to robot pose i and j. The bound depends on the norm $\|S\|$ and condition number $\mathrm{cond}(S)$ of S. The matrix norms refer to the usual spectral or 2-norm $\|S\| := \max_{|v|=1} |Sv|$ being equal to the largest eigenvalue of S (largest singular value for a non-symmetric matrix). The derived condition number is equal to the ratio between largest and smallest eigenvalue.

Theorem 1 (Demko, Moth, Smith [DMS84]). *Let S be an SPD w-banded matrix. Then for entry $(S^{-1})_{\#ij}$ of S^{-1} holds*

$$|(S^{-1})_{\#ij}| \leq \alpha\lambda^{|i-j|}, \quad \textit{with } \lambda := \left(\frac{\sqrt{\mathrm{cond}(S)}-1}{\sqrt{\mathrm{cond}(S)}+1}\right)^{\frac{2}{w}} \tag{2.61}$$

$$\textit{and } \alpha = \|S^{-1}\| \max\left\{1, \frac{\left(1+\sqrt{\mathrm{cond}(S)}\right)^{2}}{2\,\mathrm{cond}(S)}\right\} \leq 2\|S^{-1}\|. \tag{2.62}$$

Some technical lemmas are needed. Their proof is provided in appendix B. For lemma 1 an even stronger version has been proven in [Tee00].

Lemma 1. *Let S be a block diagonal SPD matrix with block bandwidth w. Then the norm $\|S\|$ is at most $2w-1$ times the norm of any diagonal block ($= \max_i \|S_{ii}\|$).*

Lemma 2. *For all $\omega \geq 0$ the following inequality holds:*

$$\frac{\sqrt{1+\frac{3}{\omega}}+1}{\sqrt{1+\frac{3}{\omega}}-1} \geq 1+\frac{4}{3}\omega \tag{2.63}$$

[3] In the usual positive definite sense: $A \leq B \iff \forall x : x^{T}Ax \leq x^{T}Bx$ (see appendix §A.1)

From theorem 1, lemma 1 and 2 follows:

Lemma 3. *Let S be a block tridiagonal SPD matrix with 3×3 blocks, smallest eigenvalue $\lambda_{\min} \geq 1$ and largest eigenvalue $\lambda_{\max} \leq 1 + \frac{3}{\omega}$. Then for $i \neq j$ the norm of block $(S^{-1})_{ij}$ of the inverse is at most $\|(S^{-1})_{ij}\| \leq 6\left(1 + \frac{4}{3}\omega\right)^{1-|i-j|}$.*

Proof. S has a bandwidth of $w = 6$. Since the condition number of S is $\mathrm{cond}(A) = \frac{\lambda_{\max}}{\lambda_{\min}} \leq 1 + \frac{3}{\omega}$, the parameter λ central to theorem 1 is

$$\lambda = \left(\frac{\sqrt{\mathrm{cond}(S)} - 1}{\sqrt{\mathrm{cond}(S)} + 1}\right)^{\frac{2}{m}} \leq \left(\frac{\sqrt{1 + \frac{3}{\omega}} + 1}{\sqrt{1 + \frac{3}{\omega}} - 1}\right)^{-\frac{1}{3}} \overset{\text{lemma 2}}{\leq} \left(1 + \frac{4}{3}\omega\right)^{-\frac{1}{3}}. \tag{2.64}$$

The block of S^{-1} corresponding to robot pose i and j has column indices $[3i - 2 \dots 3i]$ and row indices $[3j - 2 \dots 3j]$. Thus the minimal distance to the diagonal for any entry of that block is

$$\min\big|\,[3i - 2 \dots 3i] - [3j - 2 \dots 3j]\,\big| = \min\big|\,[(3i - 2) - 3j \dots 3i - (3j - 2)]\,\big| \tag{2.65}$$

$$= \min\big|\,[3(i - j) - 2 \dots 3(i - j) + 2)]\,\big| \overset{i \neq j}{=} \min\,[3|i - j| - 2 \dots 3|i - j| + 2] \tag{2.66}$$

$$= 3|i - j| - 2. \tag{2.67}$$

So by theorem 1 the size of each entry k, l of the 3×3 block $(S^{-1})_{ij}$ is at most

$$|((S^{-1})_{ij})_{\#kl}| \leq 2\|S^{-1}\| \left(1 + \frac{4}{3}\omega\right)^{-\frac{1}{3}(3|i-j|-2)} \leq 2\left(1 + \frac{4}{3}\omega\right)^{1-|i-j|}. \tag{2.68}$$

Since $(S^{-1})_{ij}$ is a 3×3 block, its norm is at most 3 times as large as the largest entry (appendix. A.1, (A.24)):

$$\|(S^{-1})_{ij}\| \leq 3 \max_{k,l=1}^{3}((S^{-1})_{ij})_{\#kl} \leq 6\left(1 + \frac{4}{3}\omega\right)^{1-|i-j|} \tag{2.69}$$

□

The next step is to derive an exponentially decaying bound for the off-diagonal entries of S^{-1}. For technical reasons in the proof of theorem 2 the matrix to be considered is $R^T S^{-1} R$ not S^{-1}. So instead of deriving a bound for $(S^{-1})_{ij}$, directly a bound for $R_{il_1}^T (S^{-1})_{ij} R_{jl_2}$ is given.

Lemma 4. *For a sequence of odometry and landmark observations with parameter ω, η, ρ, and all robot poses i, j and all landmarks $l_1 \neq l_2$ the following bound holds:*

$$\|R_{l_1 i}^T (S^{-1})_{ij} R_{jl_2}\| \leq 6\eta \left(1 + \frac{4}{3}\omega\right)^{1-|i-j|}. \tag{2.70}$$

Proof. Let $S^{\mathcal{L}} := \operatorname{diag}_i(S^{\mathcal{L}}_{ii})$ be the part of S that originates from the landmark measurements and $S^{\mathcal{O}} := S - S^{\mathcal{L}}$ be the remaining part originating from odometry. The block diagonal of $S^{\mathcal{O}}$ is $\operatorname{diag}_i(S^{\mathcal{O}}_{ii})$, but $S^{\mathcal{O}}$ itself is block tridiagonal (compare figure 2.7).

The first step is to scale $S^{\mathcal{O}}$, so matrices of comparable norm appear on the block diagonal. Let therefore $L_i L_i^T = S^{\mathcal{L}}_{ii}$ be a Cholesky decomposition of $S^{\mathcal{L}}_{ii}$ and define the inverse of $L := \operatorname{diag}_i(L_i)$ as scale matrix. This way the scaled matrix $L^{-1}S^{\mathcal{L}}L^{-1T}$ is the identity matrix and the scaled matrix $L^{-1}S^{\mathcal{O}}L^{-1T}$ is normalized relative to $S^{\mathcal{L}}$. So it is possible to bound it by ω:

Matrix $S^{\mathcal{O}}$ is block tridiagonal and L is block diagonal. Thus, $L^{-1}S^{\mathcal{O}}L^{-1T}$ is block tridiagonal, too. Further $S^{\mathcal{L}}_{ii} \geq \omega S^{\mathcal{O}}_{ii}$ by definition 1. So for each diagonal block it follows that

$$(L^{-1}S^{\mathcal{O}}L^{-1T})_{ii} = L_i^{-1}S^{\mathcal{O}}_{ii}L_i^{-1T} \overset{\text{definition 1}}{\leq} \frac{1}{\omega}L_i^{-1}S^{\mathcal{L}}_{ii}L_i^{-1T} = \frac{1}{\omega}L_i^{-1}L_iL_i^TL_i^{-1T} = \frac{1}{\omega}I. \tag{2.71}$$

The matrix $L^{-1}S^{\mathcal{O}}L^{-1T}$ is tridiagonal, so lemma 1 can be applied with $w = 2$ and the eigenvalue $\lambda_{\max}\left(L^{-1}S^{\mathcal{O}}L^{-1T}\right)$ is at most $\frac{3}{\omega}$. By construction $L^{-1}S^{\mathcal{L}}L^{-1T} = I$, so the eigenvalues of $L^{-1}SL^{-1T} = L^{-1}(S^{\mathcal{L}} + S^{\mathcal{O}})L^{-1T}$ lie in the interval $\left[1 \ldots 1 + \frac{3}{\omega}\right]$. It follows from lemma 3 that

$$\|L_i^T(S^{-1})_{ij}L_j\| = \|((L^{-1}SL^{-1T})^{-1})_{ij}\| \overset{\text{lemma 3}}{\leq} 6\left(1 + \frac{4}{3}\omega\right)^{1-|i-j|} \tag{2.72}$$

and

$$\|R_{il_1}^T(S^{-1})_{ij}R_{jl_2}\| = \|R_{il_1}^TL_i^{-1T}L_i^T(S^{-1})_{ij}L_jL_j^{-1}R_{jl_2}\| \tag{2.73}$$

$$\leq \|R_{il_1}^TL_i^{-1T}\| \, \|L_i^T(S^{-1})_{ij}L_j\| \, \|L_j^{-1}R_{jl_2}\| \tag{2.74}$$

$$\leq \|R_{il_1}^TL_i^{-1T}\| \, 6\left(1 + \frac{4}{3}\omega\right)^{1-|i-j|} \|L_j^{-1}R_{jl_2}\|. \tag{2.75}$$

Now the next step is to find a bound for $\|L_j^{-1}R_{jl_2}\|$ (which also holds for $\|R_{il_1}^TL_i^{-1T}\|$):

$$\|L_j^{-1}R_{jl_2}\|^2 = \|(L_j^{-1}R_{jl_2})^T(L_j^{-1}R_{jl_2})\| = \|R_{jl_2}^TL_j^{-1T}L_j^{-1}R_{jl_2}\| \tag{2.76}$$

$$= \|R_{jl_2}^T(S^{\mathcal{L}}_{jj})^{-1}R_{jl_2}\| \overset{(2.59)}{\leq} \|P^j_{l_2l_2}\| \overset{\text{definition 1}}{\leq} \eta. \tag{2.77}$$

It follows $\|L_j^{-1}R_{jl_2}\| \leq \sqrt{\eta}$, which is substituted into (2.75) yielding

$$\|R_{il_1}^T(S^{-1})_{ij}R_{jl_2}\| \leq 6\eta\left(1 + \frac{4}{3}\omega\right)^{1-|i-j|}. \tag{2.78}$$

□

To prove approximate sparsity of P', the norm of its off-diagonal blocks $P'_{l_1l_2}$ must be bounded relative to the corresponding diagonal blocks $\|P'_{l_1l_1}\|$ and $\|P'_{l_2l_2}\|$. Therefore, a lower bound for a diagonal block $\|P'_{ll}\|$ is derived in the following lemma:

Lemma 5. *For a sequence of odometry and landmark observations with parameter ω, η, ρ, and all landmarks l it holds that*

$$\|P'_{ll}\| \geq \rho \, |\mathcal{O}_l|. \tag{2.79}$$

Proof. Since $S \geq S^{\mathcal{L}}$ it follows that $S^{-1} \leq (S^{\mathcal{L}})^{-1}$ and

$$P' \overset{(2.55)}{=} P - R^T S^{-1} R \geq P - R^T (S^{\mathcal{L}})^{-1} R \tag{2.80}$$

$$\Rightarrow \qquad P'_{ll} \geq P_{ll} - \sum_{i,j \in \mathcal{O}_l} R_{il}^T ((S^{\mathcal{L}})^{-1})_{ij} R_{jl}. \tag{2.81}$$

$S^{\mathcal{L}}$ is block diagonal, so $(S^{\mathcal{L}})^{-1} = \operatorname{diag}_i((S^{\mathcal{L}}_{ii})^{-1})$ and $((S^{\mathcal{L}})^{-1})_{ij} = 0$ for $i \neq j$

$$P'_{ll} = P_{ll} - \sum_{i \in \mathcal{O}_l} R_{il}^T (S^{\mathcal{L}}_{ii})^{-1} R_{il} = \sum_{i \in \mathcal{O}_l} P_{ll}^i - R_{il}^T (S^{\mathcal{L}}_{ii})^{-1} R_{il} \overset{\text{definition 1}}{\geq} \rho \, |\mathcal{O}_l|. \tag{2.82}$$

□

The final step is to substitute the bound of lemma 4 into (2.57) to derive an overall bound. A sum of different powers of $(1 + \frac{4}{3}\omega)^{-1}$ appears. The exact exponents depend on the robot poses the landmark has been observed from, so it is difficult to find a closed expression. However all exponents are at least $d_{l_1 l_2}$, which is defined as the minimal distance (in number of robot poses) between observation of l_1 and l_2. Using this property the sum can be bounded by the following lemma (proof in appendix B):

Lemma 6. *Let $0 \leq \gamma < 1$ and $\mathcal{A}, \mathcal{B} \subset \mathbb{N}$ be two sets of natural numbers with a minimal distance d between elements of $\mathcal{A}$ and $\mathcal{B}$: $d \leq |i - j| \; \forall i \in \mathcal{A}, j \in \mathcal{B}$. Then the following inequality holds:*

$$\sum_{i \in \mathcal{A}, \, j \in \mathcal{B}} \gamma^{|i-j|} \leq 2 \frac{\gamma^d}{1 - \gamma} \min\{|\mathcal{A}|, |\mathcal{B}|\} \tag{2.83}$$

Theorem 2 (Information Matrix Sparsity). *Consider a sequence of odometry and landmark observations with parameter ω, η, ρ. Then the resulting SLAM information matrix of all landmarks P' is approximately sparse. The off-diagonal block $P'_{l_1 l_2}$ corresponding to two landmarks $l_1 \neq l_2$ decays exponentially with the smallest number of steps $d_{l_1 l_2}$ traveled between observation of l_1 and l_2.*

$$\frac{\|P'_{l_1 l_2}\|}{\min\left\{\|P'_{l_1 l_1}\|, \|P'_{l_2 l_2}\|\right\}} \leq \frac{\eta}{\rho}\left(24 + 16\omega + \frac{9}{\omega}\right)\left(1 + \frac{4}{3}\omega\right)^{-d_{l_1 l_2}} = O\left(\left(1 + \frac{4}{3}\omega\right)^{-d_{l_1 l_2}}\right)$$

Proof. By equation (2.57) $P'_{l_1 l_2}$ is a sum over different robot poses i, j. Each term in the sum can be bounded by lemma 4 yielding

$$\|P'_{l_1 l_2}\| \overset{(2.57)}{=} \left\| -\sum_{i \in \mathcal{O}_{l_1}} \sum_{j \in \mathcal{O}_{l_2}} R_{il_1}^T (S^{-1})_{ij} R_{jl_2} \right\| \tag{2.84}$$

$$\leq \sum_{i \in \mathcal{O}_{l_1}} \sum_{j \in \mathcal{O}_{l_2}} \left\| R_{il_1}^T (S^{-1})_{ij} R_{jl_2} \right\| \tag{2.85}$$

$$\overset{\text{lemma 4}}{\leq} 6\eta \sum_{i \in \mathcal{O}_{l_1}} \sum_{j \in \mathcal{O}_{l_2}} \left(1 + \frac{4}{3}\omega\right)^{1-|i-j|} \tag{2.86}$$

$$= 6\eta \left(1 + \frac{4}{3}\omega\right) \sum_{i \in \mathcal{O}_{l_1}} \sum_{j \in \mathcal{O}_{l_2}} \left(1 + \frac{4}{3}\omega\right)^{-|i-j|}. \tag{2.87}$$

The sum can be bounded by lemma 6 with $\mathcal{A} := \mathcal{O}_{l_1}, \mathcal{B} := \mathcal{O}_{l_2}, \gamma := \left(1 + \frac{4}{3}\omega\right)^{-1}$ and $d := d_{l_1 l_2}$:

$$\overset{\text{lemma 6}}{\leq} 6\eta \left(1 + \frac{4}{3}\omega\right) \left(\frac{2 \min\{|\mathcal{O}_{l_1}|, |\mathcal{O}_{l_2}|\}}{1 - \left(1 + \frac{4}{3}\omega\right)^{-1}} \left(1 + \frac{4}{3}\omega\right)^{-d_{l_1 l_2}} \right) \tag{2.88}$$

$$= 12\eta \left(1 + \frac{4}{3}\omega\right) \left(1 + \frac{3}{4\omega}\right) \min\{|\mathcal{O}_{l_1}|, |\mathcal{O}_{l_2}|\} \left(1 + \frac{4}{3}\omega\right)^{-d_{l_1 l_2}} \tag{2.89}$$

$$= \eta \min\{|\mathcal{O}_{l_1}|, |\mathcal{O}_{l_2}|\} \left(24 + 16\omega + \frac{9}{\omega}\right) \left(1 + \frac{4}{3}\omega\right)^{-d_{l_1 l_2}}. \tag{2.90}$$

By lemma 5 it holds that

$$\min\left\{\|P'_{l_1 l_1}\|, \|P'_{l_1 l_1}\|\right\} \geq \rho \min\{|\mathcal{O}_{l_1}|, |\mathcal{O}_{l_2}|\} \tag{2.91}$$

$$\Rightarrow \quad \frac{\|P'_{l_1 l_2}\|}{\min\left\{\|P'_{l_1 l_1}\|, \|P'_{l_1 l_1}\|\right\}} \leq \frac{\eta}{\rho} \left(24 + 16\omega + \frac{9}{\omega}\right) \left(1 + \frac{4}{3}\omega\right)^{-d_{l_1 l_2}}. \tag{2.92}$$

Even with an asymptotically increasing map size, ω, η and ρ remain constant, since they depend on the quality of the measurements such as sensor noise, typical distance to landmarks, typical number of landmarks. They do not depend on how many measurements were made. So the final asymptotic formula is

$$\frac{\|P'_{l_1 l_2}\|}{\min\left\{\|P'_{l_1 l_1}\|, \|P'_{l_1 l_1}\|\right\}} \leq O\left(\left(1 + \frac{4}{3}\omega\right)^{-d_{l_1 l_2}}\right). \tag{2.93}$$

□

The algorithmic approach taken in this thesis is to divide the map hierarchically into small parts and represent each part at each level of hierarchy by a small matrix. Implicitly, the hierarchy represents a sparse information matrix as an approximation to P'. With this approach small off-diagonal entries have to be neglected. Theorem 2 provides the certainty that not too much information is being lost thereby.

2.12 Local vs. Global Uncertainty

It can be observed that there is a qualitative difference between local and global structures of SLAM, i.e. between relations of neighboring and of distant landmarks. Roughly speaking, the local uncertainty is small but complex and depends on actual observations, whereas the global uncertainty is large, rather simple and dominated by the map's geometry. This is a consequence of theorem 2 and will be clarified in the following:

The measurements themselves define independent relations between landmarks and robot poses. For most sensors the uncertainty depends on the distance (laser scanner, stereo vision) or is even infinite in one dimension (mono vision). The information provided by the set of landmark observations from a single robot pose contains a highly coupled uncertainty originating from the uncertainty of the robot pose. From successive robot poses usually similar but different sets of landmarks are observed. So the parts of the information corresponding to different robot poses are highly coupled, but are always coupling different sets of landmarks. As a result the overall information on a local scale is also highly coupled and very complex. This corresponds to the coupling entries $P'_{l_1 l_2}$ in the information matrix being high for landmarks l_1, l_2 that are near to each other.

On a global scale the structure of the information is governed by theorem 2. The coupling entry $P'_{l_1 l_2}$ between distant landmarks is very low. So the uncertainty of the relation between them is approximately the composition of local uncertainties along the path from l_1 to l_2:

Consider the information matrix resulting from the integration of several local bits of information, for instance, the distance of each landmark to any other landmark nearby. By (2.25), this matrix is the sum of the information matrices for each bit of information. Each of them has non-zero coupling entries only for the landmarks involved. So the overall information matrix is sparse with all coupling entries being zero, except those of adjacent landmarks.

Thus, as local information corresponds to a sparse information matrix, an approximately sparse information matrix corresponds to information that can approximately be viewed as being the integration of local information. To see the uncertainty structure of such information, imagine

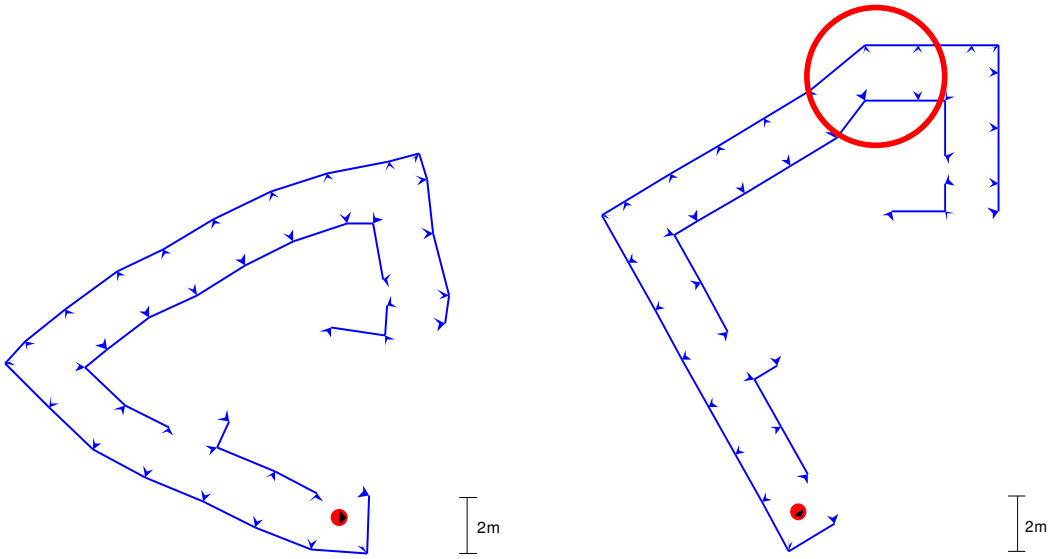


Figure 2.8: Global uncertainty generated by the uncertainty in a local region. Random outcome of a map estimate with (a) measurement uncertainty everywhere (b) measurement uncertainty only in the encircled region.

that measurement noise applies only to the measurements in a small region (Fig. 2.8b). The noise corrupts the robot position and orientation estimate when the robot moves through the region. Thus the part of the map built afterward is affected by an uncertain translation and rotation relative to the part before. The effect of the rotation around the region grows linear with the distance to the region. So globally it is much larger than the uncertain translation. The magnitude of the rotation angle depends on the local uncertainty in the region, namely on how much orientation error is being accumulated while moving through the region. Its structure, however, solely depends on the distances of the different landmarks to the region, i.e. on the map's geometry.

If all measurements are uncertain, the global effect is approximately the sum of an uncertain rotation for each local region. The resulting uncertainty structure can best be described as an *uncertain bending* of the map (Fig. 2.8a). Compared to local uncertainty it is much larger, but simpler because the maps geometry is dominating it's structure.

The main target of SLAM is modeling global uncertainty. But often representation of local uncertainty is necessary to support landmark identification or allow task planning based on objects represented in the map. The approach of decomposing the map into regions and using a

small information matrix for each region ideally suits the SLAM uncertainty structure: Local uncertainty is precisely captured in the small matrix representing a local region, whereas the global uncertainty structure does not need to be represented explicitly, since it is in very good approximation nothing more than the composition of local uncertainties.

2.13 Requirements for an Ideal Solution

In this section some properties, which an ideal SLAM solution should have, are postulated. They are based on an intentionally naive view of the problem blinding out its apparent difficulty, but asking how mapping should work when based on a common sense understanding of maps. These properties were proposed prior to the development of the algorithm [FH01]. There is a discrepancy between these intuitive requirements and the performance of most established algorithms as will be discussed in the following. This made the author presume that some absolutely specific property, which makes SLAM different from general estimation problems, has to be exploited to meet the requirements proposed. For the algorithm presented in this thesis the specific property is that typical buildings can be hierarchically divided into parts with each part having only few connections with the remainder of the building.

Quality, Storage Space and Computation Time

The requirements refer to those three important criteria for a SLAM algorithm discussed in §1.1: *Quality*, *Storage space* and *Computation time*

> **(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.*

This postulate is quite general saying all that can be known from the measurements should at least roughly be represented in the map. Consistently approximating some relations for the sake of efficiency is acceptable to the extent, relations get slightly less precise, but without losing all or almost all information on certain relations. As many relations can be known precisely from the measurements, not representing one would violate the principle stated and hence (R1) implies the ability to represent relativity. As explained above, representing relativity comprises to be able to close large loops achieving topologically consistent maps.

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

The soundness of this postulate can be seen from the following example: Imagine a building consisting of two parts, A and B, being connected by a few corridors. Then the map of the whole building consists of the map of both parts plus some information concerning the connections and should thus have a size only slightly larger than the size of an A map plus the size of a B map.

It is worth noting that simply storing all measurements will not meet (R2), since the storage space is proportional to the number of measurements m not to the number of landmarks n. Thus, the map's size would grow during motion even when repeatedly travelling through the same area.

(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 ($O(n)$).*

Establishing this postulate is more difficult than the preceding one:

Let us assume that the same setting like above holds with a measurement made in A. At first the measurement has to be incorporated into the map of A, taking into account the known effect of A on the connection between A and B. Then, the effect of these connections onto B must be computed. This is equivalent to incorporating *several* measurements concerning the connections into the map of B. However computation can be deferred until the robot actually enters B sharing the computational cost with all other measurements having generated effects on the connections that have to be integrated upto then. As the number of landmarks in the connections is small this should not take more time per measurement than incorporating the original measurement into the map of A, at least for large maps.

So the total cost for integrating a measurement into a map of A and B should not be larger than the cost of integration into A plus the cost of integration it into B, thus being linear in the number of landmarks.

(R1) states the map shall represent nearly all information contained in the measurements, thus binding the map to reality. The other postulates regard efficiency, requiring linear space and time consumption. The most important postulate from a practical point of view is (R3) limiting the amount of time spent on each measurement.

Formalization of Map Quality

Requirement (R2) and (R3) concerning storage space and computation time refer to criteria which are canonically applied to any algorithm. However with respect to map quality requirement (R1) must still be formalized appropriately.

Definition 2 (Aspect). *An* aspect *of a map is a function f mapping the landmark positions to a real number $f(x)$*

$$f : \mathbb{R}^{2n} \longrightarrow \mathbb{R}. \tag{2.94}$$

Examples for aspects of a map are: a landmark's $x-$ or $y-$ coordinate, the distance between two landmarks, the angle between three landmarks or any linear combination of landmark coordinates. Considering the SLAM uncertainty structure ("Certainty of Relations despite Uncertainty of Positions") a relation consequently is an aspect of the map being invariant under rigid body transformation of the whole map

$$f_{\text{rel}}(\text{Rot}_\alpha\, x + \text{Trans}_d) = f_{\text{rel}}(x) \quad \forall \alpha, d, \tag{2.95}$$

where Rot_α is a rotation matrix, rotating the whole map by α and Trans_d is a vector translating the whole map by d.

The uncertainty of an aspect f of a map estimate $\hat{x}$ is its standard deviation $\sqrt{\text{cov}(f(\hat{x}))}$. In terms of (R1) the "minimal uncertainty, that could be theoretically derived from the measurements" is the corresponding standard deviation of the optimal maximum likelihood estimate $\sqrt{\text{cov}(f(\hat{x}_{\text{ML}}))}$. The ratio between those uncertainties indicates how much error is induced by the estimation algorithm and how much error is caused by the sensors. So the *relative uncertainty* of an aspect f of the map

$$\text{ruc}(f) := \frac{\sqrt{\text{cov}(f(\hat{x}))}}{\sqrt{\text{cov}(f(\hat{x}_{\text{ML}}))}} \tag{2.96}$$

indicates the quality of the estimation algorithm for a particular map and aspect. It can be computed from $C := \text{cov}(\hat{x})$, $C_{\text{ML}} := \text{cov}(\hat{x}_{\text{ML}})$ and g the gradient of f as

$$\text{ruc}(f) = \frac{\sqrt{\text{cov}(f(\hat{x}))}}{\sqrt{\text{cov}(f(\hat{x}_{\text{ML}}))}} = \sqrt{\frac{\text{cov}(f(\hat{x}))}{\text{cov}(f(\hat{x}_{\text{ML}}))}} \approx \underbrace{\sqrt{\frac{g^T C g}{g^T C_{\text{ML}} g}}}_{\text{ruc}(g)}. \tag{2.97}$$

The last equation is the usual first order approximation for propagating covariances through functions with g being the gradient of f. The term "any aspect of the map" in (R1) formally

stands for "any function f". By virtue of (2.97) this can be replaced by a conceptually much more convenient expression involving "any vector g". For any given g the expression $\mathrm{ruc}(g)$ can be computed from the covariances C and C_{ML}. To systematically characterize the values for different choices of g, the so called generalized eigenvalues λ_i and eigenvectors v_i defined by

$$Cv_i = \lambda_i C_{\mathrm{ML}} v_i \tag{2.98}$$

are useful [HJ90]. They correspond to independent directions both with respect to C and C_{ML}. Their properties are

$$v_i^T C v_i = \lambda_i, \quad v_i^T C_{\mathrm{ML}} v_i = 1, \quad v_i^T C v_j = 0 \; \forall i \neq j \quad \text{and} \quad v_i^T C_{\mathrm{ML}} v_j = 0 \; \forall i \neq j. \tag{2.99}$$

If g happens to be the i-th eigenvector v_i, the relative error $\mathrm{ruc}(g)$ in this aspect of the map may easily be determined as the square root of the i-th eigenvalue:

$$\mathrm{ruc}(g) = \mathrm{ruc}(v_i) = \sqrt{\frac{v_i^T C v_i}{v_i^T C_{\mathrm{ML}} v_i}} = \sqrt{\frac{\lambda_i}{1}} = \sqrt{\lambda_i} \tag{2.100}$$

For an arbitrary aspect g, $\mathrm{ruc}(g)$ is the square root of a convex combination of the different eigenvalues. The weights of the convex combination are just the square of the coefficients μ_i used in expressing g as a linear combination $\sum_i \mu_i v_i$ of the eigenvectors.

$$\sum_i \mu_i v_i := g \tag{2.101}$$

$$\mathrm{ruc}(g) = \mathrm{ruc}\left(\sum_i \mu_i v_i\right) = \sqrt{\frac{\left(\sum_i \mu_i v_i\right)^T C \left(\sum_i \mu_i v_i\right)}{\left(\sum_i \mu_i v_i\right)^T C_{\mathrm{ML}} \left(\sum_i \mu_i v_i\right)}} \tag{2.102}$$

$$= \sqrt{\frac{\sum_i \mu_i^2 (v_i^T C v_i)}{\sum_i \mu_i^2 (v_i^T C_{\mathrm{ML}} v_i)}} = \sqrt{\frac{\sum_i \mu_i^2 \lambda_i}{\sum_i \mu_i^2}} \tag{2.103}$$

So the generalized eigenvalues of C and C_{ML} characterize the relative error compared to the optimal maximum likelihood solution. Each eigenvalue corresponds to an independent aspect of the map in which the relative error is just the square root of the corresponding eigenvalue. Especially the square root of the largest eigenvalue bounds the maximum relative error in any aspect of the map. So in order to meet requirement (R1), formally the largest eigenvalue must be a small constant $O(1)$.

It appears to be very hard to derive an analytic expression for this eigenvalue for any practical algorithm. However, it can be determined for a particular map by Monte Carlo simulation. For the algorithm presented in this thesis, this evaluation is performed by simulation experiments in §5 and by real experiments in §6.

2.14 State of the Art

Basically there were three stages of SLAM development:

Mathematical Formulation

From the first works on map building in the mid-eighties to the early nineties, the special structure of SLAM uncertainty had not been fully identified. In general the used uncertainty representations treated different parts of the map as stochastically independent. This assumption makes it possible to devise unquestionable efficient algorithms but cannot back-propagate errors and thus cannot close a loop. So these approaches are limited to relatively small environments (to give a figure: ≈ 15m).

In 1985 Moravec and Elfes [ME85, Elf89] proposed so called *evidence grids* as a representation for a map consisting of a regular grid of square cells with fixed size (typically ≈ 5cm). Each cell stores the evidence that there is an obstacle in the cell as a real number between 0 and 1. When integrating a measurement all cells in the sensors field of view are updated: If the sensor reports an obstacle in a certain direction and distance, the evidence in the corresponding cell is increased and the evidence in all cells with same direction and smaller distance is decreased. This approach is absolutely efficient and suits well to process data from noisy ultrasonic sensors with low resolution. Since ultrasonic sensors were the most prominent type of sensors at this time, this approach has been utilized and extended by many researchers [ME88, Zel91, SC94, PNDW96, YL97]. The major drawback is that the uncertainty of the robot pose cannot be incorporated because this would blur the map and make it useless.

A different approach was proposed by Brooks [Bro85]: His idea was to achieve a structured representation and build the map as a collection of objects or landmarks and a graph of uncertain relations between them [CL85, CS86, DW88, Fau89]. The uncertainty of each relation is described as Gaussian distribution. Two relations between the same pair of landmarks can be integrated into a single relation and consecutive relations can be composed analytically [Fau89]. To derive information from the graph relations along a path in the graph are composed and information derived from different paths are integrated. Much later, Frese et al. as well as Schlegel and Kämpke re-adpoted this idea using sets instead of distributions as uncertainty representation [FHBH00, SK02]. These approaches incorporate uncertainty in the robot pose but cannot generally provide a consistent estimate for all landmarks based on all measurements. However, this developement made it possible to formulate SLAM under the perspective of estimation theory.

Stochastic Map

Mathematically, SLAM was first thoroughly derived as estimation problem by Smith, Self and Cheeseman [SSC88]. All landmark positions and the robot pose were represented in a common state vector and a complete covariance matrix. So the representation could handle the uncertainty of the robot pose. In particular, the high correlation between landmarks and robot pose caused by the accumulated error of the robot pose could be represented. This representation was called *stochastic map*, basically being an Extended Kalman Filter (EKF) [Gel74]. Later, Durrant-Whyte introduced the name *simultaneous localization and mapping* [DWRN95].

This approach was extended and improved by several authors [Tar92, HBBC95, CTS97, CMNT99, New99]. But still the main challange of large computation time necessary for updating the covariance matrix ($O(n^2)$ for n landmarks) remained. Even though the computation time was significantly smaller than for ML-estimation ($O((n+p)^3)$ for p robot poses) computation had to be performed after each measurement. So SLAM was still limited to small environments (to give a figure $\lessapprox 100$ landmarks).

Several authors tried to reduce computation time by treating landmarks [GOR94, VBX96, Ren93, UJC97], or local submaps [Fed99] as stochastically independent. As discussed above, the uncertainty structure of SLAM is that relations between adjacent landmarks are precisely known, even though the absolute positions of landmarks are rather uncertain. This leads to high correlations of landmarks [CTS97, FH01]. The above mentioned approaches ignore this structure: When treating two landmarks as if they were stochastically independent, the covariance of their distance basically is the sum of the covariances of the landmarks. So it is impossible to express that relations between some landmarks are precisely known, although the landmarks' positions are very uncertain. Thus, these approaches either diverge and report too little uncertainty for the landmarks or they are strongly conservative [UJC97] giving an uncertainty much too large for the relations of adjacent landmarks.

Lu and Milios [LM97] avoided the use of EKF by directly solving the linear equations system of maximum likelihood estimation taking $O(p^3)$ computation time. They do not extract landmarks but directly derive uncertain relations between different robot poses by comparing laser scans (*consistent pose estimation*). The advantage is that average computation time is much lower because no covariance matrix has to be maintained. But when closing a loop the equation system needs to be solved taking order of one minute computation time for about $p \approx 1000$ robot poses. This approach has been improved by Gutmann and Konolige [GK99] introducing an effective scheme for detecting a loop has been closed. A further advantage is that nonlinearity problems can be solved by recomputing the Jacobians of the measurement functions, which is not possible for EKF based approaches.

Thrun et al. [TBF98, BFJ+99] used a grid based data structure both for representing local maps as evidence grids (similar to [Elf89]) and for representing likelihood distributions of different robot poses as histograms. These poses constitute reference frames for the local maps. This two layered approach allows to incorporate uncertainty about the robot pose without blurring the map as is the case for plain evidence grids. Furthermore, non-Gaussian and even multi modal distributions can be represented, which is not possible in all EKF or least squares based approaches. So this algorithm achieves a large degree of robustness even when using noisy ultrasonic sensors. This approach had actually been used for building the map for a robotic museum tour guide [BCF+99]. The most likely map is computed by the expectation maximization (EM) algorithm being rather slow (more than 30 minutes in the example shown), so the approach is robust but practical only for off-line use.

Efficient SLAM

In the last five years interest in SLAM has increased significantly and resulted in several more efficient algorithms (see [Thr02] for a broad review). In contrast to EKF based approaches most of these algorithms are efficient enough to be used in medium large environments ($n \approx 500$). Some fairly fast approaches can even be used for large environments ($n \gtrapprox 10000$) but for these algorithms the quality of the estimated map has to be regarded. Most approaches exploit that only a few landmarks local to the robot at a given time can be involved in measurements. The number k of these landmarks influences the computation time of these algorithms, practically being ≈ 10 and theoretically mostly considered constant $k = O(1)$.

Guivant and Nebot [GN01, GN02] developed a modification of EKF called *Compressed EKF* (*CEKF*) that allows accumulating measurements in a local region with k landmarks at cost $O(k^2)$ independent of the overall map size n. When the robot leaves the local region the accumulated result must be propagated to full EKF (*global update*) at cost $O(kn^2)$. The global update can be performed more efficiently by conservatively neglecting some correlations. However, this violates the uncertainty structure discussed in §2.3 and cannot be used directly. Instead, all landmarks are grouped locally into so called *constellations* and the position of all landmarks in a constellation is expressed relative to two reference landmarks. The state vector consists of the absolute positions of all reference landmarks and the relative positions of all other landmarks. For two distant landmarks of different constellations, the correlation of their absolute positions basically is the same like the correlation between their corresponding reference landmarks. Thus, the correlation between their relative positions is very small and can be conservatively neglected. With this approximation a global update requires an update of the following correlations: a) Be-

tween all landmarks and the landmarks of the current constellation and b) between all landmarks and all reference landmarks. Altogether for constellations of size c, $O(n(c + \frac{n}{c})) = O(nc + \frac{n^2}{c})$ entries have to be updated. Each entry needs $O(k)$ computation time, so the overall time for a global update is $O(knc + \frac{kn^2}{c})$. The asymptotically optimal result is achieved with $c := \sqrt{n}$ leading to $O(kn^{\frac{3}{2}})$ computation time and $O(n^{\frac{3}{2}})$ storage space. Practically, often c is chosen as $O(k)$. Then computation time is $O(k^2n + n^2)$. It turns out, that often the $O(n^2)$ part is comparable to the $O(k^2n)$ part.

Duckett et al. [DMS00, DMS02] iteratively solve the linear equations system appearing in maximum likelihood estimation. They make use of an equation solver called *relaxation*, which is also known as Gauss-Seidel iteration in numerical literature and as "Gibbs sampling with zero temperature" in the context of Markov Chain Monte Carlo methods. They decompose the map into different "places" which are linked by spatial relations derived from odometry and laser scan matching. The absolute poses of these places are used as variables to be estimated. This way the size of the representation is $O(kn)$ and does not grow like in the Lu and Milios approach if the robot moves through an already visited area. Directly solving the equation system would need $O(n^3)$ computation time, which is avoided by performing only one iteration of relaxation after each step of the robot. Thereby the computational effort is distributed while the robot is moving taking $O(kn)$ per pose. However, when closing a large loop a single iteration is insufficient, so up to $O(n)$ iterations with $O(kn^2)$ computation time are needed for fully back-propagating the error. This problem was recently solved by Frese and Duckett [FLD04] using a method called *Multilevel Relaxation*. A multilevel approach was employed that is similar to the multi grid methods used in the numerical solution of partial differential equations. Thereby the computation time could be reduced to $O(kn)$ even when closing large loops. This approach can also handle nonlinearities by recomputing the measurement Jacobians, which is not possible for EKF based approaches.

Montemerlo et al. [MTKW02] observed that the landmark estimates are conditionally independent given the robot pose. This can be seen at the diagonal structure of the matrix P in §2.11. A particle filter [DdFG01] is being used to represent the distribution of robot poses. Each particle represents a sampled robot trajectory and the conditional distribution of the landmark given the robot trajectory as well. Since the landmarks are conditionally independent, they can be represented by a small EKF for each. The robot trajectory need not actually be stored, so the representation needs $O(Mn)$ storage space for M particles. As per clever bookkeeping the integration of a measurement is done quickly with $O(M \log n)$ computation time. The unique advantage is that this algorithm can handle uncertain landmark identification. The key point for

efficiency is how many particles are needed. Particle filters integrate measurements by resampling, i.e. by choosing a subset of particles being compatible with the measurements from the set of existing particles. So for closing a large loop one of the M particles must already have closed the loop incidentally with an remaining error comparable to the error of a single landmark measurement. This is the critical point for the performance of the algorithm, because the number M of particles needed can be rather large: If the loop has n landmarks the distance travelled will be $O(n)$, the robot's orientation uncertainty may be up to $O(\sqrt{n})$ and the position uncertainty may be $O(n^{\frac{3}{2}})$. To make sure that there is probably one particle incidentally closing the loop, $O(\sqrt{n}n^{\frac{3}{2}}n^{\frac{3}{2}}) = O(n^{\frac{7}{2}})$ particles are needed. In practice it is presumably not possible to use so many particles. When reducing the number of particles precision is sacrificed for speed and some residual error will remain after closing a large loop.

Thrun et al. [TKGDW02] followed an idea also proposed by Frese and Hirzinger [FH01] using an information matrix instead of a covariance matrix to represent uncertainty. This matrix is approximately sparse. A proof was given in §2.11 of this thesis. The algorithm approximates the matrix by a sparse matrix with $O(kn)$ storage space and is therefore called *Sparse Extended Information Filter* (*SEIF*). An information matrix representation is of advantage because the entries of uninvolved landmarks do not change when integrating a measurement. So updating the representation takes $O(k^2)$. In contrast to SEIF the whole covariance matrix has to be updated in EKF based approaches. However, to derive an estimate a linear equation system with the information matrix has to be solved. This is performed iteratively by relaxation. With one iteration per measurement the result would be an $O(kn)$ algorithm similar to the approach of Duckett et al. with some problems when closing loops. Thrun et al. propose not to relax all $O(n)$ landmarks but just $O(k)$ after each measurement, thereby formally obtaining an $O(k^2)$ algorithm. However, in numerical literature relaxation is reputed to need $O(n)$ iterations, i.e. $O(kn^2)$ time to reduce the equation error by a constant factor [Bri99, PTVF92, §19.5]. For instance, having observed $O(n)$ landmarks each $O(1)$ times, the algorithm will have spent only $O(n)$ time on equation solving. So it is remaining unclear, whether this approach will suffice in general.

Comparison

Table 2.2 shows an overview of the performance of the algorithms discussed. It can be seen that only multi level relaxation and the proposed algorithm strictly fulfill all three requirements:

Requirement (R1) is completely fulfilled by Maximum Likelihood estimation (ML) and single or multi level relaxation and also additionally by EKF and CEKF, if the orientation error is small enough ($\lessapprox 15°$, see §2.9) to allow linearization. When closing a loop SEIF and fastSLAM do not fulfill (R1) due to the problems mentioned before.

	(R1)			(R2)	(R3)		
	UDA	nonlinear	quality	memory	update	global upd.	loop
ML		√	√	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
Multi level relaxation		√	√	kn	kn		
FastSLAM	√	√	?	Mn	$M \log n$		
SEIF			?	kn	k^2		
Proposed algorithm		√	√	kn	k^2	$k^3 \log n$ resp. kn	

Table 2.2: Performance of different SLAM algorithms with n landmarks, m measurements, p robot poses and k landmarks local to the robot. The proposed algorithm assumes a topologically suitable building (see §3.5). $O(k^3 \log n)$ is the computation time for a local, $O(kn)$ for a global map. FastSLAM is a particle filter approach (M particles). Compare the remarks in §2.14 concerning the quality of the estimates provided by fastSLAM and SEIF. UDA stands for 'Uncertain Data Association' meaning landmarks with uncertain identity.

Requirement (R2) is met by relaxation, fastSLAM (for $M = O(1)$) and SEIF.

Requirement (R3) is fulfilled by fastSLAM and SEIF, giving an estimation that does not always meet (R1). Among the approaches meeting (R1), relaxation and CEKF come very close to meet (R3). Relaxation needs linear computation time with the exception of closing a large loop, while CEKF only needs $O(n^{\frac{3}{2}})$ even when closing loops. The additional advantage of CEKF is that this computation is not performed after each measurement, but only when the robot leaves a local area of the map. Multi level relaxation avoids the $O(n^2)$ problem of relaxation when closing a loop and thus fulfills (R3) completely.

What is the reason for the large discrepancy between the computation time postulated by requirement (R3) and the one achieved by the algorithms discussed? [4]

Least squares estimation and incremental least squares estimation in general lead to linear equations systems which is an established and thoroughly studied area of numerical mathematics. So it is very unlikely to find a general solution being faster than EKF with $O(n^2)$. From the author's perspective the key point is to identify a property distinguishing SLAM from a general estimation problem. Indeed, all faster approaches exploit such a property: Relaxation, multi level

[4] Multi level relaxation [FLD04] has been published recently and was developed after the algorithm presented in this thesis.

relaxation and SEIF exploit sparsity, fastSLAM exploits a special factorization of the involved probability distribution and CEKF exploits some property of the correlation of distant landmarks.

The author of this thesis has tried to pursue this line of thought further and identified a property stronger than the above mentioned: A typical building can be (recursively) divided into two parts, with very few landmarks of one part being observable from the other part. The proposed algorithm exploits this property by decomposing the map into a hierarchy of regions and subregions, representing only those landmarks at a region that are also observable from outside the region. Each region stores a small information matrix of size $O(k \times k)$ because only few, i.e. $O(k)$ landmarks are represented there[5]. For integrating a measurement only the region containing the robot and all its superregions need to be updated. There are $O(\log n)$ superregions and each update takes $O(k^3)$, so an overall update takes $O(k^3 \log n)$. The proposed algorithm fulfills (R1) and (R2) and even exceeds (R3). This will be discussed in the following two chapters.

2.15 Summary

SLAM is the problem of simultaneously estimating a map and the robot pose in that map from landmark and odometry measurements. The estimation has to be performed while the robot is moving, providing a new estimate whenever a measurement occurs. Three exceptional features are distinguishing SLAM from many other estimation problems:

1. *Error accumulation*:
   When moving through an unknown area the error of the robot pose and, consequently, the global error of landmarks nearby can grow arbitrarily high. Nevertheless, relative properties of these landmarks like distances or angles are known much more precisely with an uncertainty independent from the overall uncertainty of the robot pose. This gives rise to a highly specific uncertainty structure, called the *Certainty of Relations despite Uncertainty of Positions*. Relations between nearby landmarks are known precisely although the absolute position of the landmarks is highly uncertain. On a global scale, uncertainty is a composition of local uncertainties along the path traveled. Its main source is the superposition of uncertain rotations generated at each moment by any new orientation error originating from movement. The error resulting from these rotations effects an "uncertain bending" of the map, showing a simple geometrically determined structure despite of its magnitude. Conversely, on a local scale the uncertainty is much smaller and of more complex structure.

   The algorithm presented in this thesis exploits the specific SLAM uncertainty structure by decomposing the map into small regions, each represented by a small matrix. So this com-

[5] for a class of topologically suitable buildings as discussed in §3.5

plex local uncertainty structure can be suitably represented in the matrix. As the global structure is composed of different local uncertainties there is no need of explicit representation.

2. *High dimensionality*:
   After each measurement a SLAM algorithm has to estimate the robot pose (3 DOF) and the whole map ($2n$ DOF for n landmarks). So the overall dimension $3 + 2n$ of the estimation problem is very large (> 500) and increasingly growing. Therefore, the common estimation algorithms are not efficient enough for SLAM: ML and LLS both need $O((n + p)^3)$ computation time per measurement (p number of robot poses) and EKF needs $O(n^2)$. EKF stores a full covariance matrix of all landmarks ($O(n^2)$) whereas ML has to store all previous measurements ($O(n+p)$). For LLS this can be avoided by eliminating old robot poses leading to $O(n^2)$ storage space and $O(n^3)$ computation time.

   In this chapter, a theorem has been proven (theorem 2) that assures that the information matrix resulting from elimination of old robot poses is approximately sparse. The theorem has important consequences: It allows to implement LLS with $O(n)$ storage and $O(n^2)$ computation time. It further theoretically substantiates the analysis of the global uncertainty described above. The algorithm presented in this thesis utilizes the structure established by the theorem to reduces the computation time to $O(\log n)$.[6]

3. *Nonlinearity*:
   It is common for estimation problems to have nonlinear measurement equations. For SLAM equations are nonlinear in the robot's orientation, the estimation error of which can grow unboundedly. For a sufficiently large map linearization is not suitable. This is a particular problem of SLAM not often encountered in other estimation problems. ML provides an optimal estimate even with nonlinear measurement equations. Technically, ML performs iterative least squares and re-evaluates the measurement Jacobians in each iteration. Both, LLS and EKF, are subject to linearization errors. A severe consequence is the distortion of distances between landmarks even though the distances are well known from the measurements.

   Due to the nonlinearities the measurement Jacobians are variable. When transforming them into robot coordinates by factoring out rotation by robot orientation, they are nearly constant. This shows that only nonlinearity of orientation is of relevance. The algorithm utilizes this property to change the point of linearization of measurements already integrated by applying a rotation matrix to the resulting information matrix.

[6] under the assumption of a topologically suitable building (see §3.5)

Chapter 3

Hierarchical Map Decomposition

The algorithm presented in this thesis decomposes the information provided by measurements into many small parts which are organized hierarchically. It consists of two separate components: The first component described in this chapter manipulates, decomposes and integrates these parts of information using linear algebra operations. The second component presented in §4 maintains a hierarchical decomposition of the map represented by a tree, integrates and decomposes the overall information using the subalgorithms provided by the first part and stores the results at the different nodes of the tree.

Sections 3.1 and 3.2 explain the general idea. Sections 3.3 to 3.8 present the linear algebra subalgorithms used for manipulation of information. Section 3.9 deals with how to avoid linearization problems. In section 3.5 conditions are discussed a building must satisfy for the algorithm to be efficient.

3.1 Basic Idea

The key observation about SLAM is that in all measurements only local landmarks are being involved. The robot's sensors have a limited range and in typical indoor environments the field of view is limited by walls. Furthermore, navigation and most other tasks need only information local to the robot. For instance, a path could be defined relative to local landmarks and the robot could follow this path by relocalizing itself relative to these landmarks. On the other hand, a single landmark observation can globally change the map, for instance, when closing a large loop. So SLAM cannot be completely reduced to updating independent local submaps.

Motivated by these insights, the algorithm provides an estimate only for the landmarks local to the robot. Nonetheless, it computes a consistent estimate that is identical[1] to the full least

[1] for landmark – landmark observation the estimate is identical, for landmark – robot observations approximately

squares estimate for the whole map. With a local estimate available observations of local landmarks can directly be integrated into the estimation using EKF equations without considering the global map. Similar to CEKF [GN02] integration takes a time independent of the overall size of the map.

When the robot moves every once in a while the set of local landmarks changes and a global update is necessary to compute an estimate for the new local landmarks. The central challenge for the algorithm is to maintain the information appropriately performing the update efficiently.

The basic idea 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, reconsider figure 1.5a with a building that is virtually divided into two parts. Let these parts be named A and B and consider the following question:

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 actually needed outside of B.

The idea can be applied recursively by dividing the building into a hierarchy of regions (Fig. 1.5a). 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 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

equal (see §3.6)

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 for such a global update only the condensed information of the actual region and all regions above need to be updated. All other regions remain unaffected.

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 from the top region down to the actual region.

3.2 Tree Map Data Structure

This section introduces the *tree map* data structure used by the algorithm for representing a hierarchically divided map. At first, it will be assumed that the robot's observations are landmark – landmark measurements. Under this assumption the algorithm is exact. No approximation is performed. In §3.6 the algorithm will be extended to integrate also landmark – robot and robot – robot (odometry) measurements using slight approximation.

Data Structure

In the algorithm, 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. All notions refering to information like "integration" and "decomposition" correspond to actual linear algebra operations on these matrices and vectors.

The regions corresponding to nodes are not defined in a strict geometrical sense, 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 tree map. 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. Figure 1.5b (page 45) shows the tree corresponding to the hierarchy of figure 1.5a and the landmarks represented at each node.

Furthermore, each node holds a *Substitution Information Block*[2] (SIB) containing the information about the landmarks eliminated at the node. This information is only needed when the robot is inside the region, i.e. the actual leaf is below this node. The distribution of information between the different node's CIBs and SIBs is made formal in the following definition:

Definition 3 (Information stored at a 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) that contains 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 that is not contained in the nodes CIB. For a leaf it stores the information from the BIB 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. It is the lowest node the landmark is represented exclusively below. The algorithm maintains an array of elimination nodes for the different landmarks.

The CIB contains all information that must be known about a node's region, if the robot is outside that region. It is computed recursively: A node integrates the CIB of both children and eliminates all landmarks for which it is the elimination node. The result is the node's CIB, which is stored and passed to the parent. The information about the eliminated landmarks is the node's SIB. Since all landmarks are being eliminated once, the integration of all SIBs is the complete information represented by the map and the same as the integration of all BIBs. 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.

Figure 3.1 shows the information flow at a single node and a three level tree: The information from the CIBs of both children is integrated (**+**) and then decomposed again (**S**). A detailed explanation will be given in §3.4. For the moment, the symbols (**+**) and (**S**) can be viewed as black box operations.

