Exploiting Locality by Nested Dissection For Square Root Smoothing and Mapping ($\sqrt{SAM}$)

Peter Krauthausen, Frank Dellaert, Alexander Kipp
College of Computing
Georgia Institute of Technology, Atlanta, GA
peter@krauthausen.com, dellaert@cc.gatech.edu, mail@alexanderkipp.de

Technical Report number GIT-GVU-05-26
September 2005

Abstract

The problem of creating a map given only the erroneous odometry and feature measurements and locating the own position in this environment is known in the literature as the Simultaneous Localization and Mapping (SLAM) problem. In this paper we investigate how a Nested Dissection Ordering scheme can improve the the performance of a recently proposed Square Root Information Smoothing (SRIS) approach. As the SRIS does perform smoothing rather than filtering the SLAM problem becomes the Smoothing and Mapping problem (SAM). The computational complexity of the SRIS solution is dominated by the cost of transforming a matrix of all measurements into a square root form through factorization. The factorization of a fully dense measurement matrix has a cubic complexity in the worst case. We show that the computational complexity for the factorization of typical measurement matrices occurring in the SAM problem can be bound tighter under reasonable assumptions. Our work is motivated both from a numerical / linear algebra standpoint as well as by submaps used in EKF solutions to SLAM.

1 Introduction

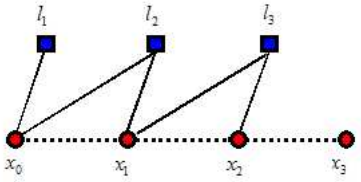
In recent years the number of applications involving robots mapping an unknown environment has increased immensely. The problem of creating a map given only erroneous odometry and feature measurements while locating the own position in this environment is known in the literature as the Simultaneous Localization and Mapping (SLAM) problem. The SLAM problem is fundamental to most robot applications. Recently it has been proposed to use Square Root Information Smoothing (SRIS) to solve this problem. As the SRIS does perform smoothing rather than filtering the SLAM problem will become the Smoothing and Mapping problem (SAM). In contrast to the solutions involving an Extended Kalman-Filter (EKF) or its dual, the Extended Information-Filter (EIF), this approach delivers a Maximum Likelihood map. Nevertheless this approach suffers from the fact that the computational complexity is dominated by the cost of transforming a matrix of all measurements into a square root form through factorization. Thus smoothing in the case of a completely dense measurement matrix has a cubic complexity in the worst case. Since the measurement matrix is usually quite sparse we can assume that cubic complexity is a very pessimistic estimation. Furthermore changing the way the matrix is factorized by ordering the variables differently can greatly reduce the number of operations needed [8]. In this paper we explain how different ordering algorithms work and how the structure of the measurement matrix is exploited to reduce the runtime. The Nested Dissection algorithm as a *divide-and-conquer* approach lends itself very well to exploiting the locality inherent in the geometrical nature of the mapping task. Given the properties of the computational complexity of the Nested Dissection orderings we obtain tighter complexity bounds for typical measurement patterns under reasonable assumptions.

In the first section we show how the SAM process can be phrased as a graphical model as well as a matrix computation. In Section 3 we will use these two representations of the problem to describe how cunningly changing the order of variables for elimination can improve the runtime performance of a factorization, thus the SRIS. Standard as well as Nested Dissection orderings are introduced. In the next section we show how the computational properties of the Nested Dissection can be used to bound the computational complexity of SRIS for typical measurement patterns. In contrast to the complexity of $O(n^3)$ for a dense matrix factorization we can obtain a complexity bound of at most $O(n\,log_2\,n)$ for the total fill-in and at most $O(n^{\frac{3}{2}})$ for the multiplication count for the factorization of a pre-ordered matrix. Nevertheless these bounds are linked to high constant factors that need to be taken into consideration when applying the algorithm. In the following two sections an application on an indoor scenario is given and an analysis of the impact of the constants in the complexity bounds is performed. We conclude this paper with a comparison of submaps - which are widely used in EKF solutions - to Nested Dissection partitions.

2 The Smoothing and Mapping Problem

Recent papers [14, 8] show how the process of a robot moving through an environment and sensing features can be understood as building a graphical model. The vertices V in the graph $G = (E, V)$ represent variables and the edges the relations between the variables. In our problem the variables are the poses of the trajectory of the robot and the features it has sensed. Whereas the relations are odometry and feature measurements. Thus we are interested in obtaining the true positions for the poses and features given our measurements.

Given such a graphical model the robot moving from pose x_i to $x_{i+1}$and sensing features l_i in this step is described by adding vertices for x_i and l_i and edges representing the odometry and feature measurements $x_i \longrightarrow x_{i+1}$and $x_i \longrightarrow l_j$ to G . Fig. 1 depicts such a graph for a robot which has moved four steps and sensed features at each time step.



$$\begin{bmatrix}
G_1^1 & & & & & & \\
F_2^1 & G_2^2 & & & & & \\
 & F_3^2 & G_3^3 & & & & \\
 & & F_4^3 & G_4^4 & & & \\
H_1^1 & & & & J_!^1 & & \\
H_2^1 & & & & & J_2^2 & \\
 & H_3^2 & & & & J_3^2 & \\
 & H_4^2 & & & & & J_4^3 \\
 & & H_5^3 & & & & J_5^3
\end{bmatrix}$$

Figure 1: Left, measurement graph of a scenario where a robot moves four steps and senses features at each step. The vertices are the robot positions (circles) and feature positions (squares). The edges represent the $x_i \longrightarrow x_{i+1}$ and $x_i \longrightarrow l_j$ measurements. Right, measurement matrix corresponding to measurement graph on the left. Note that the Jacobians are block matrices of different size depending on whether they represent measurements $x_i \longrightarrow x_{i+1}$or $x_i \longrightarrow l_j$. Each row represents one measurement whereas each column represents an unknown robot or feature position.