Summary: The purpose of the tree map is to compute estimates from measurements. The in-

[2] the name is explained in §3.3

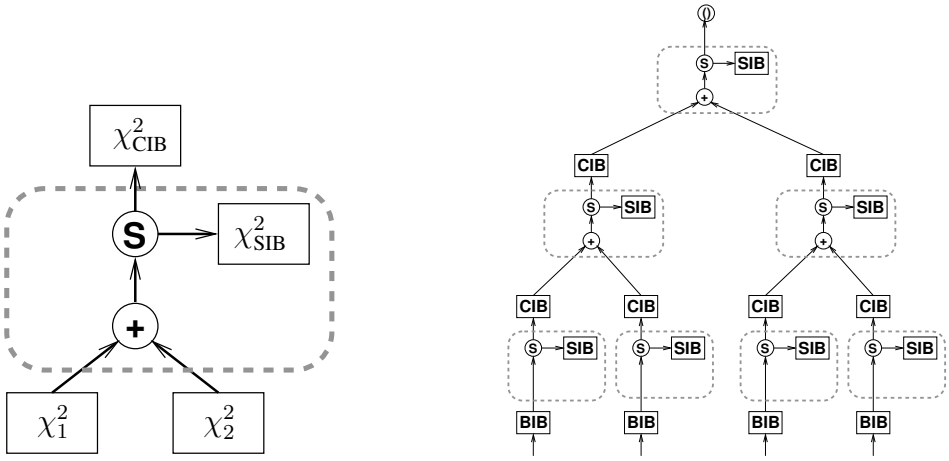


Figure 3.1: Integration and decomposition of information in a single node (oval) and a three level tree: Two IBs χ^2_1 and χ^2_2 from the nodes children are integrated (**+**) and then decomposed (**S**) into a CIB χ^2_{CIB} and a SIB χ^2_{SIB}. The CIB is passed to the parent and the SIB stored at the node.

formation from the measurements is stored in the BIBs and the estimates are computed from the information from the SIBs. So the integrated information of all BIBs equals the integrated information of all SIBs. The difference is that the information in the BIBs is distributed according to where the robot was when the measurements was made. On the contrary, all information known about a specific landmark is contained in the SIB at the elimination node of this landmark. So in the end computation of SIBs is a task of compiling information from BIBs. From this perspective the role of a CIB is that of an intermediate result: A node's CIB contains all information from the BIBs below that node that has not yet been compiled into SIBs below. It still has to be processed on higher levels and integrated with further information from other parts of the tree.

Remark: The original idea in §3.1 was to represent all landmarks that *can* be observed from outside the region. However definition 3 represents only those landmarks that *have been* observed from outside. The reason for this is that deciding whether a landmark is observable or not requires the representation of walls and obstacles in the map which is not performed by the algorithm. As a consequence, the set of landmarks represented at a node changes whenever a landmark is observed for the first time outside that node. In this case additional updates have to be performed.

Integration of a measurement

It is currently assumed that all observations are measurements of relative landmark positions (see §3.6). 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 tree map at all and its computation time 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 are not ancestors and remain unaffected, so computation is highly efficient.

Compilation of an estimate

After a new BIB is made the actual one and the global update has been performed, 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 landmarks represented at the node with the nodes' SIB.

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 stochastical information known from the measurements (§2.6). 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 $\chi^2_{\text{IB}}(x)$ refering 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 symmetric matrix A, a vector b and a constant γ as

$$\chi^2_{\text{IB}}(x) := x^T A x + x^T b + \gamma = \sum_{i,j} A_{ij}\, x_i x_j + \sum_i b_i\, x_i + \gamma. \tag{3.1}$$

This is the usual representation of a multidimensional quadratic function. The constant coefficient is γ, first order coefficients are found in b and second order coefficients in A. The error function $\chi^2(x)$ is ≥ 0 for all x, which implies that A is symmetric positive semidefinite (SPSD)[3] . 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. Matrix A is the information matrix for x given the information represented by the IB. It represents the uncertainty of the information of the IB. The information itself, say the landmarks' coordinates, is given by the minimum $\arg\min_x \chi^2(x) = A^{-1}b/2$, so it is represented in vector b but in a way that depends on the uncertainty in A.

[3] An extensive discussion of the mathematical properties of positive definite and positive semidefinite matrices can be found in [HJ90] and in a brief overview in appendix A

Notation

In this chapter the linear algebra part of the algorithm will be described. It provides the subalgorithms for manipulation and integration of IBs. These subalgorithms are then used by the bookkeeping part described in §4 to compute the IBs stored at different nodes of the tree.

Throughout §3 and §4, it will be assumed, that each vector / matrix is augmented with the information, which row / column corresponds to which random variable (robot pose: x, y, ϕ; landmark: x, y). Further, a few notational conventions will be defined as follows (see page 25 for a complete list of symbols):

Symbol	Definition
$A, P, R, S, \ldots$	Upper case letter denote matrices
$b, p, q, u, v, \ldots$	Lower case letter denote vectors
$\alpha, \beta, \gamma, \ldots$	Greek letter denote scalars
$\mathcal{A}, \mathcal{B}, \mathcal{C}, \ldots$	Calligraphy letter denote sets of landmarks
$\mathcal{L}(A), \mathcal{L}(b)$	Set of landmarks represented by matrix A or vector b
$\chi^2(x)$	Quadratic error function. The negative log-likelihood of landmark positions / robot pose x given the information represented by χ^2.
$\hat{x}, \hat{y}, \hat{z}$	Lower case letter with $\widehat{\text{hat}}$ denote estimates

3.3 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. The second step is to eliminate some landmarks by decomposing the result into two parts: 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.

This operation is a redistribution of information, since the integrated information of both input CIBs or the input BIB respectively is equal to the integrated information of the resulting CIB and SIB. The subalgorithm is shown as a structure chart[4] (Fig. 3.2, `computeCIBAndSIB`). Figure 3.1 illustrates the underlying data flow. In the following, the formulas for the integration and decomposition are derived:

[4]Structure charts are used to describe the algorithm formally. They are intended as a detailed reference and to provide the information necessary for implementing the algorithm.

Lemma 7. *Let $\chi_1^2(x)$ and $\chi_2^2(x)$ be two stochastically independent information blocks. Then the integrated information is $\chi^2(x) = \chi_1^2(x) + \chi_2^2(x)$.*

Proof. By definition a χ^2 function is the negative logarithm of the likelihood given the information represented by that function. The likelihood of x, considering independent information from χ_1^2 and χ_2^2, is the product of the individual likelihoods

$$\chi^2(x) = -\ln\left(e^{-\chi_1^2(x)} \cdot e^{-\chi_2^2(x)}\right) = \chi_1^2(x) + \chi_2^2(x). \tag{3.2}$$

□

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.

The following discussion deals with two different groups of landmarks: The landmarks that are going to be eliminated and stored in the SIB and the landmarks that are stored in the CIB and passed to the parent node. 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 block row / column 2 to remaining landmarks:

$$\chi^2(x) = x^T A x + x^T b + \gamma \tag{3.3}$$

$$= \chi^2\begin{pmatrix} y \\ z \end{pmatrix} = \begin{pmatrix} y \\ z \end{pmatrix}^T \begin{pmatrix} P & R^T \\ R & S \end{pmatrix} \begin{pmatrix} y \\ z \end{pmatrix} + \begin{pmatrix} y \\ z \end{pmatrix}^T \begin{pmatrix} c \\ d \end{pmatrix} + \gamma. \tag{3.4}$$

Vector block z and matrix block S correspond to the landmarks to be eliminated, vector block y and matrix block P to the remaining landmarks and matrix block R corresponds to the couplings between both groups of landmarks.

Lemma 8 (Schur Complement). *Let $\chi^2(\begin{smallmatrix} y \\ z \end{smallmatrix})$ be an information block as in* (3.4) *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 $\chi^2_{SIB}(0) = 0$:*

$$\chi^2_{SIB}(w) = w^T P w, \quad H = -P^{-1}R^T, \quad h = -P^{-1}c/2. \tag{3.5}$$

Proof. The IB $\chi^2(\begin{smallmatrix} y \\ z \end{smallmatrix})$ gives the negative log likelihood of positions y of the landmarks to be eliminated together with positions z of the remaining landmarks. To derive an IB for z means to compute a likelihood for z alone. Therefore for each z the minimum over all y must be taken and substituted into χ^2. So, the first step is to find the y that minimizes $\chi^2(\begin{smallmatrix} y \\ z \end{smallmatrix})$ as a function of z

$$y_*(z) := \arg\min_y \chi^2\begin{pmatrix} y \\ z \end{pmatrix}. \tag{3.6}$$

Therefore $\chi^2(x)$ is block decomposed and expanded resulting in

$$\chi^2(x) = \chi^2\begin{pmatrix} y \\ z \end{pmatrix} =: \begin{pmatrix} y \\ z \end{pmatrix}^T \begin{pmatrix} P & R^T \\ R & S \end{pmatrix} \begin{pmatrix} y \\ z \end{pmatrix} + \begin{pmatrix} y \\ z \end{pmatrix}^T \begin{pmatrix} c \\ d \end{pmatrix} + \gamma \tag{3.7}$$

$$= y^T P y + y^T R^T z + z^T R y + z^T S z + y^T c + z^T d + \gamma. \tag{3.8}$$

At the z dependent minimum $y_*(z)$, the gradient with respect to y is

$$0 = \frac{\partial \chi^2(\begin{smallmatrix} y \\ z \end{smallmatrix})}{\partial y}\big(y_*(z)\big) = 2Py_*(z) + 2R^T z + c \tag{3.9}$$

$$\Rightarrow \quad y_*(z) = P^{-1}(-R^T z - c/2) = \underbrace{-P^{-1}R^T}_{H} z \underbrace{-P^{-1}c/2}_{h}. \tag{3.10}$$

Substituting the maximum likelihood estimate $y = y_*(z)$ yields an information block $\chi^2_{\text{CIB}}(z)$ containing all information about the landmarks corresponding to z that are not eliminated:

$$\begin{aligned} \chi^2_{\text{CIB}}(z) :=& \chi^2\begin{pmatrix} y_*(z) \\ z \end{pmatrix} = \chi^2\begin{pmatrix} Hz+h \\ z \end{pmatrix} \\ =& (z^T H^T + h^T)P(Hz+h) + (z^T H^T + h^T)R^T z + z^T R(Hz+h) \\ &+ z^T S z + (Hz+h)^T c + z^T d + \gamma. \end{aligned} \tag{3.11}$$

$$\begin{aligned} =& z^T\big(H^T P H + H^T R^T + RH + S\big)z \\ &+ z^T\big(2H^T P h + 2Rh + H^T c + d\big) + \big(h^T P h + c^T h\big) + \gamma \end{aligned} \tag{3.12}$$

$$= z^T\big(S - RP^{-1}R^T\big)z + z^T\big(-RP^{-1}c + d\big) + \big(-\frac{1}{4}c^T P^{-1} c\big) + \gamma \tag{3.13}$$

The second information block is $\chi^2_{\text{SIB}}(w) = w^T P w$, with $w = Hz + h - y$. Expanding this expression yields

$$\begin{aligned} \chi^2_{\text{SIB}}(Hz+h-y) =& \big(Hz+h-y\big)^T P\big(Hz+h-y\big) \\ =& z^T H^T P H z + 2h^T P H z - 2y^T P H z \\ &+ h^T P h - 2y^T P h + y^T P y \end{aligned} \tag{3.14}$$

$$\begin{aligned} =& z^T R P^{-1} R^T z + c^T P^{-1} R^T z + 2y^T R^T z \\ &+ \frac{1}{4}c^T P^{-1} c + y^T c + y^T P y \end{aligned} \quad . \tag{3.15}$$

Adding equation (3.13) and (3.15) verifies that the information from χ^2 has indeed been decomposed as

$$\chi^2_{\text{CIB}}(z) + \chi^2_{\text{SIB}}(Hz+h-y) = \tag{3.16}$$

Figure 3.2: **computeCIBAndSIB** $\left(\chi_1^2,\ \chi_2^2,\ \mathcal{E}\right)$ *linearized*
χ_1^2, χ_2^2: IBs to be integrated; $\mathcal{E}$: set of landmarks to be eliminated

IF	$\chi_2^2 \neq 0$
THEN	$(A_1, b_1) := \chi_1^2; \quad (A_2, b_2) := \chi_2^2$
	Sort and extend A_1, A_2, b_1, b_2 so that columns and rows correspond to the same landmark. And landmarks $\mathcal{E}$ are first.
	$A := A_1 + A_2; \quad b := b_1 + b_2$
ELSE	$(A, b) := \chi_1^2$
Apply lemma 8 to compute χ^2_{CIB}, χ^2_{SIB}, H, h, P^{-1}	
return χ^2_{CIB} as CIB and χ^2_{SIB}, H, h, P^{-1} as SIB	

$$z^T S z + z^T d + 2y^T R^T z + y^T c + y^T P y = \chi^2 \begin{pmatrix} y \\ z \end{pmatrix} = \chi^2(x). \tag{3.17}$$

By construction $\chi^2_{\text{SIB}}(y) \geq 0$ holds. Since $\chi^2_{\text{SIB}}(\begin{smallmatrix} Hz+h \\ z \end{smallmatrix}) = 0$ and $\chi^2(\begin{smallmatrix} y \\ z \end{smallmatrix}) \geq 0$, it follows that $\chi^2_{\text{CIB}}(z) = \chi^2(\begin{smallmatrix} Hz+h \\ z \end{smallmatrix}) \geq 0$ and in particular that $S - R^T P^{-1} R$ is positive semidefinite. The IB χ^2_{CIB} contains all information about z. The IB χ^2_{SIB} expresses the information, that $y \approx Hz + h$ with an uncertainty of P^{-1}, where $Hz + h$ is the maximum likelihood estimate for y given z.

In order to prove the uniqueness consider any decomposition

$$\chi^2_{\text{CIB}}{}'(z) + \chi^2_{\text{SIB}}{}'(H'z + h' - y) = \chi^2 \begin{pmatrix} y \\ z \end{pmatrix} \tag{3.18}$$

fulfilling the conditions of the lemma. The minimum of (3.18) for a given z is $\chi^2_{\text{CIB}}(z)'$ at $y = H'z + h'$. The corresponding minimum of $\chi^2(\begin{smallmatrix} y \\ z \end{smallmatrix})$ is $\chi^2(\begin{smallmatrix} y_*(z) \\ z \end{smallmatrix}) = \chi^2_{\text{CIB}}(z)$ at $y = y_*(z) = Hz + h$. So $\chi^2_{\text{CIB}}{}'(z) = \chi^2_{\text{CIB}}(z)$ and $H = H', h = h'$. □

Example

Table 3.1 shows an example of the application of lemmas 7 and 8: There are 3 landmarks (a, b, c) considered one-dimensional for simplicity. The first IB (χ_1^2) contains the information of the distance between a and b with a standard deviation of 1. The second IB (χ_2^2) contains the corresponding information for landmark b and c. The two IBs are integrated into χ^2. Then landmark b is eliminated by decomposing the information into χ^2_{CIB} and χ^2_{SIB}. Table 3.1 shows different views (columns) on the different IBs (rows).

Column **1)** depicts a graphical description with straight lines representing distance information. The arrow in row χ^2_{SIB} indicates the structure of the information in the SIB, providing information about b if and only if information about a and c is given.

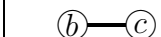
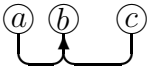
	1) Graph of Relations	**2)** Equations plus Noise	**3)** Information Block as represented by the algorithm	**4)** Information Block for (a, b, c)
χ^2_1		$b-a=1+N(1)$	$\chi^2_1\left(\begin{smallmatrix}a\\b\end{smallmatrix}\right)=(b-a-1)^2/1^2=$ $\left(\begin{smallmatrix}a\\b\end{smallmatrix}\right)^T\left(\begin{smallmatrix}1&-1\\-1&1\end{smallmatrix}\right)\left(\begin{smallmatrix}a\\b\end{smallmatrix}\right)+\left(\begin{smallmatrix}2\\-2\end{smallmatrix}\right)^T\left(\begin{smallmatrix}a\\b\end{smallmatrix}\right)+1$	$\left(\begin{smallmatrix}1&-1\\-1&1\end{smallmatrix}\right),\ \left(\begin{smallmatrix}2\\-2\end{smallmatrix}\right),\ 1$
χ^2_2		$c-b=2+N(1)$	$\chi^2_2\left(\begin{smallmatrix}b\\c\end{smallmatrix}\right)=(c-b-2)^2/1^2=$ $\left(\begin{smallmatrix}b\\c\end{smallmatrix}\right)^T\left(\begin{smallmatrix}1&-1\\-1&1\end{smallmatrix}\right)\left(\begin{smallmatrix}b\\c\end{smallmatrix}\right)+\left(\begin{smallmatrix}4\\-4\end{smallmatrix}\right)^T\left(\begin{smallmatrix}b\\c\end{smallmatrix}\right)+4$	$\left(\begin{smallmatrix}1&-1\\-1&1\end{smallmatrix}\right),\ \left(\begin{smallmatrix}4\\-4\end{smallmatrix}\right),\ 4$
χ^2		$b-a=1+N(1)$ $\wedge$ $c-b=2+N(1)$	$\chi^2\left(\begin{smallmatrix}a\\b\\c\end{smallmatrix}\right)=5+$ $\left(\begin{smallmatrix}a\\b\\c\end{smallmatrix}\right)^T\left(\begin{smallmatrix}1&-1&0\\-1&2&-1\\0&-1&1\end{smallmatrix}\right)\left(\begin{smallmatrix}a\\b\\c\end{smallmatrix}\right)+\left(\begin{smallmatrix}2\\2\\-4\end{smallmatrix}\right)^T\left(\begin{smallmatrix}a\\b\\c\end{smallmatrix}\right)$	$\left(\begin{smallmatrix}1&-1&0\\-1&2&-1\\0&-1&1\end{smallmatrix}\right),\ \left(\begin{smallmatrix}2\\2\\-4\end{smallmatrix}\right),5$
χ^2_{CIB}		$c-a=$ $3+N(\sqrt{2})$	$\chi^2_{\text{CIB}}\left(\begin{smallmatrix}a\\c\end{smallmatrix}\right)=$ $\left(\begin{smallmatrix}a\\c\end{smallmatrix}\right)^T\left(\begin{smallmatrix}\frac{1}{2}&-\frac{1}{2}\\-\frac{1}{2}&\frac{1}{2}\end{smallmatrix}\right)\left(\begin{smallmatrix}a\\c\end{smallmatrix}\right)+\left(\begin{smallmatrix}6\\-6\end{smallmatrix}\right)^T\left(\begin{smallmatrix}a\\c\end{smallmatrix}\right)+4\frac{1}{2}$	$\left(\begin{smallmatrix}\frac{1}{2}&-\frac{1}{2}\\-\frac{1}{2}&\frac{1}{2}\end{smallmatrix}\right),\ \left(\begin{smallmatrix}6\\-6\end{smallmatrix}\right),4\frac{1}{2}$
χ^2_{SIB}		$b=$ $\frac{1}{2}a+\frac{1}{2}c-\frac{1}{2}$ $+N(\sqrt{\frac{1}{2}})$	$H=\left(\begin{smallmatrix}\frac{1}{2}&\frac{1}{2}\end{smallmatrix}\right),\ h=-\frac{1}{2},\ P=2,$ $\chi^2_{\text{SIB}}(w)=w^TPw$ $\Longrightarrow\chi^2_{\text{SIB}}\big(H\left(\begin{smallmatrix}a\\c\end{smallmatrix}\right)+h-b\big)=\frac{1}{2}+$ $\left(\begin{smallmatrix}a\\b\\c\end{smallmatrix}\right)^T\left(\begin{smallmatrix}\frac{1}{2}&-1&\frac{1}{2}\\-1&2&-1\\\frac{1}{2}&-1&\frac{1}{2}\end{smallmatrix}\right)\left(\begin{smallmatrix}a\\b\\c\end{smallmatrix}\right)+\left(\begin{smallmatrix}-4\\2\\2\end{smallmatrix}\right)^T\left(\begin{smallmatrix}a\\b\\c\end{smallmatrix}\right)$	$\left(\begin{smallmatrix}\frac{1}{2}&-1&\frac{1}{2}\\-1&2&-1\\\frac{1}{2}&-1&\frac{1}{2}\end{smallmatrix}\right),\ \left(\begin{smallmatrix}-4\\2\\2\end{smallmatrix}\right),\ \frac{1}{2}$

Table 3.1: Example for integration and decomposition of information $\chi^2_1+\chi^2_2=\chi^2=\chi^2_{\text{CIB}}+\chi^2_{\text{SIB}}$.

Column **2)** shows the measurement equations with $N(\sigma)$ denoting zero-mean Gaussian noise with standard deviation σ. The equations in rows χ^2_1 and χ^2_2 are original measurement equations. Row χ^2_{CIB} is derived by adding the equations for χ^2_1 and χ^2_2, since $N(1)+N(1)=N(\sqrt{2})$ due to independence. Row χ^2_{SIB} is derived correspondingly by subtracting both equations and dividing the result by 2. Observe the structure of the equation for χ^2_{SIB}: It defines b as a function of a and c with additional uncertainty.

Column **3)** contains the actual representation of the IB by a quadratic function as used by the algorithm. Note, that χ^2_1 and χ^2_2 use only 2×2 matrices, since they do not represent c and a respectively and that χ^2_{CIB} does not represent b, which was the overall purpose in this example. χ^2_{SIB} is used by the algorithm in a factored way as w^TPw, with $w:=Hz+h-y$. This corresponds to its special structure mentioned above. The table shows both, the actually stored parameter (H, h, P) and the effective IB $\chi^2_{\text{SIB}}(Hz+h-y)$. χ^2_1 and χ^2_2 are derived from the measurement equations in the usual way by squaring the error in the equation and dividing by the measurement covariance σ^2. χ^2 is $\chi^2_1+\chi^2_2$ following lemma 7. χ^2_{CIB} and χ^2_{SIB} are computed by lemma 8. As expected they equal the IBs that could be derived from the respective equations

in column 2.

Column **4)** once again shows the information blocks as a quadratic function represented by information matrix, vector and scalar. They are extended to 3×3 with the rows and columns corresponding to landmarks a, b, c to be easily compared. It can be observed that the operation performed indeed integrates and decomposes the information: The sum of χ_1^2 and χ_2^2 equals χ^2 which in turn equals the sum of χ_{CIB}^2 and χ_{SIB}^2.

The following two technical lemmas are derived from lemma 8 and will be applied in §3.7. They deal with decomposing the expressions $x^T Ax$ and $x^T A^{-1} x$ for a 2×2 block matrix A. These expressions have the same mathematical structure as an IB, but do not represent a map. The proof for the lemmas is provided in appendix B.

Lemma 9. *Let $A = \left(\begin{smallmatrix} P & R^T \\ R & S \end{smallmatrix}\right)$ be an SPD matrix and $x = \left(\begin{smallmatrix} y \\ z \end{smallmatrix}\right)$ be decomposed accordingly, with z given. Then the minimum over y of $x^T A^{-1} x$ is $z^T S^{-1} z$ at $y = R^T S^{-1} z$ or $x = A\left(\begin{smallmatrix} 0 \\ S^{-1}z \end{smallmatrix}\right)$. The corresponding minimum over z with y given, is $y^T P^{-1} y$ at $z = RP^{-1} y$ or $x = A\left(\begin{smallmatrix} P^{-1}y \\ 0 \end{smallmatrix}\right)$.*

When using lemma 8 to build up the tree map recursively, both parts of the decomposition are used: The SIB is stored at the node and the CIB is passed to the node's parent. In §3.7 the lemma will be used to discard the information deliberately, which is contained in the SIB. This makes some entries in the matrix zero, so a simpler representation for the remaining information is achieved. This application motivates the following definition and lemma (proof in appendix B):

Definition 4. *Let A be an SPSD 2×2 block matrix. Then an elimination matrix for block row / column 1 is an SPSD matrix B with $0 \leq B \leq A$ [5] and $A - B = \left(\begin{smallmatrix} 0 & 0 \\ 0 & * \end{smallmatrix}\right)$. Such a matrix is called minimal, if $B \leq B'$ holds for all other elimination matrices B'.*

Lemma 10. *Let $A = \left(\begin{smallmatrix} P & R^T \\ R & S \end{smallmatrix}\right)$ be an SPSD 2×2 block matrix, with P being SPD. Then the unique minimal elimination matrix for block row and column 1 is $B := \left(\begin{smallmatrix} P \\ R \end{smallmatrix}\right) P^{-1} \left(\begin{smallmatrix} P \\ R \end{smallmatrix}\right)^T$. In the case of the first block being one-dimensional, $B = uu^T$ with $u = \frac{1}{\sqrt{P}} \left(\begin{smallmatrix} P \\ R \end{smallmatrix}\right)$.*

3.4 Compilation of an Estimate

When the CIB and SIB of each node in the whole tree are available an estimate with a corresponding estimation covariance for the landmarks represented in a certain BIB can be computed:

[5] See appendix A.1: $B \leq A \iff \forall x : x^T Bx \leq x^T Ax$

Start with an estimate $\hat{x}$, with covariance C for the landmarks represented at the root. Since no landmark is represented there it is empty $\hat{x} = ()$, $C = ()$. Then proceed down to the BIB. At each node use the estimate for the landmarks represented at that node and the SIB stored at the node to derive an estimate for the landmarks represented at the node's children. Lemma 11 states how to perform this computation and the whole subalgorithm is given in (Fig. 3.4, `computeEstimate`).

Lemma 11. *Let $\chi^2(x)$ be decomposed as in lemma 8 and let $\hat{z}$ be an estimate with covariance C. Then the optimal estimate for y is $\hat{y} = H\hat{z} + h$ with covariance* $\operatorname{cov}\binom{y}{z} = \begin{pmatrix} HCH^T + P^{-1} & HC \\ CH^T & C \end{pmatrix}$.

Proof. By lemma 8, $\chi^2(x)$ is decomposed as

$$\chi^2(x) = \chi^2\begin{pmatrix} y \\ z \end{pmatrix} = \chi^2_{\text{CIB}}(z) + \chi^2_{\text{SIB}}(Hz + h - y). \tag{3.19}$$

So, $\hat{y} = H\hat{z} + h$ is a maximum likelihood estimate for y, since it minimizes $\chi^2_{\text{SIB}}(Hz + h - y)$ and thus minimizes $\chi^2\binom{y}{z}$. Hence $\chi^2_{\text{SIB}}(w) = w^T P w$ expresses the information that $Hz + h = y$ with a covariance of P^{-1}. The estimation covariance for $H\hat{z} + h$ is HCH^T, because C is the covariance for z. Both are independent, so the given estimate for $Hz + h$ and the information about $(Hz + h) - y$ from the SIB can be combined by subtracting

$$y = (Hz + h) - (Hz + h - y) \tag{3.20}$$

$$\implies \operatorname{cov}\begin{pmatrix} \hat{y} \\ \hat{z} \end{pmatrix} = \operatorname{cov}\left(\begin{pmatrix} H\hat{z} + h \\ \hat{z} \end{pmatrix} - \begin{pmatrix} H\hat{z} + h - \hat{y} \\ 0 \end{pmatrix}\right) \tag{3.21}$$

$$= \operatorname{cov}\begin{pmatrix} H\hat{z} + h \\ \hat{z} \end{pmatrix} + \operatorname{cov}\begin{pmatrix} H\hat{z} + h - \hat{y} \\ 0 \end{pmatrix} = \begin{pmatrix} H \\ I \end{pmatrix} C \begin{pmatrix} H \\ I \end{pmatrix}^T + \begin{pmatrix} I \\ 0 \end{pmatrix} P^{-1} \begin{pmatrix} I \\ 0 \end{pmatrix}^T \tag{3.22}$$

$$= \begin{pmatrix} HCH^T & HC \\ CH^T & C \end{pmatrix} + \begin{pmatrix} P^{-1} & 0 \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} HCH^T + P^{-1} & HC \\ CH^T & C \end{pmatrix}. \tag{3.23}$$

□

With lemma 7, 8 and 11 the necessary tools for using a tree map to perform least squares estimation of the map are available. Figure 3.3 gives an example of the operations involved in a three level tree. Lemma 7 and 8 are used from the leaves up to the root and lemma 11 from the root down to the leaves. 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.

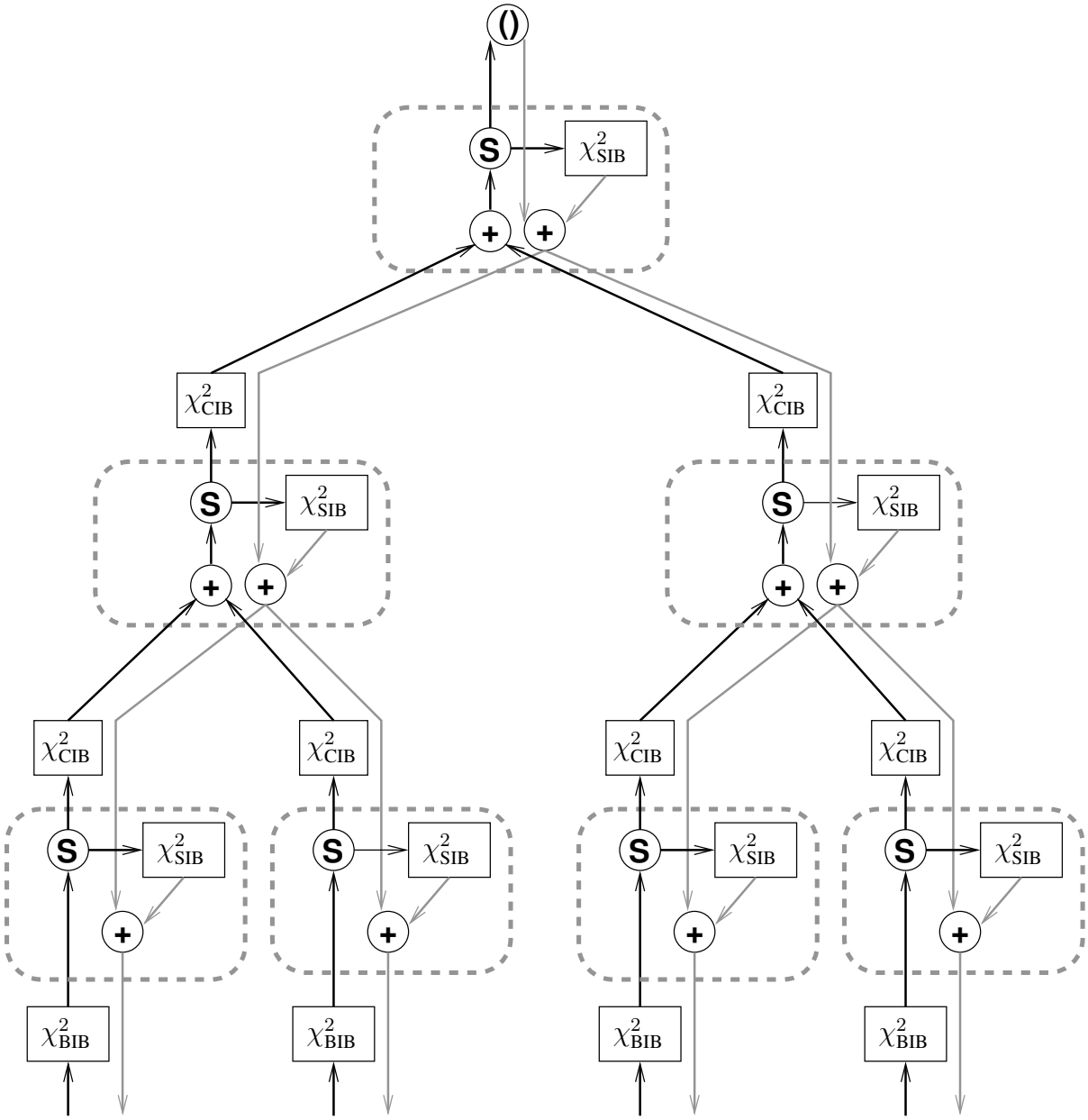


Figure 3.3: Data flow in a three level tree map: A dashed oval represents a tree node. 4 different BIBs are integrated (+) and decomposed (S) into SIBs following the tree upwards (black arrows). Estimates are generated from the stored SIBs following the tree downwards (gray arrows). The (()) mark above the root node indicates that the root node represents no landmark and, thus, the CIB computed by the root node is empty. After changing a BIB only the path from that BIB up to the root and down again needs to be recomputed. Along the upward (black) connection information is passed in information blocks, whereas along the downward connections it is passed as an estimate with corresponding covariance.

Figure 3.4: **computeEstimate** $\big((\hat{z}, C, \alpha), (H, h, P^{-1})\big)$
$\hat{z}$: estimate with covariance C; (H, h, P^{-1}): SIB; α: rotation for SIB

Remove all rows and columns of $\hat{z}, C$ that do no correspond to landmarks represented in H
nonlinear: Apply equation (3.48) with α on H, h, P^{-1} [a]
Apply lemma 11 to compute $\hat{x}, C'$
return $\hat{x}, C'$

[a] See §3.9 on relinearization by nonlinear rotation

3.5 Assumptions on Topologically Suitable Buildings

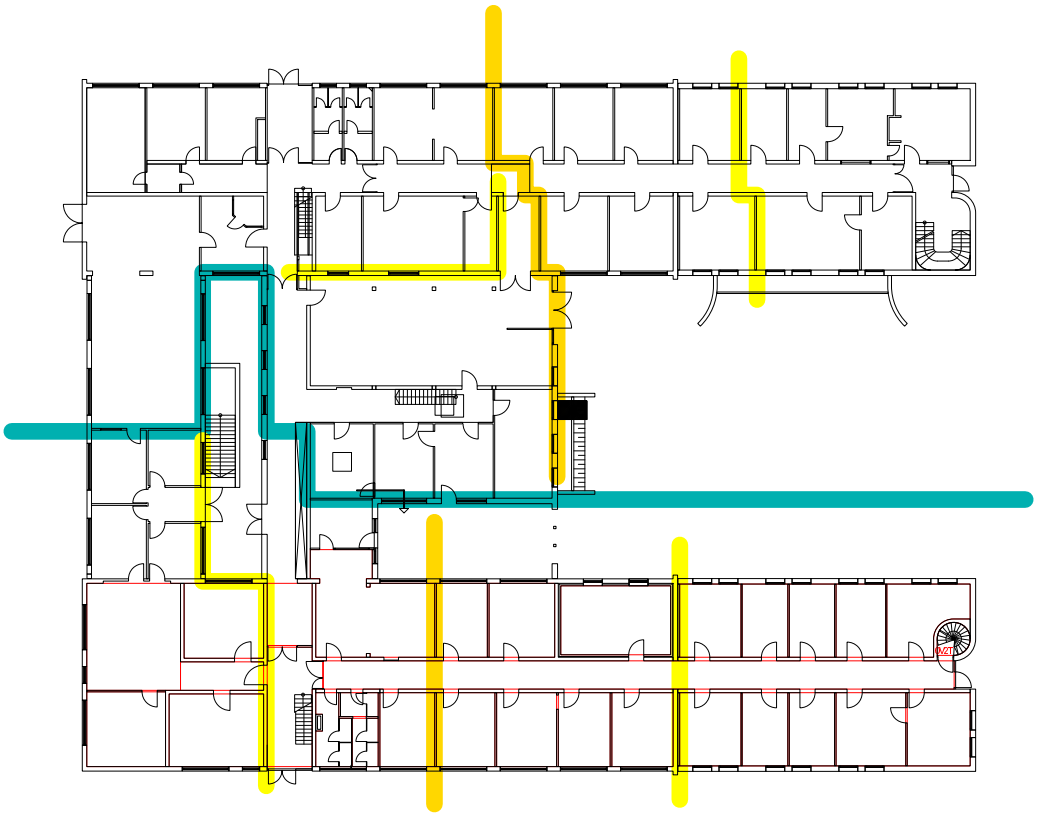
As discussed before, a node in the tree map combines the information from its childrens' CIBs to compute the SIB stored at the node and the CIB passed to the parent node. The time needed for this computation depends on the size of the matrices involved, which is determined by the number of landmarks 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.

It is not generally possible to have such a hierarchical partitioning. But as experiments and the following considerations confirm, this is possible for typical buildings, which will be called *"topologically suitable"*. Certainly, this is not a formal proof, and later an counter-example for a not topologically suitable building will be given.

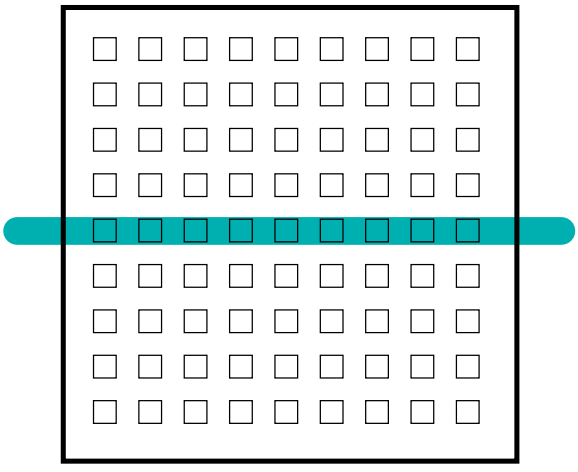
Typical buildings allow such a hierarchical partitioning because they are hierarchical themselves, consisting of floors, corridors and rooms. Different floors are only connected through a few staircases, different corridors through a few crossings and different rooms most often only through a single 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. 3.5a). For all these regions, landmarks observable from outside the region are those located at connections of the region to the rest of the map. As there are only a few connections only a few landmarks will be observable from outside.

To allow a thorough theoretical analysis of the algorithm it is formally assumed that the building is topologically suitable and the algorithm finds a proper partitioning. This is defined in the following:

Definition 5 (Topologically suitable building). *Let the building be decomposed into a hierarchy*



(a) DLR Institute of Robotics and Mechatronics.



(b) Topologically unsuitable building.

Figure 3.5: a) A typical topologically suitable building with the first three level of a suitable hierarchical partitioning. b) Counter-example for a not suitable building, for example a large storeroom. When dividing it into two regions (for instance gray line) there will always be many landmarks observable from both regions.

of regions according to definition 3. 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 and n grows to infinity.

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 (Fig. 3.5b).

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 5 is fulfilled.

Condition 1 is powerful. The fact that buildings have such a loosely connected topology indeed is a key property distinguishing SLAM from many other estimation problems. Exploiting such a property is essential, since recursive least squares is quite a fundamental problem, which has been subject of research for many decades. So one cannot expect to find a universal – not problem specific – solution to reduce computation time from $O(n^2)$ to something as fast as $O(\log n)$.

Computational Efficiency

By condition 2 there are $O(\frac{n}{k})$ nodes in the tree with each node storing matrices of dimension $O(k \times k)$ (condition 1). Thus, the storage requirement of the tree map is $O(k^2 \cdot \frac{n}{k}) = O(nk)$ being linear in the number of landmarks.

Computation time depends on the situation: 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. There are $O(k)$ landmarks represented in the actual BIB, so computation

time therefore is $O(k^2)$ and thus independent from n. This is the same performance as achieved by CEKF [GN02].

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. Each of the involved nodes represents $O(k)$ landmarks by condition 1, so the computation takes $O(k^3)$ per node, with matrix multiplication and inversion being the dominant operations. There are $O(\log n)$ nodes involved in the update, so the overall time is $O(k^3 \log n)$.

Under some special circumstances, more nodes have to be updated so additional computation is necessary for bookkeeping. This is analyzed in detail in §4.6 after the whole algorithm has been developed. The result is $O(k^3 + k^2 \log n)$ for bookkeeping and $O(\log n)$ nodes updated with $O(k^3)$ each, so total computation time of a global update is $O(k^3 \log n)$.

In order to compute an estimate not only for local but for all landmarks, estimate compilation must be performed recursively from the root down to all BIBs taking $O(kn)$. It will turn out in the experiments in §5 that the prefactor involved in this asymptotical expression is extremely small. So while from a theoretical perspective the possibility to perform updates in sublinear time is most appealing, practically the algorithm makes it possible to compute an estimate for all landmarks even for extremely large maps.

Overall the algorithm clearly meets the requirements proposed in §2.13 matching linear storage space requirement (R2) and even exceeding linear update cost requirement (R3).

The algorithm is designed for the case of an extremely large building, so n is asymptotically growing to infinity, whereas k is small and not growing. Technically, if k is not growing, it is bounded and therefore $k = O(1)$. Nevertheless, in the analysis of the algorithm's computation time, the formal dependency on k is retained. The reason therefore is best explained with the algorithms overall complexity of $O(k^3 \log n)$ as an example: The constant hidden in the O of $O(k^3 \log n)$ (about 0.38μs on a Intel Xeon, 2.67 GHz according to the experiments) is moderate. When considering k to be for instance $10 = O(1)$, the constant in the resulting complexity $O(\log n)$ is $k^3 = 1000$ times higher. While being theoretically correct to give $O(\log n)$ as an expression for the runtime complexity, assuming $k = O(1)$, it is practically misleading. Therefore throughout this thesis, k is assumed to be small but not necessarily constant.

Violation of the Assumption

The number of landmarks represented at different nodes determines the algorithm's computation time. There is no influence on the computed result, which is the maximum likelihood estimate

for the integrated information of all BIBs. So if definition 5 is "slightly violated", the algorithm simply takes some more computation time. The main reason for demanding such a strong condition is to allow an easy and theoretically sound analysis of the computation time. A slight violation does not completely break down the algorithm's performance.

If the assumption is only violated locally in a specific area of the map, efficiency will only decrease when the robot is in that area. This is quite an important advantage, since in the majority of cases a violation will be caused by a single large room but not affect the whole map: In an area where condition 1 is violated, different BIBs share more common landmarks than elsewhere in the map. So the algorithm will try to put these BIBs together in a common subtree. All nodes that represent too many landmarks lie in this subtree. The overall area which then corresponds to the the root of the subtree will only have few connections to the rest of the map as any topologically suitable building. So condition 1 holds for the subtree's root node and all other nodes not inside the subtree. Now, if the actual BIB is not inside the subtree, the algorithm will not access the subtree at all and the performance will not be influenced by the violation inside the subtree.

3.6 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. In reality they are normally 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 location relative to a previous robot location.

A possible way to integrate odometry as taken by [LM97, GK99] is to represent and estimate each robot pose as a random variable in the map. This leads to an increasing number of random variables, not only when mapping new parts of the building but even when moving through an area visited before. So this approach violates requirement (R2) (see §2.13) and appears to be impractical for long term operation. The algorithm described in this thesis takes a different approach and does not represent the robot pose, not even the actual one in the tree map at all:

If odometry can be neglected, i.e. the robot's motion is evident from the landmark observation alone, a group of landmark - robot measurements can easily be converted into an IB on the landmarks involved and integrated into a suitable BIB. But the usual situation is, that odometry cannot be neglected. Then an Extended Kalman Filter (EKF) is used as a preprocessing stage for the measurements maintaining an estimate of the robot pose and the local landmarks. When the actual BIB changes the information from the EKF state is decomposed into information about landmarks and about the robot pose. The information about the landmarks is stored in the old

Figure 3.6: **`integrateEKFOdometry`** $\left(z, C_z\right)$
z: odometry measurement; C_z: covariance of z

Apply equation (2.29) to update $\hat{x}, C$

actual BIB. The information about the robot pose is combined with the information from the new actual BIB and stored in the EKF state.

Landmark – Robot Measurements

First assume that odometry can be neglected: 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 a random variables. Then the robot pose is eliminated from the IB using Schur complement (lemma 8). 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. In fact, the information contained in the IB defines the relative location of the involved landmarks, so it is mathematically quite similar to a set of landmark – landmark measurements.

With this approach, no direct information about the robot's movement like odometry is used, i.e. the relative location of different robot poses is only indirectly defined by simultaneously observed landmarks. The precondition of this approach is that at least two common landmarks are being observed from successive robot poses. If this condition is met, the resulting accuracy is often much higher than the one obtainable from odometry alone [GK99]. It is hence no big loss to neglect odometry. Theoretically, it is quite appealing to avoid the use of odometry, since the assumption of stochastical independence between successive odometry measurements is hardly true in reality. If the condition is not met, the result, however, is quite disastrous: The two robot poses will have a completely undefined spatial relation and the map disintegrates into two unrelated parts at this point.

It is possible to heuristically prevent disintegration of the map: In the rare case, when there are less than two commonly observed landmarks, "virtual" measurements on recently observed landmarks are added. These virtual measurements are derived from the last observation of a landmark and odometry. Although this is not a theoretically thorough approach, it will presumably be a good choice in practice.

Figure 3.7: **integrateEKFObservation** (z, C_z)
z: landmark measurements; C_z: covariance of z

Apply equation (2.40), (2.35) to update $\hat{x}$, C

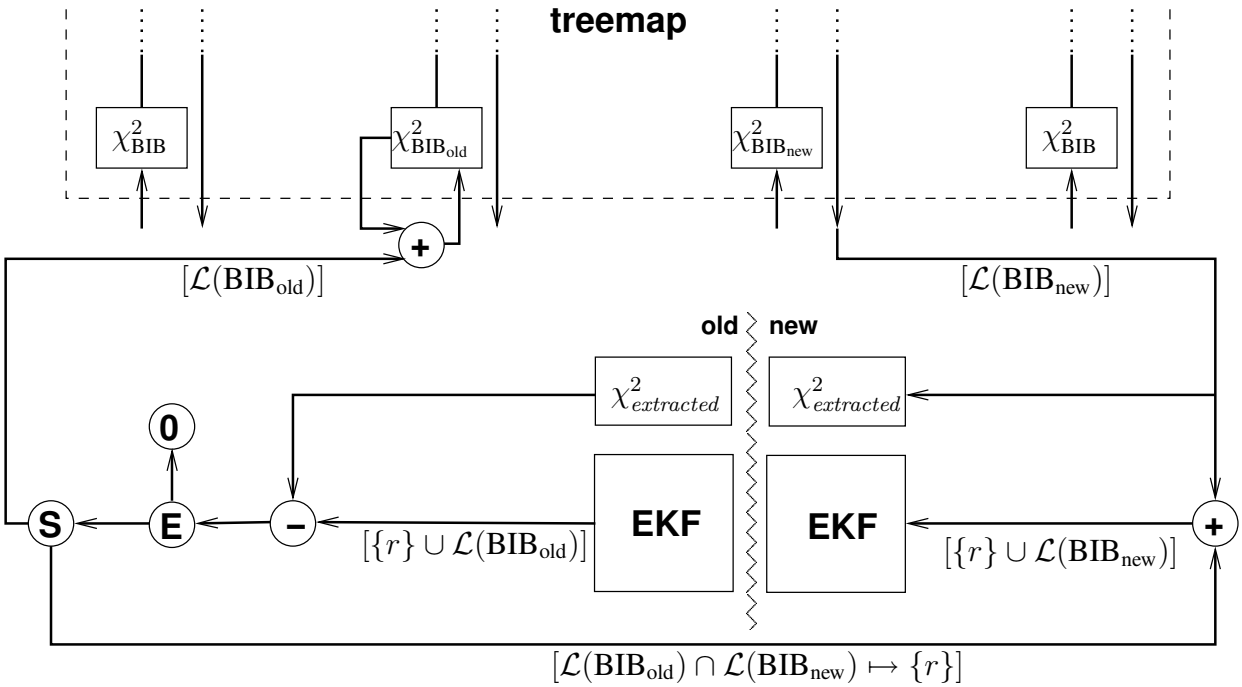


Figure 3.8: Data flow between EKF and tree map when changing the actual BIB from BIB_{old} to BIB_{new}: The information $\chi^2_{extracted}$ is subtracted (**-**) from the EKF state. The couplings between robot pose and landmarks not represented in BIB_{new} are eliminated (**E**) discarding some information (**0**). Then the information about the actual robot pose is separated (**S**). The information about the landmarks is added to BIB_{old} (**+**) and stored. Then the estimate for the landmarks represented in BIB_{new} is computed (see figure 3.3) and combined with the information about the robot pose (**+**). The result is a new EKF state. The landmarks represented in intermediate results are shown in brackets (r = robot pose, $\mathcal{L}(\text{BIB})$ denotes the landmarks represented in BIB). The term $\mathcal{L}(\text{BIB}_{\text{old}}) \cap \mathcal{L}(\text{BIB}_{\text{new}}) \mapsto \{r\}$ refers to the special structure of the separated information about the robot pose, which is a mapping from $\mathcal{L}(\text{BIB}_{\text{old}}) \cap \mathcal{L}(\text{BIB}_{\text{new}})$ to the robot pose with additional uncertainty. Black arrows depict information matrices, gray arrows covariance matrices.

Figure 3.9: **compileEKF** $\big((\hat{z}, C), (P^{-1}, H, h), \alpha\big)$
$(\hat{z}, C)$: landmark estimates; (P^{-1}, H, h): SIB for robot pose; α: rotation for SIB

Store $(z - \hat{z})^T C^{-1} (z - \hat{z})$ as $\chi^2_{extracted}$
Apply lemma 11 to compute $\hat{x}, C'$
return $\hat{x}, C'$

Figure 3.10: **`extractBIBFromEKF`** $\left(\hat{x}_{\text{EKF}}, C_{\text{EKF}}, \mathcal{N}\right)$
$\hat{x}_{\text{EKF}}, C_{\text{EKF}}$: EKF state; $\mathcal{N}$: landmarks represented in the next BIB

$\chi^2(x) := x^T C_{\text{EKF}}^{-1} x + x^T(-2C_{\text{EKF}}^{-1}\hat{x}_{\text{EKF}}) - (1-\epsilon)\chi^2_{extracted}$ [a]
Sort χ^2 to have robot pose in block 1, landmarks $\mathcal{L}(\chi^2) - \mathcal{N}$ in block 2 and landmarks $\mathcal{L}(\chi^2) \cap \mathcal{N}$ in block 3
$\chi^2 :=$ `removeCoupling` $\left(\chi^2, \hat{x}\right)$
nonlinear: Rotate $\chi^2_{actualBIB}$ by `accumulatedAngle` using (3.44) [b]
$\chi^2 := \chi^2 + \chi^2_{actualBIB} - \epsilon\chi^2_{extracted}$
Apply lemma 8 to compute $\chi^2_{\text{CIB}}, \chi^2_{\text{SIB}}, P^{-1}, H, h$. Remove 0 rows / columns of block 2.
nonlinear: Set e according to (3.50)
return (χ^2_{CIB}, e) and $(\chi^2_{\text{SIB}}, P^{-1}, H, h)$

[a] See discussion on rank deficiency in §3.7 for the role of $\epsilon \ll 1$.
[b] See §3.9 and (Fig. 4.8, `accumulatedAngle`) on nonlinear rotation

Robot-Robot Measurements (Odometry)

When odometric measurements have to be integrated, it is necessary to represent the robot pose as a random variable. Keeping all old robot poses would constantly increase the map size even if the robot moves through an area visited before. As this would violate (R2), previous robot poses have to be eliminated. Eliminating a random variable introduces coupling coefficients between all random variables that had coupling coefficients with the eliminated one. Finally the robot pose has coupling coefficients with all landmarks ever observed, ruining the representation size of the tree map.

A conservative approximation is performed to avoid this dilemma. 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. Theorem 2 (§2.11) ensures, that not too much information is discarded, since the coupling coefficients between the robot pose and a landmark decay exponentially with the distance travelled since observation of the landmark.

The measurements are integrated by an EKF as a preprocessing stage. It represents the robot pose and all landmarks represented in the actual BIB and can directly integrate odometry (Fig. 3.6, `integrateEKFOdometry`) and landmark observations (Fig. 3.7, `integrateEKFObservation`). The information about the robot pose is exclusively contained in the EKF and not transferred into the tree map. When a global update becomes necessary all coupling coefficients between the robot pose and landmarks not represented in the new actual BIB are eliminated (Fig. 3.8):

First, the EKF state is converted into an information block χ^2. Then, the information $\chi^2_{extracted}$

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 is the information gained from measurements since the last change of the actual BIB. Next, the elimination of the coupling coefficients is performed by subtracting a SPSD matrix cancelling the necessary coefficients. This means, a part of the information is deliberately discarded to give the remaining information a simpler structure. §3.7 shows how the matrix can be chosen to lose as little information as possible. After this the robot pose is eliminated from the IB by Schur complement (lemma 8) and the resulting CIB is added to the actual BIB (Fig. 3.10, `extractBIBFromEKF`) replacing it in the treemap. The corresponding SIB defines the robot pose as a function of landmarks in the actual BIB. Since all coupling coefficients to landmarks not represented in the new BIB have been eliminated, this function only depends on landmarks represented both in the old and in the new BIB. After the estimate for these landmarks has been generated by updating the tree map, the SIB can be integrated with this estimate (Fig. 3.9, `compileEKF`). 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.

3.7 Stepwise Optimal Elimination of Off-Diagonal Entries

In this section a procedure is derived to eliminate some coupling entries in an information block by subtracting a so called elimination matrix. This means deliberately discarding some information to give the remaining information a simpler structure, but to discard as little information as possible. The key idea therefore is the formalization of the term “as little as possible”, which is discussed in a mathematical analysis. The problem is reduced from a k-dimensional problem to k 1-dimensional problems by eliminating different coupling entries columnwise. For the 1-dimensional problem, a closed optimal solution is derived. The overall solution is suboptimal but should be fairly good since it is made of k optimal solutions.

In the discussion it is assumed that the rows / columns of the considered information matrix A are grouped into 3 block-rows and -columns, so that the block $A_{21} = A_{12}^T$ corresponding to row 2 and column 1 is to be eliminated. The different block-rows / columns correspond to different random variables:

Block-row / column	Random variables
1	Robot pose
2	Landmarks represented in the old but not in the new actual BIB
3	Landmarks represented in both the old and new actual BIB

The block A_{21} contains the coupling coefficients between the robot pose and the landmarks not represented in the new actual BIB. These coefficients have to be eliminated to transfer the robot pose into the new actual BIB. The following definition formalizes the desired elimination matrix:

Definition 6 (Elimination matrix). *Let A be a SPSD 3×3 block matrix. Then an elimination matrix for block A_{21} is a SPSD matrix B with $0 \leq B \leq A$ and $A - B = \begin{pmatrix} * & 0 & * \\ 0 & * & * \\ * & * & * \end{pmatrix}$, $(A-B)_{21} = 0$.*

The first step is to characterize an elimination matrix B that is minimal in the positive definite sense:

$$B \leq B' \quad \Longleftrightarrow \quad x^T B x \leq x^T B' x \; \forall x \quad \text{for all elimination matrices } B'. \tag{3.24}$$

Intuitively $B \leq B'$ means, that in any aspect B contains less information than B' and thus it is always better to use B than B' to eliminate the desired coupling coefficients.

Reduction to the 1-Dimensional Case

Three technical lemmas follow that are needed in lemma 15 and 16 to characterize a minimal elimination matrix. The implications of this characterization are discussed after derivation.

Lemma 12. *Let B, D be SPSD with $0 < D \leq B$. Then there exists a linear combination $B' := B - \lambda_* D$, with $\lambda_* > 0, 0 \leq B'$ and $\operatorname{rank}(B') < \operatorname{rank}(B)$.*

Proof. Without loss of generality, it can be assumed that B is regular because otherwise B and D can be transformed to a basis for $\operatorname{image}(B) \supset \operatorname{image}(D)$. The idea is to subtract the highest multiple of D from B for which the result remains positive semidefinite. It will be shown that the result is singular and thus of smaller rank than B. Formally consider

$$B_\lambda := B - \lambda D, \quad \lambda_* = \max\{\lambda | 0 \leq B_\lambda\}. \tag{3.25}$$

For $\lambda = 0$, $B_\lambda > 0$ and $\lambda_* > 0$. Since $D > 0$, one diagonal entry of D is > 0 and for a sufficiently large λ, $B - \lambda D$ will have a negative diagonal entry not being positive definite any more. Hence λ_* is well defined. For $\lambda > \lambda_*$ the smallest eigenvalue of B_λ will be negative and for $\lambda <= \lambda_*$ positive or 0. Thus by continuity it must be 0 for $\lambda = \lambda_*$ and $B' := B_{\lambda_*}$ is singular and of smaller rank than B. □

Lemma 13. *If B is an elimination matrix for A and there exists a SPSD matrix D, with $D_{21} = 0$ and $0 < D \leq B$, then there exists an elimination matrix $B' < B$ with $\operatorname{rank}(B') < \operatorname{rank}(B)$. This is especially the case if there exists a linear combination $u \neq 0; u \in \operatorname{image}(B)$ of columns of B with $u_1 = 0$ or $u_2 = 0$.*

Proof. Lemma 12 is applied. The result B_{λ_*} is of smaller rank. It still has to be verified that it is an elimination matrix:

$$(A - B_{\lambda_*})_{21} = (A - B + \lambda_* D)_{21} = (A - B)_{21} = 0. \tag{3.26}$$

For the special case: Since $u \in \mathrm{image}(B)$, $u = Bw$ for some w. Set $D := \frac{1}{w^T Bw}(Bw)(Bw)^T$. Since $Bw \neq 0$ it follows, that $w \notin \mathrm{kernel}(B)$, $w^T Bw > 0$ and $0 < D$. By construction $D_{21} = \frac{1}{w^T Bw}(Bw)_2(Bw)_1^T = 0$. Further it follows from Cauchy - Schwarz inequality (appendix A.1) that $D \leq B$, because for every x

$$x^T Dx = \frac{1}{w^T Bw}(x^T Bw)^2 \overset{\text{Cauchy-Schwarz}}{\leq} \frac{1}{w^T Bw}(x^T Bx)(w^T Bw) = x^T Bx. \tag{3.27}$$

So D fulfills the requirements of this lemma and the existence of an elimination matrix with smaller rank is proven. □

Lemma 14. *Let B be an elimination matrix for A and E a matrix with $\mathrm{image}(E) \subset \mathrm{image}(B)$ (for instance any selections of columns of B). If $\mathrm{rank}(E_1) < \mathrm{rank}(E)$ or $\mathrm{rank}(E_2) < \mathrm{rank}(E)$, then there exists an elimination matrix $B' < B$ with $\mathrm{rank}(B') < \mathrm{rank}(B)$.*

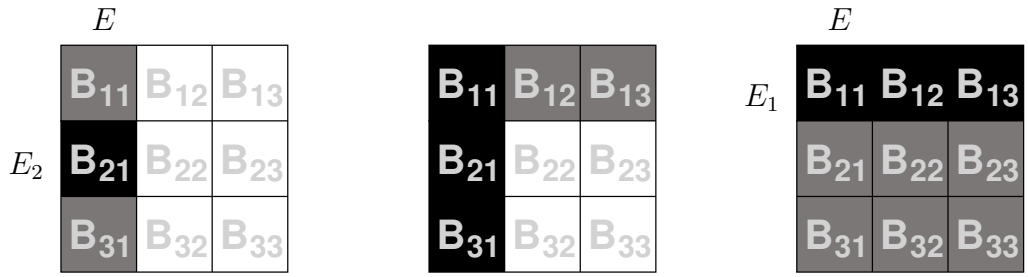
Proof. Assume $\mathrm{rank}(E_1) < \mathrm{rank}(E)$: There are $\mathrm{rank}(E)$ linear independent columns in E. The first block rows of these columns are contained in $\mathrm{image}(E_1)$ which has a dimension smaller than $\mathrm{rank}(E)$. Thus they are linear dependent and there exists a linear combination $0 \neq u \in \mathrm{image}(E)$ of these columns with a zero first block row $u_1 = 0$. Since $\mathrm{image}(E) \subset \mathrm{image}(B)$, the existence of a elimination matrix of smaller rank follows from lemma 13.

If $\mathrm{rank}(E_2) < \mathrm{rank}(E)$, the same argument holds for the second block row. □

The following two lemmas characterize the minimum elimination matrices as elimination matrices with the same rank as the submatrix of eliminated coefficients and establishes that these are incomparable (two of them are neither $\leq$ nor $\geq$).

Lemma 15 (Rank of a minimal elimination matrix). *Let B be an elimination matrix for A. Then there exists an elimination matrix B' with $\mathrm{rank}(B') = \mathrm{rank}(A_{21})$ and $B' \leq B$. In particular, there exists no elimination matrix with a rank of less than $\mathrm{rank}(A_{21})$.*

Proof. Since $B_{21} = A_{21}$, $\mathrm{rank}(B) \geq \mathrm{rank}(A_{21})$. Assume there exists no elimination matrix of rank smaller than $\mathrm{rank}(B)$. Application of lemma 14 with $E := B_{\bullet 1}$ yields that $\mathrm{rank}(B_{\bullet 1}) = \mathrm{rank}(B_{21}) = \mathrm{rank}(A_{21})$. Due to symmetry of B and since rank is invariant under transposing, it follows that $\mathrm{rank}(B_{1\bullet}) = \mathrm{rank}(B_{\bullet 1}^T) = \mathrm{rank}(A_{21})$. Another application of lemma 14 with $E := B$ yields that $\mathrm{rank}(B) = \mathrm{rank}(B_{1\bullet}) = \mathrm{rank}(A_{21})$ (Fig. 3.11). □



(a) $\mathrm{rank}(B_{\bullet 1}) = \mathrm{rank}(B_{21})$ by lemma 14 with $E := B_{\bullet 1}$

(b) $\mathrm{rank}(B_{1\bullet}) = \mathrm{rank}(B_{\bullet 1})$ by symmetry

(c) $\mathrm{rank}(B) = \mathrm{rank}(B_{1\bullet})$ by lemma 14 with $E := B$

Figure 3.11: Three steps of deducing $\mathrm{rank}(B)$ from the rank of $B_{21} = A_{21}$ in lemma 15.

Normally, A_{21} has full rank, so the rank of the sought elimination matrix is the minimum of $\mathrm{col}(A_{21})$ and $\mathrm{row}(A_{21})$. Typically there are different elimination matrices of minimal rank. The following lemma establishes that these are incomparable by $\leq$. So considering two of them the first is better in one aspect and the second is better in another aspect. Thus, an explicit quality measure must be defined to choose the best elimination matrix.

Lemma 16. *Let B, B' be two different elimination matrices for a matrix A. If $\mathrm{rank}(B) = \mathrm{rank}(B') = \mathrm{rank}(A_{21})$, then B and B' are incomparable ($B \not\leq B'$ and $B' \not\leq B$).*

Proof. Assume B and B' would be comparable, so without loss of generality $B \leq B'$. Then $D := B' - B$ satisfies the conditions of lemma 13 and an elimination matrix B'' of smaller rank would exist. This contradicts lemma 15. □

The message of lemma 15 is: When looking for the best elimination matrix for A, only those of rank equal to $\mathrm{rank}(A_{21})$ need to be considered, since every other elimination matrix is worse in *all* aspects. There are still several choices and by lemma 16 each of them is better in one aspect and worse in another aspect. In order to decide which of the minimal rank elimination matrices to choose, a quality measure has to be defined. This is quite difficult for the n-dimensional case, so the elimination is performed stepwise by eliminating A_{21} one column after the other. For the 1-dimensional problem of eliminating a single column, a quality measure can be naturally defined and the optimum elimination matrix with respect to this measure can be determined:

Optimum Solution for the 1-Dimensional Case

Assume that the block to be eliminated consists of one column $A_{21} = r$: By lemma 15, a minimum elimination matrix has rank 1 and can be written as $B = xx^T$. In order to compare different choices of x the expression $x^T A^{-1} x$ is used as a measure of the relative loss of information. This idea will be substantiated by the next lemma. It follows the formalization of requirement (R1) as discussed in §2.13 and compares the uncertainty of some aspect g of the map before and after applying the elimination matrix. The uncertainty before application is $g^T A^{-1} g$ and after application it is $g^T (A - xx^T)^{-1} g$. The result of the lemma is that uncertainty grows inverse proportionally to $1 - x^T A^{-1} x$. This shows that $x^T A^{-1} x$ measures the relative part of the information lost by subtracting xx^T from A:

Lemma 17. *Let A be an SPD matrix and x a vector. If $x^T A^{-1} x < 1$, then $A - xx^T$ is SPD and for any g, it holds*

$$g^T (A - xx^T)^{-1} g \leq \frac{1}{1 - x^T A^{-1} x} g^T A^{-1} g \tag{3.28}$$

with equality for $g = x$.

Proof. The proof applies Cauchy - Schwarz inequality (A.1) and Sherman - Morrison (A.2) formula to evaluate $(A - xx^T)^{-1}$. The formula ensures regularity of $A - xx^T$ if and only if $x^T A^{-1} x \neq 1$. So $A - xx^T$ is SPD as long as $x^T A^{-1} x < 1$ and its inverse is computed as

$$g^T (A - xx^T)^{-1} g \overset{\text{Sherman - Morrison}}{=} g^T A^{-1} g + g^T \frac{A^{-1} x \, x^T A^{-1}}{1 - x^T A^{-1} x} g \tag{3.29}$$

$$= g^T A^{-1} g + \frac{(g^T A^{-1} x)^2}{1 - x^T A^{-1} x} \overset{\text{Cauchy - Schwarz}}{\leq} g^T A^{-1} g \left(1 + \frac{x^T A^{-1} x}{1 - x^T A^{-1} x} \right) \tag{3.30}$$

$$= \frac{1}{1 - x^T A^{-1} x} g^T A^{-1} g. \tag{3.31}$$

In (3.30) and thus in (3.28) equality holds, if $x = g$. □

Minimizing the relative information loss $x^T A^{-1} x$ is a good optimization criterion, since it corresponds to requirement (R1) (§2.13) demanding the uncertainty of any aspect of the map being comparable to the uncertainty that could be derived from the measurements. In contrast to this approach, using the absolute loss of information $\operatorname{tr}(xx^T) = x^T x$ as an optimization measure violates (R1), since the same absolute information loss can only be a small fraction of the information available or nearly all information represented in a map. In the latter case uncertainty would grow so high that the map is nearly unusable.

Lemma 18. *The minimum for the expression $\begin{pmatrix} \lambda \\ \lambda^{-1} r \end{pmatrix}^T \begin{pmatrix} \psi & r^T \\ r & S \end{pmatrix}^{-1} \begin{pmatrix} \lambda \\ \lambda^{-1} r \end{pmatrix}$ is $\lambda = \sqrt[4]{\psi (r^T S^{-1} r)}$.*

The proof of this lemma can be found in appendix B. The following theorem is the central result of this section yielding the optimal elimination matrix in the sense defined above:

Theorem 3. *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*

$$x = A \begin{pmatrix} \gamma \\ \delta S^{-1} r \\ 0 \end{pmatrix}, \text{with} \quad \begin{array}{c} \alpha = r^T S^{-1} r, \quad \beta = (\psi - \alpha)^{-1}, \\ \lambda = \sqrt[4]{\psi (r^T S^{-1} r)},\ \gamma = \beta(\lambda - \lambda^{-1}\alpha),\ \delta = \beta(-\lambda + \psi\lambda^{-1}). \end{array} \tag{3.32}$$

Proof. The block r to be eliminated is 1-dimensional, so by lemma 15 all minimum elimination matrices have rank 1 and can be written as $B = xx^T$ for a vector x. Since $x_2 x_1^T = B_{21} = r$ and x_1 is 1-dimensional it follows that $x_2 = r/x_1$. So x can be parameterized as:

$$x := \begin{pmatrix} \lambda \\ \lambda^{-1} r \\ z \end{pmatrix}, \quad B := xx^T = \begin{pmatrix} \lambda^2 & r^T & \lambda z \\ r & \lambda^{-2} r r^T & \lambda^{-1} r z^T \\ \lambda z & \lambda^{-1} z r^T & z z^T \end{pmatrix} \tag{3.33}$$

The expression to be optimized is $x^T A^{-1} x$ with respect to λ and z. The first step is to optimize with respect to z by applying lemma 9, with $y = \left(\begin{smallmatrix} \lambda \\ \lambda^{-1} r \end{smallmatrix}\right)$ given. So block row 3 corresponds to the optimized part and block row 1 and 2 to the fixed part of x. In order to apply lemma 9 formally , A must be decomposed as a 2×2 block matrix

$$x = \begin{pmatrix} \left(\begin{smallmatrix} \lambda \\ \lambda^{-1} r \end{smallmatrix}\right) \\ z \end{pmatrix}, \quad A = \begin{pmatrix} \left(\begin{smallmatrix} \psi & r^T \\ r & S \end{smallmatrix}\right) & \left(\begin{smallmatrix} w^T \\ W^T \end{smallmatrix}\right) \\ \left(\begin{smallmatrix} w & W \end{smallmatrix}\right) & X \end{pmatrix}. \tag{3.34}$$

The resulting minimum is

$$y^T \begin{pmatrix} \psi & r^T \\ r & S \end{pmatrix}^{-1} y \quad \text{at} \quad x = A \begin{pmatrix} \left(\begin{smallmatrix} \psi & r^T \\ r & S \end{smallmatrix}\right)^{-1} y \\ 0 \end{pmatrix} \tag{3.35}$$

$$\text{hence} \quad \begin{pmatrix} \lambda \\ \lambda^{-1} r \end{pmatrix}^T \begin{pmatrix} \psi & r^T \\ r & S \end{pmatrix}^{-1} \begin{pmatrix} \lambda \\ \lambda^{-1} r \end{pmatrix} \quad \text{at} \quad x = A \begin{pmatrix} \left(\begin{smallmatrix} \psi & r^T \\ r & S \end{smallmatrix}\right)^{-1} \left(\begin{smallmatrix} \lambda \\ \lambda^{-1} r \end{smallmatrix}\right) \\ 0 \end{pmatrix}. \tag{3.36}$$

The minimum with respect to λ is found by applying lemma 18 and expanding $\left(\begin{smallmatrix} \psi & r^T \\ r & S \end{smallmatrix}\right)^{-1} \left(\begin{smallmatrix} \lambda \\ \lambda^{-1} r \end{smallmatrix}\right)$ using the block matrix inversion formula (appendix. A.2):

$$\alpha = r^T S^{-1} r, \quad \beta = (\psi - \alpha)^{-1}, \quad \lambda = \sqrt[4]{\psi (r^T S^{-1} r)}, \tag{3.37}$$

$$\begin{aligned}\begin{pmatrix}\psi & r^T\\ r & S\end{pmatrix}^{-1}\begin{pmatrix}\lambda\\ \lambda^{-1}r\end{pmatrix} &= \begin{pmatrix}\beta & -\beta(r^TS^{-1})\\ -(S^{-1}r)\beta & S^{-1}+(S^{-1}r)\beta(r^TS^{-1})\end{pmatrix}\begin{pmatrix}\lambda\\ \lambda^{-1}r\end{pmatrix}\\ &= \begin{pmatrix}\lambda\beta-\lambda^{-1}\beta\alpha\\ (S^{-1}r)\,(-\lambda\beta+\lambda^{-1}(1+\beta\alpha))\end{pmatrix}\\ &= \begin{pmatrix}\beta(\lambda-\lambda^{-1}\alpha)\\ (S^{-1}r)\beta\,(-\lambda+\psi\lambda^{-1})\end{pmatrix}\end{aligned} \quad . \tag{3.38}$$

The last equation follows from observing $1+\beta\alpha = \beta\psi$. The final result (3.32) is obtained by substituting (3.38) into (3.36). □

Solution for the n-Dimensional Case

Theorem 3 allows to eliminate one column of coupling coefficients. So the idea for eliminating a whole block A_{21} of coupling coefficients is to apply theorem 3 successively to each column of A_{21} (Fig. 3.12, `removeCoupling`). However, care must be taken, not introduce coefficients into columns already eliminated. This means that the corresponding entries of x must be 0 when optimizing $x^TA^{-1}x$. This is equivalent to removing those rows / columns from A^{-1}. Also equivalent, but easier to implement is the following approach iterating through all columns i of A_{21}: First compute an elimination matrix $x_ix_i^T$ for column i with block row 2 by theorem 3. Then apply lemma 10 to eliminate the rest of that column by a second elimination matrix $w_iw_i^T$. This forces the following elimination matrices to be 0 in column / row i. The final elimination matrix returned as result is $\sum_i x_ix_i^T$.

In order to apply theorem 3 it is necessary to compute A_{22}^{-1}. An efficient way for this is updating the inverse after each change in A using the Sherman-Morrison formula (appendix A.2). Subtraction of the second elimination matrix $w_iw_i^T$ does not affect A_{22}, since $w_{i2}=0$, so no update is necessary.

Rank deficiency

In the overall algorithm matrix A is the information gained since the last change of the actual BIB. Sometimes A can be rank deficient, being SPSD but not SPD anymore. This happens when some landmark is represented in the old actual BIB but has not been observed since the last change. Theoretically this is not too much of a problem: If $xx^T \le A$, x must be orthogonal to $\mathrm{kernel}(A)$. Otherwise, there would exist a $y \in \mathrm{kernel}(A)$, with $x^Ty \neq 0$ and

$$y^T(xx^T)y = (y^Tx)^2 > 0 = y^TAy \tag{3.39}$$

Figure 3.12: **removeCoupling** $(A,\ b,\ \hat{x})$

A, b: IB $x^TAx + x^Tb$, block A_{21} of A to be eliminated; $\hat{x} = -A^{-1}b/2$

IF	$\mathrm{col}(A_{21}) > \mathrm{row}(A_{21})$
THEN	exchange block row / column 1 and 2 in A, b
$W := A; \quad S := W_{22}; \quad S^I := S^{-1}$	
FOR $i = 1 \ldots \mathrm{col}(A_{21})$	
	Find elimination matrix xx^T for row i of W_2 by theorem 3. (Using S^I as S^{-1})
	$A := A - xx^T; \quad W := W - xx^T; \quad b := b - 2x(x^T\hat{x})$
	$S := S - x_2x_2^T; \quad S^I := S^I + \frac{1}{1-x_2^TS^Ix_2}(S^Ix_2)(S^Ix_2)^T$
	Find elimination matrix ww^T for row / column i of W by lemma 10
	$W := W - ww^T$
return A, b	

contradicting $xx^T \leq A$. So the problem could be reduced to the regular part of A by transforming A into a basis for $\mathrm{image}(A)$. However, such an approach is practically cumbersome, since due to round-off errors A never is exactly singular. In this thesis, a heuristical solution is applied (Fig. 3.10, `extractBIBFromEKF`): When computing the information gained through measurements only $(1-\epsilon)\chi^2_{extracted}$ instead of $\chi^2_{extracted}$ is subtracted. The result is $\geq \epsilon\chi^2_{extracted}$ and thus SPD. After performing the elimination the missing $\epsilon\chi^2_{extracted}$ is subtracted. The result can be slightly negative. The algorithm requires only the sum of all BIBs not each individual BIB to be positive definite, so this does not pose a problem. In the experiments for this thesis $\epsilon = 0.001$ was chosen.

3.8 Approximation Quality

Structure of the Discarded Information

Consider again the example of figure 3.1 with three landmarks a, b, c decomposed into $\chi^2_1(a,b)$ and $\chi^2_2(b,c)$. Assume the robot r observes the landmarks in the following sequence (Fig. 3.13):

$$\text{landmark } a \Rightarrow \text{odometry } 1 \Rightarrow \text{landmark } b \Rightarrow \text{odometry } 2 \Rightarrow \text{landmark } c$$

The combination of observation a, odometry 1 and observation b gives information on the relation $a-b$ stored in χ^2_1. To transfer the robot pose into χ^2_2 it is expressed relative to b, since a is not represented in χ^2_2. Therefore the relations $a-r$ and $b-r$ are converted into relations $a-b$ and $b-r$. Thereby some part of the information is sacrificed, since $a-b$ and $b-r$ both involve information from the original $b-r$ measurement, so the information must be split. The relation $b-r$ defines the robot pose relative to landmark b and is used to transfer this information into

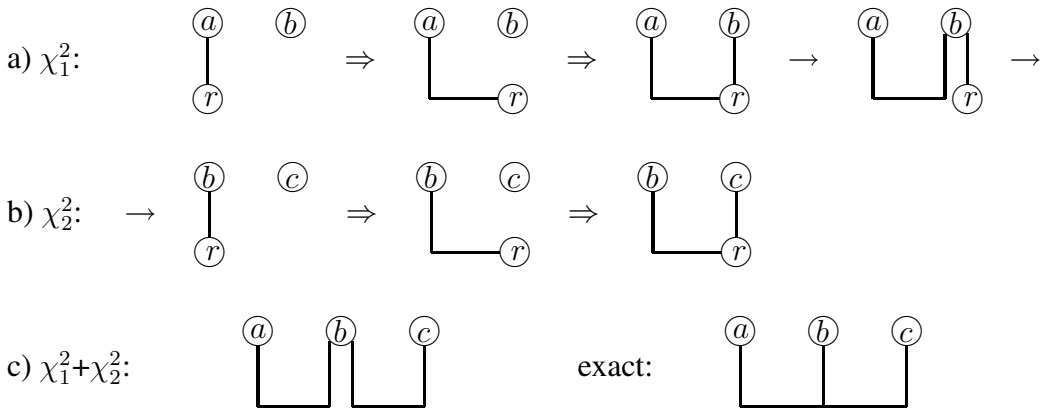


Figure 3.13: Approximation when changing a BIB: a) Operations on χ_1^2: observation of a, movement of robot r, observation of b, elimination of the coupling between a and r (the information $b-r$ is implicitly split). b) Operations on χ_2^2: transfer of r, movement, observation of c c) Information in $\chi_1^2 + \chi_2^2$ vs exact information: The uncertainty of $a - c$ in $\chi_1^2 + \chi_2^2$ includes > 2 times the uncertainty of $b - r$, which does not affect $a - c$ in an exact representation.

χ_2^2. Combining the relation with odometry 2 and observation c gives information on the relation $b - c$ stored in χ_2^2.

In comparision with [Fed99] the strength of the algorithm lies in the exactness of the remaining computation, once information has been stored in a BIB. Implicit information like the relative position $a - c$ in the example is handled correctly, thus preventing any gap or break in the map. Consider the information on a, b, c actually discarded. The information gained from observation b is split into two parts: One of them is integrated into χ_1^2 for the relation $a - b$ and the other one is integrated into χ_2^2 for the relation $b-c$. From the view of each relation the effective uncertainty of observation b is increased by a factor $\sqrt{2}$. If the odometry error is large compared to the error of observing a landmark, the additional error introduced by the approximation is comparatively small. The relation $a - c$ is affected in a similar way: It is implicitly derived by the algorithm as $(a - b) + (b - c)$ and thus contains two times the uncertainty of the split observation b, being about twice as large as its original uncertainty. Again, this additional error must be compared to the odometric error (odometry 1 and 2).

This example illustrates that the quality of the approximation depends on the ratio between the uncertainty of landmark observation and and the odometric uncertainty of a distance about the size of one region. Usually, mobile robots have pretty large odometry errors, so rather small regions should already achieve good results.

In the experiments in §5 and §6 the regions have a diameter of at most 5m and 7m.

The Necessity of Discarding Information

After separation of information about local landmarks, the remaining information about the robot pose can be uniquely expressed as a substitution by the local landmarks' positions plus additional uncertainty (lemma 8). This means that the robot pose is defined relative to the local landmarks, i.e. if an estimate for all local landmarks is given, an estimate for the robot pose can be derived. If one of the landmarks is not available, the substitution is without information. This can be verified by the following argument: If information on the robot pose is available with or without information on some landmark, then this landmark is either completely uninvolved or the information on the robot pose is redundant, generating implicit information on the landmark. This is a contradiction that all information about the landmarks has been separated in the beginning.

When changing the actual BIB, some landmarks represented in the old actual BIB are not represented in the new one. So the information on the robot pose remaining after separation of all information about the landmarks is of little use in the new BIB. In order to obtain information only involving landmarks represented in the new actual BIB some information on landmarks represented in the old actual BIB must be sacrificed. As a consequence the estimate is a conservative approximation of the optimal least squares estimate. This problem of conservatively transferring the robot pose between different regions is the fundamental problem for all submap based approaches [Fed99].

Requirement (R1)

In requirement (R1) *Bounded Uncertainty*, postulated in §2.13, it has been pointed out that approximations in a SLAM algorithm certainly may diminish precision in all aspects of the map but should not lead to a nearly complete loss of information in any aspect.

As established by lemma 17, the minimization criterion used by the algorithm is equivalent to relative growth of covariance. So each individual elimination operation respects the demand of (R1) to the largest possible extent. The remaining question is whether the sequence of elimination operations performed while the robot is moving through the building still complies with (R1). In this thesis, an experimental investigation (§5.2) of this question has been carried out. Furthermore the following general argument explains why the algorithm can be expected to behave according to (R1):

Each measurement is affected by the elimination operation only once, i.e. the next time when the actual BIB is changed. As long as the fraction of information lost in each elimination op-

eration is bounded, the fraction of information lost on the whole map holds the same bound. Consequently the map is also topologically consistent: If a pair of landmarks is being observed to be close to each other with suitable precision this information will be stored in the actual BIB with a slightly smaller but nonetheless significant precision. Once stored it will remain in the BIB and all following estimates will predict those landmarks to be close to each other. This is a highly important property of structural quality not met by several other submap based approaches, if two landmarks are represented in different submaps [Fed99, Jen01].

3.9 Relinearization by Nonlinear Rotation

SLAM, the problem of estimating a map from odometry and landmark observations is essentially nonlinear. Nearly all approaches linearize the measurement functions and are thus subject to linearization error. As pointed out in §2.9 there are two sources of nonlinearity although only the *orientation* nonlinearity poses a problem. It severely distorts the map, when the robot's orientation uncertainty gets larger than about $\approx 15^\circ$ (see §2.9).

A straightforward way to handle nonlinearities is to use an iterative nonlinear least squares algorithm like Levenberg-Marquardt [PTVF92, §12]: After computing a linearized estimate all measurement Jacobians have to be evaluated at the new estimate (*relinearized*) and the whole process is iterated until convergence. When different measurements have been integrated into the same information block or covariance matrix, it is not possible to relinearize one of them any more. At first sight this approach seems to require storing all measurements making the size of the map representation grow even when moving through an area visited before and thus violates requirement (R2) (§2.13). Furthermore, relinearizing several measurements is a type of update that cannot be performed efficiently by EKF equations. Handling nonlinearity appears to be computational expensive. This probably is the reason why the problem is not much addressed in literature [JU96, FHBH00].

The tree map data structure used in the algorithm of this thesis is ideally suited to deal with orientation nonlinearity. Since the measurements are integrated into BIBs the problem of growing map representation is avoided. As distinct BIBs are still independent, a nonlinear rotation can be applied to an individual BIB compensating for the orientation error of the BIB as a whole:

Relinearizing an Information Block

All measurement functions f are independent of translation and rotation:

$$f(x) = f(\mathrm{Rot}_\alpha\, x), \tag{3.40}$$

with Rot_α consisting of $\left(\begin{smallmatrix}\cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha\end{smallmatrix}\right)$ diagonal blocks for each landmark represented by x. By taking the derivative of the equation, it can be seen that the Jacobian of f at a rotated linearization point $\mathrm{Rot}_\alpha\, x$ is the rotated Jacobian at x:

$$\frac{\partial f(x)}{\partial x} = \frac{\partial (f(\mathrm{Rot}_\alpha\, x))}{\partial x} = \frac{\partial f(x)}{\partial x}\Big|_{\mathrm{Rot}_\alpha\, x} \cdot \mathrm{Rot}_\alpha \tag{3.41}$$

$$\frac{\partial f(x)}{\partial x}\Big|_{\mathrm{Rot}_\alpha\, x} = \frac{\partial f(x)}{\partial x} \cdot \mathrm{Rot}_{-\alpha}\,. \tag{3.42}$$

Thus, when computing a BIB using a linearization point rotated by α, the result is a BIB rotated by $-\alpha$:

$$\chi^2(x) = x^T A x + x^T b \tag{3.43}$$

$$\chi'^2(x) := \chi^2(\mathrm{Rot}_{-\alpha}\, x) = x^T \underbrace{(\mathrm{Rot}_{-\alpha}^T\, A\, \mathrm{Rot}_{-\alpha})}_{A'} x + x^T \underbrace{(\mathrm{Rot}_{-\alpha}^T\, b)}_{b'}\,. \tag{3.44}$$

Such a rotation can be applied to compensate for the landmarks of the BIB being rotated in the estimate. This way the measurements are approximately relinearized at a new estimate without actually recomputing and integrating all measurement Jacobians. Considering the linearization error, the different BIBs are now independent and the situation is like having many small maps instead of one large map. The remaining linearization error is the one each BIB would have, if seen as an individual map. This error is much smaller and probably even negligible. There is one exception to this rule i.e. the BIB containing the information of the initial robot pose being $(0, 0, 0)$. This information is absolute and hence not rotation invariant. So the BIB and all its ancestors cannot be relinearized. However, since the orientation of the BIB is fixed, it is never subject to significant linearization error anyway.

The general idea can be applied in a straightforward way by relinearizing some BIBs with large error and consequently updating the CIB and SIB of all ancestor nodes. If a large loop is closed many BIBs are rotated and are suddenly subject to linearization errors. Instantly relinearizing all these BIBs takes more than $O(k^3 \log n)$ and at most $O(k^2 n)$ computation time. In order to avoid this problem only a few BIBs are relinearized in each step. So final iteration to convergence is carried out in many small steps interleaved with the robot moving on. Thus, the SLAM algorithm is not blocked giving an preliminary estimate but it takes some time until the final result is available.

This process can be accelerated considerably by relinearizing not only BIBs but also internal nodes of the tree, removing the common orientation error of a large region. If an internal node is relinearized the orientation error of the region of that node as a whole is compensated, whereas the relative orientation error of the different subregions remains. Since the linearization error grows quadratically with the error in orientation used for computing the Jacobians, even such

a first step will reduce the error in the whole region drastically. Further relinearization steps can follow to individually relinearize subregions reducing the error even more. This way the linearization error decays more rapidly.

Normally, when rotating a node, all SIBs of descendants of that node would also have to be rotated, but this would take too much computation time. Instead, the rotation angle is stored at the node and applied to the SIB, when it is actually needed to compute an estimate applying lemma 11. The necessary computations for rotating a SIB χ^2_{SIB} described by the parameter H, h, P^{-1} are given by

$$\chi^2_{\text{SIB}}\begin{pmatrix} z \\ y \end{pmatrix} = (Hz + h - y)^T P (Hz + h - y) \tag{3.45}$$

$$\chi'^2_{\text{SIB}}(x) = \chi^2_{\text{SIB}}(\text{Rot}_{-\alpha} x) = \chi^2_{\text{SIB}}\begin{pmatrix} \text{Rot}_{-\alpha} y \\ \text{Rot}_{-\alpha} z \end{pmatrix} \tag{3.46}$$

$$= \Big(H(\text{Rot}_{-\alpha} z) + h - (\text{Rot}_{-\alpha} y)\Big)^T P \Big(H(\text{Rot}_{-\alpha} z) + h - (\text{Rot}_{-\alpha} y)\Big). \tag{3.47}$$

Since rotation matrices are orthogonal, $\text{Rot}_{-\alpha}\,\text{Rot}^T_{-\alpha} = I$ and $\chi'^2_{\text{SIB}}(x)$ is

$$= \Big(\underbrace{\text{Rot}^T_{-\alpha}\, H\, \text{Rot}_{-\alpha}}_{H'} z + \underbrace{\text{Rot}^T_{-\alpha}\, h}_{h'} - y\Big)^T \cdot \underbrace{\text{Rot}^T_{-\alpha}\, P\, \text{Rot}_{-\alpha}}_{P'} \cdot \Big(\text{Rot}^T_{-\alpha}\, H\, \text{Rot}_{-\alpha}\, z + \text{Rot}^T_{-\alpha}\, h - y\Big) \tag{3.48}$$

$$P'^{-1} = \text{Rot}^T_{-\alpha}\, P^{-1}\, \text{Rot}_{-\alpha}\,. \tag{3.49}$$

Updating the estimate after relinearizing a BIB only requires processing up to the root and down to the actual BIB again ($O(k^3 \log n)$). Since all ancestor nodes of the relinearized BIB have to be recomputed anyway, the sibling nodes of these ancestors can be relinearized without additional cost. This method is very powerful because even after closing a large loop the linearization error can be drastically reduced by a single $O(k^3 \log n)$ operation.

Tracking the Linearization Point

The algorithm keeps track of the linearization points used in the different IBs of the tree map. This is being done heuristically: Formally, the measurement functions and thus the linearization points involve the different robot poses which are not represented in the map at all. So storing the actual linearization points is useless. Instead, for each BIB the algorithm stores the estimate for landmarks' positions as reported by the EKF after integrating all measurements into this BIB. Even though the absolute orientation of the robot is usually quite uncertain, the error of the orientation relative to landmarks is quite low ($< 5°$) and – what matters most – is not accumulating

(§2.2). As discussed in §2.9 the linearization error induced thereby is very small ($< 0.4\%$) and therfore usually negligible compared to the stochastical error of the estimate. Thus, whenever two IBs are to be integrated they are both rotated before, so that the linearization point is aligned as well as possible with the most recent estimate.

For each IB χ^2 the algorithm stores a triplet $e := (x^0, w, e^0)$. The first component x^0 is the vector of landmark positions reported by the EKF while integrating measurements into this BIB. The second component w is a weight vector defining the importance of the different landmarks while aligning x^0 with the most recent estimate $\hat{x}$. The last component heuristically measures the linearization error made during computation of that IB. The triplet e is initialized from the estimate $\hat{x}$ of the EKF and from the information $\chi^2(x) = x^T Ax + x^T b$ transferred from the EKF to the actual BIB during a global update (Fig. 3.10, `extractBIBFromEKF`):

$$e := (\hat{x}, w, 0), \quad \text{with} \quad w_l := \operatorname{tr} A_{ll} \quad \forall l = 1 \dots k. \tag{3.50}$$

As usual matrix A and vector x consist of k blocks of size 2×2 and 2 respectively each representing a landmark. The expressions A_{ll} and x_l refer to the block corresponding to landmark l. The weight vector w reflects the amount of information on different landmarks in χ^2 using the trace of the diagonal block A_{ll}. This block is the information matrix for landmark l, if all other landmarks were known. So its trace is a heuristical value for defining the relative importance of landmark l. The reason for introducing a weight vector is to prevent inprecisely measured landmarks from disturbing the rotation angle.

When an IB is rotated by α and moved by d, the linearization point x^0 is rotated and moved accordingly resulting in

$$x'^0 := \operatorname{Rot}_\alpha x^0 + \operatorname{Trans}_d, \quad e' := (x'^0, w, e^0). \tag{3.51}$$

Here Trans_d is a vector of k blocks with each block equal to d. Adding Trans_d moves all landmarks by d. As the IBs are translation invariant and as translation is a linear operation, χ'^2 is not affected by d only by α as in (3.44).

For a vector x the weighted squared distance between x and x^0 defines a measure for the linearization error at x:

$$e(x) := e^0 + \sum_{i=1}^{k} w_i (x_i^0 - x_i)^T (x_i^0 - x_i). \tag{3.52}$$

Obviously, the minimum of $e(x)$ is e^0 at the linearization point x^0. When two IBs χ_1 and χ_2 are integrated the corresponding linearization error functions $e_1(x)$ and $e_2(x)$ described by

(x_1^0, w_1, e_1^0) and (x_2^0, w_2, e_2^0) are added. The result

$$e'(x) := e_1(x) + e_2(x) = \sum_{i=1}^{k} w_{1i}(x_{1i}^0 - x_i)^T(x_{1i}^0 - x_i) + \sum_{i=1}^{k} w_{2i}(x_{2i}^0 - x_i)^T(x_{2i}^0 - x_i) \quad (3.53)$$

$$\begin{aligned} &= \sum_{i=1}^{k} \underbrace{(w_{1i} + w_{2i})}_{w_i'} \left(\underbrace{\frac{w_{1i}x_{1i}^0 + w_{2i}x_{2i}^0}{w_{1i} + w_{2i}}}_{x_i'^0} - x_i \right)^T \left(\frac{w_{1i}x_{1i}^0 + w_{2i}x_{2i}^0}{w_{1i} + w_{2i}} - x_i \right) \\ &\quad + \underbrace{e_1^0 + e_2^0 + \sum_{i=1}^{k} \frac{w_{1i} + w_{2i}}{w_{1i}w_{2i}} \left(x_{1i}^0 - w_{2i}^0\right)^T \left(x_{1i}^0 - w_{2i}^0\right)}_{e'^0} \end{aligned} \quad (3.54)$$

again has the same structure which can easily be verified by expanding $e_1(x)$, $e_2(x)$ and $e'(x)$ but is omitted here for lack of space. It is compatible to (3.52) and can be described by a triplet (x'^0, w', e'^0), x'^0 being interpreted as the linearization point for the IB resulting from integration of χ_1^2 and χ_2^2. The inconsistency between x_1^0 and x_2^0 is reflected by e'^0 which therefore measures the linearization error already inherent in the resulting IB. Prior to integration both IBs are rotated, so x_1^0 and x_2^0 are both aligned with $\hat{x}$. This implicitly minimizes e'^0. [6]

Determination of the Rotation Angle

Before the integration of two IBs (§3.3) both are moved and rotated by applying (3.44) and (3.51), so the linearization error at the most recent estimate $e'(\hat{x})$ is minimal. The following lemma gives an analytical solution to this problem. The proof is given in appendix B.

Lemma 19. *Let x^0 and $\hat{x}$ be two vectors of landmark positions and w a corresponding weight vector. Let*

$$e_{\alpha,d}(\hat{x}) := \sum_{i=1}^{k} w_i(x_i'^0 - \hat{x}_i)^T(x_i'^0 - \hat{x}_i), \quad \textit{with } x'^0 := \mathrm{Rot}_\alpha\, x^0 + \mathrm{Trans}_d \quad (3.55)$$

be the weighted squared distance between $\hat{x}$ and x^0 rotated by α and moved by d. Then the α, d combination, that minimizes $e_{\alpha,d}(\hat{x})$ is

$$\bar{\hat{x}} := \frac{\sum_{i=1}^k w_i\hat{x}_i}{\sum_{i=1}^k w_i}, \quad \bar{x^0} := \frac{\sum_{i=1}^k w_i x_i^0}{\sum_{i=1}^k w_i}, \quad \alpha = \arctan2\begin{pmatrix} -\bar{\hat{x}}_x\bar{x^0}_x - \bar{\hat{x}}_y\bar{x^0}_y \\ -\bar{\hat{x}}_y\bar{x^0}_x + \bar{\hat{x}}_x\bar{x^0}_y \end{pmatrix}, \quad d = \bar{\hat{x}} - \mathrm{Rot}_\alpha\, \bar{x^0}. \quad (3.56)$$

[6] Defining the initial weights as $w_i = \mathrm{tr}\, A_{ii}$ in (3.50) is consistent with $w_i' = w_{1i} + w_{2i}$ in (3.54), because $\mathrm{tr}\, A_{ii}' = \mathrm{tr}(A_{1ii} + A_{2ii}) = \mathrm{tr}\, A_{1ii} + \mathrm{tr}\, A_{2ii}$ with A_1, A_2 and A' being the information matrices of χ_1^2, χ_2^2 and $\chi'^2 = \chi_1^2 + \chi_2^2$ respectively.

Figure 3.14: **computeCIBAndSIB** $\left((\chi_1^2, e_1),\ (\chi_2^2, e_2),\ \mathcal{E}\right)$ *nonlinear*
χ_1^2, χ_2^2: IBs to be integrated; $\mathcal{E}$: set of landmarks to be eliminated

IF	$\chi_2^2 \neq 0$	
THEN	IF	Exists a saved estimate $\hat{x}$
	THEN	Find optimal rotation $\alpha_{\swarrow}$ and translation d by lemma 19 with $\hat{x}$ (0 if not marked *rotatable*). Store $\alpha_{\swarrow}$
		Rotate and move χ_1^2, e_1 by $\alpha_{\swarrow}$, d using (3.44), (3.51)
		Find optimal rotation $\alpha_{\searrow}$ and translation d by lemma 19 with $\hat{x}$ (0 if not marked *rotatable*). Store $\alpha_{\searrow}$
		Rotate and move χ_2^2, e_2 by $\alpha_{\searrow}$, d using (3.44), (3.51)
	$(A_1, b_1) := \chi_1^2$; $(A_2, b_2) := \chi_2^2$	
	$(x_1^0, w_1, e_1^0) := e_1$; $(x_2^0, w_2, e_2^0) := e_2$	
	Sort and extend A_1, A_2, b_1, b_2, x_1^0, x_2^0, w_1, w_2, such that columns and rows correspond to the same landmark. And landmarks $\mathcal{E}$ are the first block.	
	$A := A_1 + A_2$; $b := b_1 + b_2$	
	Use (3.54) to compute $e' := (x'^0, w', e'^0)$	
ELSE	$(A, b) := \chi_1^2$; $e' := e_1$	
Apply lemma 8 to compute χ_{CIB}^2, χ_{SIB}^2, H, h, P^{-1}		
Remove rows corresponding to $\mathcal{E}$ from x'^0 and w'		
IF	χ_1^2 and χ_2^2 are marked as *rotatable*	
THEN	mark χ_{CIB}^2 as *rotatable*	
return χ_{CIB}^2, e' as CIB and $\chi_{\text{SIB}}'^2$, H, h, P^{-1} as SIB		

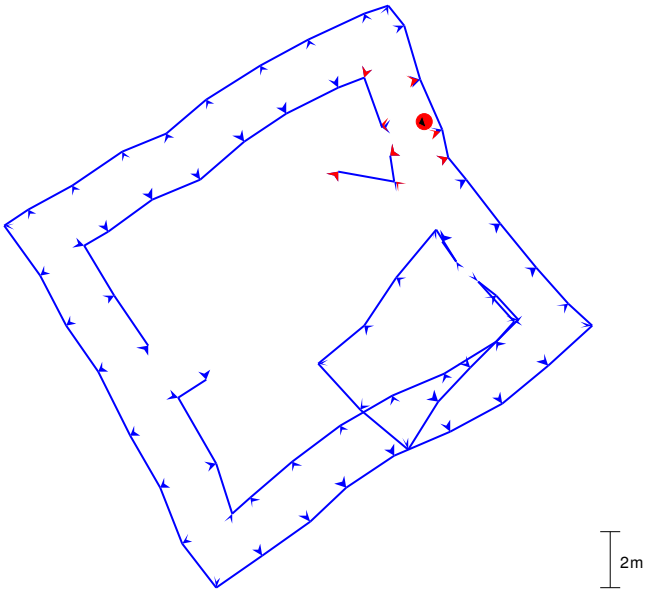


Figure 3.15: Result of the proposed algorithm for the example introduces in figure 1.3. The corresponding EKF and ML estimates are shown in figure 2.5a and 1.3c. It can be observed, that the algorithm provides a very good estimate despite the large orientation error of $\approx 140°$.

The complete procedure for combining two IBs with nonlinear rotation is described as a structure chart (Fig. 3.14, `computeCIBAndSIB`) replacing the linearized version in figure 3.2. The version of `computeCIBAndSIB` used in the algorithm is the only major difference between linear and nonlinear mode. Additionally, in nonlinear mode some further bookkeeping is necessary and in case of closing a large loop several iterations of relinearization must be performed (§4.3).

3.10 Discussion

Summary

The algorithm presented in this thesis achieves its efficiency by processing information as a collection of small *Information Blocks* (IBs). Each IB represents some uncertain information about a small set of landmarks. The decomposition of information is based on hierarchical decomposition of the map into regions represented as a tree (*tree map*). For each region *condensed* information of all measurements made in that region on landmarks observable from outside the region is computed and stored at the regions node. Whenever the robot is outside the region only condensed information needs to be known about this region. The key advantage of this data structure is that integrating a group of local landmark observations only requires an update of condensed information from the leaf which represents the corresponding region up to the root of the tree. Such an operation takes $O(k^3 \log n)$ computation time (n number of landmarks, k number of local landmarks)[7]. The tree map needs $O(kn)$ storage space meeting requirement (R2) and is exceeding requirement (R3). As long as only landmark – landmark measurements are involved, the algorithm computes the exact optimal estimate except for errors generated by linearization and numerics exceeding requirement (R1).

In order to integrate landmark – robot observations and odometry measurements an Extended Kalman Filter (EKF) representing local landmarks is used for preprocessing. Each measurement only requires $O(k^2)$ computation time independent from the overall size of the map. When the actual region is left and a new region is entered information about landmarks is transferred into the tree map requiring a *global update* ($O(k^3 \log n)$). Information about the robot pose is directly transferred into a new EKF state by conservative approximation incorporating an optimization process to lose as little information as possible. Usually, this approximation only increases the estimation error by a small constant factor. This is compliant with requirement (R1) saying that the uncertainty of any aspect of the map should not be much higher than the uncertainty that could be derived from measurements. As a consequence the map is topologically consistent: If

[7] If the building is topologically suitable as discussed in §3.5

two landmarks have been observed to be close to each other they are represented to be close to each other in the map.

The algorithm provides a nonlinear extension that allows to handle the linearization error in the measurement equations by nonlinear rotation of individual IBs. This is a way to approximate the new measurement Jacobians without actually recomputing them after a region has been rotated in the estimate. This *relinearization* procedure is applied to all regions updated anyway. When closing a loop further regions are updated. This approach takes several iterations to converge being interleaved with further global updates as the robot continues moving. However, even the first estimate after one iteration is much better than the result without relinearization at all and needs only slightly more computation time (Fig. 3.15).

This chapter introduced the subalgorithms to manipulate information blocks. They mostly perform numerical linear algebra computation and are the only place in the overall algorithm where actual data are processed. Some further remarks on implementation are provided by appendix C.1. The next chapter will deal with the bookkeeping part. It maintains the tree map representation of the hierarchy and uses the following three subalgorithms derived in this chapter to manipulate information:

- Integrate and decompose IBs (Fig. 3.2, 3.14, `computeCIBAndSIB`)
- Compile an estimate (Fig. 3.4, `computeEstimate`)
- Transfer information from the EKF into the tree map (Fig. 3.10, `extractBIBFromEKF`)

Annotation

To conclude the discussion, assume a different view on the same problem:

What does a least squares estimator finally do when providing an estimate for some landmarks based on measurements involving many more landmarks?

All measurements can contain some indirect information on the landmarks to be estimated, so an entire information matrix has to be computed. In order to provide an estimate the inverse of this matrix is required. If only a few landmarks are to be estimated, a small part of the inverse suffices. So in some sense the purpose of the algorithm is to compute a small part of the inverse of a large matrix efficiently. From this perspective when computing P^{-1} from P for the SIB (lemma 8) the actual inversion occurs. Here the information is converted from information to covariance matrix representation. This can also be seen from the units of measurement, which are inverse squares distance (m^{-2}) in BIB and CIB and square distance (m^2) in the matrix P^{-1} of SIB. Matrix H also used in SIB has no unit.

Chapter 4

Maintenance of the Hierarchy

Chapter 3 provided subalgorithms to manipulate so called Information Blocks (IBs) which are pieces of information about some landmarks. The IBs are integrated according to a hierarchical decomposition of the map represented by a tree. This chapter explains the bookkeeping part of the algorithm maintaining the tree and integrating the information using the subalgorithms derived in the preceding chapter.

The original information is stored as Basic Information Blocks (BIBs) in the leaves of the tree. By definition 3 and (4.1) each node represents those landmarks represented both in some BIBs below that node and in some BIBs not below that node. So a certain landmark is represented a) in the leaves where the corresponding BIB represents the landmark and b) in all ancestor nodes of these leaves up to the least common ancestor of all of them. This least common ancestor is called the landmarks' *elimination node*, i.e. the node the landmark is eliminated from representation and where final integrated information of this landmark is stored.

Each node contains two information blocks: One is the *Condensed Information Block* (CIB) containing the integrated information of all BIBs below this node on the landmarks represented at this node. The other one is the *Substitution Information Block* (SIB) containing the complete integrated information on the landmarks eliminated at this node.

While the robot moves through the map there is always one *actual BIB* into which the current measurements are being integrated. As long as measurements involve only landmarks represented at the actual BIB computation time does not depend on the maps size. When the actual BIB must be changed a global update has to be performed. In so far the behavior is similar to Compressed EKF [GN02]. When utilizing a tree map it is not necessary to modify the whole representation because recomputing all nodes from the actual BIB up to the root will do. This is the key advantage gained by organizing computation hierarchically using a tree.

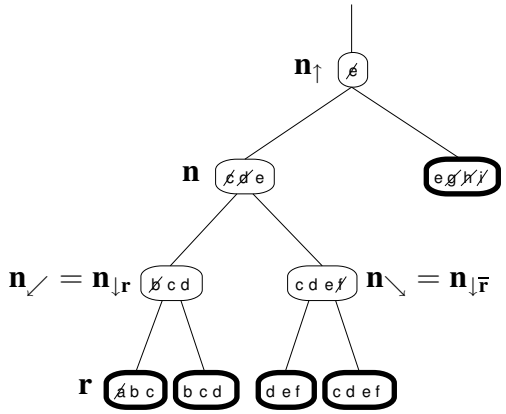


Figure 4.1: Notation of different nodes in a tree: Node ($\mathbf{n}$), parent ($\mathbf{n}_\uparrow$) and children ($\mathbf{n}_\swarrow, \mathbf{n}_\searrow$). In each node the represented landmarks are listed ($\mathsf{e}, \mathsf{g}, \ldots$). The landmarks eliminated at a node are slashed ($\not\mathsf{c}, \not\mathsf{d}$). In this example, $\mathbf{r}$ is a descendant of $\mathbf{n}_\swarrow$, then $\mathbf{n}_{\downarrow \mathbf{r}} = \mathbf{n}_\swarrow$ and $\mathbf{n}_{\downarrow \bar{\mathbf{r}}} = \mathbf{n}_\searrow$. Here $\mathcal{L}(\mathbf{n}) = \{\mathsf{e}, \mathsf{g}\}$, $\mathcal{L}(\mathbf{n}_\swarrow) = \{\mathsf{c}, \mathsf{d}\}$, $\mathcal{L}(\mathbf{n}_\searrow) = \{\mathsf{c}, \mathsf{d}, \mathsf{e}, \mathsf{g}\}$ and $\mathcal{L}(\mathbf{n}_\uparrow) = \emptyset$.

In this chapter the bookkeeping part of the algorithm is described. Its purpose is to build and maintain the tree. This is done by operating on the tree itself and using `computeCIBandSIB` (Fig. 3.2, 3.14) to update CIB and SIB stored at a node from the CIBs of its children. This way the following properties of the tree map are maintained:

1. All CIBs and SIBs are up to date
2. Each landmark's elimination node is the least common ancestor of all BIBs representing that landmark
3. The tree is balanced
4. The tree 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)

The first three items are pure bookkeeping properties that can be perfectly maintained. The fourth item is an optimization task to find the best *Hierarchical Tree Partitioning* [Vij91, HL95] which is theoretically NP-complete [GJ79]. The algorithm contains an incremental tree optimization subalgorithm that improves the partitioning represented by the tree by moving a single subtree in each optimization step. As a consequence, $O(\log n)$ additional nodes have to be updated in each step taking $O(k^3 \log n)$ computation time. Hence the asymptotical complexity is

not affected. The algorithm's overall performance critically depends on this subalgorithm being able to maintain a suitable partitioning. (see §3.5 and §4.4)

As discussed in §3.5 all statements about asymptotical complexity in this thesis depend on the mapped building being topologically suitable. This means that all nodes represent $O(k)$ landmarks and a BIB shares landmarks with at most $O(1)$ different BIBs. Under this assumption the tree map needs $O(kn)$ storage space and a global update operation recomputes $O(\log n)$ nodes with $O(k^3)$ time per node and $O(k^3 + k^2 \log n)$ additional bookkeeping overhead. So the overall complexity is $O(k^3 \log n)$. This result will be discussed in detail in §4.6 after the algorithm has been presented.

Notation

The subalgorithm described in this chapter works on the tree considering sets of landmarks being represented somewhere in the tree. In order to make the description easier to understand, a few notational conventions will be defined as follows:

For a node **n**, the terms *ancestor* of **n**, *descendant* of **n**, *below* **n** or *above* **n** shall always include **n** if not mentioned otherwise. The notation for different nodes related to a node **n** is shown in the following table and figure 4.1. A complete list of symbols can be found on page 25.

Symbols	Definition
$\mathbf{n}, \mathbf{r}, \ldots$	Boldface Roman letters correspond to nodes in the tree
$\mathbf{n}_{\text{CIB}}$, $\mathbf{n}_{\text{SIB}}$, $\mathbf{n}_{\text{BIB}}$, $\mathbf{n}_{size}$, $\mathbf{n}_{\alpha}$, $\mathbf{n}_{\hat{x}}$	Subscripts denote components stored at a specific node by the algorithm (CIB, SIB, BIB, $size$, α, $\hat{x}$)
$\mathbf{n}_{\swarrow}, \mathbf{n}_{\searrow}, \mathbf{n}_{\uparrow}$	Subscript arrows denote a nodes left child, right child and parent
$\mathbf{n}_{\downarrow \mathbf{r}}, \mathbf{n}_{\downarrow \bar{\mathbf{r}}}$	For a node **n** and a descendant node **r**, $\mathbf{n}_{\downarrow \mathbf{r}}$ denotes the child of **n** that is ancestor of **r** and $\mathbf{n}_{\downarrow \bar{\mathbf{r}}}$ denotes the other child, which is not ancestor or **r**.
root	Root of the tree
$\mathcal{L}(\mathbf{n})$	Set of landmarks represented at node **n**
$\mathbf{eN}[l]$	Elimination node of landmark l (stored in an array by the algorithm)
$\mathcal{A}, \mathcal{B}, \mathcal{C}, \ldots$	Calligraphy letters correspond to sets of landmarks

Following definition 3 in §3.2, the set of landmarks represented at a node **n** is given by

$$\mathcal{L}(\mathbf{n}) = \{l | \exists \text{ leaf } \mathbf{n}_1 \text{ below } \mathbf{n} : l \in \mathcal{L}(\mathbf{n}_{1\text{BIB}}) \ \wedge \ \exists \text{ leaf } \mathbf{n}_2 \text{ not below } \mathbf{n} : l \in \mathcal{L}(\mathbf{n}_{2\text{BIB}})\} \,. \quad (4.1)$$

Figure 4.2: **`integrateTreemapObservation`** (z, C_z)
z: measurements, C_z: covariance of z; *global:* $\hat{x}_{\text{EKF}}$: EKF estimate, C_{EKF}: covariance of $\hat{x}$

IF	`isTooLarge` $(\hat{x}_{\text{EKF}}, z) \quad \vee \quad \exists l \in \mathcal{L}(z) : \; l \notin \mathcal{L}(\hat{x}_{\text{EKF}}) \; \wedge \; \exists \mathbf{n} : l \in \mathcal{L}(\mathbf{n})$
THEN	(**newBIB**, *isNonlinearLoop*) := `findOrCreateBIB` (z)
	IF *isNonlinearLoop* THEN `integrateEKFObservation` (z, C_z)
	$(\chi^2_{\text{BIB}}, (P^{-1}, H, h))$:= `extractBIBFromEKF` $(\hat{x}_{\text{EKF}}, C_{\text{EKF}}, \mathcal{L}(z))$
	`setBIB` $(\mathbf{actBIB}, \chi^2_{\text{BIB}})$
	FOR *optHTPSteps* times
	`optimizeHTP` ()
	IF *isNonlinearLoop* THEN `iteratedRelinearize` $((P^{-1}, H, h))$
	α := `accumulatedAngle` (**actBIB**); **actBIB** := **newBIB**
	$(\hat{x}, C)$:= `compileEKF` $(\mathbf{actBIB}, (P^{-1}, H, h), \alpha)$
	IF not *isNonlinearLoop* THEN `integrateEKFObservation` (z, C_z)
ELSE	`integrateEKFObservation` (z, C_z)

4.1 Main Algorithm

A SLAM algorithm has to provide two main routines called by the application processing a) odometry measurements and b) a set of landmark observations. For EKF these are `integrateEKFOdometry` (Fig. 3.6) and `integrateEKFObservation` (Fig. 3.7).