Following [8] this graphical model is equivalent to a system of linear equations or more compact a measurement matrix $A \in \mathbb{R}^{m \times n}$ consisting of the Jacobians for the odometry measurements and the feature measurements with x_i and l_i as variables. A is full-rank *per definition* and fulfills the Strong Hall condition[1]. Note that A grows enormously as a block of rows and a block of columns are added for each new measurement. Yet, the matrix is very sparse[2].

Using the matrix A to solve the mapping problem means solving the least squares problem given by Eq. 1. Whereas θ is our current estimate of the true underlying position of the robot poses and the features.

$$\theta^* = \arg\min \|A\theta - b\|_2 \tag{1}$$

Solving Eq. 1 can be done by using QR or Cholesky factorization by solving the *normal-equations* $A^T A = A^T b$ [2]. For Cholesky factorization this yields the *Algorithm 1* and for QR factorization *Algorithm 2*.

In the case of QR factorization the RHS is most often calculated by attaching it to A. In addition Q is often not formed but replaced by a series of *Householder reflections* [7]. Note

[1]A matrix $A \in \mathbb{R}^{mxn}$has the *Strong Hall Property* (SHP) if every $m \times k$ sub-matrix, for $1 \leq k < n$, has at least $k + 1$ nonzero rows. In real applications least squares problems almost always fulfill this requirement or A can be permuted into a form that has the SHP. *[18, 30]*

[2]The block size depends on the size of the Jacobians, thus on the dimensionality of the space used. For example in a planar environment a robot pose is defined by the $2D$-Position and an angle describing the heading. A feature measurement is defined by bearing and range. Therefore the following sizes could be used. $F, G \in \mathbb{R}^{3\times3}$, $H \in \mathbb{R}^{2\times3}$and $J \in \mathbb{R}^{2\times2}$. The block sizes are fixed. For further description we refer the reader to [8].

Algorithm 1 Solving Eq. using Cholesky Factorization

(1) Form $\mathcal{I} \triangleq A^T A = R^T R$

(2) Solve $R^T y = A^T b$

(3) Solve $R\theta^* = y$

that due to the uniqueness of the factorization $R \in \mathbb{R}^{n \times n}$ is the same triangular matrix for both factorizations.[3]

Algorithm 2 Solving Eq. using QR Factorization

(1) Form $Q^T A = \begin{bmatrix} R \\ 0 \end{bmatrix}$

(2) Solve $Q^T b = \begin{bmatrix} d \\ e \end{bmatrix}$

(3) Solve $R\theta = d$

The aspect of Alg. *1* and Alg. *2* that interests us most is the computational complexity. The number of non-zero elements (NNZ) initially in the matrix A and the non-zero elements in R govern the amount of computation needed for the factorization and back-substitution. We can not influence the first quantity but we can influence the number of non-zeros z introduced by the factorization by reordering the columns of the A as we will show in Section 3. Note that the mere size of the matrix A does not determine the amount of computation. In the coarse algorithm below we show the complexity of the different steps of the algorithm [25].

[3] Some authors denote the lower triangular matrix of the Cholesky factorization as R. To avoid any confusion of the reader we keep a uniform notation throughout this paper.

Algorithm 3 Iterative solution for least squares problems, [25]

Given $A \in \mathbb{R}^{m \times n}$,

(1) Reordering $A_\pi \overleftarrow{\pi} A$- Complexity depends on the method of choice

(2) Symbolic Factorization - Compute Space needed for non-zero elements $O(z)$ Time

Repeat

(*3) Numeric Factorization* - Compute R: $O(z^3)$, if A is dense

(*4) Back-substitution* $O(z)$ Time

Until *Convergence*

(5) *Backorder solution* $\theta \quad \overleftarrow{\pi^{-1}} \ \theta_\pi$

For a least squares problem this sequence of computation has to be repeated until convergence. Yet our experiments have shown that only a very small number of iterations is needed until convergence.

3 Matrix Reordering

In this section we will show how non-zeroes are introduced in R. We will introduce column reordering schemes and show how orderings can reduce the NNZ. As we have seen in Section 2 the matrix A consists of small block matrices. In this section we will also show the effect of combining the block matrices into one variable in contrast to considering each column a separate a variable.

We will start this section by presenting standard ordering algorithms and continue with Nested Dissection orderings. For the latter we will expose how locality of the underlying matrix structure is exploited automatically. Before we continue in the next section with an analysis of the computational complexity of Nested Dissection orderings on several graph types and for typical measurement patterns.

3.1 How reordering effects the NNZ in R

Reordering the matrix $A \in \mathbb{R}^{m \times n}$means that we permute the columns. Let $\tau : \{1, .., n\} \mapsto \{1, .., n\}$be a bijective function that reorders indices. Then the following matrix

$$P \in \mathbb{R}^{n \times n}, P_c = \left(\left(\delta_{i,\tau(j)}\right)\right)$$

defines a column permutation and

$$P \in \mathbb{R}^{m \times m}, P_r = \left(\left(\delta_{\tau(i),j}\right)\right)$$

a row permutation.

Eq. 2 shows that for a Cholesky factorization row permutation is obsolete as it is canceled out in the matrix multiplication. Thus only a column permutation will change the structure of the A^TA and therefore the NNZ of the Cholesky triangle. Note that Eq. 2 also shows that the column permutation actually represents a symmetric permutation of the information matrix[4].

$$(P_rAP_c)^T(P_rAP_c) = P_c^T(A^TA)P_c \tag{2}$$

As the QR factorization operates directly on A, thus P_rAP_c, the row ordering matters for a QR factorization. Nevertheless if one uses a multifrontal QR approach all rows with the same leading non-zero elements will be gathered in the same frontal matrix block. This gives a rough row ordering as the row blocks are usually fairly small, [9]. Because of this and since both QR and Cholesky factorization will benefit from a good column reordering we will only discuss column orderings in the rest of this paper.

So far we have been abstract about how an ordering affects the factorization process. It is far out of the scope of this paper to provide the reader with a detailed understanding of the underlying mechanisms. The following paragraph shall give the reader an intuition of how the factorization works. The interested reader is referred to [25, 18] for a more detailed explanation.