The algorithm described in this thesis uses an EKF as a front end to process individual measurements in $O(k^2)$ time. Odometry measurements are directly handled by `integrateEKFOdometry`. If possible landmark observations are processed directly by the EKF. This is the case for landmarks represented in the EKF and for new landmarks as long as there are not too many landmarks represented. If a known landmark is observed and the landmark is not represented in the EKF information about this landmark has to be retrieved from the tree map and a global update is necessary.

Before integrating the measurements it has to be ensured that the actual BIB represents all landmarks involved. This may mean to make another BIB the actual one, to extend an existing BIB and make it actual or to create a completely new BIB. The heuristical criterion to choose one of these options is described in §4.2 (`findOrCreateBIB`). With regard to perform the change to the new actual BIB, all information from the EKF except about the robot pose is stored in the actual BIB as described in §3.6 (`extractBIBFromEKF`). While this is being done some updating in the tree map (§4.3, `setBIB`) is carried out. In the next step an estimate of the landmarks in the new BIB is compiled. The estimate is integrated with the information about the robot pose from the old EKF state (§4.3, `compileEstimate`) yielding a new EKF state. During these operations the algorithm performs some re-organization of its internal data structure by executing a fixed

number of optimization steps of the partitioning described by the tree (§4.4, `optimizeHTP`).

In the particular situation of closing a large loop in nonlinear mode (detected by `findOrCreateBIB`) iterative relinearization is required. The actual integration of a measurement is always performed by the EKF while the algorithm itself transfers information from the EKF to the tree map and vice versa (§3.6). If a single measurement results in a large change of the robot orientation estimate, the EKF state and thus the actual BIB will be affected by linearization errors. As this error appears in a single BIB and not between two different BIBs it cannot be corrected by the relinearization scheme proposed in §3.9. The following approach avoids this problem: In the special situation of closing a large loop the measurements are directly integrated into the EKF as if they referred to new landmarks and then a global update is performed. So the loop is closed by a global update and not by the EKF. Thus, the actual BIB is not affected by linearization errors. Instead, the linearization error occurs between the actual BIB and the other BIB containing the landmark having closed the loop. Hence it can be corrected by a relinearization scheme iteratively relinearizing until none of the rotation angles changes too much any more ($< maxAngle = 2^\circ$ in the experiments in this thesis).

A chart of the main algorithm is shown in (Fig. 4.2, `integrateTreemapObservation`). The data flow between EKF and tree map has been explained in §3.6, figure 3.8.

4.2 Heuristical BIB Changing Control

New measurements are integrated into a designated BIB called actual BIB. While the robot is moving, this BIB has to be changed from time to time through the heuristical control discussed in this section. Given a set of landmark observations, the control chooses a BIB into which the measurements can be integrated, either by extending a BIB to represent all landmarks necessary or by creating a new BIB. The control is designed to avoid BIBs with too many landmarks, a landmark being represented in too many BIBs and having to change the actual BIB too often, which all three can deteriorate the algorithms performance.

In order to choose the best new BIB, a set of five criteria is obeyed: Criterion (I) defines potential new BIBs; (II) requires which landmarks have to be represented or must be introduced by extending the new BIB; (III) limits the extension of a BIB to meet (II); Criterion (IV) defines which BIB to choose when several are possible and when to create a new BIB. Finally (V) is a special rule to trigger iterative relinearization when closing a large loop in nonlinear mode.

(I) ***Coherence:*** *New and old actual BIB must have at least two landmarks in common.*

There are two reasons to discard BIBs with less than two landmarks in common:

First: When changing the actual BIB, the information about the robot pose must be transferred from the old EKF state to the new EKF state (Fig. 3.8). So, the robot pose must be expressed relative to landmarks both represented in the old and the new actual BIB. This is not possible if both had less than two landmarks in common.

Second: The nonlinear rotation scheme proposed in §3.9 can correct errors in relative orientation of two BIBs even after measurements have been integrated into these BIBs. On the contrary correcting errors is not possible after different measurements have been integrated into the same BIB. Thus the algorithm requires that the measurements integrated into the same BIB have a relative orientation error that allows linearization ($\lessapprox 5°$). This is provided if old and new actual BIB have two landmarks in common.

(II) ***Representation:*** *The new actual BIB must be extended to represent the observed landmarks and the landmarks that are observable according to the EKF.*

The landmarks observed must be represented in the new BIB to be able to integrate the measurement. Some landmarks can be predicted to be observable from their position relative to the robot as reported by the EKF. These landmarks are required to be represented in the new BIB to avoid having to change the actual BIB too soon again.

(III) ***Size:*** *The distance between two landmarks in the same BIB may not exceed a specified limit $maxDistance$ (after a potential extension induced by (II)).*

This is the criterion actually making the algorithm divide measurements and thus the map into different BIBs. The specified distance determines the size of the BIBs and thus influences k. A rule of thumb is the double viewing range of the landmark sensor (5m and 7m in the experiments in §5 and §6).

An alternative criterion would directly limit the number of landmarks represented in a single BIB. The criterion proposed is a much better approach as can be seen in the following example: When for instance a new corridor is mapped usually some landmarks are missed and will only be mapped when the robot moves through the same region again. With a maximum distance criterion these landmarks can easily be integrated into an existing BIB. When enforcing a maximum number of landmarks criterion they would have to be stored in new BIBs which would basically correspond to the same region of some previous BIBs. This would mean to change the actual BIB very often when moving through the corridor.

The following criterion defines, which BIB to choose if several are possible by (I) – (III):

(IV) ***Optimality:*** *Among all BIBs fulfilling (I) and being extendable to meet (II) without*

Figure 4.3: **`findOrCreateBIB`** (z)
z: measurements; *global:* $\hat{x}_{\text{EKF}}$: EKF estimate

<table>
<tr><td colspan="8">Compute observable landmarks $\mathcal{O}$ from $\hat{x}_{\text{EKF}}$</td></tr>
<tr><td colspan="8">**bib** := <code>allBIBsRepresenting</code> $(\mathcal{L}(\hat{x}))$</td></tr>
<tr><td colspan="8">$\mathcal{M} := \mathcal{L}(z) \cup \mathcal{O}$; $\mathcal{N} := \mathcal{L}(z)$; $best := |\mathcal{M}|$; **bestN** := ()</td></tr>
<tr><td colspan="8">FOR All nodes **n** $\in$ **bib**</td></tr>
<tr><td></td><td>IF</td><td colspan="6">$|\mathcal{L}(\hat{x}_{\text{EKF}}) \cap \mathcal{L}(\mathbf{n})| \geq 2$</td></tr>
<tr><td></td><td>THEN</td><td colspan="6">$\mathcal{N} := \mathcal{N} - \mathcal{L}(\mathbf{n})$</td></tr>
<tr><td></td><td></td><td>IF</td><td colspan="5">not <code>isTooLarge</code> $(\mathbf{n}_{\hat{x}}, z)$</td></tr>
<tr><td></td><td></td><td>THEN</td><td colspan="5">$cost := |\mathcal{M} - \mathcal{L}(\mathbf{n})|$</td></tr>
<tr><td></td><td></td><td></td><td>IF</td><td>$cost < best$</td><td>THEN</td><td>$best := cost$; **bestN** := **n**</td></tr>
<tr><td>IF</td><td colspan="7">**bestN** $\neq$ ()</td></tr>
<tr><td>THEN</td><td colspan="7">Extend BIB **bestN** to represent all landmarks from $\mathcal{M}$ except new ones.</td></tr>
<tr><td rowspan="3">ELSE</td><td>IF</td><td>$|\mathcal{M} \cap \mathcal{L}(\hat{x})| < 2$</td><td>THEN</td><td colspan="3">$\mathcal{M} := \mathcal{M} \cup \{$last observed landmark$\}$</td></tr>
<tr><td>IF</td><td>$|\mathcal{M} \cap \mathcal{L}(\hat{x})| < 2$</td><td>THEN</td><td colspan="3">$\mathcal{M} := \mathcal{M} \cup \{$second last observed landmark$\}$</td></tr>
<tr><td colspan="6">**bestN** := <code>createEmptyBIB</code> $(\{l \in \mathcal{M} | l$ is not new$\})$</td></tr>
<tr><td colspan="8">return (**bestN**, $(\mathcal{N} \neq \emptyset)$)</td></tr>
</table>

violating (III) select the one that needs to be extended by as few landmarks as possible. If this is not is possible, create a new one.

The purpose of choosing the BIB to be extended as little as possible is to keep the BIBs small and thus computation time low. If a new BIB has to be created (I) still must be met. So if the landmarks required by definition of (II) do not contain at least two landmarks from the actual BIB, the last two observed landmarks will be added.

(V) ***Loop:*** *If a landmark is observed being neither new nor represented in any BIB fulfilling (I), a large loop has been closed and iterative relinearization must be applied.*

When two BIBs share at least two common landmarks the orientation of those BIBs is coupled by the landmarks and thus the relative orientation error is low. If for a landmark there is no BIB with this property, this landmark may close a large loop thereby leading to a large change in the robot orientation estimate. So iterative relinearization must be applied to prevent linearization errors from being integrated into the BIB.

The subalgorithm (Fig. 4.3, `findOrCreateBIB`) finds the optimal BIB following (IV) in $O(k^3 + k^2 \log n)$ time. This is done by computing the $O(k)$ different BIBs sharing a common landmark with the actual BIB, testing them against (I) – (III) and then selecting the one to be extended by the smallest number of landmarks. The technical details are given in appendix C.2.

4.3 Global Update

After BIB changing control took the decision to make a new BIB to be the actual one, accumulated information from the EKF is stored in the old actual BIB of the tree map and the EKF is initialized with information from the tree map concerning the landmarks of the new actual BIB (Fig. 3.8, §3.6). This requires another update of parts of the tree by recomputing CIBs and SIBs of certain nodes. The subalgorithms in this section determine the nodes to be recomputed. According to the overall purpose of hierarchical decomposition most of the nodes remain valid.

After a BIB's change

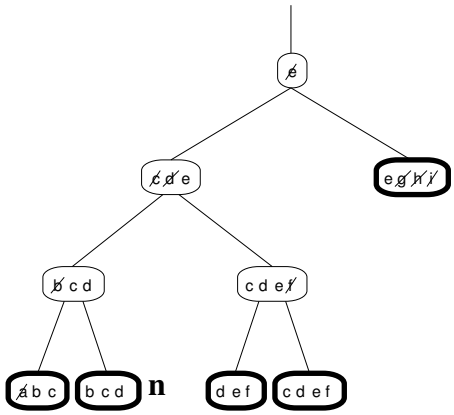
Changing the information in a BIB requires updating all CIBs and SIBs from the leaf containing the BIB up to the root (Fig. 4.4a, b). As the BIB' structure, i.e. the set of represented landmarks changes further nodes have to be updated and for each landmark two different cases appear:

In figure 4.4c the situation is described, what's happening if a landmark is not represented in the new BIB any more: The elimination node moves downward to the first node, where the landmark is still represented in both children. Such a node is called *y-node* for this landmark. This now is the least common ancestor of all BIBs representing that landmark and thus the landmark's elimination node. All nodes from the new elimination node up to the old one must be updated additionally to those from the BIB up to the root.

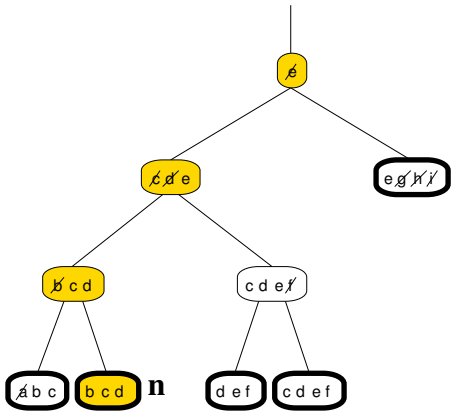
The complementary situation is, when a landmark is represented in the new BIB that has not been represented in the old BIB (Fig. 4.4d): The landmark's elimination node moves up to the least common ancestor of the old elimination node and the BIB. So all nodes from the old elimination node up to the new one have to be updated additionally to those from the BIB up to the root.

A combination of different cases appears, if some landmarks are freshly represented and some vanish from being represented. In order to simplify bookkeeping the algorithm distinguishes between *marking a node invalid* and *updating* a marked node. When a node is marked invalid a flag is set signaling recomputation of the node's state whereas the precedent state is still accessible. Not before the node is updated, its CIB and SIB are recomputed from the childrens' CIBs or from the BIB using `computeCIBandSIB`. Recomputation is performed recursively, so if one or both children have been marked invalid, they are recomputed, too.

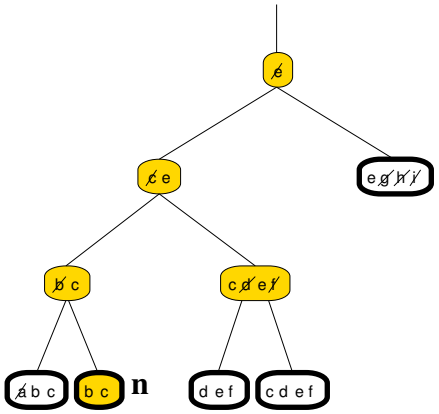
The algorithm first changes the elimination node for all recent landmarks in the new BIB while invalidating all nodes from the former elimination node to the new one and then proceeds with updating from the new BIB up to the root. If an elimination node of a vanished landmark is encountered, the elimination node is set as the next y-node below invalidating all nodes from the



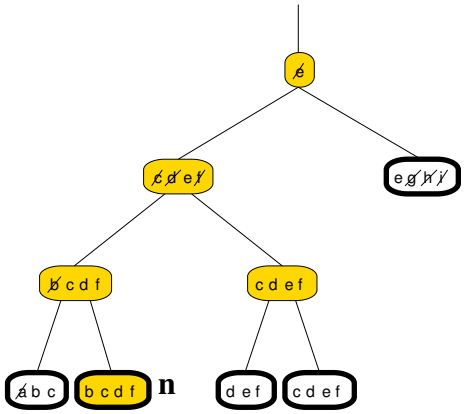
(a) Initial situation $\mathcal{L}(\mathbf{n}) = \{\mathsf{b}, \mathsf{c}, \mathsf{d}\}$



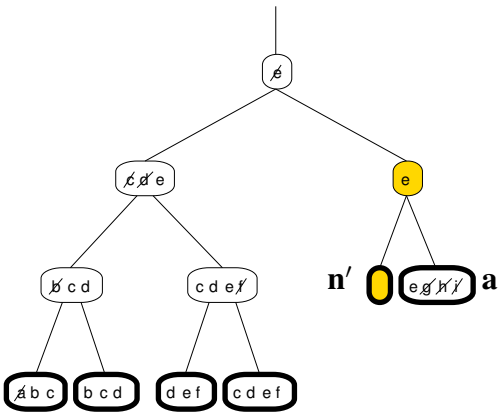
(b) The information in $\mathbf{n}_{\text{BIB}}$ changes, but $\mathcal{L}(\mathbf{n})$ remains



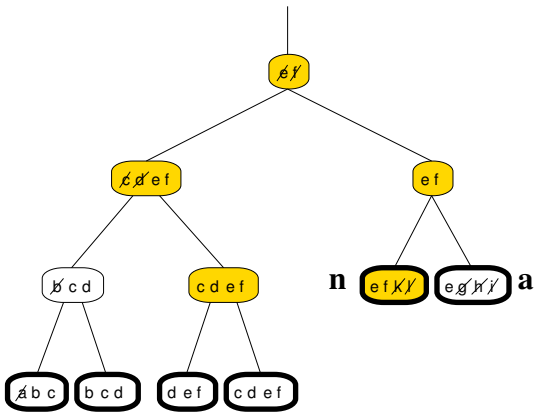
(c) $\mathcal{L}(\mathbf{n})$ changes to $\{\mathsf{b}, \mathsf{c}\}$



(d) $\mathcal{L}(\mathbf{n})$ changes to $\{\mathsf{b}, \mathsf{c}, \mathsf{d}, \mathsf{f}\}$



(e) Empty BIB $\mathbf{n}'$ inserted



(f) $\mathbf{n}'$ changes to $\mathbf{n}$, $\mathcal{L}(\mathbf{n}) = \{\mathsf{e}, \mathsf{f}, \mathsf{k}, \mathsf{l}\}$

Figure 4.4: Different cases for which node has to be updated after the BIB $\mathbf{n}_{\text{BIB}}$ has been changed. The updated nodes are shaded. In (b) only the information contained in the BIB changes. In (c) landmark d disappears from the BIB. In (d) landmark f appears in the BIB. In (e) – (f) a new BIB $\mathbf{n}$ with landmarks $\{\mathsf{e}, \mathsf{f}, \mathsf{k}, \mathsf{l}\}$ is inserted above $\mathbf{a}$, which is realized by first inserting an empty BIB $\mathbf{n}'$ (e) and then changing that BIB to $\mathbf{n}$ (f).

Figure 4.5: **`setBIB`** $\big(\mathbf{n}, (\chi^2, e)\big)$

n: leaf node in which to replace the BIB, χ^2: BIB, e: linearization error for χ^2

FOR All landmarks $l \in \mathcal{L}(\chi^2)$			
	newEN := least common ancestor of **eN** $[l]$ and **n**		
	IF	**eN** $[l] \neq$ **newEN**	
	THEN	Mark nodes **eN** $[l]$ to **newEN** (exclusively) *invalid*	
		eN $[l]$:= **newEN**	
$\mathcal{B} := \mathcal{L}(\mathbf{n}_{\text{BIB}})$; $\quad \mathbf{n}_{BIB} := (\chi^2, e)$			
FOR All landmarks $l \in \mathcal{B} - \mathcal{L}(\chi^2)$ with **eN** $[l] =$ **n**			
	eN $[l]$:= ()		
FOR **n**′ from **n** up to **root**			
	IF	**n**′ is no leaf	
	THEN	FOR All landmarks $l \in \mathcal{B} - \mathcal{L}(\mathbf{n}'_{\downarrow \mathbf{n}})$ with **eN** $[l] =$ **n**′	
			eN $[l]$:= y-node for l below (or equal) $\mathbf{n}'_{\downarrow \overline{\mathbf{n}}}$
			Mark nodes **eN** $[l]$ to **n**′ (exclusively) *invalid*
	Mark **n** *invalid*		
	`update` (**n**′) [a]		
	Mark **n**′ as *to be optimized* [b]		

[a] Optimization: Update only the list of landmarks represented in the node and defer recomputation of the CIB and SIB to `compileEstimate`

[b] Mark this note as to be processed by the HTP subalgorithm. (§4.4)

new elimination node up to the old one. Having finished a node, its CIB and SIB is recursively updated.

All updated nodes share a common landmark with the old or new BIB. According to part (2) of definition 5 there are only $O(1)$ BIBs and thus $O(\log n)$ nodes with this property. So even in a worst case scenario only $O(\log n)$ nodes need to be updated. The subalgorithm are shown in (Fig. 4.5, `setBIB`) and (Fig. 4.6, `update`).

To insert a new BIB **n** into the tree, first an empty BIB **n**′ is inserted above some node **a**. Therefore **a** is replaced by a combination node having **n**′ and **a** as children (Fig. 4.4e). Since $\mathcal{L}(\mathbf{n}') = \emptyset$, only **n**′ and $\mathbf{n}'_{\uparrow}$ need to be updated. In a second step **n**′ is changed to **n** and all affected nodes are updated (Fig. 4.4f). This is equivalent to the situation discussed when a new BIB contains landmarks not represented before.

Compiling an Estimate

If all CIBs and SIBs are updated, compiling an estimate for the landmarks represented at a certain BIB is straightforward: The SIB stored at a node allows computing an estimate of the landmarks

Figure 4.6: **update** $(\mathbf{n})$

n: node to be updated recursively

IF	**n** is marked *invalid*	
THEN	IF	**n** is no leaf
	THEN	update $(\mathbf{n}_{\swarrow})$
		update $(\mathbf{n}_{\searrow})$
		$\mathcal{E} := \{l \in \mathcal{L}(\mathbf{n}_{\swarrow}) \cup \mathcal{L}(\mathbf{n}_{\searrow}) \mid \mathbf{eN}[l] = \mathbf{n}\}$
		$(\mathbf{n}_{\text{CIB}}, \mathbf{n}_{\text{SIB}}) :=$ computeCIBandSIB $(\mathcal{L}(\mathbf{n}_{\swarrow\text{CIB}}),\ \mathcal{L}(\mathbf{n}_{\searrow\text{CIB}}),\ \mathcal{E})$
	ELSE	$\mathcal{E} := \{l \in \mathcal{L}(\mathbf{n}_{\text{BIB}}) \mid \mathbf{eN}[l] = \mathbf{n}\}$
		$(\mathbf{n}_{\text{CIB}}, \mathbf{n}_{\text{SIB}}) :=$ computeCIBandSIB $(\mathbf{n}_{\text{BIB}},\ \mathcal{E})$

Figure 4.7: **compileEstimate** $\left(\mathbf{n}, (H, h, P^{-1}), \gamma\right)$

n: leaf node to compute estimate for; (H, h, P^{-1}): SIB defining robot pose; γ:SIB rotation

$\hat{x} := ();\quad C := ();\quad \alpha := 0$	
FOR $\mathbf{n}'$ from **root** down to **n**	
	update $(\mathbf{n}')$ [a]
	$(\hat{x}, C) :=$ computeEstimate $((\hat{x}, C, \alpha),\ \mathbf{n}'_{\text{SIB}})$
	$\mathbf{n}'_{\hat{x}} := \hat{x};\quad \alpha := \alpha + \mathbf{n}'_{\alpha_{\text{CHILD}}}$ (CHILD: which child to go next)
$(\hat{x}, C) :=$ computeEstimate $\left((\hat{x}, C, \gamma),\ (H, h, P^{-1})\right)$	
return$(\hat{x}, C)$	

[a] Necessary only if recomputation of the CIB and SIB in **setBIB** has been defered.

Figure 4.8: **accumulatedAngle** $(\mathbf{n})$

n: node to compute the accumulated rotation angle for

$\alpha := 0$
FOR $\mathbf{n}'$ from **root** to **n**
$\alpha := \alpha + \mathbf{n}_{\alpha_{\text{CHILD}}}$ (CHILD: which child to go next)
return α

Figure 4.9: **iteratedRelinearize** $\left((H, h, P^{-1})\right)$

(H, h, P^{-1}): SIB defining the robot pose; *global:* **actBIB**: actual BIB

bib := allBIBsRepresenting $(\mathcal{L}(\mathbf{actBIB}))$	
	FOR All $\mathbf{n} \in$ **bib**
	FOR All $\mathbf{n}'$ from **n** to **root**
	Mark $\mathbf{n}'$ *invalid*
	compileEKF $\left(\mathbf{actBIB}, (H, h, P^{-1}), 0\right)$ [a]
UNTIL $\mathbf{n}_{\alpha}$ changed by $< maxAngle$ for all ancestors of all $\mathbf{n} \in$ **bib**	

[a] Rotation angle for robot pose SIB is unimportant, since the robot pose is not used here.

represented at the node's children from an estimate of the landmarks represented at the node (lemma 11, `computeEstimate`). The root node represents no landmark, so the estimate for the root node is empty and `computeEstimate` can be applied incrementally from the root down to the BIB the estimate is to be compiled for. If a global estimate i.e. an estimate for all landmarks is desired `computeEstimate` is applied recursively. Evaluation of the covariance in (3.23) can be omitted computing only $\hat{y} = H\hat{z} + h$ in every node. Thus computation is extremely fast despite needing asymptotically $O(kn)$ time (see §5.4).

Another point to consider is the question of nonlinear rotation (see §3.9). When rotating a CIB for relinearization (in `computeCIBandSIB`), all SIBs below this CIB would have to be rotated, too. As this is too expensive, the rotation angle is stored at the node ($\mathbf{n}_{\alpha_{\text{LEFT}}}, \mathbf{n}_{\alpha_{\text{RIGHT}}}$) and rotation is deferred until a SIB is used. So while proceeding from the root to the BIB the algorithm accumulates the rotation angle to be applied to each SIB (Fig. 4.8, `accumulatedAngle`). The estimates generated are stored in $\mathbf{n}_{\hat{x}}$ and used for relinearization. Finally, the SIB containing information about the robot pose is integrated into the estimate. The whole algorithm is shown as a structure chart in (Fig. 4.7, `compileEstimate`).

Relinearization

Whenever a node is updated using `computeCIBandSIB` the latest estimate stored at the node is used to relinearize the CIB by applying a nonlinear rotation (§3.9). So linearization error can be reduced with nearly no additional cost while the robot is moving.

This is not sufficient when closing a large loop with high orientation error. This makes it necessary to apply appropriate nonlinear rotations along the loop immediately. Otherwise, the map may become inconsistent at the place where the loop has been closed. In order to prevent this closing a nonlinear loop will explicitly be detected when changing the actual BIB (§4.2) and iterated relinearization is performed: During each iteration, all BIBs sharing common landmarks with the actual BIB and all their ancestors are updated applying a further nonlinear rotation as usual. Iteration is terminated, when no node is rotated by more than one specified angle $maxAngle$ (2° in the experiments) (Fig. 4.9, `iteratedRelinearize`).

So linearization errors can be reduced rapidly, because an update of an internal node applies a nonlinear rotation efficiently to a large piece of the map. Since a BIB shares landmarks only with $O(1)$ other BIBs, only $O(\log n)$ nodes are updated, so one single iteration takes only $O(k^3 \log n)$ time. There is good reason to assume that the number of iterations needed to close a loop depends on the orientation error of the loop but not on the length of the loop or overall size of the map.[1]

[1] Note this subalgorithm is not an iterative linear equation solver like conjugate gradients, but a Newton type nonlinear equation solver like Levenberg-Marquardt with a direct linear equation solver used in each iteration.

4.4 Hierarchical Tree Partitioning

At this point the algorithm has almost been described completely, subalgorithms to integrate measurements updating the tree map and computation of an estimate included. The remaining open question is where to insert a new BIB, thus, a new node into the tree. Actually, the algorithm could be used as described so far, if the tree was given a priori. This point is decisive because the performance of the algorithm crucially depends on the quality of the tree:

The height of the tree determines how many nodes have to be updated during a global update. The number of landmarks represented at a node determines the size of the matrices involved in updating the nodes parent and thus computation time needed. By definition 3, a node represents those landmarks that are represented both in BIBs inside and in BIBs outside the subtree below this node. So the goal is to subdivide the set of BIBs hierarchically into a balanced tree (i.e. subdivide the map into regions), such that for each subtree as few landmarks as possible are represented in BIBs outside this subtree (i.e. observable from outside the region). This is easy to achieve for the leaves of the tree because they are corresponding to small regions in the map (enforced by criterion (III), §4.2). However internal nodes correspond to large regions containing many landmarks. But to ensure that only few landmarks are being represented the region is chosen skillfully, i.e. most landmarks of a region are not observable from outside.

The computation time of the proposed algorithm is only $O(k^3 \log n)$, if the tree is *balanced* having a height of $O(\log n)$ and *well partitioned*, so that the number of landmarks represented at each node including internal nodes is $O(k)$. In reality, a good tree (i.e. a good partitioning of the map) is not known in advance and must be generated and optimized constantly while the robot is moving and new BIBs are being added. Asymptotical analysis assumes, that the tree optimization subalgorithm achieves this goal. See discussion in §3.5 and §4.6.

Relation to Graph Theory

This problem is equivalent to the *Hierarchical Tree Partitioning Problem* (HTP) known from graph theory and parallel computing. It is a hierarchical version of the *Graph Partitioning Problem* or in case of a binary tree of the *Graph Bisection Problem*, which are both known to be NP-complete [GJ79]. However, successful heuristical algorithms have been developed to solve these in practice [Vij91, FM82]. The graph bisection problem is posed as follows:

Divide the nodes of a (multi-) graph into two (approximately) equal partitions with a minimal number of edges connecting nodes of different partitions.

Minimizing the number of landmarks involved in computation at the root node of a tree map

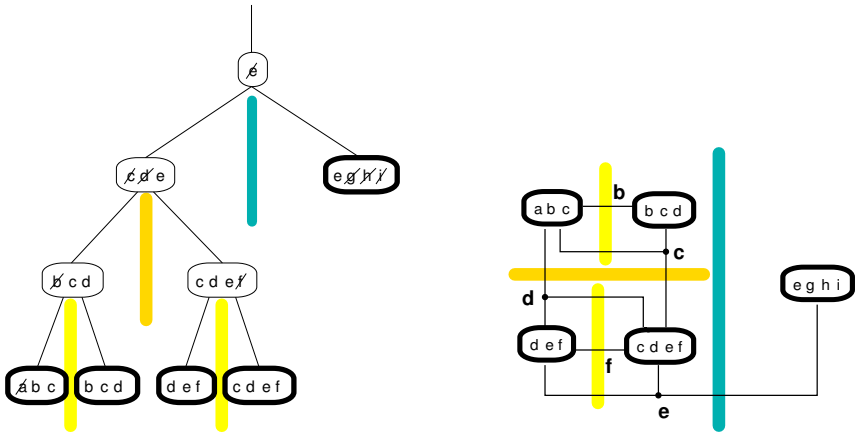


Figure 4.10: Tree and corresponding multigraph partitioning: A set of BIBs corresponds to a multigraph, where each BIB forms a node and each landmark forms a multi-edge connecting all BIBs representing this landmark. In turn a hierarchical partitioning of the graph (gray bars) corresponds to a tree with BIBs as leaves and (sub-) partitions as internal nodes. The landmarks represented in different nodes correspond to edges being incident to different partitions.

is equivalent to finding a minimum bisection for the following multigraph (Fig. 4.10): In the graph, each BIB forms a node with each landmark forming a multi-edge connecting all BIBs that represent this landmark. A bisection of this multigraph divides the BIBs into **root**$_{\swarrow}$ and **root**$_{\searrow}$. The landmarks involved in computation at the root node are those represented both below **root**$_{\swarrow}$ and **root**$_{\searrow}$. They correspond to edges connecting different partitions in the graph constructed. The bisection problem is to minimize the number of these edges, thereby yielding an optimal decomposition of the map into two regions.

There are several heuristical approaches established in parallel computing that could be used to find a good tree map for a set of BIBs [FM82, Vij91, HL95]. However, all these approaches perform an offline computation requiring $O(n)$ to $O(n \log n)$ computation time, changing the whole tree and thus necessitating a complete recomputation of all CIBs and SIBs ($O(k^2 n)$). As HTP optimization is not meant to be the dominant part of the overall computation a step by step optimization scheme has been developed: In each single step only one subtree is moved to a different location. The time per step is limited to $O(k^2 \log n)$ for finding the subtree and $O(k^3 \log n)$ for updating all necessary nodes.

Optimization criteria

Finding the best transfer step (i.e. a subtree and where to move it) in $O(k^2 \log n)$ time is quite a challenge. In fact, the exact criteria used by the algorithm have been inspired by a method called *multilevel tree partitioning* [HL95], but were tailored for efficient enforcement. The general idea

is to optimize the partitioning of a node (II) under the constraint the node is still balanced (III) giving priority to ancestor nodes (I):

(I) ***Priority:*** *Optimizing a node is more important than optimizing its descendant nodes.*

When a node is updated, its ancestor nodes are updated too so in general they are updated more often and thus more important for computation time. All recursive subdivision algorithms in graph partitioning implicitly employ this priority.

(II) ***Partitioning:*** *For a given node* $\boldsymbol{n}$ *minimize the sum of the landmarks represented in both children of this node*

$$\mathrm{par}(\boldsymbol{n}) := |\mathcal{L}(\boldsymbol{n}_{\swarrow})| + |\mathcal{L}(\boldsymbol{n}_{\searrow})|\,. \tag{4.2}$$

Maintaining a good partitioning, i.e. minimizing $\mathrm{par}(\mathbf{n})$ for all nodes $\mathbf{n}$ is the main goal of the HTP subalgorithm because it determines the computation time needed for updating $\mathbf{n}$. When succeeding, $\mathrm{par}(\mathbf{n}) = O(k) \forall \mathbf{n}$.

A more obvious definition would be $\mathrm{par}'(\mathbf{n}) := |\mathcal{L}(\mathbf{n}_{\swarrow}) \cup \mathcal{L}(\mathbf{n}_{\searrow})|$ actually being the size of the matrix A involved in computation at node $\mathbf{n}$ by equation (3.4). There is subtle reason for using $\mathrm{par}(\mathbf{n})$ instead of $\mathrm{par}'(\mathbf{n})$, which applies to landmarks both in $\mathcal{L}(\mathbf{n}_{\swarrow})$ and $\mathcal{L}(\mathbf{n}_{\searrow})$. These contribute with 1 to par' and 2 to $\mathrm{par}(\mathbf{n})$, whereas landmarks exclusively contained in $\mathcal{L}(\mathbf{n}_{\swarrow})$ or $\mathcal{L}(\mathbf{n}_{\searrow})$ contribute with 1 to both. When using $\mathrm{par}(\mathbf{n})$ the algorithm is encouraged to collect all BIBs representing a certain landmark either left or right of $\mathbf{n}$. Once this is done, a subtree containing all these BIBs may be moved to a different position, thus completely removing the landmark from representation at $\mathbf{n}$. This optimization can not be carried out in a single step, so it is of advantage to encourage the subalgorithm to take the above mentioned intermediate step.

(III) ***Balancing:*** *For a node* $\boldsymbol{n}$ *define the* size $\boldsymbol{n}_{size}$ *as the number of BIBs below this node and the balancing* $\mathrm{bal}(\boldsymbol{n})$ *by* (4.3)*. Enforce a balancing constraint of* $\mathrm{bal}(\boldsymbol{n}) \in \mathrm{Bal}(\boldsymbol{n})$*. During insertion temporarily allow violation of the balancing constraint. In this case enforce an even harder constraint in the next optimization step for* $\boldsymbol{n}$*.*

$$\mathrm{bal}(\boldsymbol{n}) := \frac{\boldsymbol{n}_{\swarrow\, size}}{\boldsymbol{n}_{size}}, \quad \mathrm{Bal}(\boldsymbol{n}) := \begin{cases} \left[\frac{2}{5} \ldots \frac{2}{3}\right] & \mathrm{bal}(\boldsymbol{n}) < \frac{1}{3} \\ \left[\frac{1}{3} \ldots \frac{2}{3}\right] & \frac{1}{3} \leq \mathrm{bal}(\boldsymbol{n}) \leq \frac{2}{3} \\ \left[\frac{1}{3} \ldots \frac{3}{5}\right] & \frac{2}{3} < \mathrm{bal}(\boldsymbol{n}) \end{cases} \tag{4.3}$$

The balancing $\mathrm{bal}(\mathbf{n})$ of a node is always in $[0 \ldots 1]$, with $\frac{1}{2}$ corresponding to a perfectly balanced node with an equal number of BIBs on each side. Figure 4.11 shows the enforced balancing constraint $\mathrm{Bal}(\mathbf{n})$ as a function of $\mathrm{bal}(\mathbf{n})$. Apart from temporal violation this criterion

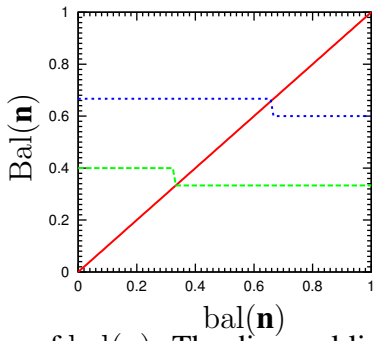


Figure 4.11: $\mathrm{Bal}(\mathbf{n})$ as a function of $\mathrm{bal}(\mathbf{n})$: The diagonal line allows to compare the balancing $\mathrm{bal}(\mathbf{n})$ with the enforced balancing constraint $\mathrm{Bal}(\mathbf{n})$. It can be seen, that the constraint is either already fulfilled (diagonal line is between lower and upper bound) or there is at least a distance of $\frac{1}{15} \approx 0.067$ between $\mathrm{bal}(\mathbf{n})$ and the allowed interval $\mathrm{Bal}(\mathbf{n})$.

guarantees that $\mathrm{bal}(\mathbf{n}) \in \mathrm{Bal}(\mathbf{n}) \subset \left[\frac{1}{3} \dots \frac{2}{3}\right]$ and thus the the height of the tree is at most $\log_{\frac{3}{2}} n \leq 1.71 \log n = O(\log n)$.

There is a good reason for not enforcing the constraint while inserting a new BIB: Maintaining the criterion would lead to new BIBs being inserted alternately left and right of a given node. Otherwise the number of leaves below one child of this node would grow and below the other it would not. This policy would result in a extremely badly partitioned tree difficult to be improved by the optimization subalgorithm. A better approach is to insert at the best position regarding criterion (I) – (II). If this violates (III) for any node this node is marked "*to be optimized*". The optimization algorithm restores the nodes balancing prioritizing it over good partitioning.

It has a major advantage to priorize partitioning over balancing during insertion and vice versa during optimization. The optimization subalgorithm can transfer arbitrary subtrees to a different position in the tree, whereas during insertion the only choice is where to insert the new BIB. Thus the optimization subalgorithm can most often find a well balanced and well partitioned solution not possible during insertion. The reason for enforcing a harder constraint ($\frac{2}{5}$ instead $\frac{1}{3}$ or $\frac{3}{5}$ instead $\frac{2}{3}$) if the balancing constraint has been temporarily violated lies in the problem of finding the optimal transfer step in $O(k^2 \log n)$ and will be explained in the following:

Finding the best transfer step

This subsection gives an overview how to find an optimal transfer step. It will be discussed, how the definitions of (I) – (III) were designed to make an efficient search possible. The key point is that a node $\mathbf{r}$ divides the tree into three parts (Fig. 4.12): below $\mathbf{r}_{\swarrow}$, below $\mathbf{r}_{\searrow}$ and not below $\mathbf{r}$. All criteria do not depend on the specific shape of these parts. They only depend on the question

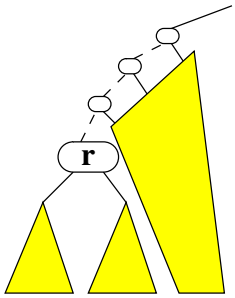


Figure 4.12: A node **r** divides the tree into three parts: below $\mathbf{r}_{\swarrow}$, below $\mathbf{r}_{\searrow}$ and not below **r**

whether a BIB or landmark is represented in the respective parts or not.

The detailed subalgorithm is found in appendix §C.3 to §C.5, its general outline is as follows:

1. Choose a node **r** that has been marked *to be optimized* for optimization
2. Find the optimum subtree **s** to move from below $\mathbf{r}_{\swarrow}$ to below $\mathbf{r}_{\searrow}$ or vice versa considering (II) and (III) for **r**
3. Find where below $\mathbf{r}_{\searrow}$ (or below $\mathbf{r}_{\swarrow}$ resp.) to move subtree **s** to considering (I), (II) and (III) for all affected nodes from **r** down to the point of insertion

In step 1 a node is chosen that is marked *to be optimized* and for which all descendants are not marked. The reason for this policy is that it is much likelier to find a good transfer step for a node, when the subtree below has already been well partitioned because the transfer step is restricted to move a single subtree only. At first sight this seems contrary to criterion (I) that demands prioritizing higher nodes. However, priority is achieved in step 2 by ensuring that the optimization of **r** does not affect ancestors of **r** but taking no care affecting descendants of **r**.

In step 2 the subtree to be transferred is determined according to (II) and (III). The subtree is always removed from below one child of **r** ($\mathbf{r}_{\swarrow}$ or $\mathbf{r}_{\searrow}$ resp.) and moved to some place below the other child ($\mathbf{r}_{\searrow}$ or $\mathbf{r}_{\swarrow}$ resp.). For **r** it does not make a difference where the subtree is moved to below the other child. What matters is that it is moved not where it is moved to. That is why the exact point of insertion can be determined in step 3 when applying the optimization criterion for the affected descendants of the other child. Transferring the subtree to a place above **r** is not allowed, since it would exert influence on ancestor nodes and violate (I). Transferring the subtree to a place below the same child will not affect **r** at all. After all, the decision taken in step 2 is:

*Which subtree **s** is best to be transferred from one side of **r** to the other.*

The search for the optimal subtree is performed by evaluating all possible candidates **s**, whether transferring the subtree below **s** to the other side of **r** (from $\mathbf{r}_{\downarrow \mathbf{s}}$ to $\mathbf{r}_{\downarrow \bar{\mathbf{s}}}$) will violate (III) and how $\mathrm{par}(\mathbf{r})$ will change. From all feasible **s** the one with the lowest $\mathrm{par}(\mathbf{r})$ is selected. Criteria (II) and (III) have been designed, so that there are only $O(k \log n)$ candidates that need to be evaluated. All these are on $O(k)$ paths from **r** down to some BIB, so checking all candidates is not too expensive. Two cases arise:

1. *The node **r** is already balanced*:
   Then it is a valid transfer step to change nothing at all, so by criterion (II) the best transfer step must decrease $\mathrm{par}(\mathbf{r})$ and thus either $|\mathcal{L}(\mathbf{r}_{\swarrow})|$ or $|\mathcal{L}(\mathbf{r}_{\searrow})|$. It must involve a subtree, whose root represents some landmarks of $\mathcal{L}(\mathbf{r}_{\swarrow})$ or $\mathcal{L}(\mathbf{r}_{\searrow})$. As discussed in §4.2 there only exist $O(k)$ BIBs with this property. The ancestors up to **r** of all these BIBs possibly represent the same landmark, so there are $O(k \log n)$ candidates to be evaluated.
2. *The node **r** is not balanced*:
   The priority is to restore the balancing constraint $\mathrm{bal}(\mathbf{r}) \in \mathrm{Bal}(\mathbf{r})$. This constraint is even harder than in case 1, requiring $\mathrm{bal}(\mathbf{r}) \geq \frac{2}{5}$, if $\mathrm{bal}(\mathbf{r})$ was $< \frac{1}{3}$ or $\mathrm{bal}(\mathbf{r}) \leq \frac{3}{5}$ if $\mathrm{bal}(\mathbf{r})$ was $> \frac{2}{3}$. $\mathrm{Bal}(\mathbf{r})$ has been defined this way to ensure that the size of the subtree to be transferred is at least $\frac{2}{5} - \frac{1}{3} = \frac{2}{3} - \frac{3}{5} = \frac{1}{15}$ of the number of BIBs below **r** (Fig. 4.11). Obviously there are at most $15 = O(1)$ disjoint subtrees as large as $\frac{\mathbf{r}_{size}}{15}$. The roots of these disjoint subtrees and all their ancestors up to **r** are the only possible candidates for a transfer step and at most $O(\log n)$.

   The purpose of the definition of $\mathrm{Bal}(\mathbf{r})$ in (4.3) is to ensure a minimum size for the subtree to be moved, so there are only few possible candidates. If $\mathrm{Bal}(\mathbf{r})$ was defined as $\left[\frac{1}{3} \ldots \frac{2}{3}\right]$, a subtree of size 1 may be sufficient to restore the balancing. In the worst case $O(n)$ candidates to would have to be checked, but this would take too long.

The fact of only a few subtrees to be evaluated when optimizing a node is the key property for the $O(k^2 \log n)$ optimization algorithm used in this thesis.

In step 3, the new location of the subtree is determined. This is the same procedure as used for inserting a completely new BIB. The algorithm optimizes criterion (II) and (III) for different affected nodes prioritizing ancestor nodes following (I). For a node **n** it only matters, whether the subtree is inserted above **n**, below $\mathbf{n}_{\swarrow}$ or below $\mathbf{n}_{\searrow}$. It has no influence where exactly the subtree is inserted. It only matters where relative to **n**. Given this property and criterion (I) the optimal insertion point can be determined recursively:

For a node **n** the three possibilities *directly above **n***, *somewhere below* $\boldsymbol{n}_{\swarrow}$ and *somewhere below* $\boldsymbol{n}_{\searrow}$ are compared considering $\mathrm{par}(\mathbf{n})$. If the best option is *directly above **n*** this is the

final insertion point otherwise recursion is applied to $\mathbf{n}_{\swarrow}$ or $\mathbf{n}_{\searrow}$ respectively. The balancing criterion (III) only limits the choice *directly above* ***n***: If the number of BIBs below the subtree to be inserted is smaller than half the number of BIBs below **n**, it is not allowed to insert directly above **n** since this would violate (III). As mentioned above, (III) is not enforced considering the choice between $\mathbf{n}_{\swarrow}$ and $\mathbf{n}_{\searrow}$ since this would lead to extremely badly partitioned trees. Instead all nodes down to the insertion point get marked as *to be optimized*. If one of them violates the balancing constraint it will be repaired when the node is optimized next time.

Figure 4.13 shows an example for optimizing a tree. The technical details of implementation can be found in appendix C.

4.5 Transfer of a Subtree

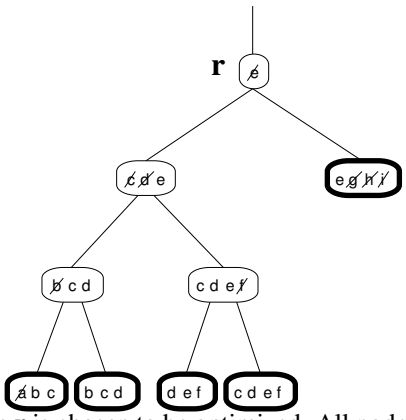
This section describes a subalgorithm for transferring a whole subtree (below a node **s**) from one place in the tree to another place (above **a**) for executing the best optimization step determined by the HTP optimization algorithm described in the previous section.

Before it is possible to move the subtree **s** to above **a**, a new node must be inserted there. This node has **a** and **s** as children and $\mathbf{a}_{\uparrow}$ as parent. Conversely, after moving the subtree, the former parent $\mathbf{s}_{\uparrow}$ must be deleted. The algorithm consists of three steps (Fig. 4.15):

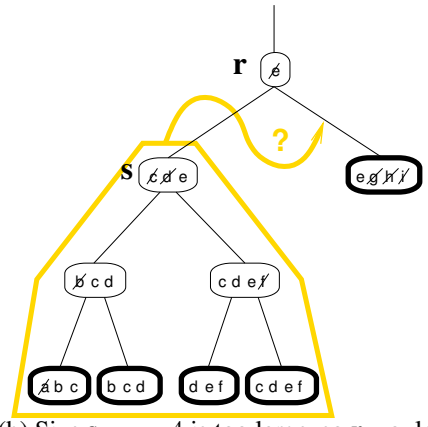
1. Insert a new node above **a** with an empty dummy BIB **d** (one representing no landmark) as other child
2. Swap **d** and **s**
3. Remove **d** and the former parent of **s**, which is now the parent of **d**

For step 1 and 3 no updating is required except in **d** and its parent $\mathbf{d}_{\uparrow}$. Step 2 has to update all nodes invalidated due to the subtree of **s** being moved. It is important to note that the number of these nodes is extremely limited: The only nodes invalidated are those from **s** and **a** to the least common ancestor **lca** of both (to **root** in nonlinear mode). The reason for this is obvious from the definition of a CIB (Definition 3): A node's CIB is only invalidated if either the set of BIBs below that node or the set of BIBs not below that node is changing. When transferring a subtree no BIB is added or removed but the BIBs below **s** alterate their position in the tree:

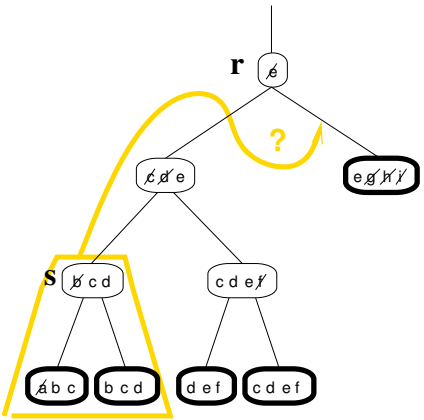
1. The nodes (CIBs and SIBs) from $\mathbf{s}_{\uparrow}$ to **lca** exclusively are invalidated since **s** is below those nodes before and not afterwards
2. The nodes (CIBs and SIBs) from $\mathbf{a}_{\uparrow}$ to **lca** exclusively are invalidated, since **s** is below those nodes afterward but not before



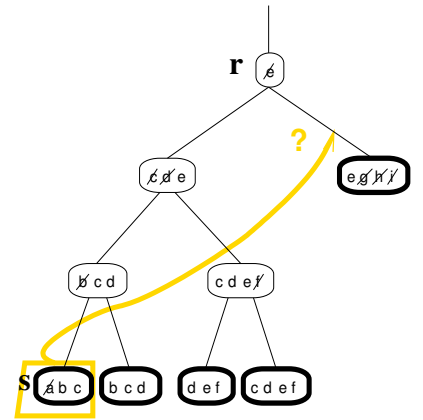
(a) Node **r** is chosen to be optimized. All nodes below have been optimized by prior HTP steps. **r** is unbalanced $\mathrm{bal}(\mathbf{r}) = \frac{4}{5}$, so **s** may have size $\mathbf{s}_{size} \in [1 \dots 3]$.



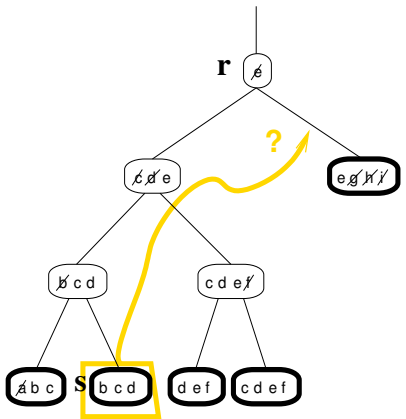
(b) Size $\mathbf{s}_{size} = 4$ is too large, so **r** would be unbalanced if **s** was transfered to $\mathbf{r}_{\searrow}$ and **s** is not feasible.



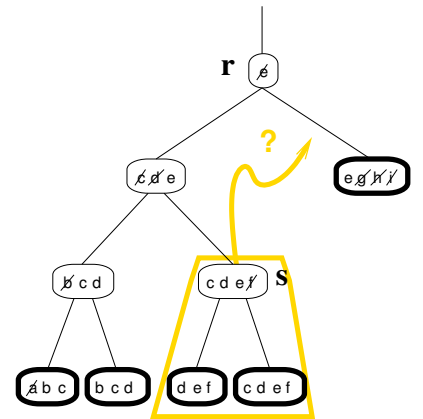
(c) Transferring **s** would make $\mathrm{bal}(\mathbf{r}) = \frac{2}{5}$, $\mathcal{L}(\mathbf{r}_{\swarrow}) = \mathcal{L}(\mathbf{r}_{\searrow}) = \{\mathsf{c}, \mathsf{d}, \mathsf{e}\}$, $\mathrm{par}(\mathbf{r}) = 6$, so **s** is feasible.



(d) Transferring **s** would make $\mathrm{bal}(\mathbf{r}) = \frac{3}{5}$, $\mathcal{L}(\mathbf{r}_{\swarrow}) = \mathcal{L}(\mathbf{r}_{\searrow}) = \{\mathsf{b}, \mathsf{c}, \mathsf{e}\}$, $\mathrm{par}(\mathbf{r}) = 6$, so **s** is feasible.

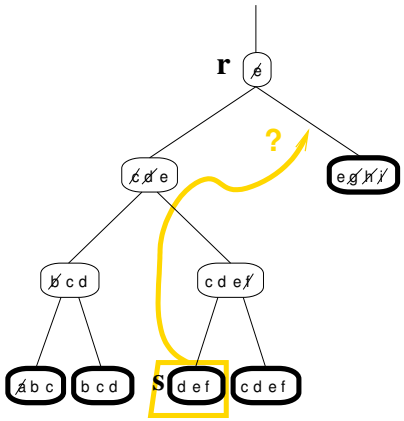


(e) Transferring **s** would make $\mathrm{bal}(\mathbf{r}) = \frac{3}{5}$, $\mathcal{L}(\mathbf{r}_{\swarrow}) = \mathcal{L}(\mathbf{r}_{\searrow}) = \{\mathsf{b}, \mathsf{c}, \mathsf{d}, \mathsf{e}\}$, $\mathrm{par}(\mathbf{r}) = 8$, so **s** is feasible but worse than (c).

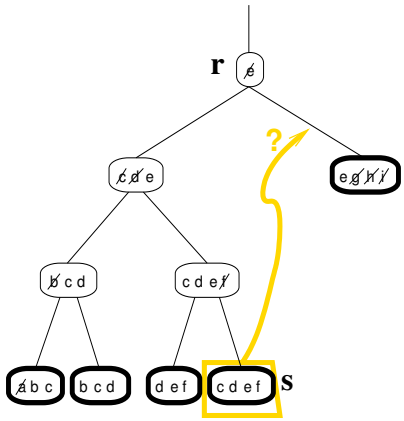


(f) Transferring **s** would make $\mathrm{bal}(\mathbf{r}) = \frac{2}{5}$, $\mathcal{L}(\mathbf{r}_{\swarrow}) = \mathcal{L}(\mathbf{r}_{\searrow}) = \{\mathsf{c}, \mathsf{d}\}$, $\mathrm{par}(\mathbf{r}) = 4$, so **s** is feasible and better than (c).