Let $A \in \mathbb{R}^{m\times n}$ be a symbolic matrix, A^TA its information matrix and let the graph G be constructed as described in Section 2. Vertices in G represent a variable whereas edges represent the dependencies between them. An edge between two variables means that they can be expressed as a linear combination of the other and their respective dependant variables. Furthermore elimination of the matrix can be understood graph-theoretically as a recursive application of a function on G that eliminates one variable at a time. The application of this function will eliminate variables in G until there is only one variable left. Eliminating a variable means that we express this variable through the variables that it is a linear combination of. When we remove a variable we thus introduce new dependencies into the graph as we have to link every depending variable of the variable to be eliminated with the variable it will from now on be expressed by. Graph-theoretically this means that we remove a node from the G and then add edges to the reduced graph. We make the set of nodes adjacent to the removed variable a completely connected subgraph, a clique. In matrix terms this means that we add non-zeros to the R triangle. Thus the optimal solution is to find an ordering of the variables for the elimination that will result in the least increase in dependencies overall. Unfortunately finding the next unknown to be eliminated is an NP-complete problem [1].

3.2 Standard orderings

As we have shown the problem of finding an optimal ordering is NP-complete. This problem has been known for a long time by the linear algebra and scientific computation community [21, 18, 17, 20]. A lot of heuristic approaches have been developed to tackle this problem. In this section we will present the two most widely used sparse matrix ordering algorithms for matrix oriented scientific computation on standard desktop machines. Both approaches are based on the heuristic of eliminating the least constrained variables of G, thus A^TA. The family of algorithms based on this heuristic is known as the Minimum Degree algorithms (MD).

[4]We will not discuss any algorithms for obtaining symmetric permutations as to our knowledge the same performance can be achieved with orderings presented. We refer the interested reader to [6].

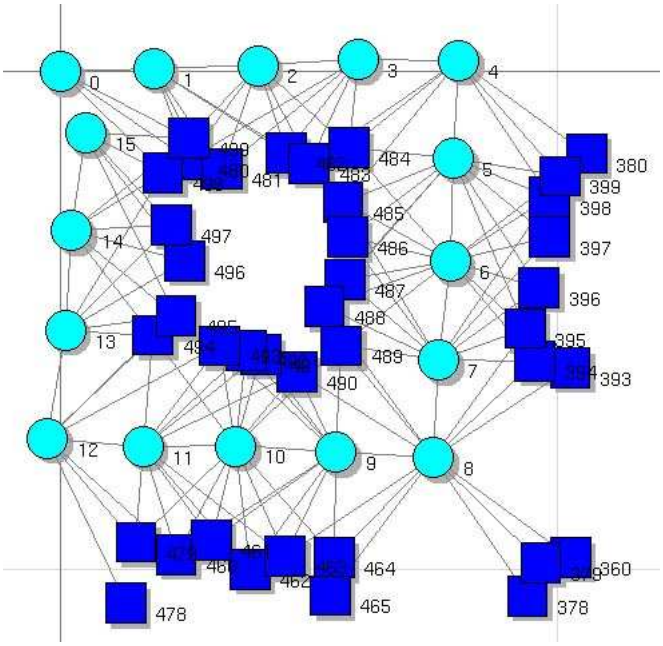
A widely used approximation of the original MD heuristic is to eliminate multiple variables in one call of the elimination function, (MMD). Thus if c is the lowest node degree in G, all nodes with degree c will be eliminated in the next step. In addition *indistinguishable* nodes are eliminated. These nodes do not introduce an additional set of dependencies as they are subsumed by the set produced by another elimination that will be performed in the same step. MMD saves time on updating the graph and determining the next elimination candidates [27].

Another approach is to save time on the computation of the exact degrees when eliminating one or more variables. This is done by collecting nodes into cliques so that the bookkeeping does not need to be done for each node separately but is rather approximated. Therefore this algorithm is known as *Approximate Minimum Degree* (AMD) method[1, 6]. AMD is a widely used ordering. For example this reordering technique is used in Matlab's *colamd* command. In comparison to MMD this method is supposed to be faster though delivering competitive orderings. To illustrate the effect of these standard orderings the following series of figures shows an example.

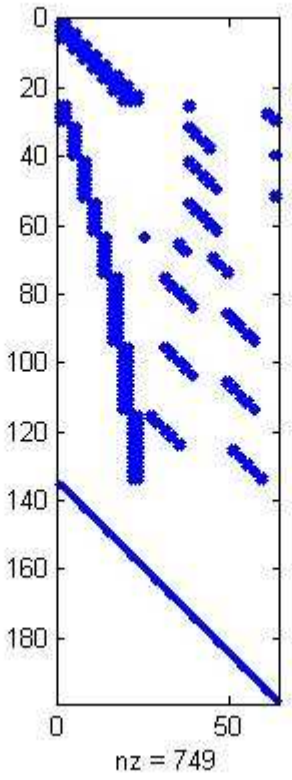
The example scenario consists of a robot walking around a block (Fig. 2(a)) and sensing landmarks. For simplicity we used only the first 8 poses of this walk for the further example. Fig. 2(b) shows the measurement matrix and Fig. 2(c) a straightened up graph of Fig. 2 (a). The NNZ of R for various orderings are given in Fig. 3. In this figure a *column-ordering* means that every single column and row of A is considered independent of all others. In a *block-ordering* the blocks formed by the Jacobians are reduced to a symbolic unit block. As already described in [8] it makes a huge difference in the NNZ whether AMD is applied on the symbolic block structure of A or if every column considered standalone. Apparently AMD does not perform overly well on the *column-ordered* A. In contrast MMD works better on A. We attribute this to the fact that the multiple elimination of indistinguishable nodes works quite well for matrices that have a block structure. Fig. 3 (f) shows that it is possible to obtain even better orderings. Only a few manually chosen permutations of AMD ordering were necessary to decrease the NNZ in R. Summarizing we can confirm the results in [8] and it can be said that already the standard orderings work very well on the measurement structure underlying the measurement matrix. For the rest of this paper we will only work on the symbolic block structure of A.

3.3 Nested Dissection orderings

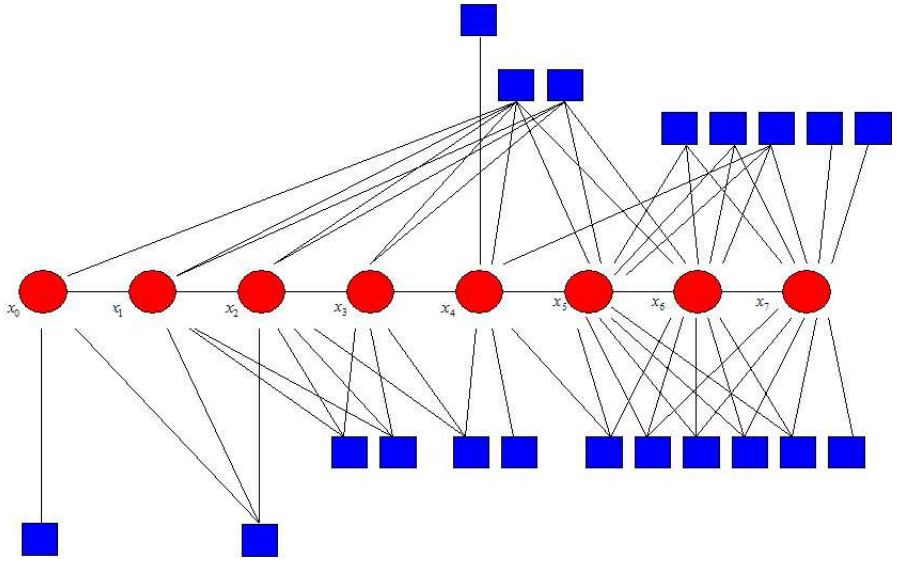
All ordering techniques presented in Section 3.2 try to find an ordering by repeatedly searching the whole graph for the next elimination candidate. A fundamentally different approach is to apply the *divide-and-conquer* paradigm. This is feasible as most of the nodes will only induce new constraints to a set of originally direct neighbors. Nested Dissection orderings try to exploit this fact by recursively partitioning the graph and returning a *post-fix* notation of the partitioning tree as the ordering. This makes Nested Dissection orderings a natural choice for exploiting locality in graphs. *Algorithm 4* shows how the basic Nested Dissection works.



(a) Simulated walk with 15 poses

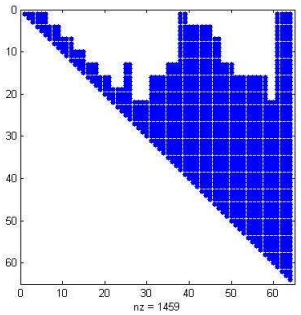


(b) Measurement Matrix for the first 8 poses in (a)

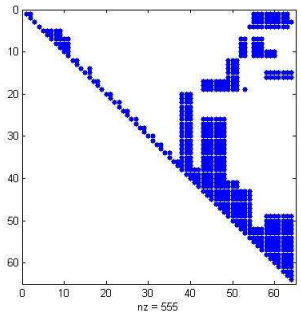


(c) Straightened subgraph of the first 8 poses of (a)

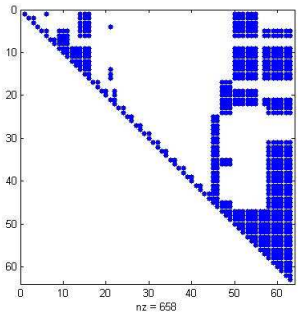
Figure 2: Example Scenario



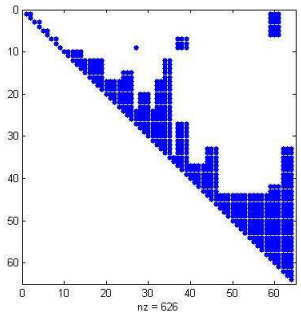
(a) No Ordering - 1459 NNZ



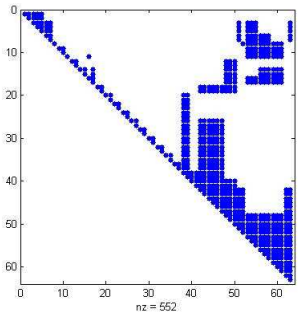
(b) Column MMD - 555 NZ



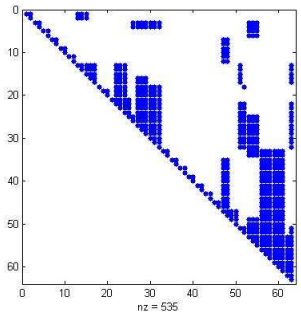
(c) Block MMD - 658 NNZ



(d) Column AMD - 626

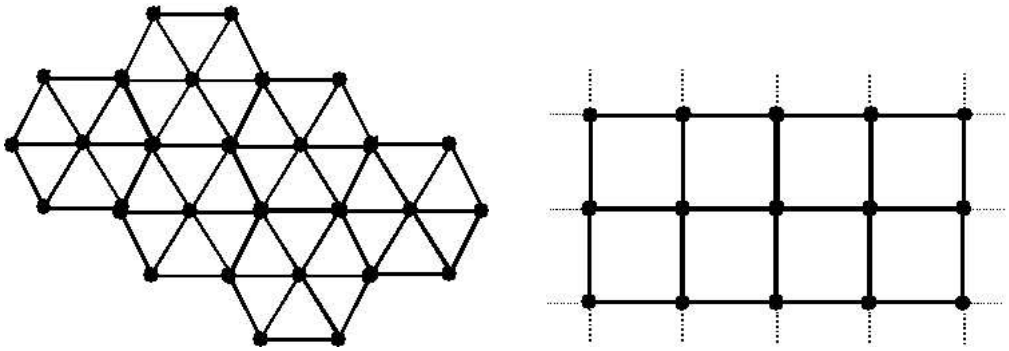


(e) Block AMD - 552 NNZ



(f) Hand Tweaked - 535 NNZ

Figure 3: Column permutations and their effect on the NNZ in the R triangle



(a) (b)

Figure 4: Planar Graph and Finite Element Mesh

Algorithm 4 Nested Dissection

Let $G = (V, E)$ be a graph, with a set of vertices V and a set of edges E and$|G| = n$.