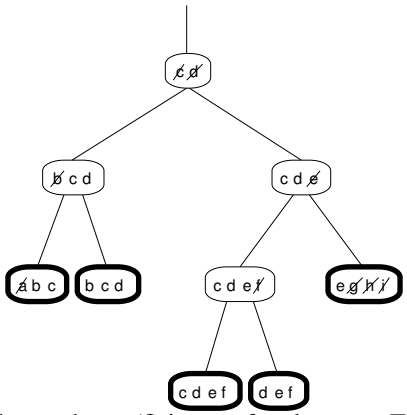
Figure 4.13: Example of an HTP optimization step followed by insertion of a new BIB.



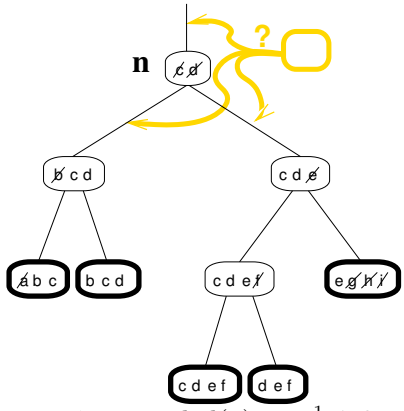
(g) Transferring **s** would make $\mathrm{bal}(\mathbf{r}) = \frac{3}{5}$, $\mathcal{L}(\mathbf{r}_{\swarrow}) = \mathcal{L}(\mathbf{r}_{\searrow}) = \{\mathsf{d}, \mathsf{e}, \mathsf{f}\}$, $\mathrm{par}(\mathbf{r}) = 6$, so **s** is feasible but worse than (f).



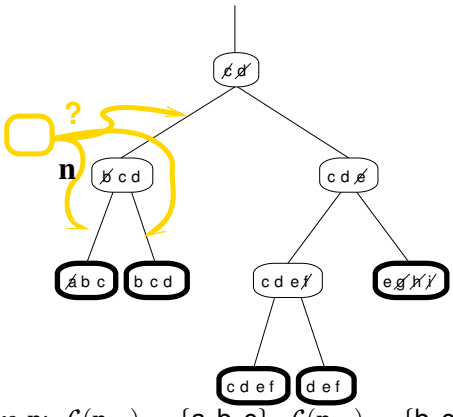
(h) Transferring **s** would make $\mathrm{bal}(\mathbf{r}) = \frac{3}{5}$, $\mathcal{L}(\mathbf{r}_{\swarrow}) = \mathcal{L}(\mathbf{r}_{\searrow}) = \{\mathsf{c}, \mathsf{d}, \mathsf{e}, \mathsf{f}\}$, $\mathrm{par}(\mathbf{r}) = 8$, so **s** is feasible but worth than (f).



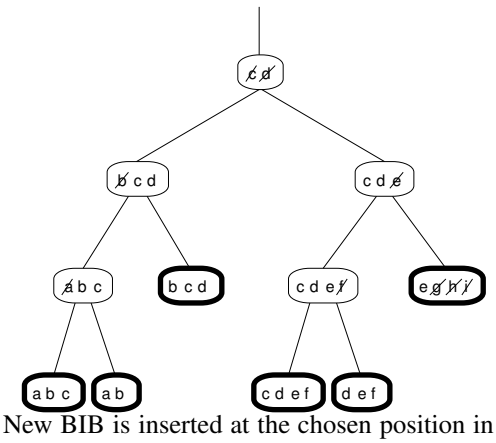
(i) The best subtree (f) is transfered to $\mathbf{r}_{\searrow}$. There is only one option for where to insert **s** here. As next operation a new BIB with landmarks $\{\mathsf{a}, \mathsf{b}\}$ will be inserted.



(j) Insert above **n**: $\mathrm{bal}(\mathbf{n}) = \frac{1}{6}$ infeasible; left below: $\mathcal{L}(\mathbf{n}_{\swarrow}) = \mathcal{L}(\mathbf{n}_{\searrow}) = \{\mathsf{c}, \mathsf{d}\}$, $\mathrm{par}(\mathbf{n}) = 4$; right below: $\mathcal{L}(\mathbf{n}_{\swarrow}) = \mathcal{L}(\mathbf{n}_{\searrow}) = \{\mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{d}\}$, $\mathrm{par}(\mathbf{n}) = 8$; left below is chosen.



(k) above **n**: $\mathcal{L}(\mathbf{n}_{\swarrow}) = \{\mathsf{a}, \mathsf{b}, \mathsf{c}\}$, $\mathcal{L}(\mathbf{n}_{\searrow}) = \{\mathsf{b}, \mathsf{c}, \mathsf{d}\}$, $\mathrm{par}(\mathbf{n}) = 6$; left below: $\mathcal{L}(\mathbf{n}_{\swarrow}) = \{\mathsf{b}, \mathsf{c}\}$, $\mathcal{L}(\mathbf{n}_{\searrow}) = \{\mathsf{b}, \mathsf{c}, \mathsf{d}\}$, $\mathrm{par}(\mathbf{n}) = 5$; right below: $\mathcal{L}(\mathbf{n}_{\swarrow}) = \{\mathsf{a}, \mathsf{b}, \mathsf{c}\}$, $\mathcal{L}(\mathbf{n}_{\searrow}) = \{\mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{d}\}$, $\mathrm{par}(\mathbf{n}) = 7$; left below is chosen.



(l) New BIB is inserted at the chosen position in the tree.

Figure 4.14: **`transferSubtree`** $(\mathbf{s}, \mathbf{a})$
s subtree to transfer; **a** insert above this node

d := new empty leaf; **ap** := new node with **d** and **a** as children; **sp** := $\mathbf{s}_\uparrow$						
Make **ap** child of $\mathbf{a}_\uparrow$ replacing **a**						
`update` (**ap**)[a]						
lca := least common ancestor of **a** and **s**						
All landmarks $l \in \mathcal{L}(\mathbf{s})$						
	IF	**eN** $[l]$ = **lca**				
	THEN	**y** := next y-node for l below $\mathbf{lca}_{\downarrow \mathbf{s}}$				
		IF	**s** is below **y**			
		THEN	**y** :=next y-node for l below $\mathbf{lca}_{\downarrow \mathbf{a}}$			
			eN $[l]$:= least common ancestor of **y** and **ap**			
	ELSE	IF	**lca** is ancestor of **eN** $[l]$	THEN	**eN** $[l]$:= **lca**	
Swap **s** and **a**						
Mark nodes from $\mathbf{s}_\uparrow$ and $\mathbf{a}_\uparrow$ to **lca** (*nonlinear:* to **root**) *invalid* and *to be optimized*						
`update` (**lca**); Delete **sp** and **d** [a]						
Update $\mathbf{n}_{size}$ for all **n** from **s**, **a** to the root. [b]						

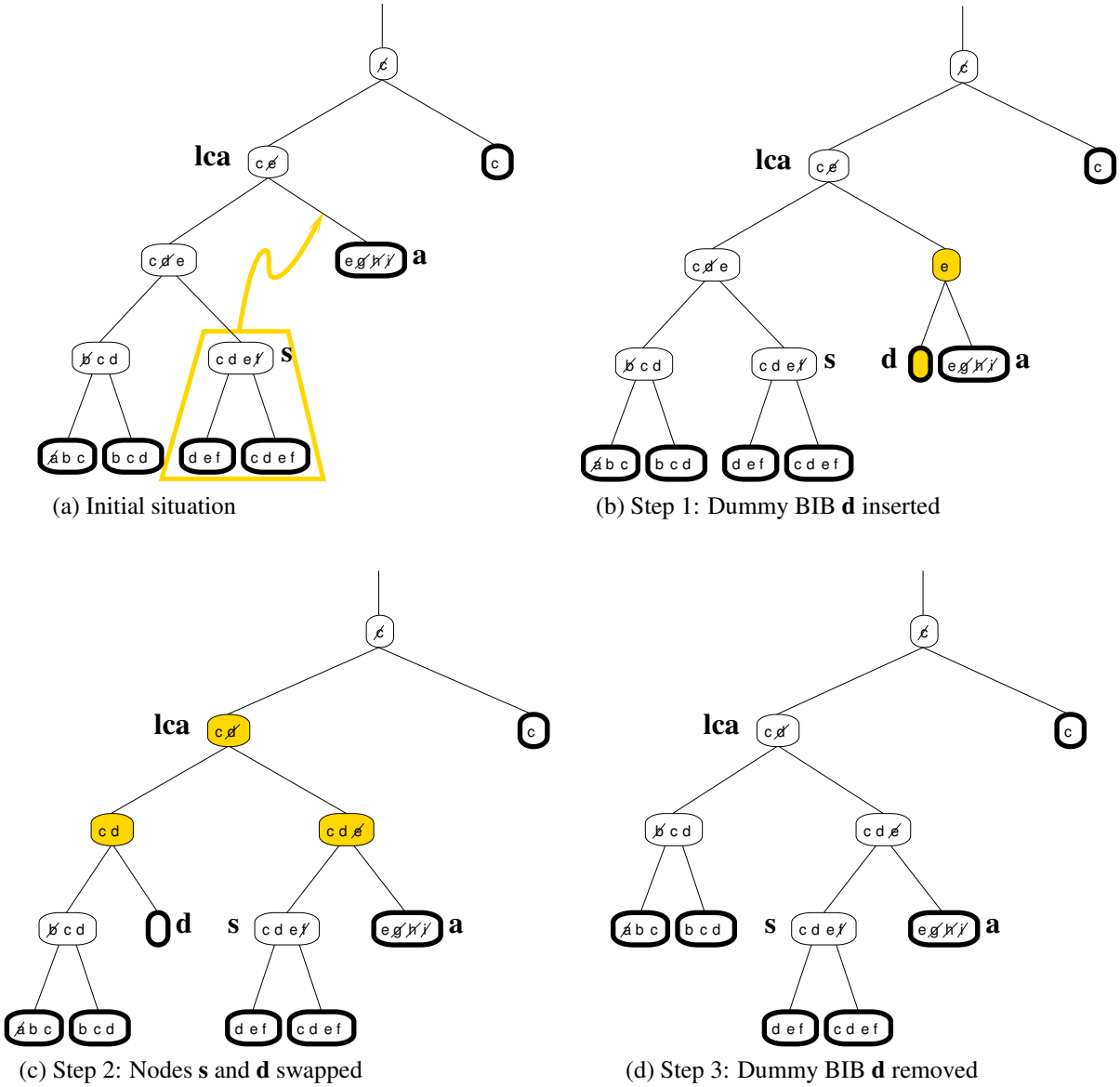
[a] Optimization: Update only the list of landmarks represented in the node and defer recomputation of the CIB and SIB to `compileEstimate`

[b] Entry $\mathbf{n}_{size}$ is the number of BIBs below **n**. It is used by the HTP subalgorithm (§4.4).

3. The SIB of **lca** is invalidated being computed from the CIBs of its children
4. In nonlinear mode: The nodes (CIBs and SIBs) from **lca** to **root** get invalid because **lca** and further nodes below have been recomputed using a new linearization point

For all landmarks l represented at **s** the elimination node may change when **s** is transferred. This does not lead to additional nodes becoming invalid like when a new BIB is inserted. Three different cases depending on the relation between **eN** $[l]$ and **lca** arise:

1. ***eN*** $[l]$ *is a proper ancestor of* ***lca***:
   eN $[l]$ will not change. No BIB moves outside or inside the subtree of **lca** and the same holds for **eN** $[l]$. (landmark `c` in figure 4.15)
2. ***eN*** $[l] =$ ***lca***:
   eN $[l]$ remains the same or moves down to some ancestor of **a**. Which is the case depends where landmark l is represented in the subtrees of both children of **lca**. If **s** is the only node below $\mathbf{lca}_{\downarrow \mathbf{s}}$ representing l, l is only represented below $\mathbf{lca}_{\downarrow \mathbf{a}}$ after the tree has moved and **eN** $[l]$ must be set to the least common ancestor of these nodes. Whether this is the case can be decided by finding the next y-node below $\mathbf{lca}_{\downarrow \mathbf{s}}$. If it is above **s**, another BIB



(a) Initial situation

(b) Step 1: Dummy BIB **d** inserted

(c) Step 2: Nodes **s** and **d** swapped

(d) Step 3: Dummy BIB **d** removed

Figure 4.15: Three steps of transferring subtree **s** to above **a**. The nodes that have to be updated are marked gray.

represents l and $\mathbf{eN}[l]$ remains the same. Otherwise $\mathbf{eN}[l]$ must be set to the least common ancestor of $\mathbf{a}_\uparrow$ and the next y-node below $\mathbf{lca}_{\downarrow\mathbf{a}}$. (landmark e in figure 4.15)

3. ***eN*** $[l]$ *is proper descendant of* ***lca***:
   Since landmark l is represented in **s** but not in **lca**, it must be represented in some BIB which is below **lca** but not below **s**. After the move of the subtree, landmark l will still be represented there and will additionally be represented at the new location of **s**. So the elimination node must be set to the least common ancestor **lca** (landmark d in figure 4.15)

A structure chart is shown in (Fig. 4.14, `transferSubtree`) and an example in figure 4.15.

4.6 Computational Efficiency

The algorithm's computation time depends on two parameters n and k. n is the number of landmarks and k is the number of landmarks represented in each BIB. n depends on the size of the building mapped and grows as the map gets larger. k is indirectly controlled by the parameter $maxDistance$ (see `isTooLarge`, §4.2) used by BIB changing control (5m and 7m in the experiments in this thesis). The algorithm groups landmarks into one BIB that have a mutual distance of at most $maxDistance$. The resulting k depends on the density of landmarks in the building and is a qualitative parameter for the building / landmark / sensor combination that does not grow when the map gets larger. It is assumed that each landmark is only contained in $O(1)$ different BIBs, so the overall number of BIBs is $O(\frac{n}{k})$.

Altogether, the algorithm is designed for the typical situation when k is small (≈ 10) and n is very large (> 1000). The algorithm uses an EKF as front end to process odometry and landmark observations. It maintains an estimate for the local landmarks represented in the actual BIB. Processing odometry and observations of these landmarks is efficiently performed by the EKF without having to consider the overall map. Computation time for this is $O(k)$ and $O(k^2)$ per measurement. This behavior is the same as shown by the Compressed EKF algorithm [GN02] and allows fully exploitation of sensors with high observation frequency.

When a landmark is observed not being represented in the actual BIB the actual BIB has to be changed and information transferred from the EKF into the tree map and vice versa. This is the main step of the algorithm. It takes $O(k^3 \log n)$ computation time as will be discussed in the following. This is much faster than the corresponding "global update" step in CEKF taking $O(kn^{\frac{3}{2}})$. There are three major factors contributing to the computation time needed:

1. *The linear algebra computation performed at each node*:
   The number of landmarks represented at a node's children determines the size of the ma-

trices involved in computation at this node. The time needed to perform computation for a single node is cubic in this number. To minimize the time spent the HTP subalgorithm (§4.4) optimizes the tree to represent as few landmarks in each node as possible. As discussed in §3.5 for topologically suitable buildings the algorithm can be expected to find a tree representing only $O(k)$ landmarks in each node. This assumption is further confirmed by the experimental result presented in §5 and §6. As a consequence the linear algebra computation performed at a single node takes $O(k^3)$ time.

2. *The number of nodes updated*:
   Basically, a node is computed from its children. If a node changes, only the ancestors of this node have to be updated. This is the fundamental advantage gained from representing information in a hierarchy. As the tree is balanced the algorithm never has to update more than $O(\log n)$ nodes, taking $O(k^3 \log n)$ computation time. In some cases additional nodes have to be updated (see §4.3) although never more than $O(\log n)$.

3. *The bookkeeping*:
   The algorithm performs a considerable amount of bookkeeping mainly to determine which nodes to update, to find an optimal point to insert a new BIB and to improve the tree to make further computation more efficient. The last point is a heuristical approximation to the hierarchical tree partitioning problem which is NP-complete in theory. Nevertheless, the bookkeeping part only uses $O(k^3 + k^2 \log n)$ computation time. This is asymptotically the same with respect to n and even less with respect to k than linear algebra computations.

The tree has $O(\frac{n}{k})$ leaves, so $O(\frac{n}{k})$ nodes. By definition 5 each node contains a fixed number of $O(k \times k)$ matrices. The amount of storage space needed is $O(nk)$. Normally, the algorithm provides an estimate for the landmarks represented in the actual BIB with $O(k^2)$ resp. $O(k^3 \log n)$ computation time. If an estimate for all landmarks is desired, `computeEstimate` must be recursively applied to all nodes. Evaluation of covariance in (3.23) can be omitted only computing $\hat{y} = H\hat{z} + h$ with $O(k^2)$ time. Since there are $O(\frac{n}{k})$ nodes it takes $O(nk)$ with an extremely small prefactor (table 5.2, 6.1) to compute an estimate for the whole map.

Table 4.1 shows the calling hierarchy of the algorithm together with the time needed for each subalgorithm. The time spend in linear algebra computation (`computeCIBandSIB`, `compileEstimate`, `removeCoupling`) is marked boldface. The implementation of `optimizeHTP` is described in appendix C, and the corresponding calling hierarchy in table C.1.

An overview of the performance of different algorithms established in literature is given in §2.14, table 2.2. Among the algorithms computing a sufficiently good estimate for closing loops the proposed algorithm is asymptotically fastest. When comparing the performance it has to be

integrateTreemapObservation	$O(k^3 \log n)$
isTooLarge	$O(k^2)$
findOrCreateBIB	$O(k^3 + k^2 \log n)$
allBIBsRepresenting	$O(k^2 \log n)$
$O(k)$ times	$O(k^3)$
isTooLarge	$O(k^2)$
createEmptyBIB	$O(k \log n)$
initLandmarkState	$O(k \log n)$
findBestInsertionPoint	$O(k \log n)$
$O(\log n)$ times	$O(k \log n)$
$\cdots$	$O(k)$
updateLandmarkState	$O(k)$
extractBIBFromEKF	$O(k^3)$
removeCoupling	$\mathbf{O(k^3)}$
setBIB	$O(k^3 \log n)$
$\cdots$	$O(k \log n)$
$O(\log n)$ times	$O(k^3 \log n)$
update	$O(k^3)$
computeCIBandSIB	$\mathbf{O(k^3)}$
O(1) times	$O(k^3 \log n)$
optimizeHTP	$O(k^3 \log n)$
$\cdots \implies$ See appendix C, table C.1.	
When closing a loop: O(1) times	$O(k^3 \log n)$
iteratedRelinearize	$O(k^3 \log n)$
$O(\log n)$ times	$O(k^3 \log n)$
update	$O(k^3)$
computeCIBandSIB	$\mathbf{O(k^3)}$
compileEstimate	$O(k^3 \log n)$
$O(\log n)$ times	$O(k^3 \log n)$
computeEstimate	$\mathbf{O(k^3)}$
compileEKF	$\mathbf{O(k^3)}$
integrateEKFObservation	$O(k^2)$

Table 4.1: Calling hierarchy of all subalgorithms with computation time needed. The calling hierarchy for `optimizeHTP` is shown in appendix C, table C.1.

kept in mind that the proposed algorithm only computes an estimate for the local landmarks not for the complete map, which takes $O(kn)$ computation time. However the involved prefactor is extremely small and normally only a local estimate is necessary anyway. Thus, it is a great advantage being able to compute a local estimate more efficiently than a complete map. Furthermore, the algorithm needs equal or less storage space than all other algorithms. Comparing to the requirements stated in §2.13 it may be concluded:

> *The tree map uses $O(kn)$ storage space being compliant to requirement (R2).*
>
> *Odometry and observation of a local landmark are integrated using $O(k)$ and $O(k^2)$ computation time. Every once in a while a global update is necessary needing $O(k^3 \log n)$ computation time for topologically suitable buildings. The time is spent in $O(\log n)$ linear algebra operations taking $O(k^3)$ each and $O(k^3 + k^2 \log n)$ bookkeeping overhead. If k is constant, the algorithm needs $O(1)$ for local measurements and $O(\log n)$ for a global update, which even goes beyond requirement (R3).*

4.7 Discussion

Summary

The bookkeeping part maintains the hierarchy by optimizing the tree and calling linear algebra routines to update nodes when necessary. A main task is to determine, which nodes have to be recomputed during a global update.

A precondition for efficient computation is, that the tree is balanced and partitions the BIBs so only $O(k)$ landmarks are represented at each node. In order to maintain this property the algorithm employs a hierarchical tree partitioning (HTP) subalgorithm. In theory, the HTP problem is NP-complete. For keeping computation time bounded the algorithm performs one optimization step per global update, optimally moving a single subtree to a different location. Three criteria formalize the idea of a balanced and well partitioned tree. They were designed in a way that allows to compute the best optimization step in $O(k^2 \log n)$.

While updating the algorithm applies nonlinear rotations to the affected nodes to reduce linearization error. In case of closing a large loop relinearization is performed several times until the result converges. With this strategy the linearization error is reduced to a negligible extent.

A tree map needs $O(nk)$ storage space and computation time needed by the algorithm is: $O(k)$ for odometry, $O(k^2)$ for observation of a local landmark and $O(k^3 \log n)$ for a global update. This complies with requirements (R2), surpasses (R3) and is much faster than CEKF which needs $O(kn^{\frac{3}{2}})$.

Annotations

How does the tree depend on the measurements?

It is interesting to note that the algorithm is oblivious apart from some small exceptions: The computation applied to measurements and measurement uncertainties is independent of their actual values. The shape of the tree and whether a landmark is represented at a node or not only depends on which landmarks are observed when a certain BIB is the actual BIB. The only exception is the control used for changing the actual BIB depending on the estimated position of the landmarks. A practical advantage of such a property is that the bookkeeping part can be verified without actually working on data.

What happens, when moving through an already known area?

By the nature of least squares estimation moving through an already known area will give independent information on the same landmarks and thus reduce the overall map uncertainty. Also in this case the algorithm does more than just localizing the robot. However, most of the bookkeeping work is only necessary while the map is extended, i.e. new landmarks are being observed: No new BIBs and no new landmarks have to be inserted, so the elimination nodes do not change and there is no necessity of optimizing the tree. The only operation performed is an update from the actual BIB to the root and compilation of an estimate. So in this case the algorithm becomes faster by a considerable constant factor.

How can it be possible to reduce computation time from $O(n^2)$ to $O(\log n)$?

Assuming $k = O(1)$, the algorithms computation time $O(\log n)$ appears to be surprisingly low compared to $O(n^2)$ of EKF. The main operation of EKF basically is the update of the inverse of a matrix after a change of small rank [Hag89]. This is a general well known problem most likely not be expected to be solved in significantly less computation time. Presumably the only way to reduce the computation time is to exploit a specific property of the SLAM problem making SLAM different from general least squares estimation.

The presented algorithm follows this idea exploiting the specific structure of SLAM in two ways: First it is only necessary to provide an estimate for local landmarks. Second at least typically, the map can be decomposed into a hierarchy in a very sparse and loosely connected way. These two are powerful properties making it possible to devise a much more efficient algorithm for this special case than it would be possible for general least squares estimation.

Chapter 5

Simulation Experiments

This chapter presents the simulation experiments conducted to verify the algorithm with respect to the requirements (R1)-(R3) proposed in §2.13. The simulation environment computes exact measurements from a CAD model of the simulated building and robot trajectory and corrupts the measurements with artificial Gaussian noise before passing them to the estimation algorithm. The simulation experiments focus on behavior of the algorithm as an approximate least square or ML estimator. It takes for granted that least square estimation is indeed a reasonable way to create a map, which has been proven in literature. The experiments evaluate map quality (R1), storage space (R2) and computation time (R3) of the proposed algorithm for three different scenarios: a typical building, a scenario with large measurement noise and a large scale map. For this purpose, a simulation approach is advantageous allowing to repeat the same experiment with identical measurements but new independent measurement noise. This way statistical quantities e.g. error covariance matrix can be determined by Monte Carlo simulation. With these quantities, the quality of *any aspect* of the map can be evaluated as demanded by requirement (R1) "bounded uncertainty" (§2.13).

Evaluation of practical feasibility of the algorithm including real data, unreliable landmark detection and problems of landmark identification will be performed in §6.

All simulations have been conducted on an Intel Xeon, 2.67 GHz, 512 MB, `LINUX`, `gcc 2.95.3`, options `-O3`.

5.1 Scenario

The simulation program uses an a-priori CAD model of the building and of the robot trajectory to be simulated. In this chapter the same fictional building as in §2 is used. Simulation of the robot's motion is composed of alternating movement and observation steps. For each movement

Scenario	Landmark sensor				Odometry		
	distance noise[a]	distance bias[b]	angular noise	field of view	velocity noise[c]	orient. bias[d]	robot radius
Small noise	2.5%		2°	±70°, 3m	$0.01\sqrt{\text{m}}$		0.3m
Large noise	10%	$4.5\frac{\text{mm}}{\circ}$	5°	±70°, 3m	$0.07\sqrt{\text{m}}$	$5\frac{\circ}{m}$	0.3m
Large scale map	2.5%		2°	±70°, 3m	$0.01\sqrt{\text{m}}$		0.3m

Table 5.1: Artificial measurement noise parameter: [a]proportional to distance, [b]proportional to observation angle, [c]proportional to square root of speed, [d]proportional to distance traveled

step the relative pose is passed over to the algorithm. In contrast, measurements are simulated for each landmark in the simulated field of view for an observation step. The ideally simulated measurements of position relative to the robot (see §2.1) are corrupted by artificial Gaussian noise and passed on to the SLAM algorithm including the landmarks' identities. The noise parameters are displayed in table 5.1. Artifical noise is generated with random numbers using the routines `ran4` and `gasdev` as recommended by [PTVF92, §7].

Constant angle noise and proportional distance noise is an appropriate noise model for the landmark sensor. The noise model for odometry is assuming a two wheeled robot with wheel revolution measurement corrupted by continuous Gaussian noise. Basically, experience shows that the error accumulated while traveling a certain distance is independent of traveling speed. According to continuous noise theory this is achieved, when noise is proportional to square root of velocity. Consequently, accumulated noise is proportional to square root of distance traveled.

To give a figure: If the proportionality constant is $0.01\sqrt{\text{m}}$, as in the small error experiments, after traveling 4m, each wheel has accumulated an error of $0.01\sqrt{\text{m}}\cdot\sqrt{4\text{m}} = 0.02$m. If the wheels have a distance of 0.3m to the robot's center, this leads to an orientation error of $\frac{\sqrt{2}\cdot 0.02\text{m}}{2\cdot 0.3\text{m}} \approx 2.7°$.

In the large noise simulation experiment, an additional bias both on landmark sensor and odometry is assumed. Unequal wheel diameters or calibration errors can be a cause for such bias. It is probably the most common reason for excessively large orientation error of a mobile robot. The bias is multiplied by a random number $\in [-1 \ldots +1]$. It is constant for each experiment but varies randomly over several runs.

The first two experiments ("small noise", "large noise") are based on the same building also used in illustrations in §2 but with a longer trajectory (Fig. 5.1a). They were designed to analyze map quality in case of low and high measurement error. The third experiment ("large scale map") is a large 10×10 cell grid, each grid cell being a copy of the building used before (Fig. 5.4). It contains $n = 11300$ landmarks and was designed to analyze asymptotical behavior of storage space and computation time.

The algorithm's parameters are $optHTPSteps = 5$ steps of tree optimization per global

update and $maxDistance = 5$m or $maxDistance = 7$m as maximum diameter of a region. For easier understanding walls are included in the drawings. In a real implementation this would certainly require the landmark sensor to detect these walls.

5.2 Small Noise Experiment

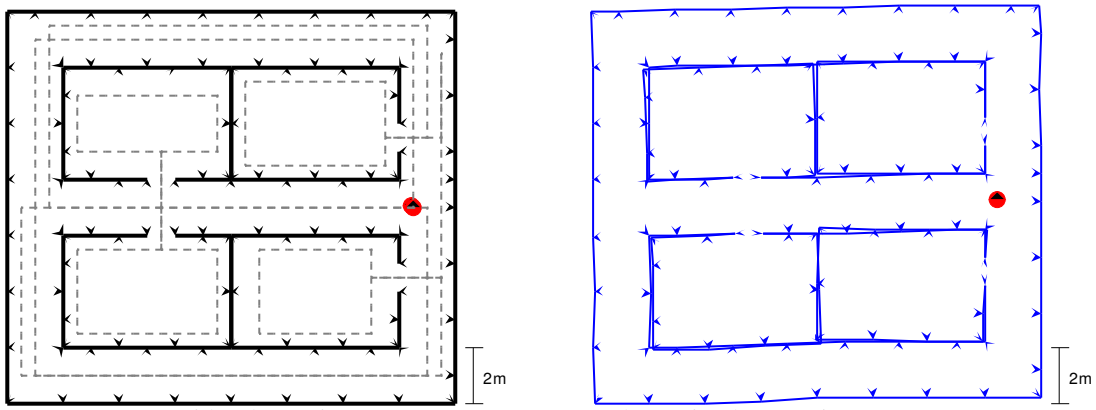
The small noise simulation experiment allows statistical evaluation of the estimation error and comparison with EKF and ML, verifying that requirement (R1) "bounded uncertainty" is met. Figure 5.1 shows the result using the same $16\text{m} \times 14\text{m}$ map as throughout §2.

The optimal ML estimate (Fig. 5.1b) shows little error being realistic for a well calibrated robot and a map of that size. It can be observed that room and corridors slightly overlap. Since all landmark based algorithms just estimate positions and have no notion of occupied and free space this result was expected. Of all the algorithms in §2.14 fastSLAM [TBF98] is the only one to incorporate this kind of information. The EKF estimate (Fig. 5.1c) resembles the ML estimate at visual inspection with the orientation error being comparatively small and inducing no visible linearization problems. In contrast, rigorous comparison of covariances of the algorithms involved shows that there is still a significant difference between ML and EKF estimate. Finally, the estimate computed by the proposed algorithm (Fig. 5.1d) also looks similar only with the upper left room being more tilted than in other estimates. At visual inspection the three estimates basically appear of same quality and are perfectly usable for navigation.

The tree representation of the map internally used by the proposed algorithm is shown in figure 5.2a indicating that the tree is well balanced and well partitioned, i.e. no node represents too many landmarks. Conclusively, the building is indeed topologically suitable in the sense discussed in §3.5.

Figure 5.2b compares the relative error in the three estimates in *all aspects* of the map. The error covariances $C = E\left((\hat{x} - x_{\text{true}})(\hat{x} - x_{\text{true}})^T\right)$ for the proposed algorithm, C_{EKF} for EKF and C_{ML} for ML are approximatively determined by Monte Carlo simulation with 1000 runs. The number of necessary runs is growing with the size of the covariance matrix. So only the covariance of eight selected landmarks, one in each corner and one in each room is evaluated. The covariance C is compared to C_{EKF} as well as C_{ML} considering all aspects of the map as discussed in §2.13. To give an example: If a particular aspect i.e. a landmark coordinate or distance between two landmarks has a relative error of 110%, this means that the error of this aspect in the estimate from the proposed algorithm is 10% larger than in the ML estimate.

Comparison for all aspects of the map is performed by computing generalized eigenvectors corresponding to independent aspects. The eigenvalue corresponding to an eigenvector gives the



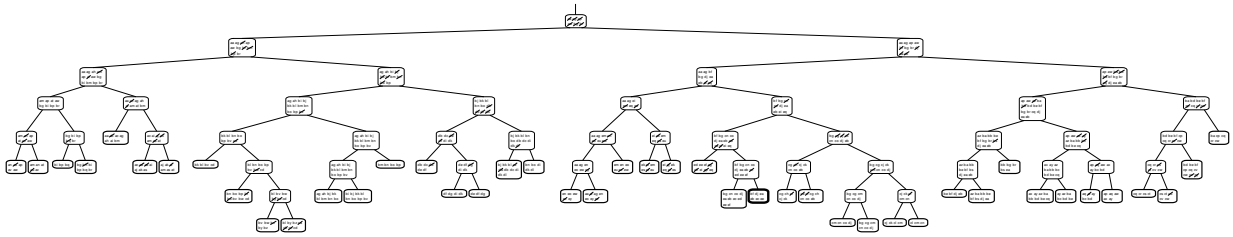
(a) True map with robot trajectory.

(b) Optimal ML estimate.

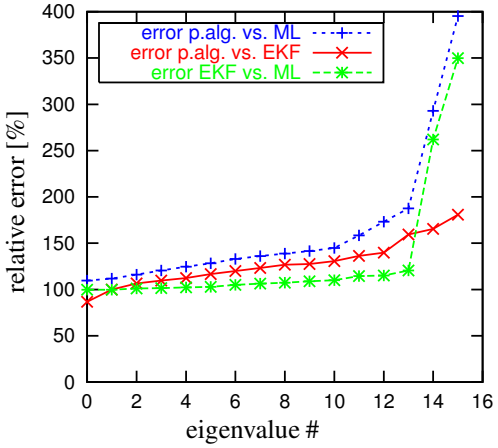
(c) EKF estimate.

(d) Estimate generated by the proposed algorithm.

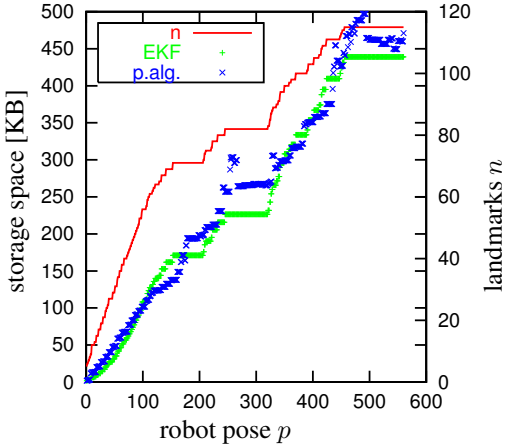
Figure 5.1: Small noise simulation experiment: True map and estimates generated by different algorithms. The robot starts at the position indicated by the circle/triangle symbol (Fig. 5.1a) moves up-left-down-right along the outer corridors, maps the lower-right room, maps the upper-right room, moves up-left-down around the outer corridor again and through the central corridor, mapping the upper-left and lower-left room on the way. Finally, it performs a second down-left-up loop along the outer corridor and back through the central corridor to the starting position.



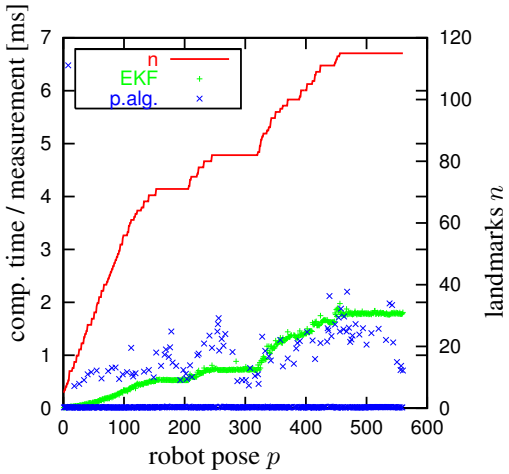
(a) Tree representation of the map. Size of the node ovals is proportional to number of represented landmarks.



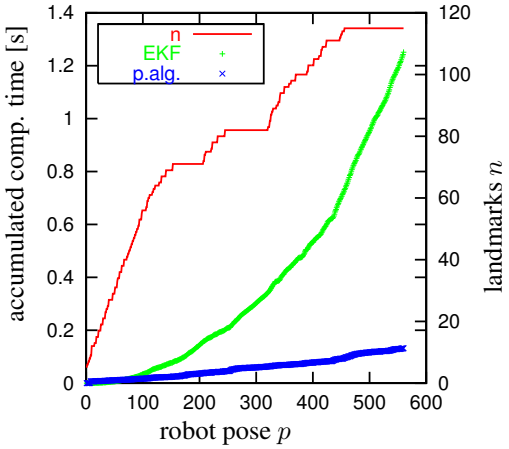
(b) Relative error spectrum compare to ML and EKF.



(c) Storage space.



(d) Computation time per measurement. Observe the local updates ≈ 0.02ms.



(e) Accumulated computation time.

Figure 5.2: Small noise simulation experiment: (a) Internal tree representation used by the proposed algorithm, (b) average relative error compared to ML and EKF map, (c) storage space, (d) computation time per measurement, (e) accumulated computation time.

relative error in the two estimates for the aspect corresponding to the eigenvector. The sorted eigenvalues are shown in figure 5.2b. The smallest eigenvalue is 110% (87% vs. EKF) and the largest 395% (181% vs. EKF). This means that the map estimate computed by the proposed algorithm 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 the proposed algorithm. Observed by Frese and Duckett [FD03] linearization errors in the relative landmark-robot position, though small, can create apparent orientation information perturbing orientation of the overall map or individual rooms.

The overall result meets requirement (R1) very well. Any real structural problem in an algorithm would lead to relative errors beyond $\gtrapprox 10000\%$: For instance, the error of an open loop has a magnitude of some meters, whereas the error of a closed loop has a magnitude of some centimeters, so any algorithm that can not close loops would have a relative error of $\approx 10000\%$.

The next figure plots storage space requirement of the proposed algorithm and EKF over robot pose p (Fig. 5.2c). Further the number of landmarks n is plotted over p. With 497KB and 439KB the proposed algorithm and EKF have similar memory requirements. Apparently, the asymptotical difference between $O(kn)$ and $O(n^2)$ is not yet strong enough for $n = 115$. The result is plotted w.r.t. p instead of n because when the robot moves through an already known area, n does not increase. Thus, it is difficult to compare the plot and the respective $O(k^2)$, $O(kn)$ and $O(k^3 \log n)$ asymptotical expressions. Comparability is facilitated in a large scale map experiment (§5.4) because in this experiment n is basically growing monotonically with p and all plots are w.r.t. n. Thus, the discussion about requirement (R2) and (R3) is deferred to the large scale map experiment.

Figure 5.2d and 5.2e show the computation time for a single measurement and accumulated computation time respectively. Computation time per measurement for the EKF grows up to 9.05ms for $n = 115$. The corresponding plot for the proposed algorithm is more difficult to interpret: Most operations are local updates and require 0.02ms time. The time needed for a global update is 1.12ms in average but grows upto 6.48ms depending on the number of nodes updated (scattered points in the plot). This is comparable to the time needed by the EKF but this number alone does not give the whole picture: A global update is only performed after several meters of traveling and there is never more than one global update for a single robot pose. Consequently, computation time per robot pose and accumulated computation time are much lower for the proposed algorithm than for EKF (Fig. 5.2e).

5.3 Large Noise Experiment

The large noise simulation experiment evaluates the quality of the map estimate with respect to requirement (R1) when a large accumulated orientation error ($140°$) occurs before closing a loop. It is the same example as used before, except for larger artificial sensor noise and bias. The magnitude of the error is shown in figure 2.3b before closing the first loop. When closing a loop relinearization is performed (see §3.9,§4.1) until no BIB changes more than $maxAngle = 2°$.

Figures 5.3a-c show the estimates provided by different algorithms. The EKF estimate is clearly unusable, whereas the estimate of the proposed algorithm appears to be slightly worse than the ML estimate showing that the proposed algorithm still works extremely well, especially when compared to the result of EKF.

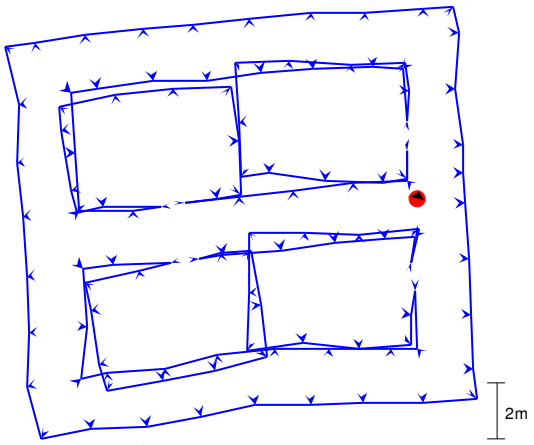
The relative error spectrum is shown in figure 5.3d ranging from 41% to 1229%. It can be seen that the ML estimate is not optimal any more since the smallest eigenvalue 41% is below 1. This is due to the sensor bias not considered in the ML likelihood model. The typical (median) relative error is 187%, whereas the two outliers have increased to 654% and 1229%. This is consistent with linearization effects causing this behavior, since larger sensor noise leads to larger linearization errors. In general, the estimate is still excellent when taking into account that all other nonlinear algorithms need much more computation time. Indeed problems generated by linearization are largely ignored in literature.

Computation time is basically similar to the previous example with local update below 0.02ms and global update 2.30ms with some outliers where loops are closed and several iterations are required to reduce the linearization error. Considering that nonlinear problems usually are much more difficult to solve than linear ones, it is extremely surprising that such a good estimate can be computed with so little additional effort.

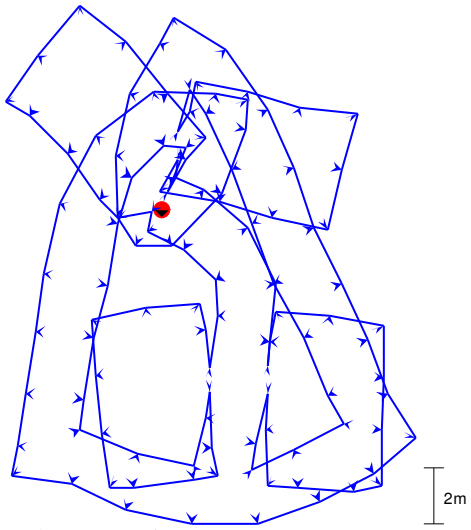
5.4 Large Scale Map Experiment

The third experiment uses an extremely large map consisting of a 10×10 cell grid with each cell being a copy of the building used in previous experiments. The overall experiment encompasses $n = 11300$ landmarks, $m = 312020$ measurements and $p = 63974$ robot poses. The intention is to show performance of the algorithm on large scale maps and to determine the prefactors involved in the asymptotical $O(k^2)$, $O(k^3 \log n)$ and $O(kn)$ expressions for computation time and storage space. Figure 5.4 shows the true map with the robot moving columnwise through the grid using the same trajectory (Fig. 5.1a) in every cell. The result is shown in figure 5.5.

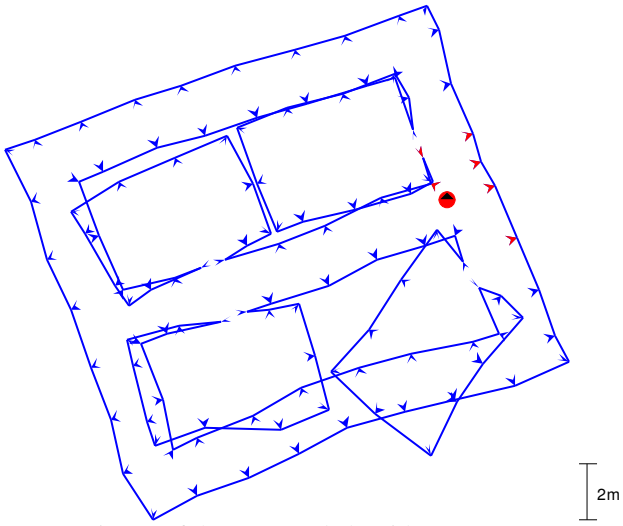
The tree representation stores an average number of $k = 5.81$ landmarks in each leaf (BIB) of the tree. In figure 5.6a the storage space consumption of the proposed algorithm and EKF is



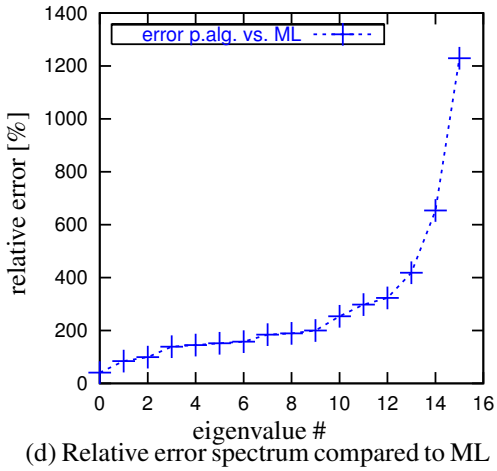
(a) ML estimate.



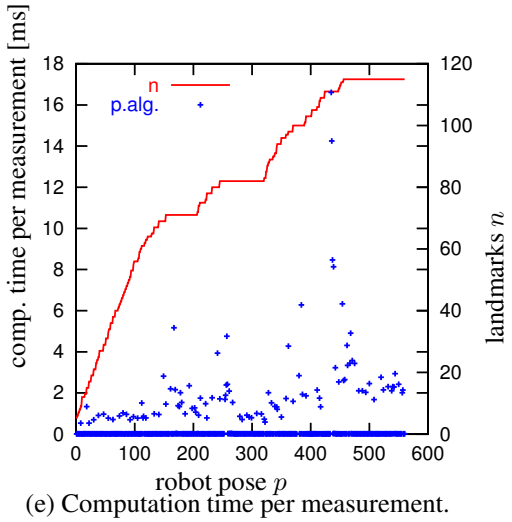
(b) EKF estimate.



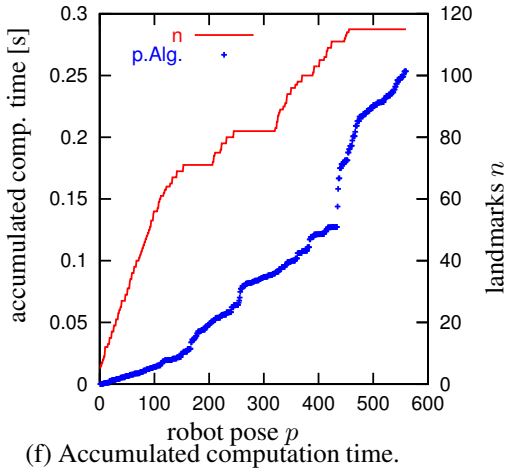
(c) Estimate of the proposed algorithm.



(d) Relative error spectrum compared to ML



(e) Computation time per measurement.



(f) Accumulated computation time.

Figure 5.3: Large noise simulation experiment: (a)-(c) Estimates generated by different algorithms, (d) relative error compared to the ML map, (e)-(f) computation time.

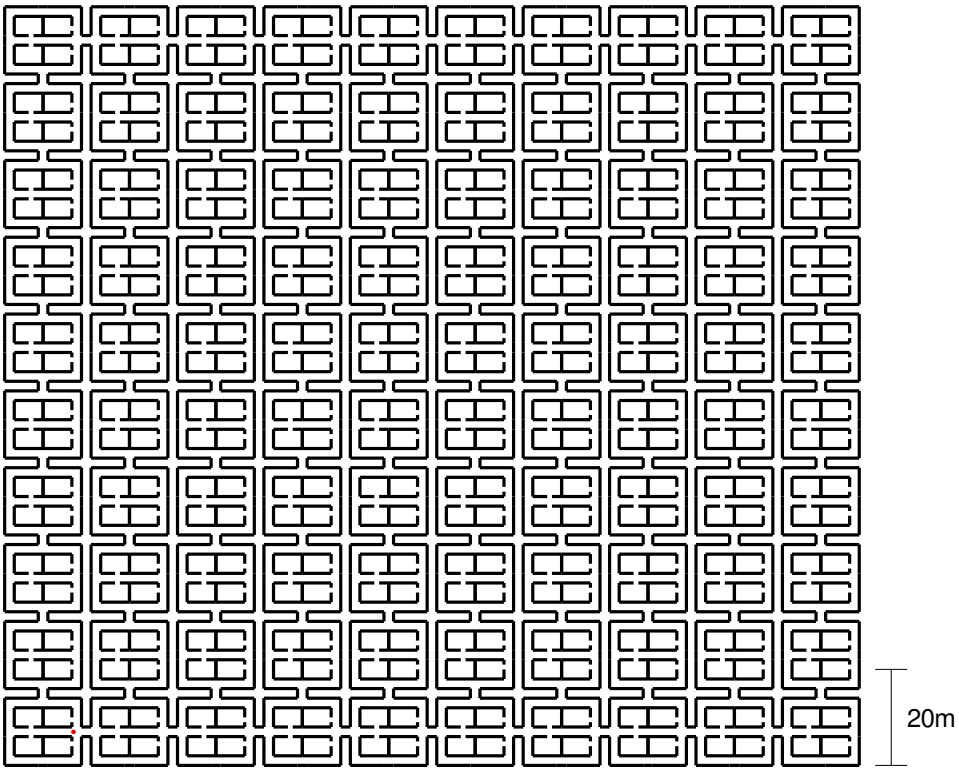


Figure 5.4: Large scale simulation experiment: True map. The building consists of a 10×10 cell grid, where each cell is a copy of the building shown in figure 5.1a. The overall experiment encompasses $n = 11300$ landmarks, $m = 312020$ measurements and $p = 63974$ robot poses.

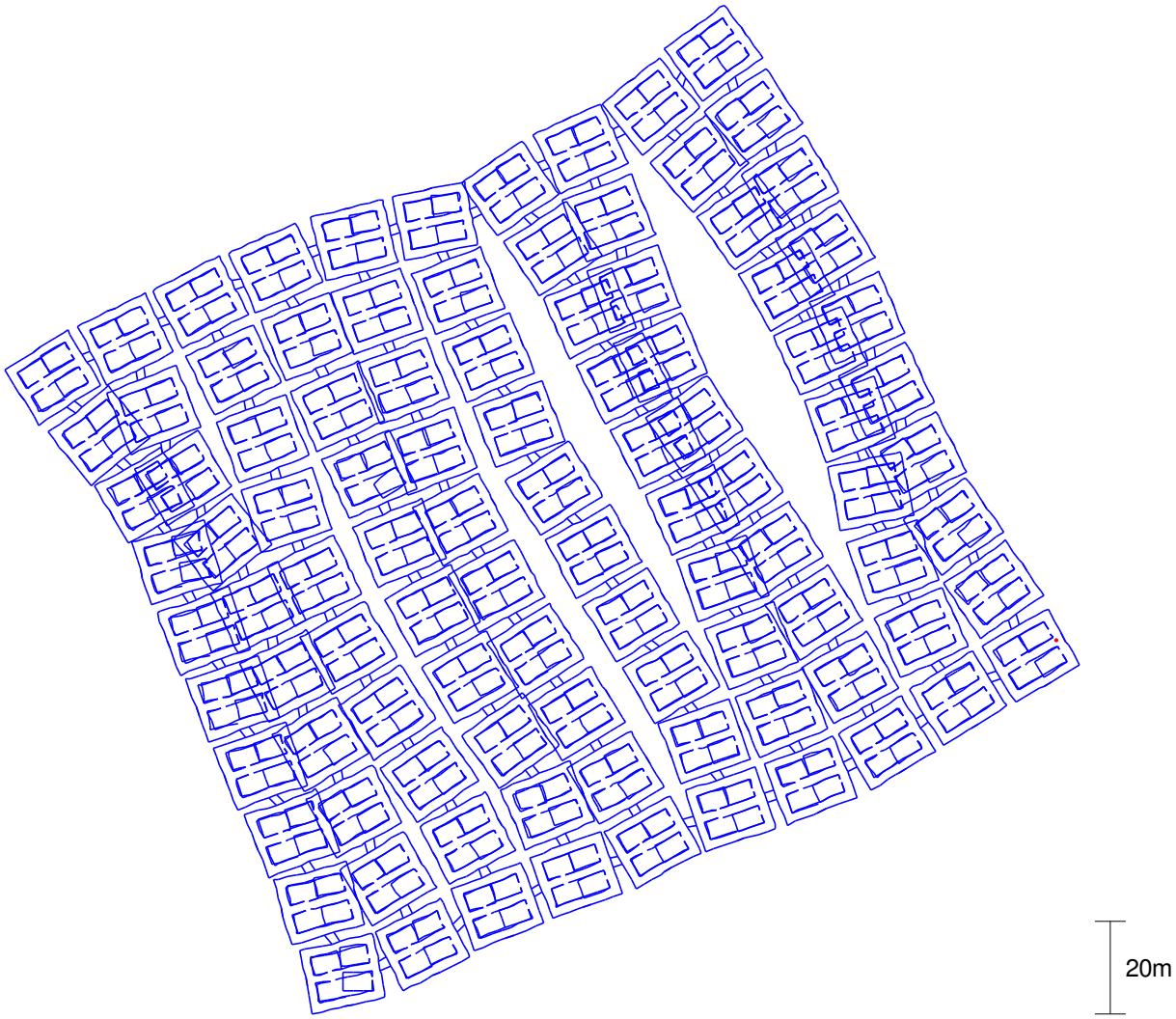


Figure 5.5: Large scale simulation experiment: Map estimate. As in previous experiments, several corridors overlap and the overall orientation has a large error both caused by the fact that errors are accumulating.

computation time			storage space
local update	global update	global map	
$1.21\mu\text{s} \cdot k^2$	$0.38\mu\text{s} \cdot k^3 \log n$	$0.15\mu\text{s} \cdot kn$	$679\text{B} \cdot kn$
0.02ms	2.51ms	5.62ms	25002KB

Table 5.2: Average computation time, storage space and corresponding average prefactors for asymptotical expressions (Intel Xeon, 2.67 GHz, $k \approx 5.81$, $n = 0 \dots 11300$).

plotted over n. The EKF experiment was aborted earlier due to large computation time. It is evident that storage space consumption is linear for the proposed algorithm ($O(kn)$) and super-linear ($O(n^2)$) for EKF. The prefactor involved in $O(kn)$ is shown in table 5.2.

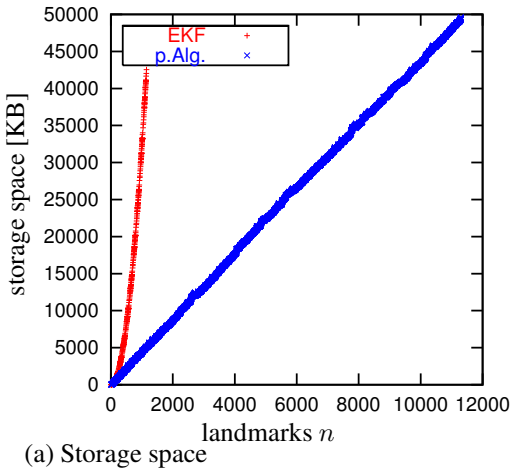
Computation time per measurement and accumulated computation time are shown in figures 5.6b-c. Time for three different computations is given: Local updates (dots below < 0.5ms), global updates computing a local map (scattered dots above 0.5ms) and the additional cost for computing a global map are plotted w.r.t. n. The algorithm is extremely efficient updating an $n = 11300$ landmark map in 12.37ms and on average 0.20ms per measurement. As predicted by theory, computation time for a local update $O(k^2)$ does not grow, computation time for a global update $O(k^3 \log n)$ grows slowly and time for computing a global map $O(kn)$ grows linear in n. All corresponding prefactors are shown in table 5.2. In particular, the prefactor for computing a global map is extremely small being the most impressive result from a practical perspective.

Figure 5.6d shows the height of the internal tree representation plotted over n. Its growth is moderate with n being a prerequisite for efficient computation.

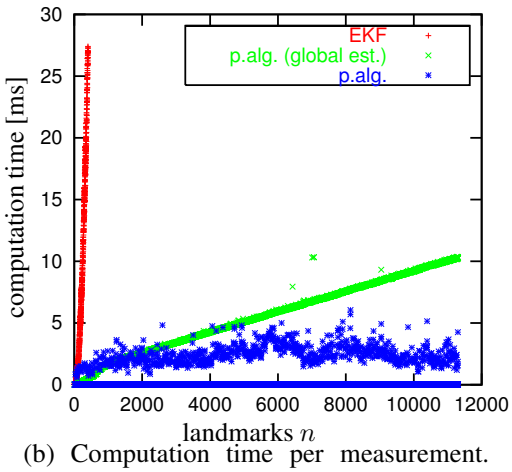
5.5 Discussion

The simulation experiments described in this chapter clearly show the algorithm's extreme efficiency when computing large maps. Even in situations subject to large orientation error a high quality estimate is generated. At visual inspection all maps look excellent in comparison to optimal maximum likelihood estimates. When strictly applied the relative error criterion (§3.5) reveals two aspects in which the algorithm produces an estimate significantly inferior to the ML estimate. However, this problem is also apparent in the EKF estimate assumed being caused by linearization. Besides, comparison of the algorithms confirms that the error is only slightly larger (median relative error 137% and 187% with large orientation error) higher than the minimal error possible. Storage space consumption has been found to be indeed linear in n although with a fairly high prefactor.

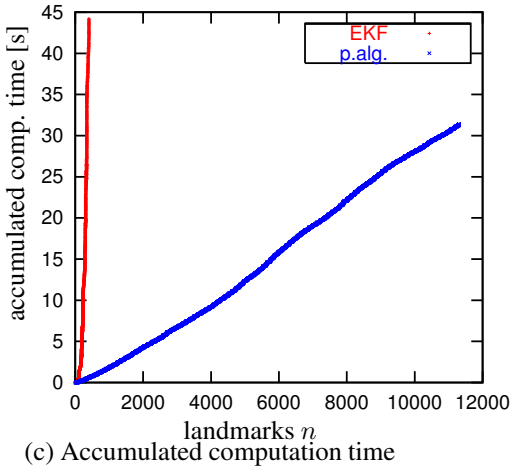
Conclusively, the algorithm meets all three requirement proposed in §2.13 as far as these experiments show.



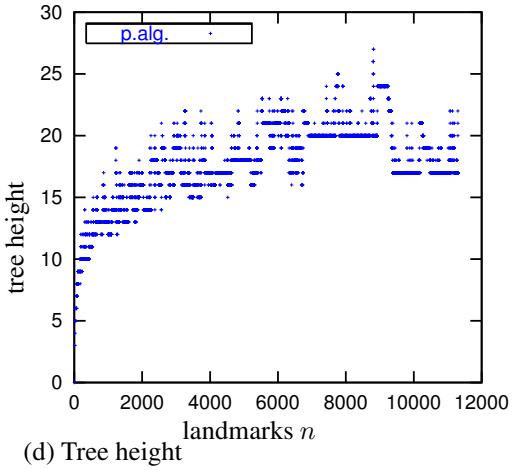
(a) Storage space



(b) Computation time per measurement. Observe the local updates ≈ 0.02ms.



(c) Accumulated computation time



(d) Tree height

Figure 5.6: Large scale simulation experiment: (a)-(c) Storage space and computation time over number of landmarks n. Only minimal additional computation time is needed to compute a global map instead of only a local one. (d) Tree height over number of landmarks.

Chapter 6

Real World Experiments

The real world experiments reported in this chapter are used to demonstrate how to apply the algorithm proposed in practice by mapping the DLR Institute of Robotics and Mechatronics' building (Fig. 6.1). It is used as an example for a typical office building and indeed "topologically suitable" as defined in §3.5. The algorithm is generating a balanced and well partitioned tree representation online.

In the first experiment the 60m $\times$ 45m building is mapped with three large circles. It is one of the largest maps reported in literature. The second experiment is a U - shaped trajectory of similar size being repeated 17 times to allow statistical evaluation of map quality. Finally, results from autonomously navigating with a SLAM acquired map will be presented.

All simulations were conducted on an Intel Xeon, 2.67 GHz, 512 MB, `LINUX`, `gcc 2.95.3`, options `-O3`.

6.1 Scenario

Mobile Robot System

In the experiments a mobile robot (Fig. 6.2a) developed by Hanebeck et al. [HSFS00] in the DIROKOL research project[1] was used. The robot is equipped with four wheels, each with one motor for steering and another one for moving. Since wheel axis and steering axis intersect, the robot is able to move omni-directionally but is not holonomic. Internal robot control estimates the robot's velocity with corresponding covariance. The estimate is based on the wheels' angular velocities measured by incremental encoders and the nonholonomic constraint that the wheel

[1] "Dienstleistungsroboter in kostengünstiger Leichtbauweise" funded by "Bayerische Forschungsstiftung"

moves perpendicularly to its axis. From the estimated velocity, the robot's odometric position with covariance is integrated by equations (2.1) and (2.7).

The robot is equipped with a stereo camera system[2] mounted on a pan-tilt unit[3] at a height of 1.55m. Both cameras have wide angle conversion lenses ($\times 0.45$) achieving a field of view of $\pm 45°$. In the experiments, only one camera and a fixed pan / tilt position is used and the camera is calibrated in that position with respect to the floor plane using a pinhole camera model with third order radial distortion[4]. In order to avoid problems with interlacing while the robot is moving only a single frame (every even line of the image) is processed.

Building

All experiments were conducted on data recorded in the DLR Institute of Robotics and Mechatronics' building, a typical office environment. Building and trajectory are shown in figure 6.1. The size is 60m $\times$ 45m with 29 rooms included. The trajectory's total length is 505.01m and encompasses three large loops. For closing the largest loop the robot moved outdoor from one part of the building to the other.

SLAM Algorithm

Since the accumulated orientation error in the real experiments is not too large, the algorithm is used in linear mode. Like in the simulation experiments 5 HTP optimization steps (see §4.4) are performed in each global update. The maximal diameter of a region is slightly larger (7m) than in the simulation experiments because of the robot's wider observation range.

6.2 Computer Vision for Landmark Detection

In literature a lot of different types of landmarks and methods for landmark detection are proposed. They are distinguished as artificial landmarks, i.e. landmarks deployed for the purpose of localization, and natural landmarks, i.e. landmarks already present in the environment[5]. Borenstein et al. [BEF96, §6] give an overview of different types of landmarks and summarize "Artificial landmark detection methods are well developed and reliable. By contrast, natural landmark navigation is not sufficiently developed yet [...].". In recent years, laserscanners became most

[2] A standard PAL cam corder zoom / iris / auto focus camera module: Sony EVI-371DG

[3] Directed Perception PTU-46-17.5

[4] Radial distortion model: $r \mapsto r + \frac{a_2 r^2 + a_3 r^3}{1 + b_1 r + b_2 r^2 + b_3 r^3}$

[5] By this definition, walls, edges, door, etc. are natural landmarks in an office environment.

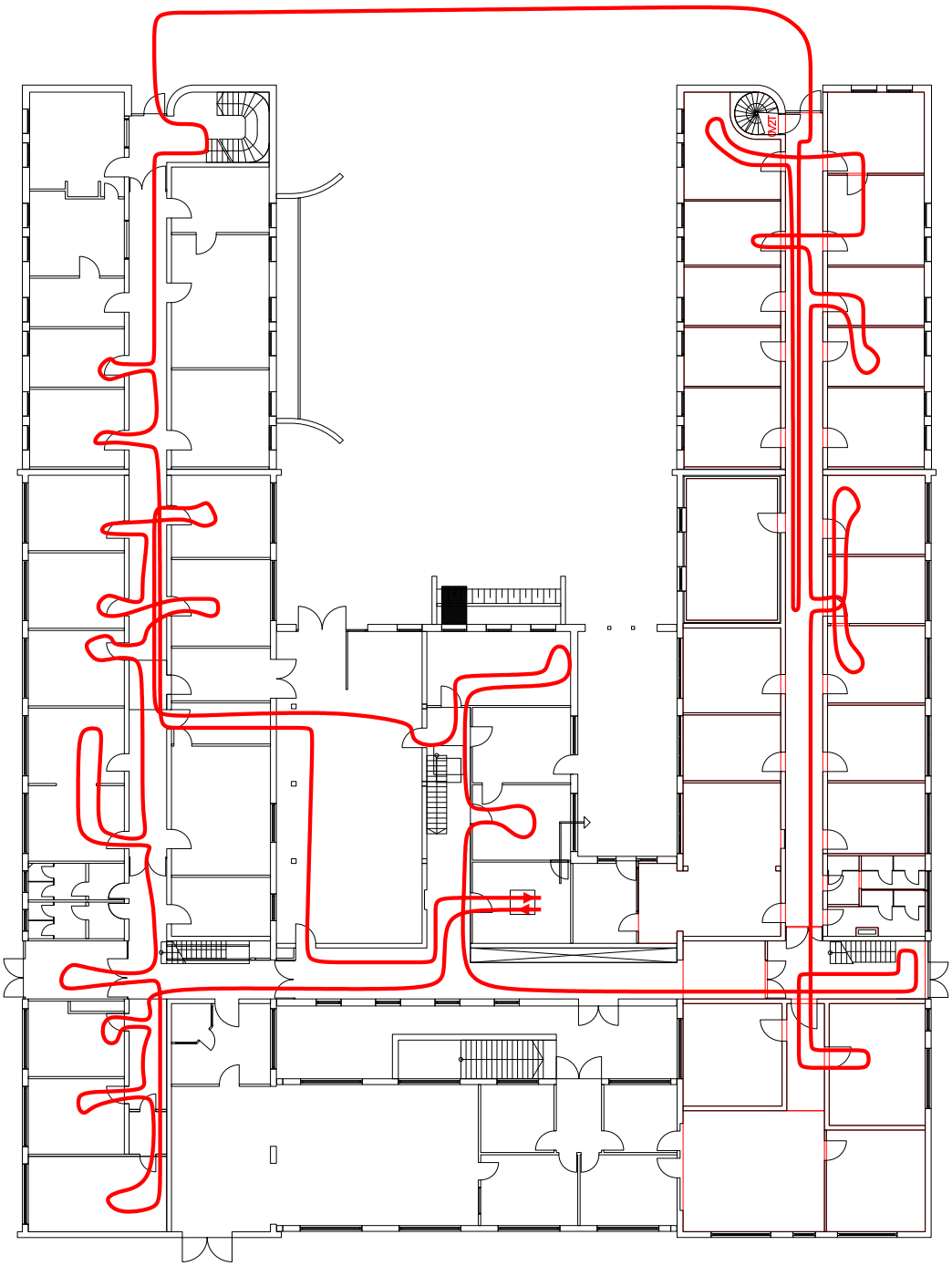
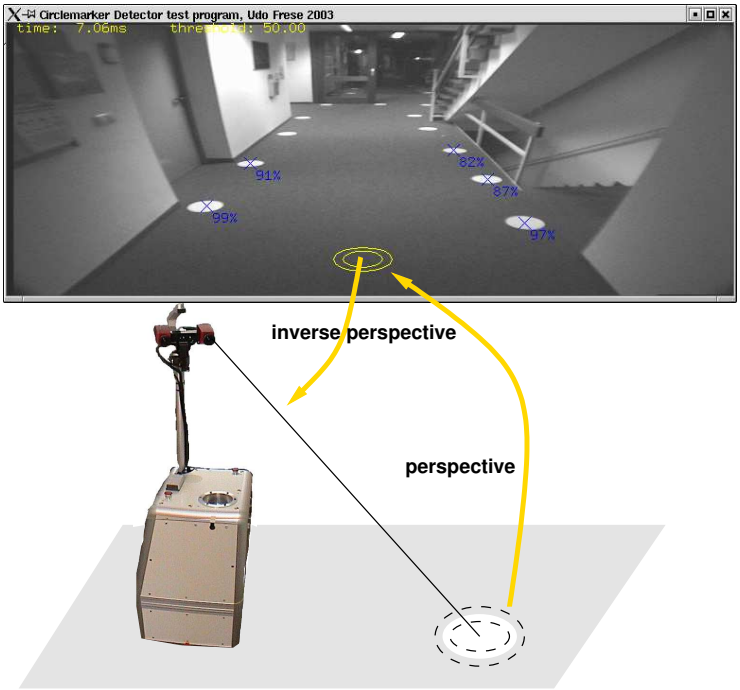


Figure 6.1: DLR building ($60\text{m} \times 45\text{m}$) used for real experiments with a sketch of the robot trajectory (505.01m). Start and end of the trajectory are indicated by triangles in the small room in the center of the building. The robot leaves the building at the right upper corner and re-enters at the left upper corner. It was necessary to travel outdoors in order to close a large loop.

(a) View of robot and building with landmarks.



(b) Landmark detection

Figure 6.2: (a) Artificial circular landmarks used in the experiments. (b) Landmark detection: For each image pixel the perspective outline of two circles centered at the corresponding point on the floor is computed and stored in a lookup table. Detected landmarks as shown by dark crosses together with computed quality measure.

popular type of sensor for SLAM and navigation. Commercially available scanners [SIC04] provide high quality distance measurements with an error of a few centimeters. So detection of walls or edges as landmarks has become much easier. Often, explicit landmark detection can even be completely avoided by comparing different scans with a scan matching algorithm [LM97] and treating the whole scan as a single landmark.

Although laser scans are much easier to process, visual landmark detection is a most active area of research because computer vision is a cheap multi-purpose sensor suitable for most other tasks including obstacle detection, object recognition, scene analysis and human-robot interaction. Different ways to detect landmarks have been proposed in literature: Kortenkamp and Weymouth [KW94] use vertical lines found at wall edges, doors and cabinets. Basri and Rivlin extract general lines and edges [BR95]. Zobel et al. [ZDH+01] as well as Bellini et al. [BPP02] use ceiling lamps which have the particular advantage not to be subject to occlusion. A more general approach is pursued by Winters et al. [WGLSV00] using eigenimages and Se et al. [SLL02] using scale invariant features, both being able to detect a larger class of landmarks. Fraundorfer employs Gabor wavelets [FDF02] to detect salient features as landmarks.

Summarizing, there are several approaches that can be used for computer vision based natural landmark detection. However, employing a reliable system is far from trivial because performance of all detectors often depends on the actual environment being present.

6.3 Detection of Circular Artificial Landmarks

Throughout the building circular artificial landmarks (Fig. 6.2a) were set in order to simplify conduction of the experiments. The detection algorithm is sketched in figure 6.2b. It is based on Hough transform [Nie89, §3.4.2] and a gray-level variance criterion similar to Otsu automatic thresholding [Ots79].

For a given image pixel it is necessary to determine, whether this pixel is the center of a circular landmark: The key observation is that there is only a single possible circle with that center, since the radius is known and circles are invariant under rotation in the circle's plane. Since the camera is calibrated with respect to robot coordinates and floor plane, the image outline of this circle can be computed (Fig. 6.2b): The optical ray corresponding to the considered pixel is computed in robot coordinates using camera calibration parameters. This ray is intersected with the floor plane. The result defines the center for two circles, one with a radius slightly smaller than the radius of the landmark and one with a radius which is slightly larger. Both circles are mapped back into the image. If, in fact, there is a circular landmark located at the pixel considered, the projection of the smaller circle will be inside the landmarks image and the

projection of the larger circle will be outside the landmarks image. Thus, along the circumference of each circle the gray-level is basically constant, whereas it is different comparing both circles.

For checking, gray-level variance statistic is computed along the circumference of the smaller and larger circle as well as along both circles together. The gray-level variance for both circles is a sum differentiated into parts: The variance along the inner circle, the variance along the outer circle (*intra class variances*) and a term proportional to the squared difference of both means (*inter class variance*). For an ideal image the intra class variance should be zero and the overall variance equal to inter class variance. Thus, according to Otsu [Ots79] inter class variance divided by overall variance is a non dimensional quality measure $\in [0 \dots 1]$, invariant under affine illumination changes.

The projected circles' outlines for each pixel being the hypothetical center of a landmark are precomputed and stored. The radius of inner circle, landmarks and outer circle is 6cm, 10cm and 14cm respectively. Each outline is represented by 16 equally spaced samples to reduce computation time. A pixel is accepted as a landmark if the quality exceeds a pre-defined threshold (60% in the experiments) and is higher than the quality of all neighbor pixels. For each accepted pixel the corresponding location relative to the robot is passed as a landmark to the landmark identification algorithm. Computation time is 8ms for a 768×288 image on a Intel Xeon, 2.67 GHz.

6.4 Landmark Identification

Overview

The task of the landmark identification algorithm is to recognize a detected landmark as a landmark already represented in the map. In other words, the algorithm matches landmark observations with landmarks in the map including the decision to define unmatched landmarks as new. In general, *not identifying* a landmark observation is a harmless error only resulting in a duplicate landmark being introduced in the map. This may possibly lead to some problems when using the map but does not affect the map in general. In contrast *false identification* of a landmark, i.e. confusing it with another landmark, often ruins the map completely because wrong information like two different places being the same is integrated. This severely distorts map and robot pose estimates, often making further landmark identification impossible.

Three types of information can be used by a landmark identification algorithm: Appearance of observed landmarks, relative location of a group of landmarks and bounds on the error in robot pose accumulated after the last observation of a landmark. It is obvious that the effort of landmark identification strongly depends on landmarks, sensors and environment:

In many settings the distance between confusable landmarks is larger than the robot's pose uncertainty accumulated after last observation of the landmarks [GN01, CMNT99]. Then the map only contains a single matching candidate for each landmark observation and identification is easy. This situation is encountered in small maps with sparse landmarks or in case high precision or long distance sensors are used. More efforts are necessary, if there are several possible matching candidates for each individual observation in the map. In this case a whole set of observations is matched simultaneously by iterating through all matching combinations being compatible with the bounds on the accumulated error in the robot's pose. The plausibility of a possible match can be evaluated by summing up the squared distance between matched observation and landmark to which the observation is matched. A much better approach is to use Mahalanobis distance based on the covariance matrix of the considered landmarks delivered by the SLAM algorithm [NT00]. Again, sensor noise, map size and arrangement of landmarks decide, whether observations in a single image can be uniquely matched to the map. If a single image is not unique enough, as an alternative a local map patch around the robot can be matched with the remaining map [GK99]. If even this approach does not suffice, multi hypothesis tracking must be utilized to defer the decision about the identification of landmarks until further observations provide enough evidence. This technique has proven to be a robust solution for localization in an a-priori map [DN01], however, tracking many map hypotheses in real time is not yet possible with currently available SLAM algorithms.

Although multi hypotheses tracking is beyond the scope of this thesis, the proposed algorithm in general is well suited for this approach: Spawning a new hypothesis corresponds to closing a loop, being performed by the algorithm with extreme efficiency. Furthermore, the tree data structure allows *lazy-copying* similar to fastSLAM [MTKW02]: When spawning a new hypothesis only the part of the tree actually modified must be copied while storing a link to the unmodified part.

Algorithm used for the experiments

The landmark identification algorithm used for the experiments employs two different strategies the local identification of landmarks and the detection of loop closure. Local identification is performed by simultaneously matching all observations from a single robot pose to the map. Matching is based on a least square approach taking into account both error in each landmark observation and error in the robot pose. This means, for a potential matching the most likely robot pose is computed and the matching is accepted, if the resulting error both in robot pose and in landmark observations is below a pre-specified threshold (0.3m, 7.5°)[6]. Landmarks that could

[6] 1° if only a single landmark is observed to prevent false identification of that landmark.

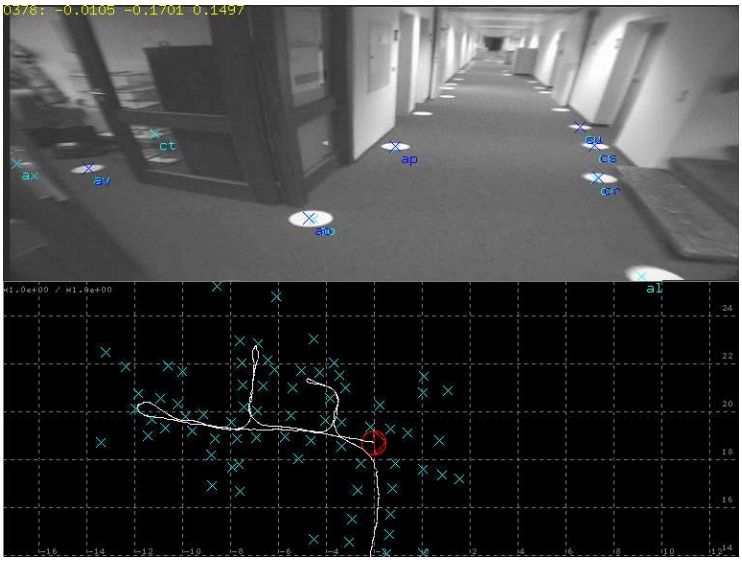


Figure 6.3: Screen shot of the SLAM implementation mapping the DLR building.

not be matched in this approach are introduced as new landmarks into the map. Significantly more robust results could be achieved by allowing errors both in observations and robot pose. In order to avoid duplicate landmarks, a new landmark is not introduced, if the distance to an existing landmark is below 0.5m.

Considerable difficulties were encountered in detecting closure of a loop: Before closing the largest loop the accumulated robot pose error was 16.18m (Fig. 6.4, 6.5) and the average distance between adjacent landmarks was $\approx$ 1m. Furthermore, the landmarks used were indistinguishable. So 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. Since the patch is already part of the map it is necessary to change the identity of the landmarks of the patch when a loop is closed. This is performed by formally changing the identification in all BIBs and updating all nodes from these BIBs up to the root as described in §4.3. Since arrangement of landmarks is comparatively similar throughout the building, even a whole map patch can sometimes be ambiguous. Thus, the map patch is only accepted for closing a loop when 90% of all landmarks of the patch have been successfully matched.

6.5 Large Map Experiment

In the large map experiment an operator navigates the robot through the building mentioned above (Fig. 6.1, 60m $\times$ 45m, 29 rooms, 505.01m traveled, three large loops) and the robot maps

computation time			storage space
local update	global update	global map	
$0.77\mu\text{s} \cdot k^2$	$0.02\mu\text{s} \cdot k^3 \log n$	$0.04\mu\text{s} \cdot kn$	$317\text{B} \cdot kn$
0.07ms	3.87ms	0.41ms	2686KB

Table 6.1: Average computation time, storage space and corresponding average prefactors for asymptotical expressions (Intel Xeon, 2.67 GHz, $k \approx 16.39$, $n = 0 \dots 725$).

the building while moving. The resulting map shows 725 landmarks, 29142 measurements and 3297 robot poses. The purpose of the experiment was to demonstrate that algorithm and overall system work in a real world scenario of considerable size. Figure 6.3 shows a screen shot of the program at work. Figure 6.4 shows the estimated map before closing the largest loop having an error of 16.18m, while figure 6.5 shows the final map estimate. It can be observed that the error in the loop mainly arises in the right upper corner of the map when the robot leaves the building. Odometry is perturbed by a small door step and the vision system is affected by change of illumination between indoors and outdoors. This highlights 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 6.6a shows the internal tree representation used by the algorithm. On the average there are $k \approx 16.39$ landmarks represented in each BIB in contrast to $k \approx 5.55$ in the simulation scenario, because more landmarks were visible from a single robot position than in the simulation. 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 §3.5. This is confirmed by figure 6.6e plotting tree height w.r.t. n.

Storage space requirements are increasing linearly in the number of landmarks reaching 5369KB for all $n = 725$ landmarks (Fig. 6.6b). In contrast EKF storage space is super-linear, progressively exceeding the proposed algorithm for $n \gtrapprox 200$ landmarks. As in the simulation experiments computation time is extremely low (0.07ms per measurement) if, only a local update is performed as is the case most often. The average time for a global update is 3.87ms (Fig. 6.6c) and for computing an additional global estimate 0.41ms. Accumulated computation time is shown in figure 6.6d. It is clearly visible that the proposed algorithm is much faster than EKF.

Table 6.1 lists average prefactors for the asymptotical expressions. The prefactors are considerably smaller than in the simulation experiments (Tab. 5.2) due to k being much larger. For instance, computation time needed for a global update consists of an asymptotically dominant part proportional to $k^3 \log n$ and further parts proportional to $k^2 \log n$, $k \log n$ and $\log n$. As often

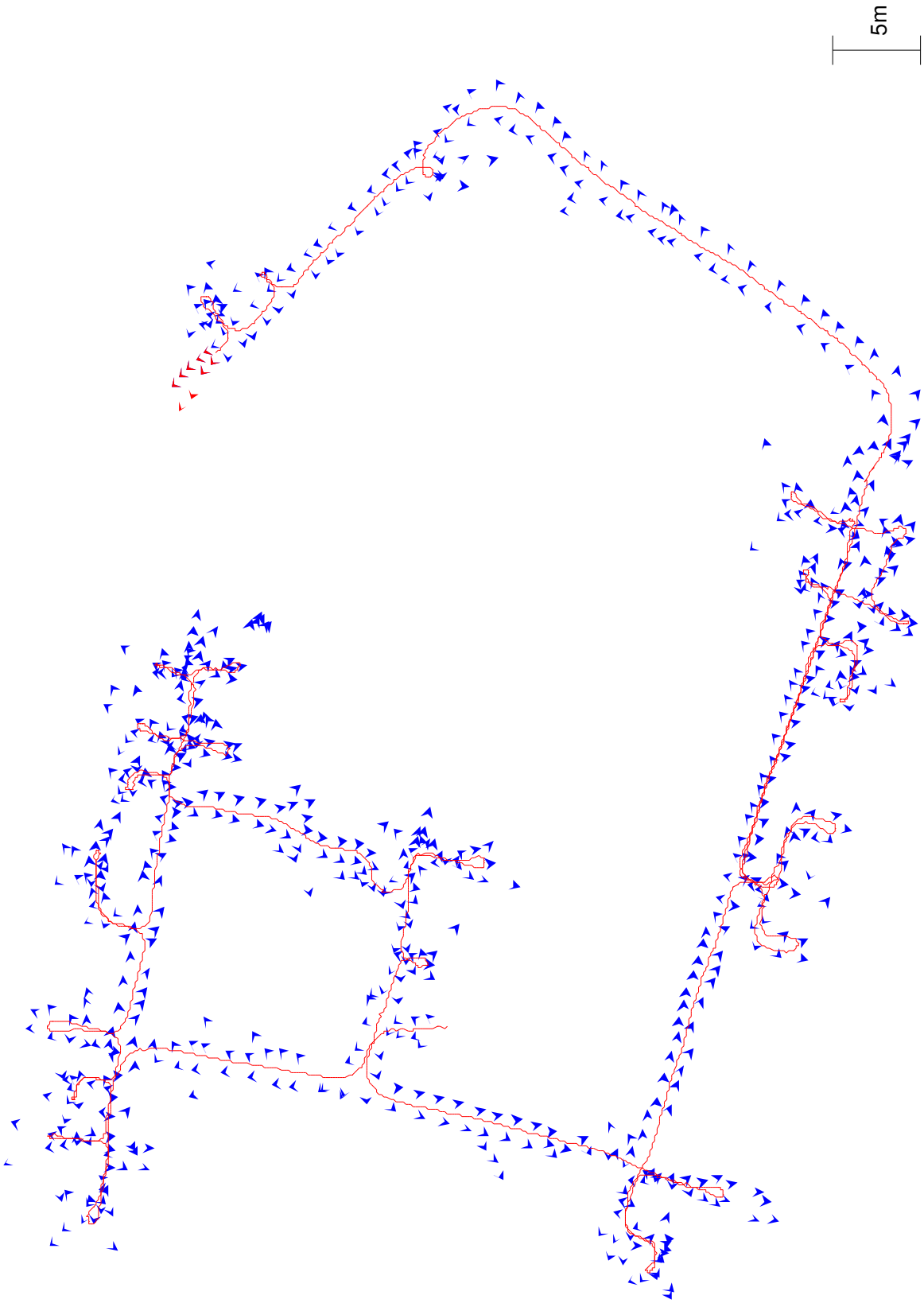


Figure 6.4: Map estimate before closing the large loop having an accumulated error of 16.18m mainly caused by the robot leaving the building in the right upper corner.

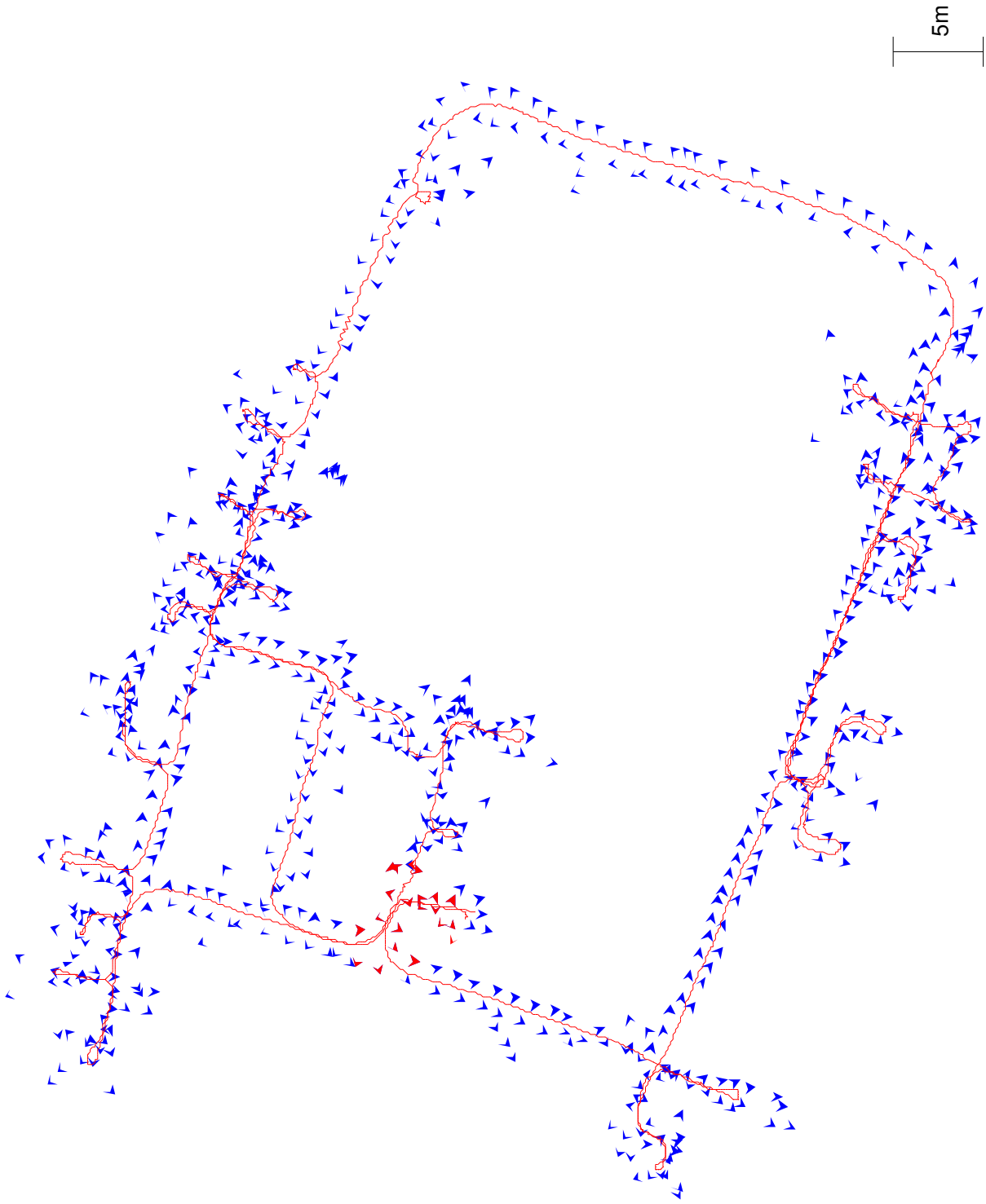
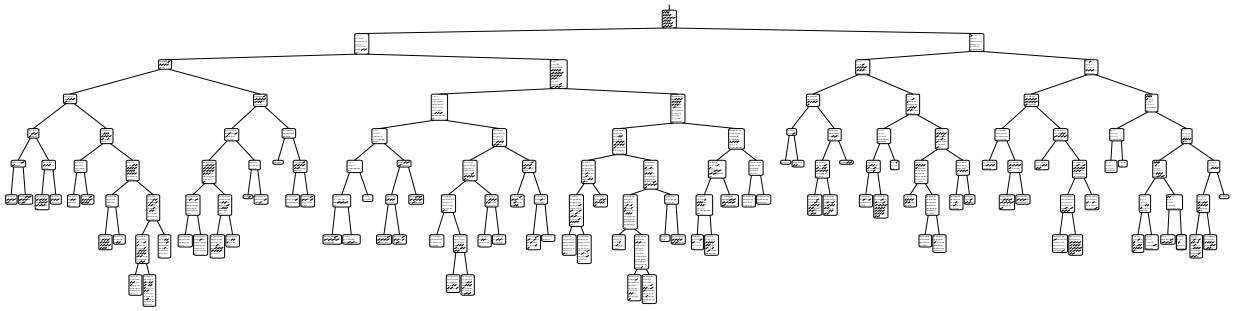
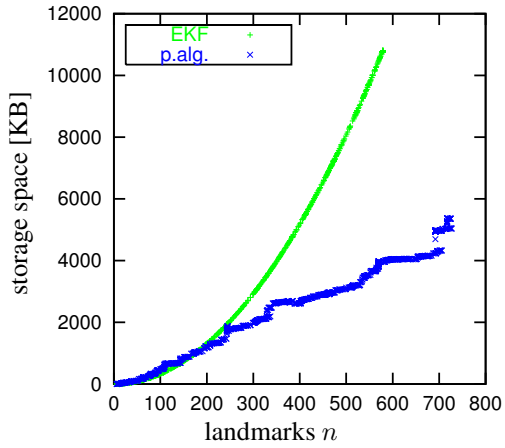


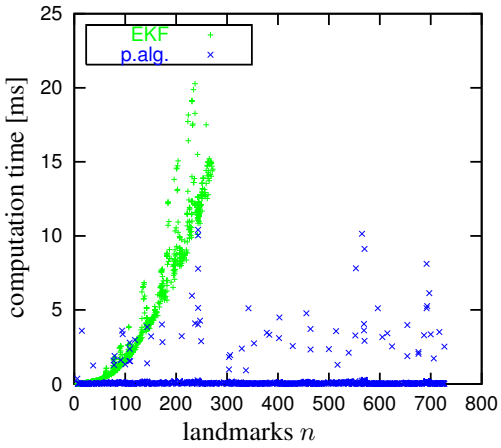
Figure 6.5: Final map estimate.



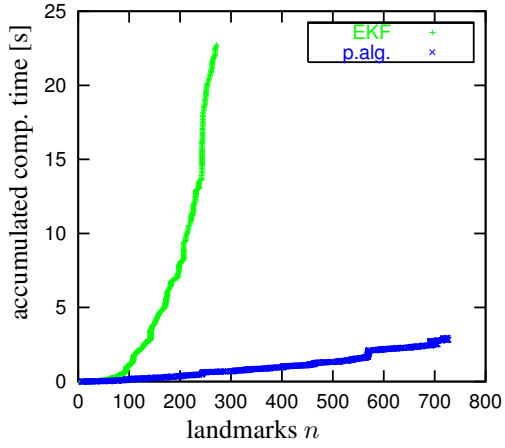
(a) Tree representation of the map. Size of the node ovals is proportional to number of represented landmarks.



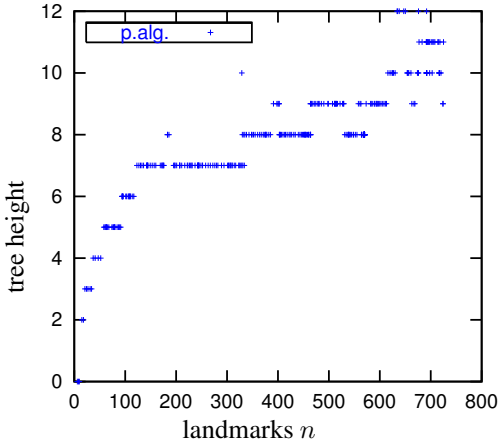
(b) Storage space.



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



(d) Accumulated computation time.



(e) Tree height.

Figure 6.6: Real experiment performance: (a) Internal tree representation, (b) storage space, (c)-(d) computation time, (e) tree height

encountered for elaborate algorithms, the prefactor of $k^3 \log n$ is much smaller[7] than the prefactor of $k^2 \log n$, $k \log n$ or $\log n$. So for k as small as 16.39 or 5.81 the asymptotically dominant term $O(k^3 \log n)$ is not actually responsible for most of the computation time yet. When determining prefactors by dividing computation time by asymptotical expression the result is k dependent for small k and should converge to a constant for $k \to \infty$.

6.6 Statistical Evaluation Experiment

During the statistical evaluation experiment the same environment was mapped independently 17 times by repeatedly moving the robot along the same trajectory to allow statistical evaluation. The trajectory was a large $50\text{m} \times 40\text{m}$ "U" along the two long corridors and the connection outside the building (Fig. 6.1). In order to compute a relative error spectrum (§2.13) like in simulation (§5.2, 5.3) the true location of four landmarks in the corners of the trajectory was measured manually by tape. Figure 6.7 shows the resulting estimates of the proposed algorithm, whereas figure 6.8 shows the corresponding Maximum Likelihood results. Again, it is observable that most error occurred in the right upper corner when passing the buildings entrance. At casual inspection, the estimates of the proposed algorithm appear to be slight though not dramatically worse than the ML estimates.

For all 17 maps figure 6.9a shows the mean absolute error over the four selected landmarks i.e. the distance between estimated and true location. It is plotted both for the proposed algorithm and for the maximum likelihood solution. The error of the proposed algorithm was varying more. On the average it is 12.40m and a bit higher than the ML error of 11.43m (ratio: 108%).

The relative error spectrum is shown in figure 6.9b. The lowest relative error is only 50% and the highest 876%, which is rather inconceivable. Any relative error below 1 means that the ML estimate is not optimal. Certainly it can only be optimal for real world experiments to the extent the statistical model defining likelihood is met in reality. But then the same model is employed in the proposed algorithm, so in any case the lowest relative error should be $\gtrapprox 1$. The highest relative error is 876% and thus more than 2 times larger than the relative error encountered in the small error simulation experiment. Furthermore, the plot does not show the shape encountered in the simulation experiments with low average value and two outliers. Overall, the results are not as expected from the simulation experiments.

The author conjectures that 17 runs are too few for estimating a 8×8 covariance matrix distorting the relative error spectrum. Indeed, when using less than 8 runs the covariance matrices are rank deficient resulting in the lowest relative error being 0 and the highest being ∞. In

[7] Inner loop of matrix multiplication and inversion. All other computations are atmost $O(k^3 + k^2 \log n)$, see §4.6.

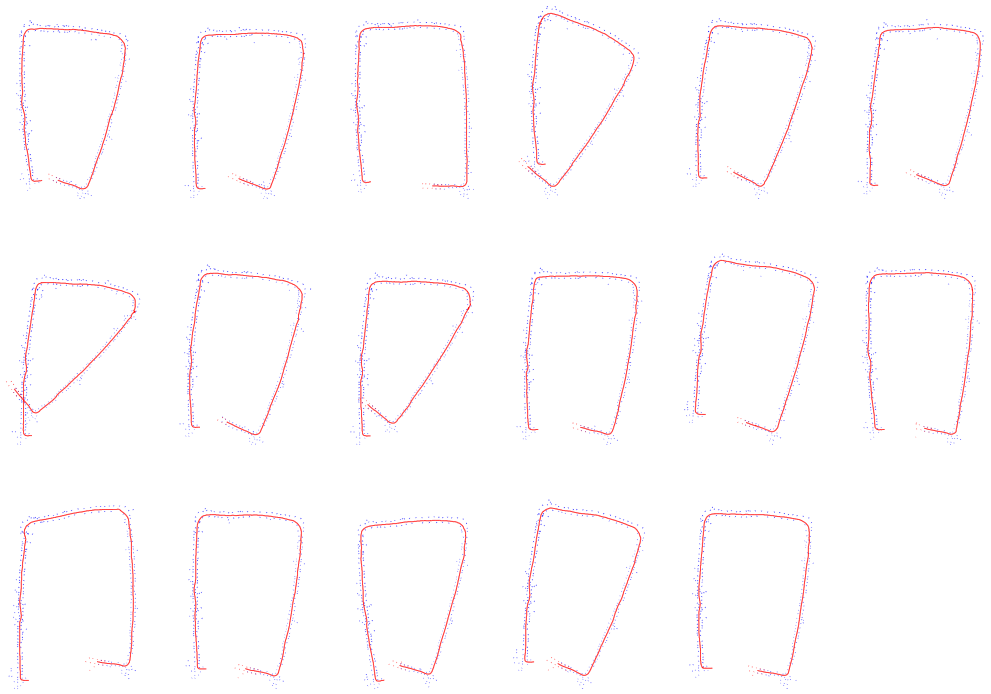


Figure 6.7: Several map estimates performed by the proposed algorithm.

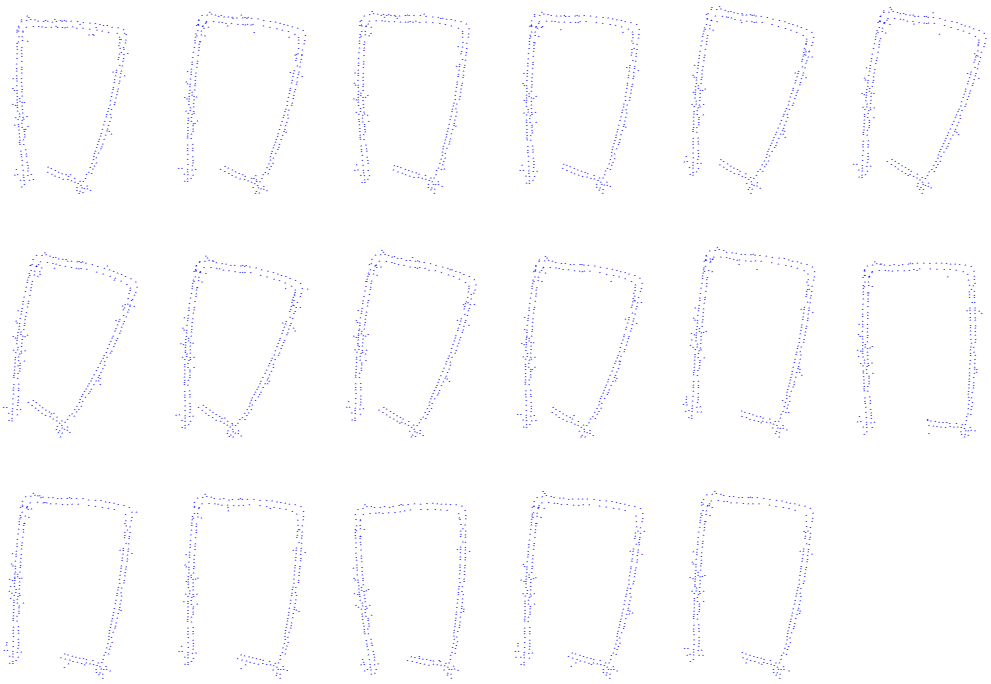
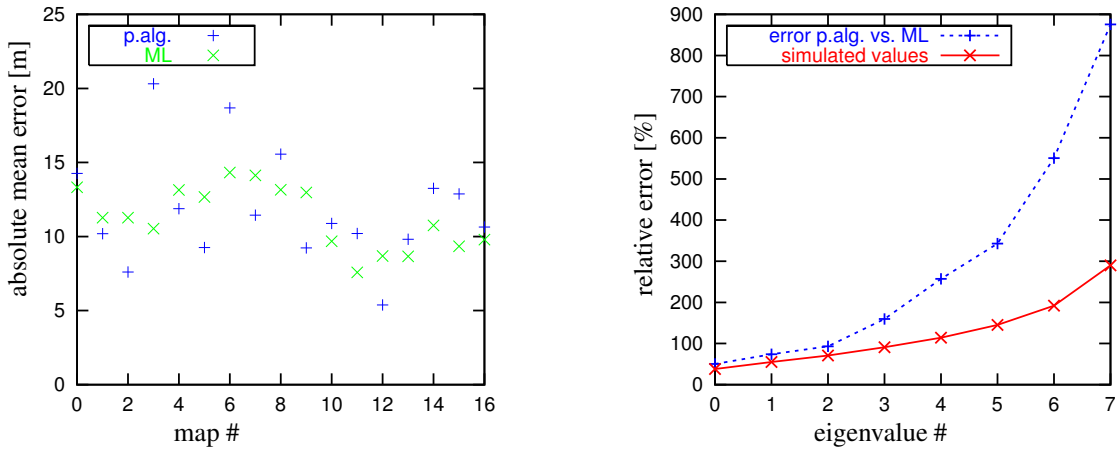


Figure 6.8: Corresponding Maximum Likelihood estimates.



(a) Error of estimates for different maps.

(b) Relative error spectrum (see comment).

Figure 6.9: Error of the proposed algorithm compared to ML estimate.

general, the error made by estimating the covariance matrix from too few samples decreases the lower eigenvalues and increases the higher eigenvalues making the overall spectrum look more extreme. For investigating the plausibility of this explanation, a simulation was performed assuming both algorithms had identical error distributions. The result is also plotted in figure 6.9b, showing that the eigenvalues range from 38.4% to 290.1%, although the correct result (achieved for infinite many samples) is 1. Even with 1000 samples as used in the simulation experiments the eigenspectrum still ranges from 90.2% to 110.9%.

Two observations follow: Apparently, requirement (R1), i.e. relative error spectrum, is a much stricter criterion for evaluating map quality because it reveals critical aspects of the map estimate a simple absolute error criterion does not consider. Secondly, due to the large number of runs needed, verifying requirement (R1) in real world experiments appears to be impossible, highlighting once more the benefit of simulation experiments.

6.7 Navigation Experiment

Despite being far from precise in an absolute sense the last experiment demonstrates that an estimated map can be used for autonomous navigation. Again, part of the building was mapped and passed to an autonomous navigation system using the same landmark detector and employing a multi hypothesis tracking approach to increase robustness. Figure 6.10 shows the map and the trajectory (A-B-C-D-B) autonomously navigated at a length of 106.11m as logged by the navigation system. Figure 6.2 is an image taken from the video reporting the experiment.

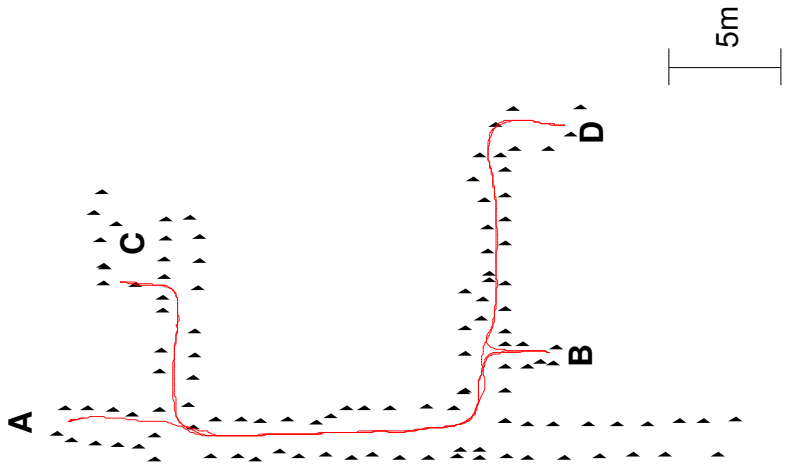


Figure 6.10: Map used for navigation with corresponding trajectory A-B-C-D-B (106.11m). The map has been rotated to be aligned with the floor plan in figure 6.1.

6.8 Discussion

The experiments reported in this chapter clearly show that the proposed algorithm works as required performing in real world environments with high efficiency. In particular, the experiment of mapping the 60m $\times$ 45m DLR building with $n = 725$ landmarks is one of the largest maps reported in literature. Computation time for integrating a measurement was ≤ 13.15ms and thus by far no issue for overall performance.

The main challenge in conducting this experiment was not identified in the estimation algorithm. Least square estimation is a very clearly defined mathematical problem. From the authors perspective there was little doubt that the encouraging results found in simulation experiments would be transferable to a real world scenario. It turned out that landmark identification was the limiting factor for map size instead. A considerable amount of effort was needed to make the landmark identification algorithm close the large loop in the first experiment. For future research it appears worthwhile investigating multi hypothesis tracking to be tolerant to false identification as well as investigating landmarks to be distinguished by appearance.

When comparing the last two chapters, it can be said that simulation and real world experiments complement one another. Simulation experiments allowed to investigate map quality and computation time in experiments larger than practical to be conducted in reality. Performance of the algorithm with respect to the three proposed criteria could be evaluated thoroughly, but only under the assumptions made in the simulation model. In contrast, real world experiments allowed for validation that these assumptions are close enough to reality to be useful and to show "that it is actually working".

Chapter 7

Conclusion

7.1 Summary

This thesis provides two main contributions to the Simultaneous Localization and Mapping problem: A thorough theoretical analysis and an efficient $O(\log n)$ algorithm. The analysis focuses on the structure of uncertainty inherent in a map estimate, being phrased as "*certainty of relations despite uncertainty of positions*". The argument is based both on informal reasoning and on formally proving approximate sparsity of the information matrices occurring in SLAM. Moreover, three requirements for a SLAM algorithm were proposed from an intuitive perspective: *Bounded uncertainty*, *linear storage space*, and *linear update cost*. These conditions affect map quality, storage space, and computation time, respectively, being the most important issues for a SLAM algorithm.

The SLAM algorithm proposed in this thesis 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 $O(k^3 \log n)$[1] time. Practically, it is equally important that a global map can be computed in $O(kn)$ additional time. With the prefactor involved being extremely low (0.15μs) this allows computation of a map with $n = 11300$ landmarks in 12.37ms on a Intel Xeon, 2.67 GHz. Furthermore, the algorithm can handle much larger orientation errors than, for instance, EKF because it can reduce linearization error by applying "nonlinear rotations" to parts of the map. Thus even with errors as large as $140°$, excellent estimates can be provided.

With respect to the three proposed criteria the algorithm was verified theoretically, by simulation experiments, and by experiments with a real robot. These experiments included challenging

[1] n landmarks, k local landmarks

setups: The simulations comprise a map with $n = 11300$ landmarks and a map with 140° orientation error. In real experiments a $60\text{m} \times 45\text{m}$ building with $n = 725$ landmarks was mapped, containing 3 large loops, now being one of the largest maps reported in literature. Overall, the algorithm meets the requirements concerning map quality and storage space, and even needs less than linear time for computing a local map surpassing requirement (R3).

A precondition of achieving these results is a topologically suitable building as explained in §3.5. Indeed, typical buildings are topologically suitable. For the DLR Institute of Robotics and Mechatronics' building mapped in the experiments, in particular, the algorithm had no problems to find a suitable partitioning.

7.2 Outlook

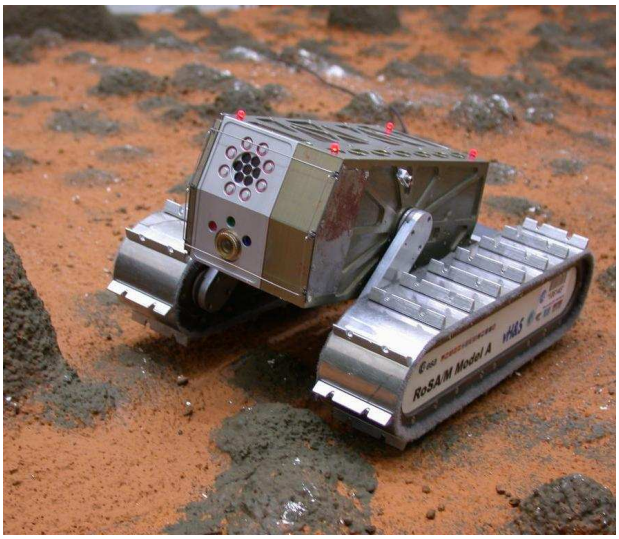
SLAM evolved over a decade of research and has reached a level of maturity by now that it can be used in medium or even large environments. This development triggered an impressive amount of publications on SLAM in all major robotics conferences [FLS+03, LY03] in the past year. Undoubtedly, in the future the focus will shift towards applying SLAM as a component in larger systems and handling the challenges of very large, dynamical, and outdoor environments. This poses new problems, both concerning the core algorithm and possible applications.

Algorithms

For many applications it is more important to compute a global map in $O(n)$ time with a small prefactor than to asymptotically achieve $O(\log n)$ time. This opens up possibilities to simplify the algorithm for easier implementation while preserving its outstanding performance when computing a global map. For instance, most of the algorithms bookkeeping could be replaced by a standard graph-partitioning algorithm like Kernighan-Lin [HL95] running in $O(n)$ instead of $O(\log n)$ time.

Apart from computation time, the most important challenge is landmark identification[2]. This is critical, especially for detecting loop closure. Failure in landmark identification often completely ruins the map, so it would be worthwhile devising an algorithm able to handle uncertain landmark identification. A promising approach is Multi-Hypothesis Tracking, that is allowing for several plausible landmark identifications simultaneously and postponing the final decision until sufficient evidence will be accumulated. By this approach, efficiency of the core algorithm becomes even more crucial as it has to handle all hypotheses simultaneously, multiplying the

[2] more generally called data association



(a)

(b)

Figure 7.1: Potential applications for an efficient SLAM algorithm: (a) ESA Mars rover "Nanokhod", (b) DIROKOL service robot consisting of mobile platform, lightweight robot arm, and dextrous robot hand.

computation time needed.

Visual landmarks pose a hard problem when mapping large environments, especially in outdoor environments. Using cameras and computer vision, artificial landmarks like circular discs or retroreflective stripes can be detected rather easily but setting them is time-consuming. Natural landmarks like vertical lines or corners are also easy to detect but highly ambiguous, so identification is nearly impossible. Thus, developing reliable visual landmark recognition for landmarks of appearance significant enough to facilitate identification is a highly promising research topic.

Applications

A particularly interesting outdoor application for a SLAM algorithm is autonomous exploration of Mars [SABL01] (Fig. 7.1a). With a round-trip time between 9 and 42 minutes depending on celestial position of Earth and Mars, direct tele-operation of a vehicle from Earth is not possible, but SLAM could provide the autonomous behavior required. Since space-qualified computers are much slower and have much less memory than terrestrial ones, efficiency is of utmost importance. The ultimate vision may be airdropping a large number of small mobile robots that collectively explore parts of the planet's surface, interchange gathered information, and finally provide an overall map of the desired area.