(1) Partition G into subgraphs A, B and C, with $|A|, |B| \leq \frac{2}{3}n$ and $|C| \leq 2\sqrt{2}\sqrt{n}$

(2) Repeat *Step (1)* until $|A|, |B| \leq \epsilon$ or $|A|, |B| = 1$

Obtain ordering by putting the binary tree of the recursive partitioning in post-order, with the nodes of the separating set C last for every triple of sets.

The *divide-and-conquer* approach will only be efficient if the partition of the graph and the elimination in the local subgraphs can be performed without adding much computation. There is a lot of literature about balanced graph partitioning. This is the case especially for planar graphs or finite element graphs (Fig.4) . These are used in physics and mechanics for large scale simulations. [26] may serve as a good introduction to graph partitioning and delivers fundamental properties of graph partitions. Improvements to this $\sqrt{n}/f(n)$ *separator theorem* mainly focus on finding the separating subgraphs more efficiently. Approaches to this include spectral analysis of the adjacency structure [32], using partially applied minimum degree orderings as indicators for good partitions [28], pyramidal coarsening and refining techniques and non-recursive k-way partitioning methods [21]. Other variants will stop the recursion at a certain level a coarseness and then order the vertices in the graph arbitrarily or apply one of the standard ordering algorithm described in Section 3.2 on the larger subgraphs.

$k-$way partitioning is popular as one might save time on the recursion as well as partitions that might otherwise not influence each other in the k-way partitioning process interact as they are not hidden in different parts of the recursion. Improving the efficiency of the partitioning of non-spectral methods is usually done by coarsening the graph down to a certain number of nodes while preserving the topological features. This allows for an efficient search of a near optimal graph partitioning. This partition of the coarse graph is then refined

in a multilevel un-coarsening process of the shrunk graph.

In all Nested Dissection algorithms the separating subgraphs (separator) are of central importance. As is obvious in the light of how the fill-in comes into existence in the elimination game the size of the separator is crucial to keeping the NNZ low. The aim is to maximize the number of nodes that are mutually independent. This can only be achieved with small separators. Therefore most algorithms mentioned so far will consist of a two step approach of determining the separators in the graph. In the first step the algorithms will try to find good areas for a cut that preserve the balance which is very important in the context of parallelizing the calculations on several computers or robots. In a second step a refinement algorithm like [22] or [13] will be applied. These algorithms can be understood as optimized variants of bipartite graph matching algorithms as they try to find the minimal cut between a set of border nodes of the so far determined subgraphs.

3.4 Comparison of AMD and Nested Dissections orderings for SAM

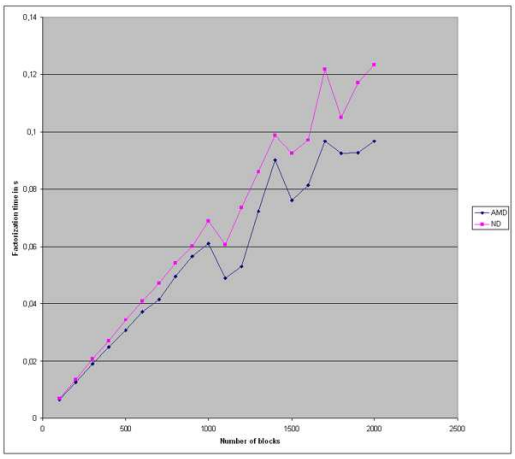
According to the ordering literature AMD and Nested Dissection are the most widely used reordering techniques [21, 28, 27, 1]. In this section we compare the performance of the AMD implementation of [6, 5] to the Nested Dissection implementation of [21] on measurement matrices of the kind presented in 2. The measurement matrices were produced using simulated block-worlds. The result of a walk around one block can be seen in Fig. 2 (a).

In Fig. 5 (a) we can see the results of a walk straight down a hallway of blocks to the left and right. At each step the robot has sensed between 8 and 12 features. As the robot needs 4 steps to pass a block this means that if we walked down a 1,000 blocks we will obtain a measurement matrix with about 4,000 rows and about 40,000 non zero entries. Fig. 5 (b) and (c) shows the results for walks in a square world of blocks. The number of blocks denote the length in blocks of one side of the square. The difference between the two sub-figures is that in Fig. 5 (b) the robot walked pass the given number of blocks. In the lower sub-figure the number of blocks the robot passed are square to the number of blocks forming one side of the square. Thus we take into consideration the square gain in the number of blocks.

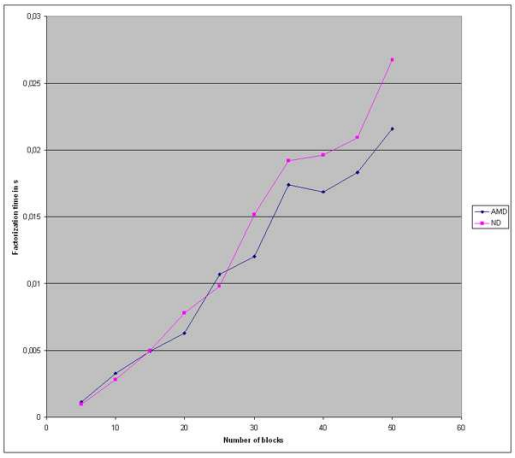
Note the scale of the factorization times. For the straight walk in Fig. 5 (a) and for the random walk in Fig. 5 (b) are roughly the same. Whereas in Fig. 5 (c) the factorization time is one order of magnitude bigger. The reason for this increase is on the one hand the length of the walk and on the other hand the density of the measurement graph. The latter aspect will be described in more detail in the following sections. Nevertheless the main result of this chapter is that for the SAM measurement matrices up to the given maximum size and density the orderings work equally well.

4 Computational Complexity

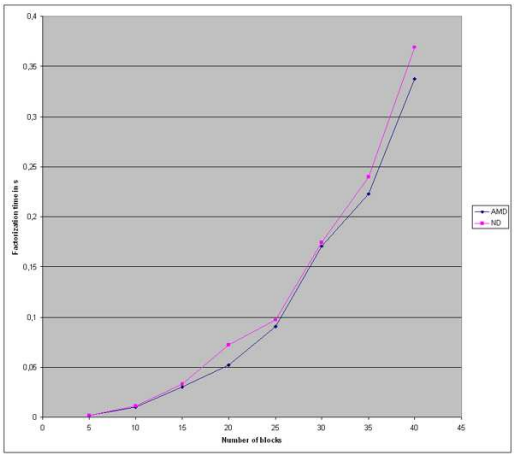
In this section we will present the complexity bounds already known for Nested Dissection orderings and show how they can be used to derive the computational complexity of a factorization of the measurement matrix A of the SAM problem. At first we will introduce complexity bounds for planar graphs and general classes of graphs. We will then show how graph partitions can be computed and how the size of the separators is bound by typical measurement patterns. For these typical measurement pattern we will derive complexity



(a) Straight walk with blocks to the left and right



(b) Random walk through a square block world



(c) Long Random walk through a square block world

Figure 5: Factorization times over number blocks. In the case of (b) and (c) the number of blocks denotes the number of blocks of one side of the square block world.

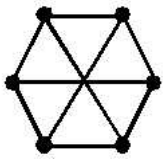
bounds. Finally we will show how this is applicable in a standard indoor scenario.

4.1 Properties of Nested Dissection orderings

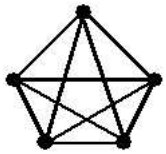
As described in the last section, the Nested Dissection algorithm is based on the ability to recursively partition graphs into subgraphs of roughly equal size with a very small separating subgraph efficiently. The fundamental work about Nested Dissection orderings [25] is based on the so-called *separator theorem* [26] and the $f(n)$-*separator theorem*.

SEPARATOR THEOREM: *Let G be any $n-$ planar vertex graph. The vertices of G can be partitioned into three sets, A, B, C such that no edge joins a vertex in A with a vertex in B, neither do A nor B contain more than $\frac{2}{3}n$ vertices and C contains no more than $\sqrt{2}\sqrt{n}$ vertices. This partition can be found in $O(n)$ time.*

The Separator theorem is a very restrictive statement as it only holds for planar and finite element graphs. Typical planar graphs or finite element graphs are depicted in Fig. 4. This class of graphs and meshes is found frequently in scientific simulations [21]. A criterion for the planarity of a graph is given with the Kuratowski Theorem.



(a) A complete bipartite graph of two sets of three vertices



(b) Kuratowski graph

Figure 6: It is sufficient for a graph to have one of the above graphs as a subgraph or being reducible to one of these, to show non-planarity and vice versa.

KURATOWSKI THEOREM: *A graph is planar if and only if it contains neither a complete bipartite graph on two sets of three vertices, Fig.* 6 *(a), nor a complete graph on five vertices, Fig.* 6 *(b).*

A general separator theorem for a given class of graphs S is the $f(n)$- *separator theorem*.

$f(n)$-SEPARATOR THEOREM: *A graph is $f(n)$- separable if there exist constants α, β with $\alpha < 1, \beta > 0$ such that if G is any $n-$vertex graph S, the vertices of G can be partitioned into three sets A, B, C such that no edge joins a vertex in A with a vertex in B, neither A nor B contains more than αn vertices and C contains no more than $\beta f(n)$ vertices.*

For all classes of graphs for which a $f(n)$-theorem holds and a partition of such a kind can be found fast an efficient *divide-and-conquer* ordering is possible [26]. This is espe-

cially the case if the graphs are planar and $f(n) = \sqrt{n}, \alpha = \frac{2}{3}$ and $\beta = \sqrt{2}$. In [25] the $\sqrt{n}$-separator theorem was used to prove computational bounds for the complexity of matrix factorizations of sparse matrices whose matrix structure equals a planar graph.

PLANAR NESTED DISSECTION THEOREM: *Let G be any planar graph. Then G has an elimination ordering which produces a fill-in of size $c_1 n \log n + O(n)$ and a multiplication count of $c_2 n^{3/2} + O(n\,(\log n)^2)$, where $c_1 \leq 129$ and $c_2 \leq 4002$. Such an ordering can be found in $O(n \log n)$ time.*

For a wider class of graphs exchanging the $\sqrt{n}$- theorem for the planar graph with the general $f(n)-$ theorem yields the following complexity bounds [25].

RELAXED NESTED DISSECTION THEOREM 1: *Let S be any class of graphs closed under the subgraph criterion on which an n^σ separator theorem holds for $\sigma > \frac{1}{2}$. Then for any $n-$vertex graph G in S, there is an elimination ordering with $O(n^{2\sigma})$ fill-in size and $O(n^{3\sigma})$ multiplication count.*

Depending on which statements we can make about $f(n)$ the resulting complexity bounds for the factorization will be weaker or tighter. It is also assumed that a partitioning can be found in $O(n^\eta)$ with $\eta \leq f(n)$. This means that the complexity for finding the partition ought not to be higher than the complexity for the factorization.

This means that in order to apply a Nested Dissection Theorem for a certain class of graphs we need to show that the graph can be partitioned according to a *separator theorem* and then show that obtaining the partition is a less complex process than computing the ordering.

4.2 Complexity Bounds for Factorizing the Measurement Matrix

The factorization of a dense matrix has a computational cost of $O(n^3)$ when using Cholesky factorization. The factorization of a sparser matrix requires less computational effort. Yet determining complexity bounds for sparse matrices is difficult. As we have seen in Section 2 the measurement matrix A is quite sparse due to the way it is constructed. The last section shows that knowledge of the structure of the matrix can be used to obtain tighter bounds for the complexity of sparse matrix factorizations. In this section we will show how we can apply the *separator theorems* with our measurement matrix A and give tighter complexity bounds for typical measurement patterns.

Let G be a measurement graph, let the number of poses be x and of the features be l. If S is a separator of G and x_S the number of poses and l_S the number of features contained in S. We require the following assumptions to hold.