It is widely acknowledged that service robotics promises the largest number of useful applications for a mobile robot. Here mapping and navigation in large dynamical environments certainly are essential skills, but the actual job the robot is supposed to do most often requires more than just moving around. Generally, a service robot must open doors and manipulate objects requiring both a robot arm and a robot gripper. To date there are very few commercially available robot arms that are small and light enough to be mounted on a mobile platform. This observation lead Hirzinger et al. to developing several mechatronically integrated light-weight robot arms [HASH+01]. These arms feature joint-torque sensors and Cartesian impedance control [GSASH03] for sensitive and safe interaction with the environment (*compliant motion control*), a prerequisite for operation outside a delimited working cell.

An example for development towards an integrated system is the service robot built in the research project DIROKOL[3] and further improved in the research project MORPHA[4] (Fig. 7.1b). This service robot is composed of a mobile platform (Hanebeck et al. [HSFS00]), a lightweight robot arm (Hirzinger et al. [HASH+01]) and a dextrous robot hand (Hirzinger et al. [BGLH01]), equipped with a visual object-recognition system (Niemann et al. [DMD02]), an a-priori map-based navigation system (Hanebeck et al. [HS98]), and a precursor of the SLAM algorithm proposed in this thesis (Frese et al. [FH01]). Recently, the system was extended to incorporate object manipulation skills like pouring drinks and wiping tables [HBBH04].

In a long-term perspective the greatest challenge will undoubtly be intelligence and behavior. To be a universal domestic aid service robots will need many more abilities than they have now: They must understand the environment they work in, communicate with humans, and plan their actions. The required techniques of 3D-scene analysis and object recognition [HN00, HK98], human-machine interaction [FKL+03, MOR03], and action planning [MNPW98, BHG+02] have been a subject of research for a long time. Once integrated into a comprehensive mechatronic system, they could lead to functionality far beyond current state of the art.

With growing complexity of future service robots, the traditional approach to explicitly program desired behaviors and skills will become more and more difficult. Similar in spirit to SLAM, where an operator shows the environment to the robot and the robot "learns" the map, the future service robot could learn behaviors and skills from human demonstration. This is another area of intense research [Koe01, SA94] again drawing heavily on 3D scene analysis, human-machine interaction and action planning.

Apparently service robotics is at the verge of being marketed. However, the vision of a universal domestic robot helper will still require years of intense research and interdisciplinary cooperation.

[3] "Dienstleistungsroboter in kostengünstiger Leichtbauweise" funded by "Bayerische Forschungsstiftung"

[4] "BMBF Leitprojekt Intelligente Antropomorphe Assistenzsysteme" (Intelligent Anthropomorphic Systems) [MOR03]

Appendix A

Positive Definite Matrices

A.1 Properties

This section list some properties of symmetric positive (semi-) definite matrices that are used throughtout §2 and §3. An extensive discussion can be found in [HJ90]. Let $A = \left(\begin{smallmatrix} P & R^T \\ R & S \end{smallmatrix}\right)$, $A' = \left(\begin{smallmatrix} P' & R'^T \\ R' & S' \end{smallmatrix}\right)$ and A'' be symmetric positive definite matrices, X and X' arbitrary matrices and x and y vectors. Then the following holds:

$$A = A^T, \quad 0 < x^T A x \; \forall \; x \tag{A.1}$$

$$\text{all eigenvalues are} > 0 \tag{A.2}$$

$$\text{all diagonal entries are} > 0 \tag{A.3}$$

$$P, S, A^{-1}, A + A', \lambda A \; \forall \; \lambda > 0 \text{ are SPD} \tag{A.4}$$

$$P - RS^{-1}R^T, S - R^T P^{-1} R \text{ are SPD (Schur - complement)} \tag{A.5}$$

$$X^T A X \text{ is SPSD and SPD if } X \text{ if } \operatorname{kernel}(X) = \{0\} \tag{A.6}$$

$$A = LL^T \text{ for a lower triangular matrix } L \text{ (Cholesky - decomposition)} \tag{A.7}$$

$$(x^T A y)^2 \leq x^T A x \, y^T A y \text{ (Cauchy-Schwarz inequality)} \tag{A.8}$$

$$\tag{A.9}$$

For symmetric positive semidefinite matrices similar properties hold:

$$A = A^T, \quad 0 <= x^T A x \; \forall \; x \tag{A.10}$$

$$\text{all eigenvalues are} \geq 0 \tag{A.11}$$

$$\text{all diagonal entries are} \geq 0 \tag{A.12}$$

$$P, S, A + A', \lambda A \; \forall \; \lambda \geq 0 \text{ are SPSD} \tag{A.13}$$

$$P - RS^{-1}R^T, S - R^TP^{-1}R \text{ are SPSD, if existing} \tag{A.14}$$

$$X^TAX \text{ is SPSD} \tag{A.15}$$

$$A = LL^T \text{ for a lower triangular matrix } L \text{ with } \mathrm{col}(L) = \mathrm{rank}(A) \tag{A.16}$$

$$(x^TAy)^2 \leq x^TAx\, y^TAy \text{ (Cauchy-Schwarz inequality)} \tag{A.17}$$

$$xx^T \text{ is SPSD and has rank } 1 \tag{A.18}$$

Some inequalities on 2-norm of matrices:

$$\|X + Y\| \leq \|X\| + \|Y\| \tag{A.19}$$

$$\|XY\| \leq \|X\|\|Y\| \tag{A.20}$$

$$\|A\| = \max_{|x|=1} x^TAx = \max_{|x|=1} |Ax| = \lambda_{\max}A \tag{A.21}$$

$$\|R\|^2 \leq \|P\| \cdot \|S\| \tag{A.22}$$

$$\|X\|_2 \leq \|X\|_E := \sqrt{\sum_{i,j} X_{ij}^2} \text{ (Frobenius norm)} \tag{A.23}$$

$$\|X\|^2 = \|XX^T\| = \|X^TX\| \tag{A.24}$$

Symmetric matrices A, A' not necessarily positive definite can be compared defining a relation $\leq$:

$$A' \leq A \iff x^TA'x \leq x^TAx\ \forall x \quad \text{(Definition)} \tag{A.25}$$

$$A' < A \iff A' \leq A \text{ and not } A \leq A' \quad \text{(Definition)} \tag{A.26}$$

$$x^TA'x < x^TAx\ \forall x \implies A' < A \text{ (not conversely)} \tag{A.27}$$

$$A' \leq A \iff A' - A \text{ is SPSD} \tag{A.28}$$

$$A' \leq A \wedge A'' \leq A' \implies A'' \leq A \tag{A.29}$$

$$A' \leq A \wedge A \leq A' \implies A = A' \tag{A.30}$$

$$A' \leq A \implies \lambda A' \leq \lambda A\ \forall \lambda \geq 0 \tag{A.31}$$

$$A' \leq A \implies \lambda A \leq \lambda A'\ \forall \lambda \leq 0 \tag{A.32}$$

$$0 \leq A' \leq A \implies 0 \leq A' + A'' \leq A + A'' \tag{A.33}$$

$$A' \leq A \implies A^{-1} \leq A'^{-1} \tag{A.34}$$

$$A' \leq A \implies X^TA'X \leq X^TAX \tag{A.35}$$

$$A' \leq A \implies P' \leq P \wedge S' \leq S \tag{A.36}$$

$$0 \leq A' \leq A \implies \mathrm{image}\, A' \subset \mathrm{image}\, A \tag{A.37}$$

A.2 Block Matrix Formulas

Block Matrix Inversion Formula

The so called block matrix inversion formula [PTVF92, §2.7] gives the inverse of a 2×2 block matrix using expressions involving the matrix blocks.

$$\begin{pmatrix} P & R^T \\ R & S \end{pmatrix}^{-1} = \begin{pmatrix} \left(P - R^T S^{-1} R\right)^{-1} & -\left(P - R^T S^{-1} R\right)^{-1} \left(R^T S^{-1}\right) \\ -\left(S^{-1} R\right) \left(P - R^T S^{-1} R\right)^{-1} & S^-1 + \left(S^{-1} R\right) \left(P - R^T S^{-1} R\right)^{-1} \left(R^T S^{-1}\right) \end{pmatrix} \tag{A.38}$$

$$= \begin{pmatrix} P^{-1} + \left(P^{-1} R^T\right) \left(S - R P^{-1} R^T\right) \left(R P^{-1}\right) & -\left(P^{-1} R^T\right) \left(S - R P^{-1} R^T\right) \\ -\left(S - R P^{-1} R^T\right) \left(R P^{-1}\right) & \left(S - R P^{-1} R^T\right) \end{pmatrix} \tag{A.39}$$

Woodbury Formula

The Woodbury formula [PTVF92, §2.7][Hag89] allows to update the inverse of a matrix after change of a small rank:

$$\left(P + R^T S^{-1} R\right)^{-1} = P^{-1} - \left(P^{-1} R^T\right) \left(S - R P^{-1} R^T\right)^{-1} \left(R P^{-1}\right) \tag{A.40}$$

$$\left(P^{-1} - R^T S^{-1} R\right)^{-1} \left(R^T S^{-1}\right) = \left(P^{-1} R^T\right) \left(S - R P^{-1} R^T\right)^{-1} \tag{A.41}$$

Sherman-Morrison Formula

If S is a 1×1 matrix, i.e. a scalar σ and $R = r^T$ correspondingly a vector, the result is the Sherman-Morrison fomula:

$$\left(P + \sigma r r^T\right)^{-1} = P^{-1} - \left(P^{-1} r\right) \frac{\sigma}{1 - \sigma r^T P^{-1} r} \left(r^T P^{-1}\right) \tag{A.42}$$

Appendix B

Technical Proofs

Lemma 1 (Page 72). *Let S be a block diagonal SPD matrix with block bandwidth w. Then the norm $\|S\|$ is at most $2w-1$ times the norm of any diagonal block ($=\max_i \|S_{ii}\|$).*

Proof. Let $\alpha := \max_i \|S_{ii}\|$. It has to be proven, that $|Sv| \leq (2m-1)\alpha|v|$ for all v:

$$|Sv|^2 = \sum_i |S_i v|^2 = \sum_i \left| \sum_j S_{ij} v_j \right|^2 \leq \sum_i \left(\sum_j |S_{ij} v_j| \right)^2 \tag{B.1}$$

Since S is a m block band matrix, $S_{ij} = 0$ if $|i-j| \geq m$

$$= \sum_i \left(\sum_{j=i-m+1}^{i+m-1} |S_{ij} v_j| \right)^2 \leq \sum_i \left(\sum_{j=i-m+1}^{i+m-1} \|S_{ij}\| \cdot |v_j| \right)^2 \tag{B.2}$$

By (A.23), $\|S_{ij}\|^2 \leq \|S_{ii}\|\|S_{jj}| \leq \alpha^2$ for all i, j:

$$\leq \sum_i \left(\sum_{j=i-m+1}^{i+m-1} \alpha |v_j| \right)^2 = \alpha^2 \sum_i \sum_{j=i-m+1}^{i+m-1} \sum_{k=i-m}^{i+m} |v_j| \cdot |v_k| \tag{B.3}$$

$$\leq \frac{1}{2} \alpha^2 \sum_i \sum_{j=i-m+1}^{i+m-1} \sum_{k=i-m}^{i+m} \left(|v_j|^2 + |v_k|^2 \right) = \alpha^2 (2m-1) \sum_i \sum_{j=i-m+1}^{i+m-1} |v_j|^2 \tag{B.4}$$

$$\leq \alpha^2 (2m-1)^2 \sum_i |v_i|^2 = \alpha^2 (2m-1)^2 |v|^2 \tag{B.5}$$

□

Lemma 2 (Page 72). *For all $\omega \geq 0$ the following inequality holds:*

$$\frac{\sqrt{1+\frac{3}{\omega}}+1}{\sqrt{1+\frac{3}{\omega}}-1} \geq 1+\frac{4}{3}\omega \tag{B.6}$$

Proof. It is easy to verify, that for all a, b

$$\left(\sqrt{a}+\sqrt{b}\right)^2 = a+2\sqrt{ab}+b \geq 3\min\{a,b\}+\max\{a,b\}. \tag{B.7}$$

So it follows, that

$$\frac{\sqrt{1+\frac{3}{\omega}}+1}{\sqrt{1+\frac{3}{\omega}}-1} = \frac{\left(\sqrt{1+\frac{3}{\omega}}+1\right)^2}{\left(\sqrt{1+\frac{3}{\omega}}+1\right)\left(\sqrt{1+\frac{3}{\omega}}-1\right)} = \frac{\left(\sqrt{1+\frac{3}{\omega}}+1\right)^2}{1+\frac{3}{\omega}-1} \tag{B.8}$$

$$= \left(\sqrt{1+\frac{3}{\omega}}+1\right)^2 \frac{\omega}{3} \overset{\text{(B.7)}}{\geq} \left(3+\left(1+\frac{3}{\omega}\right)\right)\frac{\omega}{3} = \left(4+\frac{3}{\omega}\right)\frac{\omega}{3} = \frac{4}{3}\omega+1. \tag{B.9}$$

□

Lemma 6 (Page 75). *Let $0 \leq \gamma < 1$ and $\mathcal{A}, \mathcal{B} \subset \mathbb{N}$ be two sets of natural numbers with a minimal distance d between elements of $\mathcal{A}$ and $\mathcal{B}$: $d \leq |i-j|\ \forall i \in \mathcal{A}, j \in \mathcal{B}$. Then the following inequality holds:*

$$\sum_{i\in\mathcal{A},\, j\in\mathcal{B}} \gamma^{|i-j|} \leq 2\frac{\gamma^d}{1-\gamma}\min\{|\mathcal{A}|, |\mathcal{B}|\} \tag{B.10}$$

Proof. Let without loss of generality be $|\mathcal{A}| \leq |\mathcal{B}|$. Then it follows

$$\sum_{i\in\mathcal{A},\, j\in\mathcal{B}} \gamma^{|i-j|} = \sum_{i\in\mathcal{A}}\sum_{j\in\mathcal{B}} \gamma^{|i-j|} = \sum_{i\in\mathcal{A}}\left(\sum_{j\in\mathcal{B},\, j\leq i} \gamma^{i-j} + \sum_{j\in\mathcal{B},\, j>i} \gamma^{j-i}\right) \tag{B.11}$$

$$\overset{|i-j|\geq d}{=} \sum_{i\in\mathcal{A}}\left(\sum_{j\in\mathcal{B},\, j\leq i-d} \gamma^{i-j} + \sum_{j\in\mathcal{B},\, j\geq i+d} \gamma^{j-i}\right) \tag{B.12}$$

$$\leq \sum_{i\in\mathcal{A}}\left(\sum_{j\leq i-d} \gamma^{i-j} + \sum_{j\geq i+d} \gamma^{j-i}\right) \leq \sum_{i\in\mathcal{A}}\left(\sum_{j=-\infty}^{i-d} \gamma^{i-j} + \sum_{j=i+d}^{\infty} \gamma^{j-i}\right) \tag{B.13}$$

$$= \sum_{i\in\mathcal{A}}\left(\sum_{k=d}^{\infty}\gamma^k + \sum_{k=d}^{\infty}\gamma^k\right) = \sum_{i\in\mathcal{A}}\left(2\frac{\gamma^d}{1-\gamma}\right) = 2\frac{\gamma^d}{1-\gamma}|\mathcal{A}|. \tag{B.14}$$

□

Lemma 9 (Page 102). *Let $A = \left(\begin{smallmatrix} P & R^T \\ R & S \end{smallmatrix}\right)$ be an SPD matrix and $x = \left(\begin{smallmatrix} y \\ z \end{smallmatrix}\right)$ be decomposed accordingly, with z given. Then the minimum over y of $x^T A^{-1} x$ is $z^T S^{-1} z$ at $y = R^T S^{-1} z$ or $x = A\left(\begin{smallmatrix} 0 \\ S^{-1}z \end{smallmatrix}\right)$. The corresponding minimum over z with y given, is $y^T P^{-1} y$ at $z = RP^{-1}y$ or $x = A\left(\begin{smallmatrix} P^{-1}y \\ 0 \end{smallmatrix}\right)$.*

Proof. Consider the information block $\chi^2(x) := x^T A^{-1} x$ with $A^{-1} =: \left(\begin{smallmatrix} P_I & R_I^T \\ R_I & S_I \end{smallmatrix}\right)$. In this context it is just a mathematical expression not actual information on landmarks. Application of Schur complement (lemma 8) yields a decomposition as

$$\begin{gathered} \chi^2(x) = \chi^2_{\text{CIB}}(z) + \chi^2_{\text{SIB}}(Hz + h - y), \\ \text{with } \chi^2_{\text{CIB}}(z) = z^T\big(S_I - R_I P_I^{-1} R_I^T\big) z,\ \chi^2_{\text{SIB}}(w) = w^T P_I w,\ H = -P_I^{-1} R_I^T,\ h = 0. \end{gathered} \tag{B.15}$$

$$x^T A^{-1} x = z^T\big(S_I - R_I P_I^{-1} R_I^T\big) z + \big(-P_I^{-1} R_I^T z - y\big)^T P_I \big(-P_I^{-1} R_I^T z - y\big) \tag{B.16}$$

From block matrix inversion formula (App. A.2) applied to $\left(\begin{smallmatrix} P_I & R_I^T \\ R_I & S_I \end{smallmatrix}\right)$ it can be seen, that

$$S^{-1} = S_I - R_I P_I^{-1} R_I^T \tag{B.17}$$

$$R^T S^{-1} = \big(-S(R_I P_I^{-1})\big)^T S^{-1} = -P_I^{-1} R_I^T \tag{B.18}$$

$$x^T A^{-1} x = z^T S^{-1} z + \big(R^T S^{-1} z - y\big)^T P_I \big(R^T S^{-1} z - y\big). \tag{B.19}$$

The minimum over y of this expression is $z^T S^{-1} z$ at $y = R^T S^{-1} z$ or equivalently at

$$x = \begin{pmatrix} R^T S^{-1} z \\ z \end{pmatrix} = \begin{pmatrix} R^T S^{-1} z \\ S S^{-1} z \end{pmatrix} = \begin{pmatrix} P & R^T \\ R & S \end{pmatrix} \begin{pmatrix} 0 \\ S^{-1} z \end{pmatrix} = A \begin{pmatrix} 0 \\ S^{-1} z \end{pmatrix} \tag{B.20}$$

The corresponding expressions for the minimum over z with y given are derived by exchanging the role of the first and second block ($y \leftrightarrow z,\ P \leftrightarrow S,\ R \leftrightarrow R^T$). □

Lemma 10 (Page 102). *Let $A = \left(\begin{smallmatrix} P & R^T \\ R & S \end{smallmatrix}\right)$ be an SPSD 2×2 block matrix, with P being SPD. Then the unique minimal elimination matrix for block row and column 1 is $B := \left(\begin{smallmatrix} P \\ R \end{smallmatrix}\right) P^{-1} \left(\begin{smallmatrix} P \\ R \end{smallmatrix}\right)^T$. In the case of the first block being one-dimensional, $B = uu^T$ with $u = \frac{1}{\sqrt{P}} \left(\begin{smallmatrix} P \\ R \end{smallmatrix}\right)$.*

Proof. Similar to (B.16), $\chi^2(x) := x^T A x$ is decomposed using lemma 8 as

$$x^T A x = \chi^2 \begin{pmatrix} y \\ z \end{pmatrix} = z^T\big(S - RP^{-1}R^T\big) z + \big(-P^{-1}R^T z - y\big)^T P \big(-P^{-1}R^T z - y\big). \tag{B.21}$$

Terms involving y and z are expressed using x

$$-P^{-1}R^Tz - y = \begin{pmatrix} -I & -P^{-1}R^T \end{pmatrix} x, \quad \begin{pmatrix} 0 & I \end{pmatrix} x = z \tag{B.22}$$

thereby yielding a matrix decomposition for A as

$$\begin{aligned} x^TAx =& x^T \begin{pmatrix} 0 & 0 \\ 0 & S - RP^{-1}R^T \end{pmatrix} x + x^T \begin{pmatrix} -I & -P^{-1}R^T \end{pmatrix}^T P \begin{pmatrix} -I & -P^{-1}R^T \end{pmatrix} x && \text{(B.23)} \\ =& x^T \left(\begin{pmatrix} 0 & 0 \\ 0 & S - RP^{-1}R^T \end{pmatrix} + \begin{pmatrix} I \\ RP^{-1} \end{pmatrix} P \begin{pmatrix} I \\ RP^{-1} \end{pmatrix}^T \right) x && \text{(B.24)} \\ =& x^T \left(\begin{pmatrix} 0 & 0 \\ 0 & S - RP^{-1}R^T \end{pmatrix} + \begin{pmatrix} P \\ R \end{pmatrix} P^{-1} \begin{pmatrix} P \\ R \end{pmatrix}^T \right) x && \text{(B.25)} \\ \implies A =& \begin{pmatrix} P & R^T \\ R & S \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & S - RP^{-1}R^T \end{pmatrix} + \underbrace{\begin{pmatrix} P \\ R \end{pmatrix} P^{-1} \begin{pmatrix} P \\ R \end{pmatrix}^T}_{B}. && \text{(B.26)} \end{aligned}$$

Since $S - RP^{-1}R^T \geq 0$, it follows that $B \leq A$ and B is indeed an elimination matrix for block row / column 1 of A. If P is one-dimensional, $P^{-1} = \frac{1}{\sqrt{P}} \cdot \frac{1}{\sqrt{P}}$ and

$$B = \underbrace{\left(\frac{1}{\sqrt{P}} \begin{pmatrix} P \\ R \end{pmatrix} \right)}_{u} \left(\frac{1}{\sqrt{P}} \begin{pmatrix} P \\ R \end{pmatrix} \right)^T. \tag{B.27}$$

That B is minimal can be seen by the following argument: Another elimination matrix B' must be of the form $B' = \left(\begin{smallmatrix} P & R^T \\ R & S' \end{smallmatrix}\right)$. Applying this lemma to B' yields the same elimination matrix B as applying it to A, since the definition of B in (B.26) is independent from S. So B is not only an elimination matrix for A but also for B' and thus $B \leq B'$.

A minimal elimination matrix is always unique, since if B and B' are two minimal elimination matrices, $B \leq B'$ and $B' \leq B$ implies $B = B'$. □

Lemma 18 (Page 117). *The minimum for the expression* $\left(\begin{smallmatrix} \lambda \\ \lambda^{-1}r \end{smallmatrix}\right)^T \left(\begin{smallmatrix} \psi & r^T \\ r & S \end{smallmatrix}\right)^{-1} \left(\begin{smallmatrix} \lambda \\ \lambda^{-1}r \end{smallmatrix}\right)$ *is* $\lambda = \sqrt[4]{\psi(r^TS^{-1}r)}$.

Proof. First $\left(\begin{smallmatrix} \psi & r^T \\ r & S \end{smallmatrix}\right)^{-1}$ is computed using block matrix inversion formula (A.2):

$$\alpha := r^TS^{-1}r, \quad \beta := (\psi - \alpha)^{-1} \tag{B.28}$$

$$\begin{pmatrix} \lambda \\ \lambda^{-1} r \end{pmatrix}^T \begin{pmatrix} \psi & r^T \\ r & S \end{pmatrix}^{-1} \begin{pmatrix} \lambda \\ \lambda^{-1} r \end{pmatrix} \tag{B.29}$$

$$= \begin{pmatrix} \lambda \\ \lambda^{-1} r \end{pmatrix}^T \begin{pmatrix} \beta & -\beta(r^T S^{-1}) \\ -(S^{-1} r)\beta & S^{-1} + (S^{-1} r)\beta(r^T S^{-1}) \end{pmatrix} \begin{pmatrix} \lambda \\ \lambda^{-1} r \end{pmatrix} \tag{B.30}$$

$$= \lambda^2 \beta - 2(r^T S^{-1} r)\beta + \lambda^{-2} \left(r^T S^{-1} r + (r^T S^{-1} r)\beta(r^T S^{-1} r) \right) \tag{B.31}$$

$$= \lambda^2 \beta - 2\alpha\beta + \lambda^{-2} \left(\alpha + \alpha^2 \beta \right) \tag{B.32}$$

To find the minimum, the derivative with respect to λ must be

$$0 = 2\lambda\beta - 2\lambda^{-3} \left(\alpha + \alpha^2 \beta \right) \tag{B.33}$$

$$\lambda^4 = \frac{\alpha + \alpha^2 \beta}{\beta} = \frac{\alpha}{\beta} + \alpha^2 = \alpha \left(\psi - \alpha \right) + \alpha^2 = \psi\alpha = \psi \, r^T S^{-1} r. \tag{B.34}$$

□

Lemma 19 (Page 127). *Let x^0 and $\hat{x}$ be two vectors of landmark positions and w a corresponding weight vector. Let*

$$e_{\alpha,d}(\hat{x}) := \sum_{i=1}^{k} w_i (x_i'^0 - \hat{x}_i)^T (x_i'^0 - \hat{x}_i), \quad \textit{with } x'^0 := \mathrm{Rot}_\alpha \, x^0 + \mathrm{Trans}_d \tag{B.35}$$

be the weighted squared distance between $\hat{x}$ and x^0 rotated by α and moved by d. Then the α, d combination, that minimizes $e_{\alpha,d}(\hat{x})$ is

$$\bar{\hat{x}} := \frac{\sum_{i=1}^{k} w_i \hat{x}_i}{\sum_{i=1}^{k} w_i}, \quad \bar{x^0} := \frac{\sum_{i=1}^{k} w_i x_i^0}{\sum_{i=1}^{k} w_i}, \quad \alpha = \arctan2 \begin{pmatrix} -\bar{\hat{x}}_x \bar{x^0}_x - \bar{\hat{x}}_y \bar{x^0}_y \\ -\bar{\hat{x}}_y \bar{x^0}_x + \bar{\hat{x}}_x \bar{x^0}_y \end{pmatrix}, \quad d = \bar{\hat{x}} - \mathrm{Rot}_\alpha \, \bar{x^0}. \tag{B.36}$$

Proof.

$$e'(\hat{x}) = \sum_{i=1}^{k} w_i \big((\mathrm{Rot}_\alpha \, x^0 + \mathrm{Trans}_d)_i - \hat{x}_i \big)^T \, \big((\mathrm{Rot}_\alpha \, x^0 + \mathrm{Trans}_d)_i - \hat{x}_i \big) \tag{B.37}$$

$$= \sum_{i=1}^{k} w_i \left(\mathrm{Rot}_\alpha \, x_i^0 + d - \hat{x}_i \right)^T \, \left(\mathrm{Rot}_\alpha \, x_i^0 + d - \hat{x}_i \right) \tag{B.38}$$

$$= \sum_{i=1}^{k} w_i \left({x_i^0}^T x_i^0 + 2(d - \hat{x}_i)^T \, \mathrm{Rot}_\alpha \, x_i^0 + (d - \hat{x}_i)^T (d - \hat{x}_i) \right). \tag{B.39}$$

The minimum for a given α is found by taking the derivative with respect to d:

$$0 \;=\; \frac{\partial e'(\hat{x})}{\partial d} \;=\; 2\sum_{i=1}^{k} w_i \left(\mathrm{Rot}_\alpha\, x_i^0 + (d - \hat{x}_i)\right) \;=\; 2\sum_{i=1}^{k} w_i \left(\mathrm{Rot}_\alpha\, x_i^0 - \hat{x}_i\right) + 2\left(\sum_{i=1}^{k} w_i\right) d \tag{B.40}$$

$$\arg\min_d e'(\hat{x}) = -\frac{\sum_{i=1}^{k} w_i \left(\mathrm{Rot}_\alpha\, x_i^0 - \hat{x}_i\right)}{\sum_{i=1}^{k} w_i} \;=\; \underbrace{\frac{\sum_{i=1}^{k} w_i \hat{x}_i}{\sum_{i=1}^{k} w_i}}_{\bar{\hat{x}}} - \mathrm{Rot}_\alpha \underbrace{\frac{\sum_{i=1}^{k} w_i x_i^0}{\sum_{i=1}^{k} w_i}}_{\bar{x^0}} \tag{B.41}$$

$$\min_d e'(\hat{x}) = \sum_{i=1}^{k} w_i {x_i^0}^T x_i^0 - \left(\sum_{i=1}^{k} w_i\right) \left(\bar{\hat{x}} - \mathrm{Rot}_\alpha\, \bar{x^0}\right)^T \left(\bar{\hat{x}} - \mathrm{Rot}_\alpha\, \bar{x^0}\right). \tag{B.42}$$

The remaining task is to minimize (B.42) with respect to α. Therefore the expression is expanded omitting terms independent from α and resulting in

$$= \;\mathrm{const} + 2\left(\sum_{i=1}^{k} w_i\right) \bar{\hat{x}}^T \mathrm{Rot}_\alpha\, \bar{x^0} \;=\; \mathrm{const} + \mathrm{const} \cdot \left(\bar{\hat{x}}^T \mathrm{Rot}_\alpha\, \bar{x^0}\right). \tag{B.43}$$

Decomposing Rot_α, $\bar{\hat{x}}$ and $\bar{x^0}$ into x- and y- components yields an expression linear in $\cos\alpha$, $\sin\alpha$:

$$\bar{\hat{x}}^T \mathrm{Rot}_\alpha\, \bar{x^0} = \begin{pmatrix} \bar{\hat{x}}_x \\ \bar{\hat{x}}_y \end{pmatrix}^T \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix} \begin{pmatrix} \bar{x^0}_x \\ \bar{x^0}_y \end{pmatrix} \tag{B.44}$$

$$= \cos\alpha \left(\bar{\hat{x}}_x \bar{x^0}_x + \bar{\hat{x}}_y \bar{x^0}_y\right) + \sin\alpha \left(\bar{\hat{x}}_y \bar{x^0}_x - \bar{\hat{x}}_x \bar{x^0}_y\right). \tag{B.45}$$

The expression (B.45) is a scalar product and becomes minimal when both vectors are antiparallel

$$\begin{pmatrix} \cos\alpha \\ \sin\alpha \end{pmatrix} \;\Big\|\; -\begin{pmatrix} \bar{\hat{x}}_x \bar{x^0}_x + \bar{\hat{x}}_y \bar{x^0}_y \\ \bar{\hat{x}}_y \bar{x^0}_x - \bar{\hat{x}}_x \bar{x^0}_y \end{pmatrix} \tag{B.46}$$

$$\Rightarrow \quad \alpha = \mathrm{arctan2} \begin{pmatrix} -\bar{\hat{x}}_x \bar{x^0}_x - \bar{\hat{x}}_y \bar{x^0}_y \\ -\bar{\hat{x}}_y \bar{x^0}_x + \bar{\hat{x}}_x \bar{x^0}_y \end{pmatrix}, \quad d = \bar{\hat{x}} - \mathrm{Rot}_\alpha\, \bar{x^0}. \tag{B.47}$$

□

Appendix C

Implementation

C.1 Implementation of the Linear Algebra Part

Throughout this thesis the algorithm is presented using textual explanation and structure charts to provide a medium level of abstraction. To implement the algorithm some details have to be considered that have not been made explicit up to now:

The matrices and vectors involved in the computation are small ($< 30 \times 30$) and dense, so a dense column major implementation (compatible to `LAPACK`) has been used. Since the EKF equations are numerically difficult, real values are stored as `double`. Most matrices and vectors are annotated by an identification for the random variables they represent (for the landmarks: x, y coordinate; for the robots: x, y, ϕ coordinate). Each row / column corresponds to one random variable. The landmarks are identified by consecutive indices. It is probably advantageous to assign two indices for each landmark: one for the x and one for the y coordinate and similar reserve 3 special indices for the robots x, y, ϕ coordinates. This would allow to assign an index to each individual row / column treating the different coordinates of the same landmark essentially as different random variables. The advantage of such an implementation is, that it is possible to treat matrices with and without robot pose by the same routines. The implementation used for this thesis proceedes differently, annotating each matrix by a list of landmarks, each one corresponding to two columns / rows. This has the disadvantage of generating a lot of special cases when treating matrices with robot poses.

When combining two matrices, usually the resulting matrix has to represent the union of the landmarks / random variables represented by both matrices. Therefore index tables mapping from rows / columns of the original matrices to rows / column of the resulting matrix have to be computed. All this computation can be performed in $O(k)$, if the landmarks / random variables are stored by ascending indices. Then it is possible to operate on the landmarks of the union in an

ascending order by simultaneously stepping through all involved lists. The technique is similar to the merge operation in merge sort [Sed92, §12].

It is advantageous if the matrix / vector implementation is capable of handling 0-dimensional matrices and vectors. For instance during `computeEstimate` applied to the root node, the 0-dimensional estimate $\hat{z}$ taken from the not existing parent node is multiplied by a $n \times 0$ matrix H, yielding an n-dimensional vector $\vec{0}$ as result. If the implementation does not support such matrices, additional special cases have to be considered when implementing the subalgorithms.

For bookkeeping reasons, there exists IBs that represent a certain set of landmarks but do not (yet) contain information about these landmarks ($\chi^2 = 0$). These IBs have an undefined linearization point x^0 with 0 weight vector $w = 0$, which can in turn lead to some landmarks having zero weight and undefined linearization point in derived CIBs. Sometimes the corresponding rotation angle α is undefined and must be set to zero. The implementation of `computeCIBandSIB` must be able to cope with these special cases.

C.2 Implementation of BIB Changing Control

The purpose of this subalgorithm is to find the optimal BIB to integrate a set of landmark measurements (see §4.2 for the definition of the optimization criteria (I) – (V)). It proceeds as follows (Fig. 4.3, `findOrCreateBIB`): First all BIBs sharing a landmark with the actual BIB are computed, then those BIBs are tested against criterion (I) – (III) and then the one which has to be extended by the smallest number of landmarks is chosen:

To find all BIBs representing a landmark, one starts at the landmarks elimination node stored as an array in the treemap and recursively scans each child that represents the landmarks (Fig. C.3, `recursiveCheck`). Checking, whether a node represents a landmark takes $O(k)$. If a node represents the landmark, there is indeed a BIB below that node that represents the landmark. So the number of nodes checked is at most $O(\log n)$ for each BIB that actually represents the landmarks. Finding all these nodes takes $O(k \log n)$ and all nodes for all landmarks $O(k^2 \log n)$.

Then each BIB found is tested against criterion (I) – (III) taking $O(k^2)$ per BIB and $O(k^3)$ in toto. So the overall computation time for finding the best node is $O(k^3 + k^2 \log n)$.

To check criterion (III) the distance between any landmark in the BIB and any measured landmark must be computed, where the location of the measured landmark is derived from the robot pose estimate and the measurement. When the current robot pose estimate (as reported by the EKF) is used, an up to date estimate for the BIB must be computed, because two different estimates can in general be inconsistent. This requires updating the tree from the root down to the BIB under consideration and is thus much too expensive. The solution is to use the last estimate

Figure C.1: **`isTooLarge`** $(\hat{x}, z)$
$\hat{x}$ last estimate from a considered BIB, z measurements, *global:* $\hat{x}_{\text{EKF}}$ EKF estimate

Compute transform T from $\hat{x}_{\text{EKF}}$ to $\hat{x}$ based on the landmarks in $\mathcal{L}(\hat{x}) \cap \mathcal{L}(\hat{x}_{\text{EKF}})$		
z' := Transform z to global coordinates using robot pose from $\hat{x}_{\text{EKF}}$ and T		
FOR All landmarks $l_1 \in \mathcal{L}(z)$		
	FOR All landmarks $l_2 \in \mathcal{L}(\hat{x})$	
		Compute distance d from l_1 in z' to l_2 in $\hat{x}$
	IF	$d > maxDistance$
	THEN	return true
return false		

Figure C.2: **`allBIBsRepresenting`** $(\mathcal{M})$
$\mathcal{M}$ set of landmarks, the BIBs representing which are to be found

bib := ()		
FOR All landmarks $l \in \mathcal{M}$		
	n := **eN** $[l]$	
	IF	**n** is no leaf
	THEN	**bib** := **bib** $\cup$ `recursiveCheck` $(\mathbf{n}_{\swarrow}, l)$
		bib := **bib** $\cup$ `recursiveCheck` $(\mathbf{n}_{\searrow}, l)$
	ELSE	**bib** := **bib** $\cup \{\mathbf{n}\}$
return **bib**		

computed for that BIB. To derive a robot pose estimate consistent with that estimate, a translation and rotation is applied to the robot pose estimate from the EKF, that aligns the landmark estimate from the EKF with the landmark estimate from the BIB (Fig. C.1, `isTooLarge`).

C.3 Insertion

When inserting a new BIB, the algorithm tries to find an optimal insertion point. This is done by moving down from the root and deciding at each node **n** whether to insert the BIB here, or to go to $\mathbf{n}_{\swarrow}$ or $\mathbf{n}_{\searrow}$. The decision is based on the cost function $cf(\mathbf{n})$ *after* insertion of the BIB. By definition $cf(\mathbf{n}) = |\mathcal{L}(\mathbf{n}_{\swarrow})| + |\mathcal{L}(\mathbf{n}_{\searrow})|$. So the task is to compute the number of landmarks that would be represented in $\mathbf{n}_{\swarrow}$ and $\mathbf{n}_{\searrow}$, when the new BIB was inserted at the different positions ($\swarrow, \searrow, \uparrow$). It is much too inefficient to really insert the BIB and perform all updates necessary just to decide whether to go to $\mathbf{n}_{\swarrow}$ or $\mathbf{n}_{\searrow}$, so the number of landmarks must be computed directly.

The approach taken is to compute the information, which landmark is represented in BIBs below $\mathbf{n}_{\swarrow}$ ($\mathcal{A}_0$), below $\mathbf{n}_{\searrow}$ ($\mathcal{A}_1$) or outside the subtree of **n** ($\mathcal{B}$). (See figure C.4a). This informa-

Figure C.3: **recursiveCheck** $(\mathbf{n}, l)$
l landmark, the BIBs containing which are to be found, **n** node below which to search

IF	**n** is no leaf	
THEN	IF	$\mathbf{n}_{\swarrow}$ represents l
	THEN	**bibL** := `recursiveCheck` $(\mathbf{n}_{\swarrow}, l)$
	ELSE	**bibL** := ()
	IF	$\mathbf{n}_{\searrow}$ represents l
	THEN	**bibR** := `recursiveCheck` $(\mathbf{n}_{\searrow}, l)$
	ELSE	**bibR** := ()
	return **bibL** $\cup$ **bibR**	
ELSE	return $\{\mathbf{n}\}$	

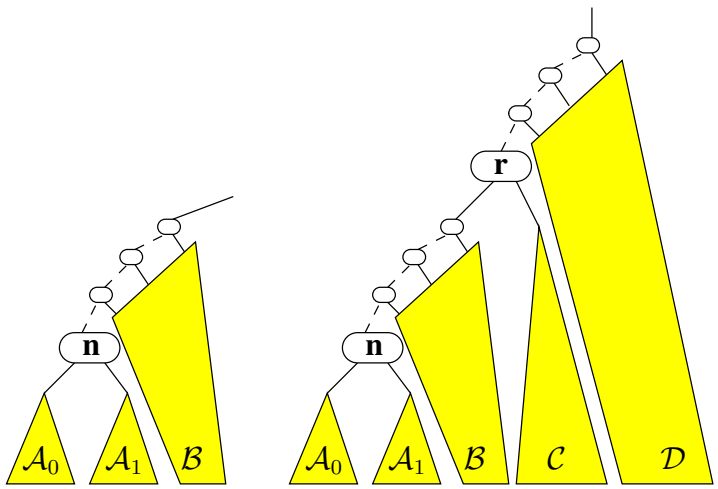
tion can be updated when **n** moves down. Additionally the set of landmarks $\mathcal{M}$ of the BIB to be inserted is needed. From these sets, the effect of inserting the BIB in the different positions can be computed as shown in the following table:

	$\mathcal{L}(\mathbf{n}_{\swarrow})$	$\mathcal{L}(\mathbf{n}_{\searrow})$
before insertion	$\mathcal{A}_0 \cap (\mathcal{A}_1 \cup \mathcal{B})$	$\mathcal{A}_1 \cap (\mathcal{A}_0 \cup \mathcal{B})$
insert BIB $\swarrow$	$(\mathcal{A}_0 \cup \mathcal{M}) \cap (\mathcal{A}_1 \cup \mathcal{B})$	$\mathcal{A}_1 \cap (\mathcal{A}_0 \cup \mathcal{M} \cup \mathcal{B})$
insert BIB $\searrow$	$\mathcal{A}_0 \cap (\mathcal{A}_1 \cup \mathcal{M} \cup \mathcal{B})$	$(\mathcal{A}_1 \cup \mathcal{M}) \cap (\mathcal{A}_0 \cup \mathcal{B})$
insert BIB $\uparrow$	$\mathcal{A}_0 \cap (\mathcal{A}_1 \cup \mathcal{M} \cup \mathcal{B})$	$\mathcal{A}_1 \cap (\mathcal{A}_0 \cup \mathcal{M} \cup \mathcal{B})$

Some sets are quite large, since the union $\mathcal{A}_0 \cup \mathcal{A}_1 \cup \mathcal{B}$ contains all landmarks. So it is not possible to compute the complete sets efficiently. However, it can be seen from the formulas that only landmarks are relevant that are contained in at least two of the sets $\mathcal{A}_0, \mathcal{A}_1, \mathcal{B}, \mathcal{M}$, thus all landmarks contained only in one set can be omitted completely.

Computing and maintaining such reduced sets can be done efficiently but is by no way a straightforward task. The same information is also needed in two circumstances for the hierarchical tree partitioning algorithm. Thus it is expressed as a subalgorithm (section C.5) clearly separating the task of maintaining the information from using it. Computing the sets for a new node **n** takes $O(k^2 \log n)$ (**initLandmarkState**), whereas updating the sets when **n** moves to a child of **n** can be done in $O(k)$ (**updateLandmarkState**).

Apart from this, the algorithm needs the size (=number of BIBs below) of all nodes readily available to evaluate the balancing constraint (III). It is stored in the nodes ($\mathbf{n}_{size}$) and updated whenever the tree changes. The subalgorithms for inserting an empty BIB is shown as a structure chart in (Fig. C.6, `createEmptyBIB`) and (Fig. C.5, `findBestInsertionPoint`).



(a) parts relative to **n** (b) parts relative to **n**, **r**

set	landmarks of BIBs
$\mathcal{A}_0$	...below $\mathbf{n}_{\swarrow}$
$\mathcal{A}_1$	...below $\mathbf{n}_{\searrow}$
$\mathcal{B}$	...below $\mathbf{r}_{\downarrow \mathbf{n}}$ and not below **n**
$\mathcal{C}$	...below $\mathbf{r}_{\downarrow \overline{\mathbf{n}}}$
$\mathcal{D}$	...not below **r**

(c) Definitions

Figure C.4: a) A fixed node **n** divides the tree into three parts $\mathcal{A}_0, \mathcal{A}_1, \mathcal{B}$. Knowledge, which landmark is represented in which part, allows to compute the effect of inserting a BIB / subtree on **n**.
b) Two fixed nodes **n**, **r** divide the tree into five parts $\mathcal{A}_0, \mathcal{A}_1, \mathcal{B}, \mathcal{C}, \mathcal{D}$. Knowledge of the represented landmarks allows to compute the effect on **r** of moving **n** to the other side of **r**.
c) Definition of the different parts.

Figure C.5: `findBestInsertionPoint` $\big((\mathbf{n}, \mathcal{A}_0, \mathcal{A}_1, \mathcal{B}, \mathcal{M}),\ size\big)$

WHILE **n** is no leaf			
	$\gamma_{\swarrow} :=\ \lvert(\mathcal{A}_0 \cup \mathcal{M}) \cap (\mathcal{A}_1 \cup \mathcal{B})\rvert + \lvert\mathcal{A}_1 \cap (\mathcal{A}_0 \cup \mathcal{M} \cup \mathcal{B})\rvert$		
	$\gamma_{\searrow} :=\ \lvert\mathcal{A}_0 \cap (\mathcal{A}_1 \cup \mathcal{M} \cup \mathcal{B})\rvert + \lvert(\mathcal{A}_1 \cup \mathcal{M}) \cap (\mathcal{A}_0 \cup \mathcal{B})\rvert$		
	IF	$size \geq \frac{1}{2}\mathbf{n}_{size}$	
	THEN	$\gamma_{\uparrow} := \lvert\mathcal{A}_0 \cap (\mathcal{A}_1 \cup \mathcal{M} \cup \mathcal{B})\rvert + \lvert\mathcal{A}_1 \cap (\mathcal{A}_0 \cup \mathcal{M} \cup \mathcal{B})\rvert$	
	ELSE	$\gamma_{\uparrow} := \infty$	
	IF	$\gamma_{\uparrow} \leq \gamma_{\searrow} \ \wedge\ \gamma_{\uparrow} \leq \gamma_{\swarrow}$	
	THEN	return **n**	
	ELSE	IF	$\gamma_{\swarrow} \leq \gamma_{\searrow}$
		THEN	$(\mathbf{n}, \mathcal{A}_0, \mathcal{A}_1, \mathcal{B}, \mathcal{M}) :=$ `updateLandmarkState` $(\mathbf{n}, \mathcal{A}_0, \mathcal{A}_1, \mathcal{B}, \mathcal{M}, \swarrow)$
		ELSE	$(\mathbf{n}, \mathcal{A}_0, \mathcal{A}_1, \mathcal{B}, \mathcal{M}) :=$ `updateLandmarkState` $(\mathbf{n}, \mathcal{A}_0, \mathcal{A}_1, \mathcal{B}, \mathcal{M}, \searrow)$
return **n**			

Figure C.6: **createEmptyBIB** $(\mathcal{M})$

$(\mathbf{n}, \mathcal{A}_0, \mathcal{A}_1, \mathcal{B}, \mathcal{M}) :=$ `initLandmarkState` $(\mathbf{root}, (), \mathcal{M})$
$\mathbf{a} :=$ `findBestInsertionPoint` $((\mathbf{n}, \mathcal{A}_0, \mathcal{A}_1, \mathcal{B}, \mathcal{M}), 1)$
$\mathbf{b} :=$ new empty leaf
$\mathbf{ap} :=$ new node with **a** and **b** as children
Make **ap** child of $\mathbf{a}_\uparrow$ replacing **a**
Update $\mathbf{n}_{size}$ for **n** from **b** up to the root
Mark **b** and all ancestors *to be optimized*

C.4 Determination of a Transfer Step

This section explains how to efficiently determine a suitable transfer step based on the criteria discussed in §4.4. The subalgorithm employs the three step strategy described there:

1. Find the node to be optimized
2. Find the subtree to be transfered
3. Find the insertion point for the subtree.

The algorithmic approach is similar to the one used in the preceding section:

Step 1: Find node to be optimized

As first step, a node **r** to be optimized is determined. The policy which nodes are marked *to be optimized* (**setBIB**, **transferSubtree**, **createEmptyBIB**) has the property that whenever a node is marked, all its ancestors are marked too. So a maximally low node to be optimized can be found by a greedy approach: Start at the root and move downward in the tree; as long as a child of the node under consideration is marked, proceed to the child. An important heuristic is employed, when the node under consideration is ancestor of the actual BIB and both children are marked. Then the algorithm chooses the child, that is not ancestor of the actual BIB. The reason therefor is, that the part of the tree containing the actual BIB can be expected to change soon, so time used to optimize this part is probably wasted (Fig. C.7, `findNodeToBeOptimized`).

Step 2: Find subtree to be transfered

Step 2 is to find the optimal subtree below **r** to move from $\mathbf{r}_\swarrow$ to $\mathbf{r}_\searrow$ or vice versa. This is performed by recursively scanning through all possible candidates ($O(k \log n)$) for the subtree root **n**. For each **n** the effect of moving the subtree of **n** to the other child of **r** is evaluated.

Figure C.7: **`findNodeToBeOptimized`** ()

r := **root**; **p** := path from **root** to **actBIB**				
IF	**r** is marked *to be optimized*			
THEN	WHILE $\mathbf{r}_{\swarrow}$ or $\mathbf{r}_{\searrow}$ are marked *to be optimized*			
	Remove first element from **p**			
	IF	$\mathbf{r}_{\swarrow}$ and $\mathbf{r}_{\searrow}$ are marked *to be optimized*		
	THEN	IF	$\mathbf{r}_{\swarrow} = \mathbf{p}\,[0]$	
		THEN	$\mathbf{r} := \mathbf{r}_{\searrow}$	
		ELSE	$\mathbf{r} := \mathbf{r}_{\swarrow}$	
	ELSE	IF	$\mathbf{r}_{\swarrow}$ is marked *to be optimized*	
		THEN	$\mathbf{r} := \mathbf{r}_{\swarrow}$	
		ELSE	$\mathbf{r} := \mathbf{r}_{\searrow}$	
	return **r**			
ELSE	return ()			

To make the evaluation efficient, a similar approach as in the previous section is taken. (figure C.4b) The algorithm computes five sets of landmarks: Landmarks represented in BIBs below $\mathbf{n}_{\swarrow}$ ($\mathcal{A}_0$), below $\mathbf{n}_{\searrow}$ ($\mathcal{A}_1$), not below **n** but below $\mathbf{r}_{\downarrow\mathbf{n}}$ ($\mathcal{B}$), below $\mathbf{r}_{\downarrow\overline{\mathbf{n}}}$ ($\mathcal{C}$) and not below **r** ($\mathcal{D}$). The situation is more complicated, since the position relative to both nodes **n** and **r** is important not only relative to **n** as in the previous section. With these sets the landmarks represented in $\mathbf{r}_{\swarrow}$ and $\mathbf{r}_{\searrow}$ and the cost function can be expressed as

$$\mathcal{L}(\mathbf{r}_{\downarrow\mathbf{n}}) = \big(\mathcal{A}_0 \cup \mathcal{A}_1 \cup \mathcal{B}\big) \cap \big(\mathcal{C} \cup \mathcal{D}\big); \quad \mathcal{L}(\mathbf{r}_{\downarrow\overline{\mathbf{n}}}) = \mathcal{C} \cap \big(\mathcal{A}_0 \cup \mathcal{A}_1 \cup \mathcal{B} \cup \mathcal{D}\big) \tag{C.1}$$

$$cf(\mathbf{r}) = \Big|\big(\mathcal{A}_0 \cup \mathcal{A}_1 \cup \mathcal{B}\big) \cap \big(\mathcal{C} \cup \mathcal{D}\big)\Big| + \Big|\mathcal{C} \cap \big(\mathcal{A}_0 \cup \mathcal{A}_1 \cup \mathcal{B} \cup \mathcal{D}\big)\Big|. \tag{C.2}$$

After **n** has been transfered from below $\mathbf{r}_{\downarrow\mathbf{n}}$ to below $\mathbf{r}_{\downarrow\overline{\mathbf{n}}}$ the situation is

$$\mathcal{L}'(\mathbf{r}_{\downarrow\mathbf{n}}) = \mathcal{B} \cap \big(\mathcal{A}_0 \cup \mathcal{A}_1 \cup \mathcal{C} \cup \mathcal{D}\big); \quad \mathcal{L}'(\mathbf{r}_{\downarrow\overline{\mathbf{n}}}) = \big(\mathcal{A}_0 \cup \mathcal{A}_1 \cup \mathcal{C}\big) \cap \big(\mathcal{B} \cup \mathcal{D}\big) \tag{C.3}$$

$$cf'(\mathbf{r}) = \Big|\mathcal{B} \cap \big(\mathcal{A}_0 \cup \mathcal{A}_1 \cup \mathcal{C} \cup \mathcal{D}\big)\Big| + \Big|\big(\mathcal{A}_0 \cup \mathcal{A}_1 \cup \mathcal{C}\big) \cap \big(\mathcal{B} \cup \mathcal{D}\big)\Big|. \tag{C.4}$$

As in the previous section, only landmarks are relevant that are contained at least in two of the sets $\mathcal{A}_0, \mathcal{A}_1, \mathcal{B}, \mathcal{C}, \mathcal{D}$, so all other landmarks can be omitted when computing the sets.

The value of (C.4) is compared for the different feasible nodes **n** and the lowest one is chosen. Feasible means, that transferring the subtree below **n** will not violate the balancing constraint (III). The allowed sizes for the subtree to be transfered are computed depending on the balancing condition $\mathrm{bal}(\mathbf{r})$ and $\mathrm{Bal}(\mathbf{r})$ and stored in $[s_{low} \ldots s_{high}]$.

As discussed in section 4.4 the search for the optimal subtree can be limited to $O(k \log n)$ candidates. This limitation is realized by using the following two bounding conditions to terminate recursion:

1. If $\mathbf{n}_{size}$ is smaller than s_{low}, recursive search can be terminated, since all lower nodes have even smaller sizes and are thus infeasible. As discussed in section 4.4, if $\mathbf{r}$ is not balanced, the choice of $[s_{low} \ldots s_{high}]$ guarantees, there are at most $O(\log n)$ subtrees larger than s_{low}. So this bounding condition ensures that no more than $O(\log n)$ nodes have to be processed.

2. The following expression is a lower bound for (C.4) that holds for all nodes below $\mathbf{n}$, since it assumes $\mathcal{A}_0 = \mathcal{A}_1 = \emptyset$:

$$cf'(\mathbf{r}) \geq \Big|\mathcal{B} \cap \big(\mathcal{C} \cup \mathcal{D}\big)\Big| + \Big|\mathcal{C} \cap \big(\mathcal{B} \cup \mathcal{D}\big)\Big| \tag{C.5}$$

   If this value is greater or equal than the current best cost function value, no node below $\mathbf{n}$ will be a better choice and recursive search can be terminated. Especially if $\mathcal{L}(\mathbf{n}) \cap \mathcal{L}(\mathbf{r}_{\downarrow\mathbf{n}}) = \emptyset$ it follows that:

$$(\mathcal{A}_0 \cup \mathcal{A}_1) \cap (\mathcal{C} \cup \mathcal{D}) = \emptyset \tag{C.6}$$

$$\mathcal{A}_0 \cap \mathcal{C} = \mathcal{A}_1 \cap \mathcal{C} = \mathcal{A}_0 \cap \mathcal{D} = \mathcal{A}_1 \cap \mathcal{D} = \emptyset. \tag{C.7}$$

   So the bound (C.5) equals (C.2) and transferring any subtree below $\mathbf{n}$ will not improve $cf(\mathbf{r})$. If $\mathbf{r}$ is balanced, (C.2) corresponds to the feasible solution of doing nothing, so recursive search can be terminated. As discussed in section 4.4 there are only $O(k \log n)$ nodes, that share a represented landmark with $\mathbf{r}$, so by using (C.5) as bound no more than $O(k \log n)$ nodes have to be processed.

The recursive subalgorithm is shown as a structure chart in (Fig. C.8, `findBestTransfer`) and (Fig. C.9, `recursiveBest`).