1. A sensor has a bounded range.
2. For every partition with S the following holds $\frac{x_S}{l_S} = \frac{x}{l} = r$.
3. The number of poses d seen from each landmark is even over the whole graph.
4. For every partition the most distant poses of a connected trajectory are given.

The example in Fig. 7 shows that in the case that a robot uses a forward looking sensor only and that the robot never walks along an area twice the measurement graph will be planar.

This assumes that the forward looking sensor is bound in range such that from pose x_{i-1} it will at maximum sense the features that x_i will sense that are closest to x_{i-1}.

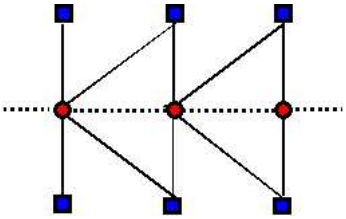


Figure 7: Measurement graph for a robot walking three steps and sensing landmarks to both sides. The circles depict poses and the squares features.

Nevertheless in the general case measurement scenarios do not correspond to planar measurement graphs. Fig. 8 depicts a small measurement scenario for a not tightly range restricted omni-directional sensor.

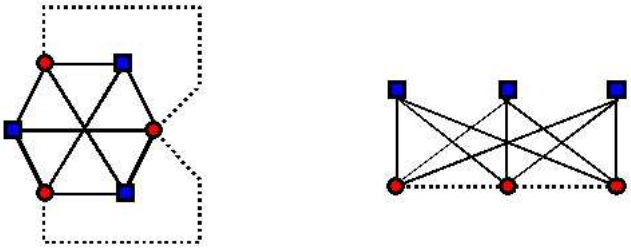


Figure 8: Example of a non-planar measurement graph - the left graph shows the graph Fig. 16(a) with squares denoting features l and circles denoting robot poses x. The original edges represent only x to l measurements. The dotted lines that were added represent odometry measurements between the poses. The right graph is isomorph to (a) but laid out to show the trajectory. From each pose three features are sensed.

The denser the set of measurement will become the *less planar* our graph will be. In the rest of this section we will show that it is possible to obtain graph partitions efficiently for typical indoor measurement scenarios. Thus under reasonable assumptions we can provide a *partitioning algorithm* comparable to the algorithm in [26] allowing us to apply the $f(n)$-separator theorem. We will now give upper bounds for the cardinality of separators of typical measurement graphs.

SEPARATOR SIZE FOR ONE-WAY WALK: *Let G be the measurement graph of a robot sensing omni-directionally with an even radius of its sensor and bounded in range. If a robot walks straight and never returns to an already visited environment it is possible to find a partition of the graph into subgraphs A, B, C such that C contains no more than $2d + 1$ vertices.*

To prove the above statement consider a straight walk of a robot as in Fig. *9.* In worst case the robot senses in all directions and detects features to the both sides. If we want to

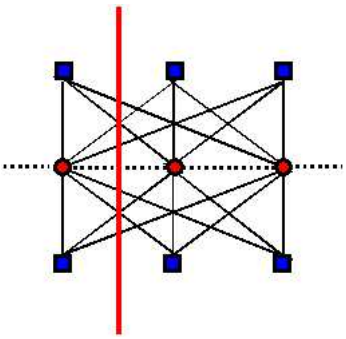


Figure 9: Edge cut of a measurement graph with average landmark node degree of 3.

partition the graph by simply placing one cut as shown in the Fig. *9* we cut on average $2d$ of the $x - l$ measurement edges and one odometry edge. Making this edge separator a vertex separator means finding all nodes on one side of the cut who have one edge cut.

SEPARATOR SIZE FOR k-WAY WALK: *Let G be the measurement graph of a robot sensing omni-directionally with an even radius of its sensor and bounded in range. If a robot walks straight and returns k- times to an already visited environment it is possible to find a partition of the graph into subgraphs A, B, C such that C contains no more than $(2d + 1)k$ vertices .*

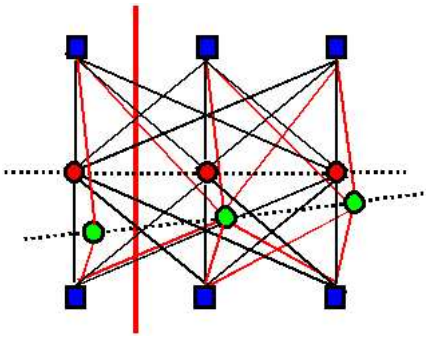


Figure 10: Edge cut of a measurement graph with average landmark node degree of 3 where the robot saw the environment twice.

The argument for the k-Way statement is analogue to the One-Way statement but we now have to find the separator over k-trajectories as shown in Fig. *10.*

SEPARATOR SIZE FOR k-CROSSING: *Let G be the measurement graph of a robot sensing omni-directionally with an even radius of its sensor and bounded in range. If a robot walks k-times over a crossing such that features are seen in common it is possible to find a partition of the graph into subgraphs A, B, C such that C contains no more than $(2d + 1)k$ vertices .*

Consider two straight walks crossing. As the sensor is bounded in range we know that

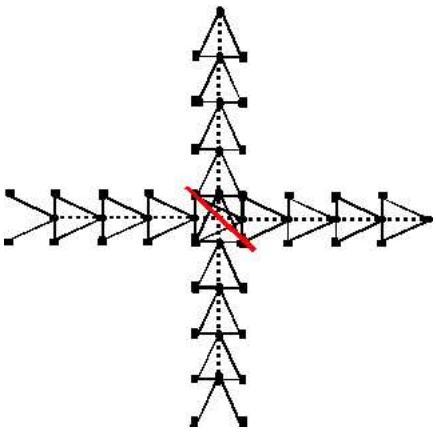


Figure 11: Edge cut of a measurement graph with a crossing that is visited k-times and features are seen in common.

on average at the crossing point the robot of the first walk will have seen $2d$ features. To cut this part of the robot trajectory we add one vertex. From the crossing c features have been seen from as much as 2 poses of the crossing trajectory. Whereas $1 \leq c \leq 4d$. Note that these features need not be identical with the already observed features as features usually are not equally good visible from all sides. Again we cut the robot trajectory and therefore add a vertex. Thus we now have a separator size of $(2d+1)(4d+1) \leq (4d+1)2$. For every further crossing in this point we can assume that at maximum another $4d+1$ vertices have to be added to the separator yielding a size of $(4d+1)k$ after k crossings.