Step 3: Find insertion point for subtree

The third step consists of finding the optimal point below $\mathbf{r}_{\downarrow\overline{\mathbf{n}}}$ to insert the subtree $\mathbf{n}$. This can be achieved by the same subalgorithm used to insert a new BIB (**findBestInsertionPoint**). The subalgorithm is required to find a insertion point below $\mathbf{r}_{\downarrow\overline{\mathbf{n}}}$ and it uses $\mathbf{n}_{size}$ instead of 1 as size of the node to be inserted when evaluating the balancing condition. The subalgorithm uses the sets $\mathcal{A}'_0, \mathcal{A}'_1, \mathcal{B}'$ to evaluate optimization criterion (II) for the different possible insertion points. These sets must be computed as they were, if $\mathbf{n}$ had already been removed from below $\mathbf{r}_{\downarrow\mathbf{n}}$:

$$\mathcal{A}'_0 = \mathcal{B}; \quad \mathcal{A}'_1 = \mathcal{C}; \quad \mathcal{B}' = \mathcal{D} \qquad \text{if } \mathbf{r}_{\swarrow} \text{ is ancestor of } \mathbf{n} \tag{C.8}$$

$$\mathcal{A}'_0 = \mathcal{C}; \quad \mathcal{A}'_1 = \mathcal{B}; \quad \mathcal{B}' = \mathcal{D} \qquad \text{else} \tag{C.9}$$

After the best insertion point has been determined, the subtree is actually transfered as described in section 4.5. The *to be optimized* flag on node $\mathbf{r}$ is not removed, since there may be another

Figure C.8: **findBestTransfer** $(\mathbf{r})$

$(\mathbf{n}, \mathcal{A}_0, \mathcal{A}_1, \mathcal{B}, \mathcal{C}, \mathcal{D}) :=$ initLandmarkState $(\mathbf{r}, \mathbf{r}, \emptyset)$	
IF	$\frac{1}{3}\mathbf{r}_{size} \leq \mathbf{r}_{\swarrow size} \leq \frac{2}{3}\mathbf{r}_{size}$
THEN	$s_{low} := \lceil \mathbf{r}_{\swarrow size} - \frac{2}{3}\mathbf{r}_{size} \rceil$; $s_{high} := \lfloor \mathbf{r}_{\swarrow size} - \frac{1}{3}\mathbf{r}_{size} \rfloor$
	best := (); $bestValue := \lvert\mathcal{L}(\mathbf{r}_{\swarrow})\rvert + \lvert\mathcal{L}(\mathbf{r}_{\searrow})\rvert$
ELSE	$s_{low} := \lceil \mathbf{r}_{\swarrow size} - \frac{3}{5}\mathbf{r}_{size} \rceil$; $s_{high} := \lfloor \mathbf{r}_{\swarrow size} - \frac{2}{5}\mathbf{r}_{size} \rfloor$
	$bestValue := \infty$
IF	$s_{high} > 0$
THEN	recursiveBest $((\mathbf{n}, \mathcal{A}_0, \mathcal{A}_1, \mathcal{B}, \mathcal{C}, \mathcal{D}), \swarrow, [s_{low} \dots s_{high}], \textbf{best}, bestValue)$
IF	$s_{low} < 0$
THEN	recursiveBest $((\mathbf{n}, \mathcal{A}_0, \mathcal{A}_1, \mathcal{B}, \mathcal{C}, \mathcal{D}), \searrow, [-s_{high} \cdots - s_{low}], \textbf{best}, bestValue)$
return **best**	

Figure C.9: **recursiveBest** $\big((\mathbf{n}, \mathcal{A}_0, \mathcal{A}_1, \mathcal{B}, \mathcal{C}, \mathcal{D}), child, [s_{low} \dots s_{high}], \textbf{best}^a, bVal^a\big)$

$(\mathbf{n}, \mathcal{A}_0, \mathcal{A}_1, \mathcal{B}, \mathcal{C}, \mathcal{D}) :=$ updateLandmarkState $(\mathbf{n}, \mathcal{A}_0, \mathcal{A}_1, \mathcal{B}, \mathcal{C}, \mathcal{D}, child)$	
IF	$\mathbf{n}_{size} < s_{low}$
THEN	return
$lowerlimit := \big\lvert\mathcal{B} \cap (\mathcal{C} \cup \mathcal{D})\big\rvert + \big\lvert\mathcal{C} \cap (\mathcal{B} \cup \mathcal{D})\big\rvert$	
IF	$lowerlimit \geq bVal$
THEN	return
$val := \big\lvert\mathcal{B} \cap (\mathcal{A}_0 \cup \mathcal{A}_1 \cup \mathcal{C} \cup \mathcal{D})\big\rvert + \big\lvert(\mathcal{A}_0 \cup \mathcal{A}_1 \cup \mathcal{C}) \cap (\mathcal{B} \cup \mathcal{D})\big\rvert$	
IF	$\mathbf{n}_{size} \in [s_{low} \dots s_{high}] \wedge val < bVal$
THEN	**best** := $(\mathbf{n}, \mathcal{A}_0, \mathcal{A}_1, \mathcal{B}, \mathcal{C}, \mathcal{D})$; $bVal := val$
IF	**n** is no leaf
THEN	recursiveBest $((\mathbf{n}, \mathcal{A}_0, \mathcal{A}_1, \mathcal{B}, \mathcal{C}, \mathcal{D}), \swarrow, [s_{low} \dots s_{high}], \textbf{best}, bVal)$
	recursiveBest $((\mathbf{n}, \mathcal{A}_0, \mathcal{A}_1, \mathcal{B}, \mathcal{C}, \mathcal{D}), \searrow, [s_{low} \dots s_{high}], \textbf{best}, bVal)$

[a] **best** and $bVal$ are passed by reference. The best result is returned in these parameter.

Figure C.10: **optimizeHTP** ()

r := `findNodeToBeOptimized` ()	
IF	**r** = ()
THEN	return
(**s**, $\mathcal{A}_0, \mathcal{A}_1, \mathcal{B}, \mathcal{C}, \mathcal{D}$) := `findBestTransfer` (**r**)	
IF	**s** = ()
THEN	Mark **r** as not *to be optimized*
	return
IF	**s** is descendant of $\mathbf{r}_{\swarrow}$
THEN	(**n**, $\mathcal{A}_0$, $\mathcal{A}_1$, $\mathcal{B}$, $\mathcal{M}$) := `updateLandmarkState` ($\mathbf{r}, \mathcal{B}, \mathcal{C}, \mathcal{D}, \mathcal{A}_0 \cup \mathcal{A}_1, \swarrow$)
ELSE	(**n**, $\mathcal{A}_0$, $\mathcal{A}_1$, $\mathcal{B}$, $\mathcal{M}$) := `updateLandmarkState` ($\mathbf{r}, \mathcal{C}, \mathcal{B}, \mathcal{D}, \mathcal{A}_0 \cup \mathcal{A}_1, \searrow$)
a := `findBestInsertionPoint` ($(\mathbf{n}, \mathcal{A}_0, \mathcal{A}_1, \mathcal{B}, \mathcal{M}), \mathbf{n}_{size}$)	
`transferSubtree` (,**a**)	

transfer step further improving $cf(\mathbf{r})$. Only if the search for the best subtree to be transfered reveals, that it is best to do nothing, the flag on **r** is removed. (Fig. C.10, `optimizeHTP`)

C.5 Tracking the State of Landmarks

The subalgorithms for hierarchical tree partitioning described in the previous sections all have to evaluate the effect of inserting a BIB or transferring a subtree on the cost function of a node. Based on this criterion they choose the best subtree or insertion point. To perform the evaluation they need information about each landmark, where it is represented relative to the node (**n**) or nodes (**r**, **n**) under consideration (figure C.4). The subalgorithm for landmark tracking described in this section efficiently provides this information (**initLandmarkState**). It further updates the information when **n** is replaced by $\mathbf{n}_{\swarrow}$ or $\mathbf{n}_{\searrow}$ (**updateLandmarkState**).

Specifically the subalgorithm provides five sets of landmarks $\mathcal{A}_0, \mathcal{A}_1, \mathcal{B}, \mathcal{C}, \mathcal{D}$, that indicate, which landmarks are represented in BIBs at certain positions relative to **n** and **r**. The union of all five contains every landmark represented in the whole map, so it is clearly impossible to efficiently compute the whole sets. Fortunately for the hierarchical tree partitioning subalgorithms only those landmarks are relevant, that are contained in at least two of those sets. So the landmark tracking subalgorithm omits all landmarks completely that are only contained in one of the sets. An external set of landmarks $\mathcal{M}$ which are never omitted can be specified.

The set $\mathcal{M}$ is used when inserting a new BIB. In this case it is necessary to always know the state of landmarks represented in the new BIB. Thus $\mathcal{M}$ is defined as the set of landmarks

represented in the BIB. When searching for the optimal transfer step for a subtree, $\mathcal{M}$ is empty.

Initial computation

For the initial computation it can be assumed that $\mathbf{r}$ is either not considered $\mathbf{r} = ()$ (**createEmptyBIB**) or equal to $\mathbf{n}$ (**findBestTransfer**). The case $\mathbf{r} = ()$ will be assumed now, so only the relation of the different landmarks to $\mathbf{n}$ has to be evaluated and $\mathcal{C} = \mathcal{D} = \emptyset$:

By definition the landmarks represented by a node are those represented in some BIB below that node and in some BIB not below that node. So it can be assigned:

$$\mathcal{A}_0 := \mathcal{L}(\mathbf{n}_{\swarrow}); \quad \mathcal{A}_1 := \mathcal{L}(\mathbf{n}_{\searrow}); \quad \mathcal{B} := \mathcal{L}(\mathbf{n}) \tag{C.10}$$

A landmark from $\mathcal{L}(\mathbf{n}_{\swarrow})$ is represented by some BIB below $\mathbf{n}_{\swarrow}$ and thus must be element of $\mathcal{A}_0$. However, not all landmarks with this property are contained in $\mathcal{L}(\mathbf{n}_{\swarrow})$. Landmarks that are represented in BIBs below $\mathbf{n}_{\swarrow}$ and only in those BIBs are not represented by $\mathbf{n}_{\swarrow}$ since there elimination node is below $\mathbf{n}_{\swarrow}$. As discussed before, these landmarks can be safely omitted as they would only be contained in one of the five sets. The same holds for $\mathcal{A}_1$ compared to $\mathcal{L}(\mathbf{n}_{\searrow})$ and $\mathcal{B}$ compared to $\mathcal{L}(\mathbf{n})$.

An exception are the landmarks specified in $\mathcal{M}$. They are not allowed to be omitted, even if only contained in one of the five sets. So for each landmark $l \in \mathcal{M}$ it is explicitly determined in which of the sets it must be contained. This is done by looking at the landmarks elimination node $\mathbf{eN}[l]$ and its relation to $\mathbf{n}$. Four cases arise:

1. $\boldsymbol{eN}[l] = \boldsymbol{()}$:
   Landmark l is represented nowhere.

2. $\boldsymbol{eN}[l] = \boldsymbol{n}$:
   Landmark l is represented both below $\mathbf{n}_{\swarrow}$ and $\mathbf{n}_{\searrow}$ and already correctly treated by (C.10).

3. $\boldsymbol{eN}[l]$ *is a descendant of* $\boldsymbol{n}_{\swarrow}$ *(*$\boldsymbol{n}_{\searrow}$ *resp.)*:
   Landmark l is represented below and only below $\mathbf{n}_{\swarrow}$ ($\mathbf{n}_{\searrow}$ resp.) and must be added to $\mathcal{A}_0$ ($\mathcal{A}_1$ resp.). The path from $\mathbf{n}$ down to the elimination node is stored (as $\mathbf{p}[l]$) since it is needed when the sets must be updated after $\mathbf{n}$ has changed. (**updateLandmarkState**).

4. $\boldsymbol{eN}[l]$ *is ancestor or unrelated to* $\boldsymbol{n}$:
   There exists a BIB representing landmark l not below $\mathbf{n}$, so l must be added to $\mathcal{B}$. It might be, that l is also represented somewhere below $\mathbf{n}_{\swarrow}$ or $\mathbf{n}_{\searrow}$. In such a case it would be contained in two different sets and thus already be correctly treated by (C.10), so it is not necessary to determine, whether this is the case or not.

Figure C.11: **initLandmarkState** $(\mathbf{n}, \mathbf{r}^{a}, \mathcal{M})$

$\mathcal{D} := \mathcal{C} := \emptyset$; $\mathbf{p} := ()$				
IF	**n** is no leaf			
THEN	$\mathcal{A}_0 := \mathcal{L}(\mathbf{n}_{\swarrow})$; $\mathcal{A}_1 := \mathcal{L}(\mathbf{n}_{\searrow})$; $\mathcal{B} := \mathcal{L}(\mathbf{n})$			
ELSE	$\mathcal{A}_0 := \mathcal{A}_1 := \mathcal{L}(\mathbf{n}_{\text{BIB}})$; $\mathcal{B} := \mathcal{L}(\mathbf{n})$			
FOR All landmarks $l \in \mathcal{M}$ with **eN** $[l] \neq ()$				
	IF	**eN** $[l]$ is descendant of **n**		
	THEN	**p** $[l]$:= path from **n** to **eN** $[l]$		
		Remove first element from **p** $[l]$		
		IF	**p** $[l]$ is not empty	
		THEN	IF	**p** $[l]$ $[0] = \mathbf{n}_{\swarrow}$
			THEN	$\mathcal{A}_0 := \mathcal{A}_0 \cup \{l\}$
			ELSE	$\mathcal{A}_1 := \mathcal{A}_1 \cup \{l\}$
	ELSE	$\mathcal{B} := \mathcal{B} \cup \{l\}$		
IF	$\mathbf{r} = \mathbf{n}$			
THEN	$\mathcal{D} := \mathcal{B}$; $\mathcal{B} := \emptyset$			
return $(\mathbf{n}, \mathbf{r}, \mathcal{A}_0, \mathcal{A}_1, \mathcal{B}, \mathcal{C}, \mathcal{D}, \mathcal{M}, \mathbf{p})$				

[a] Node **r** must be either $= \mathbf{n}$ or $= ()$.

If $\mathbf{r} = \mathbf{n}$ instead of $()$, the only difference is, that $\mathcal{D}$ replaces $\mathcal{B}$. In this case $\mathcal{B}$ and $\mathcal{C}$ are not well defined, since **n** is not on one side of **r**, so it is advantageous to define both to be $\emptyset$ and the three nonempty sets as $\mathcal{A}_0$, $\mathcal{A}_1$ and $\mathcal{D}$.

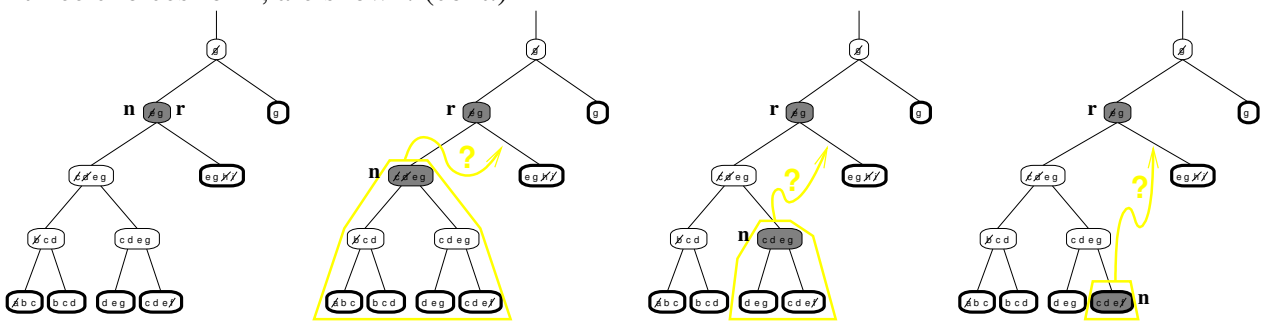
The whole operation takes $O(k + |\mathcal{M}| \log n)$. Stepping through $\mathcal{L}(\mathbf{n}), \mathcal{L}(\mathbf{n}_{\swarrow}), \mathcal{L}(\mathbf{n}_{\searrow})$ takes $O(k)$, since all sets have $O(k)$ elements. The additional computation for $\mathcal{M}$ takes $O(\log n)$ per element with the dominant part being the computation of a path from **eN** $[l]$ to **n**. In the overall algorithm the subalgorithm is used with $|\mathcal{M}| = O(k)$, so the computation time is $O(k \log n)$.

The subalgorithm is shown as a structure chart in (Fig. C.11, `initLandmarkState`).

Update

The hierarchical tree partitioning algorithm scans through several nodes evaluating optimization criterion (II) and choosing the one with the best result. To do that, the five sets $(\mathcal{A}_0, \mathcal{A}_1, \mathcal{B}, \mathcal{C}, \mathcal{D})$ have to be computed efficiently for each node. It processes $O(k \log n)$ different nodes, so performing the computation described above for each node would take $O(k^2 \log^2 n)$ overall which is too long. Fortunately the algorithm does not process arbitrary nodes, but recursively scans part of the tree. So it first processes a node and than (possibly) the nodes children. This provides the opportunity to use an updating subalgorithm. It takes the five sets computed for $(\mathbf{r}, \mathbf{n})$ as input

Figure C.12: Example for tracking the state of landmarks: The algorithm tries to improve **r** by moving a subtree from the left side of **r** to the right side. To find the best subtree, it scans the nodes below $\mathbf{r}_{\swarrow}$ recursively evaluating all possible subtree roots **n**. In this example only the first three choices for **n**, are shown. (cont.)



---

and returns the corresponding sets for $(\mathbf{r}, \mathbf{n}_{child})$, where *child* is $\swarrow$ or $\searrow$. The subalgorithm works quite similar to the previous one:

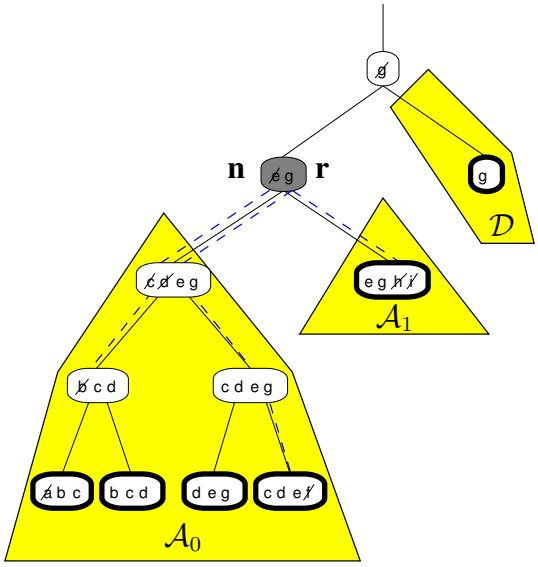
In general, when **n** is replaced by $\mathbf{n}' := \mathbf{n}_{\swarrow}$ ($\mathbf{n}_{\searrow}$ resp.), $\mathcal{A}_1$ ($\mathcal{A}_0$ resp.) must be added to $\mathcal{B}$ and $\mathcal{A}_0$ and $\mathcal{A}_1$ recomputed as $\mathcal{L}(\mathbf{n}'_{\swarrow})$ and $\mathcal{L}(\mathbf{n}'_{\searrow})$. A special case is the first update, when $\mathbf{n} = \mathbf{r}$. Then $\mathcal{A}_1$ ($\mathcal{A}_0$ resp.) is added to $\mathcal{C}$ instead $\mathcal{B}$. Similar to the situation before, all landmarks, that are contained in at least 2 of the sets are already correct and all other landmarks can be omitted, if they are not $\in \mathcal{M}$. So as before, the next step is for all landmarks $l \in \mathcal{M}$ to look at the path from $\mathbf{n}'$ to the elimination node $\mathbf{eN}[l]$.

The path for landmark l has been saved as $\mathbf{p}[l]$, so it is not necessary to compute it. (Which would be too time consuming) If the path is already removed or $\mathbf{n}'$ does not lie on the path anymore, the landmark does not affect the situation for $\mathbf{n}'$ or any of its descendant, so the whole path is removed. If $\mathbf{n}'$ lies on the path, the path is shortened by one node. If it is empty now, $\mathbf{n}'$ is the elimination node $\mathbf{eN}[l]$, so l is already contained in $\mathcal{A}_0, \mathcal{A}_1$ and nothing has to be done anymore. Otherwise the next step in the path is either $\mathbf{n}'_{\swarrow}$ or $\mathbf{n}'_{\searrow}$. Depending on which, landmark l is either represented below $\mathbf{n}'_{\swarrow}$ or $\mathbf{n}'_{\searrow}$ resp. and added to $\mathcal{A}_0$ or $\mathcal{A}_1$ resp.

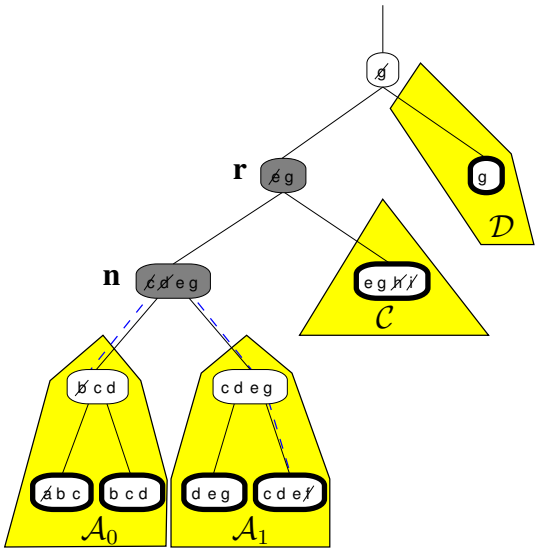
Finally landmarks from $\mathcal{B}$ that are not contained in $\mathcal{A}_0, \mathcal{A}_1, \mathcal{C}, \mathcal{D}$ or $\mathcal{M}$ are removed. This is necessary, since otherwise $|\mathcal{B}|$ may grow to $O(k \log n)$ making processing too slow. Since all involved sets are $O(k)$ and the operations performed are $O(1)$ for each landmark, the overall computation time is $O(k)$. This is very important for the performance of **findBestTransfer** that processes $O(k \log n)$ nodes, thus taking only $O(k^2 \log n)$ time.

Figure C.12 shows an example of the landmark state operations involved in finding the best tree to be transfered. The subalgorithm is shown as a structure chart in (Fig. C.13,

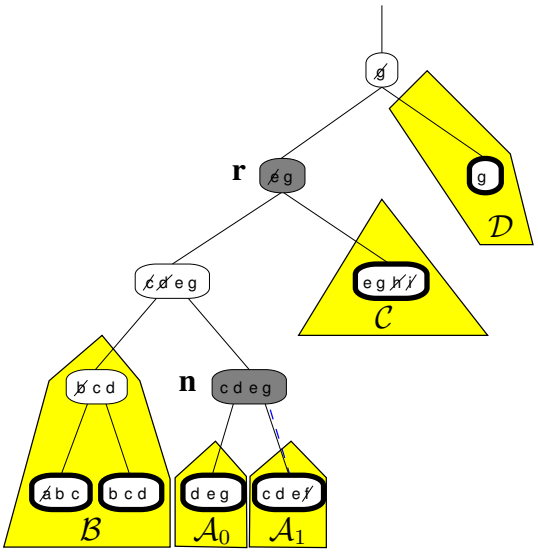
Figure C.12 cont.: Parts of the tree, corresponding sets of landmarks and cost function $cf(\mathbf{r})$: The state is initialized with $\mathbf{n} = \mathbf{r} = \mathbf{root}_{\swarrow}$, $\mathcal{M} = \{\mathsf{b}, \mathsf{f}, \mathsf{g}, \mathsf{h}\}$. Landmarks $\mathsf{a}, \mathsf{c}, \mathsf{d}, \mathsf{i}$ are omitted, since they are only contained in one part. Normally when searching for a subtree to be transfered, $\mathcal{M} = \emptyset$ and landmarks $\mathsf{b}, \mathsf{f}, \mathsf{h}$ would have been omitted too. In this example $\mathcal{M} = \{\mathsf{b}, \mathsf{f}, \mathsf{g}, \mathsf{h}\}$. So to add the members of $\mathcal{M}$ to the appropriate parts, the paths from $\mathbf{n}$ to their elimination nodes are computed. For all elimination nodes below $\mathbf{n}$ the path is saved (dashed edges).
The original tree (a) and (b) both violate the balancing constraint, so (c) is chosen as best.



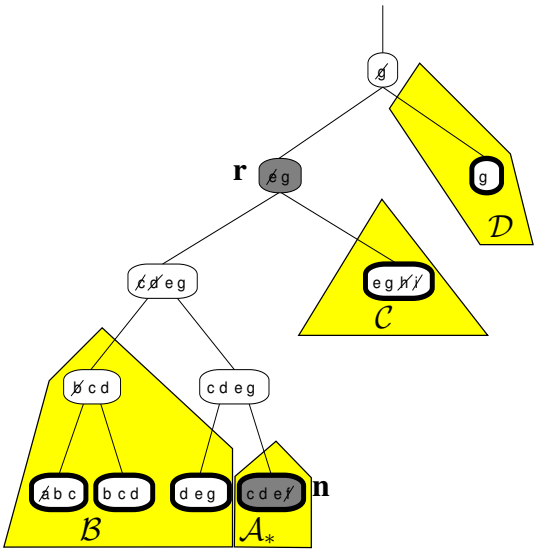
(a) Initialization $\mathbf{n} = \mathbf{r} = \mathbf{root}_{\swarrow}$, $\mathcal{M} = \{\mathsf{b}, \mathsf{f}, \mathsf{g}, \mathsf{h}\}$: $\mathcal{A}_0 = \{\mathsf{b}, \mathsf{e}, \mathsf{f}, \mathsf{g}\}$, $\mathcal{A}_1 = \{\mathsf{e}, \mathsf{g}, \mathsf{h}\}$, $\mathcal{D} = \{\mathsf{g}\}$. $cf(\mathbf{r}) = 4$



(b) update $\mathbf{n} = \mathbf{n}_{\swarrow}$: $\mathcal{A}_1$ becomes $\mathcal{C}$. Path to **eN** [h] is not needed anymore: $\mathcal{A}_0 = \{\mathsf{b}, \mathsf{c}, \mathsf{d}\}$, $\mathcal{A}_1 = \{\mathsf{c}, \mathsf{d}, \mathsf{e}, \mathsf{f}, \mathsf{g}\}$, $\mathcal{C} = \{\mathsf{e}, \mathsf{g}, \mathsf{h}\}$, $cf'(\mathbf{r}) = 1$.



(c) update $\mathbf{n} = \mathbf{n}_{\searrow}$: $\mathcal{A}_0$ becomes $\mathcal{B}$. Path to **eN** [b] is not needed any more: $\mathcal{A}_0 = \{\mathsf{d}, \mathsf{e}, \mathsf{g}\}$, $\mathcal{A}_1 = \{\mathsf{c}, \mathsf{d}, \mathsf{e}, \mathsf{f}\}$, $\mathcal{B} = \{\mathsf{b}, \mathsf{c}, \mathsf{d}\}$, $cf'(\mathbf{r}) = 5$



(d) update $\mathbf{n} = \mathbf{n}_{\searrow}$: $\mathcal{A}_0$ is added to $\mathcal{B}$. $\mathcal{A}_* = \mathcal{A}_0 = \mathcal{A}_1 = \{\mathsf{c}, \mathsf{d}, \mathsf{e}, \mathsf{f}\}$, $\mathcal{B} = \{\mathsf{b}, \mathsf{c}, \mathsf{d}, \mathsf{e}, \mathsf{g}\}$, $cf'(\mathbf{r}) = 8$

Figure C.13: **`updateLandmarkState`** $\big((\mathbf{n}, \mathbf{r}, \mathcal{A}_0, \mathcal{A}_1, \mathcal{B}, \mathcal{C}, \mathcal{D}, \mathcal{M}, \mathbf{p}), \mathit{child}\big)$

IF	$\mathbf{n} = \mathbf{r}$					
THEN	IF	$\mathit{child} = \swarrow$				
	THEN	$\mathbf{n} := \mathbf{n}_{\swarrow}$; $\mathcal{C} := \mathcal{A}_1$				
	ELSE	$\mathbf{n} := \mathbf{n}_{\searrow}$; $\mathcal{C} := \mathcal{A}_0$				
ELSE	IF	$\mathit{child} = \swarrow$				
	THEN	$\mathbf{n} := \mathbf{n}_{\swarrow}$; $B := B \cup \mathcal{A}_1$				
	ELSE	$\mathbf{n} := \mathbf{n}_{\searrow}$; $B := B \cup \mathcal{A}_0$				
IF	**n** is no leaf					
THEN	$\mathcal{A}_0 := \mathcal{L}(\mathbf{n}_{\swarrow})$; $\mathcal{A}_1 := \mathcal{L}(\mathbf{n}_{\searrow})$					
	FOR All landmarks $l \in \mathcal{M}$					
		IF	$\mathbf{p}\,[l]$ exists			
		THEN	IF	$\mathbf{p}\,[l]\,[0] = \mathbf{n}$		
			THEN	Remove first element of $\mathbf{p}\,[l]$		
				IF	$\mathbf{p}\,[l]$ is not empty	
				THEN	IF	$\mathbf{p}\,[l]\,[0] = \mathbf{n}_{\swarrow}$
					THEN	$\mathcal{A}_0 := \mathcal{A}_0 \cup \{l\}$
					ELSE	$\mathcal{A}_1 := \mathcal{A}_1 \cup \{l\}$
			ELSE	Remove $\mathbf{p}\,[l]$		
ELSE	$\mathcal{A}_0 := \mathcal{A}_1 := \mathcal{L}(\mathbf{n}_{\text{BIB}})$					
$\mathcal{B} := \mathcal{B} \cap (\mathcal{A}_0 \cup \mathcal{A}_1 \cup \mathcal{C} \cup \mathcal{D} \cup \mathcal{M})$						
return $(\mathbf{n}, \mathbf{r}, \mathcal{A}_0, \mathcal{A}_1, \mathcal{B}, \mathcal{C}, \mathcal{D}, \mathcal{M}, \mathbf{p})$						

`updateLandmarkState`). An analysis of the runtime complexity of (Fig. C.10, `optimizeHTP`) is given in table C.1.

C.6 Implementation of the Bookkeeping Part

There are some details to consider when implementing the bookkeeping part of the algorithm presented in §4 and this appendix:

The tree map as main data structure is implemented by a pointer linked binary tree as usual with dynamically allocated node objects. To be able to build external references, it is useful to assign ids to the nodes and use an index map (for instance `STL::map<>`) to quickly access a node with a given index. The elimination nodes can be stored in a plain array given that the landmarks are numbered consecutively.

The algorithm often has to manipulate sets of landmarks. These sets should be implemented as an ascending array of indices to be compatible to the matrix row / column description dis-

optimizeHTP	$O(k^3 \log n)$
findNodeToBeOptimized	$O(\log n)$
findBestTransfer	$O(k^2 \log n)$
initLandmarkState	$O(k \log n)$
$O(k \log n)$ times	$O(k^2 \log n)$
$\cdots$	$O(k)$
updateLandmarkState	$O(k)$
updateLandmarkState	$O(k)$
findBestInsertionPoint	$O(k \log n)$
$O(\log n)$ times	$O(k \log n)$
$\cdots$	$O(k)$
updateLandmarkState	$O(k)$
transferSubtree	$O(k^3 \log n)$
$\cdots$	$O(k \log n)$
$O(\log n)$ times	$O(k^3 \log n)$
update	$O(k^3)$
computeCIBandSIB	$\mathbf{O(k^3)}$

Table C.1: Calling hierarchy of `optimizeHTP`

cussed in section C.1. All set operations ($\cup, \cap, -$, etc.) can be performed by simultaneously iterating through all involved arrays using one iterator / index for each and processing all elements in ascending order. (Like the merge operation in merge sort [Sed92, chapter 12])

The only exception are the sets $\mathcal{A}_0, \mathcal{A}_1, \mathcal{B}, \mathcal{C}, \mathcal{D}$ describing the state of landmarks with respect to some nodes (section C.5). Here it is probably advantageous to use a single record (`struct`) with boolean flags for each landmark and maintain the whole state as an array of records for the different landmarks sorted by ascending index. This way the rather complex operations (like for instance equation C.4) can be implemented by simple boolean operators.

It is worth noting, that the whole bookkeeping part except section 4.2 and `relinearizeInTree` work independent from the actual data stored in the map. So it is possible to implement the bookkeeping part alone, without any actual measurements. Conversely the linear algebra part presented in chapter 3 can be implemented and tested without the bookkeeping part by manually specifying the sequence of operations the bookkeeping part would apply in a specific example.

Bibliography

[BCF+99] W. Burgard, A. Cremers, D. Fox, D. Hahnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun, *Experiences with an interactive museum tour-guide robot*, Artificial Intelligence **114** (1999), no. 1 – 2, 3 – 55.

[BEF96] J. Borenstein, B. Everett, and L. Feng, *Navigating mobile robots: Systems and techniques*, A. K. Peters, Ltd., Wellesley, 1996.

[BFJ+99] W. Burgard, D. Fox, H. Jans, C. Matenar, and S. Thrun, *Sonar-based mapping with mobile robots using EM*, Proceedings of the 16th International Conference on Machine Learning, San Francisco, 1999, pp. 67 – 76.

[BGLH01] J. Butterfass, M. Grebenstein, H. Liu, and G. Hirzinger, *DLR-Hand II: Next generation of dextrous robot hand.*, In Proceedings of the IEEE Conf. on Robotics and Automation, Seoul, 2001, pp. 109 – 114.

[BHG+02] M. Beetz, J. Hertzberg, L. Guibas, M. Ghallab, and M.E. Pollack (eds.), *Advances in plan-based control of robotic agents, lecture notes in artificial intelligence 2466*, Springer, 2002.

[BPP02] C. Bellini, S. Panzieri, and F. Pascucci, *A real-time architecture for low-cost vision based robots navigation*, Proceedings of the 15th IFAC World Congress, Barcelona, 2002.

[BR95] R. Basri and E. Rivlin, *Localization and homing using combinatins of model view*, Artificial Intelligence **78** (1995), no. 1-2.

[Bri99] W.L. Briggs, *A multigrid tutorial*, Ninth Copper Mountain Conference On Multigrid Methods, April 1999, (http://www.llnl.gov/CASC/people/henson/mgtut/ps/mgtut.pdf).

[Bro85] R.A. Brooks, *Visual map making for a mobile robot*, Proceedings of the IEEE International Conference on Robotics and Automation, St. Louis, 1985, pp. 824 – 829.

[CL85] R. Chatila and J.P. Laumond, *Position referencing and consistent world modeling for mobile robots*, Proceedings of the IEEE International Conference on Robotics and Automation, St. Louis, 1985, pp. 138 – 145.

[CMNT99] J.A. Castellanos, J. Montiel, J. Neira, and J.D. Tardos, *The SPmap: A probablistic framework for simultaneous localization and map building*, IEEE Transactions on Robotics and Automation **15** (1999), no. 5, 948 – 952.

[CS86] R. Cheeseman and P. Smith, *On the representation and estimation of spatial uncertainty*, International Journal of Robotics **5** (1986), 56 – 68.

[CT00] Jose A. Castellanos and J.D. Tardos, *Mobile robot localization and map building: A multisensor fusion approach*, Kluwer Academic Publishers, Boston/ Dorfdrecht/ London, 2000.

[CTS97] J.A. Castellanos, J.D. Tardos, and G. Schmidt, *Building a global map of the environment of a mobile robot: The importance of correlation*, Proceedings of the IEEE International Conference on Robotics and Automation, Albuquerque, 1997, pp. 1053 – 1059.

[CW90] I. J. Cox and G. T. Wilfong (eds.), *Autonomous robot vehicles*, Springer Verlag, New York, 1990.

[DdFG01] A. Doucet, J.F.G de Freitas, and N.J. Gordon, *Sequential monte carlo methods in practice*, Springer Verlag, New York, 2001.

[DMD02] C. Drexler, F. Mattern, and J. Denzler, *Generic hierarchic object models and classification based on probabilistic pca*, Proceedings of Machine Vision Applications 2002, Nara, 2002, pp. 435–438.

[DMS84] S. Demko, W.F. Moss, and P.W. Smith, *Deacy rates for inverses of band matrices*, Mathematics of Computation **43** (1984), no. 168, 491 – 499.

[DMS00] T. Duckett, S. Marsland, and J. Shapiro, *Learning globally consistent maps by relaxation*, Proceedings of the IEEE International Conference on Robotics and Automation, San Francisco, 2000, pp. 3841–3846.

[DMS02] T. Duckett, S. Marsland, and J. Shapiro, *Fast, on-line learning of globally consistent maps*, Autonomous Robots **12** (2002), no. 3, 287 – 300.

[DN00] T. Duckett and U. Nehmzow, *Performance comparison of landmark recognition systems for navigating mobile robots*, Proceedings of the AAAI National Conference on Artificial Intelligence, Austin, 2000, pp. 826–831.

[DN01] T. Duckett and U. Nehmzow, *Mobile robot self-localisation using occupancy histograms and a mixture of gaussian location hypotheses*, Robotics and Autonomous Systems **34** (2001), no. 2 – 3, 119 – 130.

[DNC+01] M.W.M.G. Dissanayake, P. Newman, S. Clark, H.F. Durrant-Whyte, and M. Csorba, *A solution to the simultaneous localization and map building (SLAM) problem*, IEEE Transactions on Robotics and Automation **17** (2001), no. 3, 229 – 241.

[DNDW+99] M.W.M.G. Dissanayake, P. Newman, H.F. Durrant-Whyte, S. Clark, and M. Csobra, *An experimental and theoretical investigation into simultaneous localisation and map building*, Proceedings of the 6.th International Symposium on Experimantal Robotics, Sydney, 1999, pp. 265–274.

[DW88] H.F. Durrant-Whyte, *Uncertain geometry in robotics*, IEEE Transactions on Robotics and Automation **4** (1988), no. 1, 23 – 31.

[DWMdB+01] H.F. Durrant-Whyte, S. Majumder, M. de Battista, S. Thrun, and S. Scheding, *A bayesian algorithm for simultaneous localisation and map building*, Proceedings of the International Symposium of Robotics Research, Lorne Victoria, 2001.

[DWRN95] H.F. Durrant-Whyte, D. Rye, and E. Nebot, *Localization of autonomous guided vehicles*, Proceedings of the 8th International Symposium on Robotics Research (G. Hirzinger and G. Giralt, eds.), Springer Verlag, New York, 1995, pp. 613 – 625.

[Elf89] A. Elfes, *Occupancy grids: A probabilistic framework for robot perception and navigation*, Ph.D. thesis, Department of Eletrical Engineering, Carnegie Mellon Univrsity, 1989.

[Eur02] European Network for Climbing and Walking Robots, *Clawar home page*, October 2002, (http://www.uwe.ac.uk/clawar/).

[Fau89] O. Faugeras, *Maintaining representations of the environment of a mobile robot*, IEEE Transactions on Robotics and Automation (1989), 804.

[FC03] J. Folkesson and H. Christensen, *Outdoor exploration and SLAM using a compressed filter*, Proceedings of the IEEE International Conference on Robotics and Automation, Taipei, 2003, pp. 419–426.

[FD03] U. Frese and T. Duckett, *A multigrid approach for accelerating relaxation-based SLAM*, Proceedings of the IJCAI Workshop Reasoning with Uncertainty in Robotics, Acapulco, 2003, pp. 39–46.

[FDF02] E. Frazzoli, M. Dahleb, and E. Feron, *Real time motion planning for agile autonomous vehicles*, AIAA Journal on Guidance, Control and Dynamics (2002).

[Fed99] H.J. Feder, *Simultaneous stochastic mapping and localization*, Ph.D. thesis, MIT, Cambridge, 1999.

[FH01] U. Frese and G. Hirzinger, *Simultaneous localization and mapping - a discussion*, Proceedings of the IJCAI Workshop on Reasoning with Uncertainty in Robotics, Seattle, August 2001, pp. 17 – 26.

[FHBH00] U. Frese, M. Hörmann, B. Bäuml, and G. Hirzinger, *Globally consistent visual localization without a-priori map (german)*, automatisierungstechnik **3** (2000), 273 – 280.

[FKL+03] J. Fritsch, M. Kleinehagenbrock, S. Lang, T. Plötz, G.A. Fink, and G. Sagerer, *Multi-modal anchoring for human-robot-interaction*, Journal of Robotics and Autonomous Systems, Special issue on Anchoring Symbols to Sensor Data in Single and Multiple Robot Systems **43** (2003), no. 2, 133–147.

[FLD04] U. Frese, Per Larsson, and T. Duckett, *A multigrid algorithm for simultaneous localization and mapping*, IEEE Transactions on Robotics (2004), (to appear).

[FLS+03] Li-Chen Fu, C.S.G. Lee, B. Siciliano, B. Lee, and S. Hirose (eds.), *Proceedings of the IEEE International Conference on Robotics and Automation, Taipei*, IEEE, 2003.

[FM82] C.M. Fiduccia and R.M. Mattheyses, *A linear-time heuristic for improving network partitions*, Proceedings of the 19th ACM/IEEE Design Automation Conference, Las Vegas, June 1982, pp. 175–181.

[Gau21] C.F. Gauss, *Theoria combinationis observationum erroribus minimis obnoxiae*, Commentationes societatis regiae scientiarum Gottingensis recentiores **5** (1821), 6–93.

[Gel74] A. Gelb (ed.), *Applied optimal estimation*, MIT Press, Cambridge, 1974.

[GFS03] J.S. Gutmann, M. Fukuchi, and K. Sabe, *Environment identification by comparing maps of landmarks*, Proceedings of the IEEE International Conference on Robotics and Automation, Taipei, 2003, pp. 662 – 667.

[GJ79] M.R. Garey and D.S. Johnson, *Computers and intractability: Guide to the theory of NP-completeness*, C. Freeman, San Francisco, 1979.

[GK99] J.S. Gutmann and K. Konolige, *Incremental mapping of large cyclic environments*, Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation, Monterey, 1999.

[GN01] J.E. Guivant and E.M. Nebot, *Optimization of the simultaneous localization and map-building algorithm for real-time implementation*, IEEE Transactions on Robotics and Automation **17** (2001), no. 3, 242 – 257.

[GN02] J.E. Guivant and E.M. Nebot, *Solving computational and memory requirements of feature based simultaneous localization and map building algorithms*, Tech. report, Australian Centre for Field Robotics, University of Sydney, Sydney, 2002.

[GOR94] J. González, A. Ollero, and A. Reina, *Map building for a mobile robot equipped with a 2D laser rangefinder*, Proceedings of the IEEE International Conference on Robotics and Automation, San Diego, 1994, pp. 1904 – 1909.

[GSASH03] G. Grunwald, G. Schreiber, A. Albu-Schäffer, and G. Hirzinger, *Programming by touch: The different way of human-robot interaction*, IEEE Transaction on industrial electronics **50** (2003), no. 4.

[Hag89] W.W. Hager, *Updating the inverse of a matrix*, SIAM Review **31** (1989), no. 2, 221 – 239.

[HASH+01] G. Hirzinger, A. Albu-Schäffer, M. Hähnle, I. Schaefer, and N. Sporer, *On a new generation of torque controlled light-weight robots*, Proceedings of the IEEE International Conference on Robotics and Automation, Seoul, 2001, pp. 3356–3363.

[HBBC95] P. Hebert, S. Betge-Brezetz, and R. Chatila, *Probabilistic map learning, necessity and difficulty*, Proceedings of the International Workshop Reasoning with Uncertainty in Robotics, 1995, pp. 307 – 320.

[HBBH04] U. Hillenbrand, B. Brunner, C. Borst, and G. Hirzinger, *Towards service robots for the human environment: the robutler*, Proceedings of the IEEE International Conference on Robotics and Automation, New Orleans, 2004.

[HJ90] R.A. Horn and C.R. Johnson, *Matrix analysis*, 2 ed., Cambridge University Press, Cambridge, 1990.

[HK98] Mei Han and Takeo Kanade, *Homography-based 3d scene analysis of video sequences*, Proceedings of the DARPA Image Understanding Workshop, 1998.

[HL95] B. Hendrickson and R. Leland, *A multilevel algorithm for partitioning graphs*, Proceedings of the ACM International Conference on Supercomputing, Sorrento, 1995, pp. 626–657.

[HN00] J. Hornegger and H. Niemann, *Probabilistic modeling and recognition of 3-d objects*, International Journal of Computer Vision **39** (2000), no. 3, 229–251.

[Hon90] J. Honerkamp, *Stochastische Dynamische Systeme*, VCH Verlagsgesellschaft, Weinheim, 1990.

[HS98] U.D. Hanebeck and G. Schmidt, *Mobile robot localization based on efficient processing of sensor data and set-theoretic state estimation*, The Fifth International Symposium on Experimental Robotics, Barcelona, 1998, pp. 385–396.

[HSFS00] U. D. Hanebeck, N. Saldic, F. Freyberger, and G. Schmidt, *Modulare Radsatzsysteme für omnidirektionale mobile Roboter*, Robotik 2000 Tagung (VDI/VDE-Gesellschaft Mess- und Automatisierungstechnik), VDI Berichte 1552, Berlin, 2000, pp. 39–44.

[Jen01] P. Jensfelt, *Approaches to mobile robot localization in indoor environments*, Ph.D. thesis, Royal Institute of Technology, Stockholm, 2001.

[JU96] S.J. Julier and J. Uhlmann, *A general method for approximating nonlinear transformations of probability distributions*, Tech. report, Dept. of Engineering Science, University of Oxford, Oxford, 1996.

[Koe01] R.H. Koeppe, *Robot compliant motion based on human skill*, Ph.D. thesis, Swiss Federal Institute of Technology ETH Zürich, 2001.

[KS03] J.H. Kin and S. Sukkarieh, *Airborne simultaneous localisation and map building*, Proceedings of the IEEE International Conference on Robotics and Automation, Taipei, 2003, pp. 406–411.

[KW94] D. Kortenkamp and T. Weymouth, *Topological mapping for mobile robtos using a combination of sonar and vision sensing*, Proceedings of the National Conference on Artificial Intelligence, Seattle, 1994.

[LDW92] J.J. Leonard and H.F. Durrant-Whyte, *Directed sonar sensing for mobile robot navigation*, Kluwer Academic Publishers, Boston, 1992.

[LF99] J.J. Leonard and H.J.S. Feder, *Decoupled stochastic mapping*, Tech. report, MIT, Cambridge, December 1999.

[LM97] F. Lu and E. Milios, *Globally consistent range scan alignment for environment mapping*, Autonomous Robots **4** (1997), 333 – 349.

[LY03] C.S.G. Lee and J. Yuh (eds.), *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Las Vegas*, IEEE, 2003.

[ME85] H.P. Moravec and A. Elfes, *High resolution maps from wide angle sonar*, Proceedings of the IEEE International Conference Robotics and Automation, St. Louis, 1985, pp. 116 – 121.

[ME88] L. Matthies and A. Elfes, *Integration of sonar and stereo range data using a grid-based representation*, Proceedings of the IEEE International Conference on Robotics and Automation, Philadelphia, 1988, pp. 727 – 733.

[Min88] C. Ming Wang, *Location estimation and uncertainty analysis for mobile robots*, Proceedings of the IEEE International Conference on Robotics and Automation, Philadelphia, 1988, pp. 1230 – 1235.

[MNPW98] Nicola Muscettola, P. Pandurang Nayak, Barney Pell, and Brian C. Williams, *Remote agent: To boldly go where no AI system has gone before*, Artificial Intelligence **103** (1998), no. 1-2, 5–47.

[MOR03] MORPHA, *Morpha home page*, 2003, (http://www.morpha.de/).

[MTKW02] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, *FastSLAM: A factored solution to the simultaneous localization and mapping problem*, Proceedings of the AAAI National Conference on Artificial Intelligence, Edmonton, 2002, pp. 593–598.

[New99] P.M. Newman, *On the structure and solution of the simultaneous localisation and map building problem*, Ph.D. thesis, Deptartment of Mechanical and Mechatronic Engineering, Sydney, 1999.

[Nie89] H. Niemann, *Pattern analysis and image understanding*, Springer-Verlag, Berlin, 1989.

[NT00] J. Neira and J.D. Tardos, *Robust and feasible data asssociation for simultaneous localization and map building*, ICRA Workshop SLAM, San Francisco, 2000.

[NT01] J. Neira and J. Tardós, *Data association in stochastic mapping using the joint compatibility test*, IEEE Transactions on Robotics and Automation **6** (2001), no. 17, 890 – 897.

[Ots79] N. Otsu, *A threshold selection method from gray-level histograms*, IEEE Transactions on Systems, Man and Cybernetics **9** (1979), no. 1, 62 – 66.

[PNDW96] D. Pagac, E.M. Nebot, and H.F. Durrant-Whyte, *An evidential approach to probabilistic map-building*, Proceedings of the IEEE International Conference on Robotics and Automation, Minneapolis, 1996, pp. 745 – 750.

[PTVF92] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery, *Numerical recipes, second edition*, Cambridge University Press, Cambridge, 1992.

[Ren93] W.D. Rencken, *Concurrent localization and map building for mobile robots using ultrasonic sensors*, Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Yokohama, 1993.

[SA94] S. Schaal and C. Atkeson, *Memory-based robot learning*, Proceedings of the IEEE International Conference on Robotics and Automation, San Diego, 1994, pp. 2928 – 2933.

[SABL01] B.M. Steinmetz, K. Arbter, B. Brunner, and K. Landzettel, *Autonomous vision-based navigation of the nanokhod rover*, Proceedings of the 6th International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS), Montreal, 2001.

[SC94] B. Schiele and J. Crowley, *A comparison of position estimation techniques using occupancy grids*, Proceedings of the IEEE International Conference on Robotics and Automation, San Diego, 1994, pp. 1628 – 1634.

[Sed92] R. Sedgewick, *Algorithms in C++*, Addison Wesley Publishing Company, Reading, 1992.

[SIC04] SICK AG, *Sick homepage*, 2004, (http://www.sick.de/).

[SK02] C. Schlegel and T. Kämpke, *Filter design for simultaneous localization and mapping (SLAM)*, Proceedings of the IEEE International Conference on Robotics and Automation, Washington D.C., 2002, pp. 2737 – 2742.

[SLL02] S. Se, D. Lowe, and J. Little, *Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks*, International Journal of Robotics Research **21** (2002), no. 8, 735–758.

[SSC88] R. Smith, M. Self, and P. Cheeseman, *Estimating uncertain spatial relationships in robotics*, Autonomous Robot Vehicles (I.J. Cox and G.T. Wilfong, eds.), Springer Verlag, New York, 1988, pp. 167 – 193.

[Tar92] J.D. Tardos, *Representing partial and uncertain sensorial information using the theory of symmetries*, Proceedings of the IEEE International Conference on Robotics and Automation, Nice, 1992, pp. 1799 – 1804.

[TBF98] S. Thrun, W. Burgard, and D. Fox, *A probabilistic approach to concurrent mapping and localization for mobile robot*, Machine Learning **31** (1998), no. 5, 1 – 25.

[TDH03] S. Thrun, M. Diel, and D. Hähnel, *Scan alignment and 3D surface modeling with a helicopter platform*, Proceedings of the International Conference on Field and Service Robotics, Lake Yamanaka, 2003.

[Tee00] G.J. Tee, *Bounds for eigenvalues of positive-definite band matrices*, Australian Mathematical Society Gazette **27** (2000), 155 – 157.

[THF+03] S. Thrun, D. Hähnel, D. Ferguson, M. Montemerlo, R. Triebel, W. Burgard, C. Baker, Z. Omohundro, S. Thayer, and W. Whittaker, *A system for volumetric robotic mapping of abandoned mines*, Proceedings of the IEEE International Conference on Robotics and Automation, Taipei, 2003, pp. 4270–4275.

[Thr02] S. Thrun, *Robotics mapping: A survey*, Tech. report, School of Computer Science, Carnegie Mellon University, February 2002.

[TKG+02] S. Thrun, D. Koller, Z. Ghahramani, H. Durrant-Whyte, and Ng. A.Y., *Simultaneous mapping and localization with sparse extended information filters: Theory and initial results*, Proceedings of the Fifth International Workshop on Algorithmic Foundations of Robotics, Nice, 2002.

[TKGDW02] S. Thrun, D. Koller, Z. Ghahmarani, and H. Durrant-Whyte, *SLAM updates require constant time*, Tech. report, School of Computer Science, Carnegie Mellon University, Pittsburgh, 2002.

[UJC97] J.K. Uhlmann, S.J. Julier, and M. Csorba, *Nondivergent simultaneous map building and localization using covariance intersection*, Proceedings of the SPIE Conference on Navigation and Control Technologies for Unmanned Systems II, vol. 3087, 1997, pp. 2 – 11.

[VBX96] J. Vandorpe, H. Van Brussel, and H. Xu, *Exact dynamic map building for a mobile robot using geometrical primitives produced by a 2D range finder*, Proceedings of the IEEE International Conference on Robotics and Automation, Minneapolis, 1996, pp. 901 – 908.

[Vij91] G. Vijayan, *Generalization of min-cut partitioning to tree structures and its applications*, IEEE Transactions on Computers **40** (1991), no. 3, 307 – 314.

[WDDW02] S.B. Williams, G. Dissanayake, and H.F. Durrant-Whyte, *Field deployment of the simultaneous localisation and mapping algorithm*, 15th IFAC World Congress on Automatic Control, Barcelona, June 2002.

[WGLSV00] N. Winters, J. Gaspar, G. Lacey, and J. Santos-Victor, *Omni-directional vision for robot navigation*, Proceedings of the IEEE Workshop on Omni-directional Vision, Hilton Head Island, 2000.

[YL97] B. Yamauchi and P. Langley, *Place recognition in dynamic environments*, Journal of Robotic Systems, Special Issue on Mobile Robots **14** (1997), no. 2, 107–120.

[ZDH+01] M. Zobel, J. Denzler, B. Heigl, E. Nöth, D. Paulus, J. Schmidt, and G. Stemmer, *Mobsy: Integration of vision and dialogue in service robots*, Proceedings Second International Workshop on Computer Vision Systems, Vancouver, 2001, pp. 50 – 62.

[Zel91] A. Zelinski, *Mobile robot map making using sonar*, Journal of Robotic Systems **8** (1991), no. 5, 557 – 577.

Index

Curriculum Vitae

Udo Frese was born in Minden, Germany in 1972. He received the Diploma degree in computer science from the University of Paderborn in 1997. From 1998 to 2003 he was a Ph.D. student at the German Aerospace Center (DLR) in Oberpfaffenhofen. In 2004 he joined the Bremen Institute of Safe Systems at university of Bremen. His research interests are mobile robotic, simultaneous localization and mapping and semantic interpretation of maps.