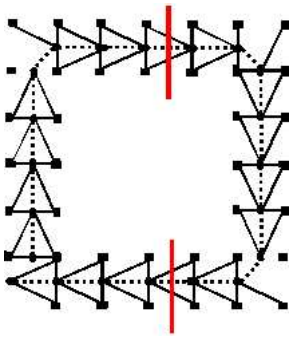


Figure 12: Edge cut of a measurement graph with average landmark node degree of 2 where the robot walks a loop .

SEPARATOR SIZE FOR ONE-k-LOOP: *Let G be the measurement graph of a robot sensing omni-directionally with an even radius of its sensor and bounded in range. If a robot walks a k-times in a loop it is possible to find a partition of the graph into subgraphs A, B, C such that C contains no more than $(2d+1)2k$ vertices .*

The statement follows direct from the separator size for a k-way walk. An example is depicted in Fig. 12. The general case where m loops have to be cut is given as follows.

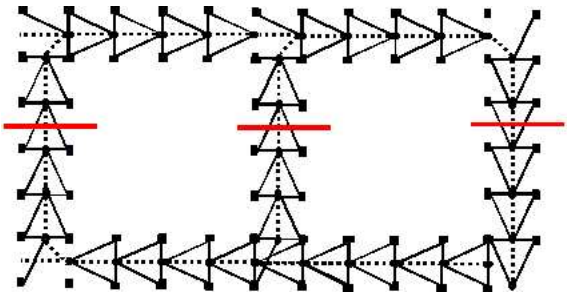


Figure 13: Edge cut of a measurement graph with average landmark node degree of 2 where the robot walks two loops.

SEPARATOR SIZE FOR M-LOOPS: *Let G be the measurement graph of a robot sensing omni-directionally with an even radius of its sensor and bounded in range. If a robot walks in m loops where the robot maximally sees an environment k-times it is possible to find a partition of the graph into subgraphs A, B, C such that C contains no more than $(2d+1)k(m+1)$ vertices .*

This statement is a result of the last statements. We want to draw the readers attention to one very useful fact about separator sizes in conjunction with *separator theorems.* The theorems only demand that the two large subgraphs have a size of αn with $\alpha < 1$. In case $\frac{x_S}{l_S}$ is high for less than $(1-\alpha)$ nodes we might shift the cut without violation of the balance restriction and obtain a smaller separator. But note that we might pay for this with a higher α constant which results in higher constants for the complexity bounds.

So far we have shown upper bounds for the separator sizes of several standard measurement patterns. What remains to be shown is that one can find these separators in a reasonable number of steps. For all measurement patterns except for the *M-loops* we can find roughly the point of the cut as follows. Let us assume c_S is the size of the separator for G. Using Assumption 4 and Assumption 2 we might choose one of the endpoints of G and then start following the trajectory numbering the poses and connected features until we reach the pose number closest to $\alpha\,|G| - c_S = p$. Note that all vertices to be numbered still need to be inside G. In the case that we do not have a connected subgraph we can still pick an endpoint and start numbering. Either we will be able to number until we reach the start pose for the separator or a complete disjoint part of G will be part of this subgraph and we can continue to number from an endpoint of another disjoint part of G. This algorithm has similarities to *region growing algorithms* and k-way partitioning [21]. Indeed for the *M-loops* we need to apply a more sophisticated algorithm for determining the poses x_p. These poses can be found by coarsening the graph as described in [21], then determining the cutting points easily and then refining the coarse graph. As shown in the separator size statements above such a c_S can be determined. Nevertheless we still need to show that the separator itself can be determined exactly. Considering the *one-way-walk* we can determine the partition starting with pose x_p. The separator will then consist of x_p and all features l_i that are seen from x_p. In regard to the *k-way-walk* we extend the separator constructed for the *one-way-walk* to contain the last numbered poses x_k of the k tracks that are parallel.

We then add all features l_k that are seen from each x_k. In regard to a $k - crossing$ we can take the separator of the *k-way-walk* and add all the poses $x_c$that are adjacent to features already in the separator. In addition we add the roughly $4dx_c$ features that are adjacent to the newly added poses. In case of a *k-loop* we can start at a random pose in the loop and then proceed as if it was a *k-way-walk*. In regard to the *M-loops* it is obvious that we can find the separator just as we did with the *k-way-walk* once the set of the poses x_p is known.

In the last paragraph we have shown that it is possible to determine the separators in a time linear to the number of vertices of G for the measurement patterns given. We believe that there are more sophisticated ways of finding these partitions. Our experience is that the software package based on [21] delivers partitions that are suitable for our purposes. Given that we can obtain partitions in linear time we still need to determine reasonable α and β for the $f(n)-$separator theorem. Let G be a measurement graph and S the maximal separator of G and $c_S = |S|$. Then for every $f(n) = n^\sigma$ in the sense of the $f(n)-$separator theorem, $\beta = c_S^\sigma$ can be calculated as c_S is constant. Given all these assumptions we can make the following statements about computational complexity. For planar graphs the next statement follows directly from the $\sqrt{n}-$separator theorem.

PLANAR MEASUREMENT GRAPH BOUND: *Given that the assumption holds that the measurement graph is planar the bounds given in the Planar Nested Dissection Theorems can be obtained. This means that we can factorize the measurement matrix of the measurement graph with a fill-in of size* $c_1 n \log n + O(n)$ *and* $c_2 O(n^{\frac{3}{2}})$ *multiplications where* $c_1 \leq 129$ *and* $c_2 \leq 4002$*.*

CONSTANT MAXIMAL SEPARATOR THEOREM: *Let* G *be the measurement graph of a robot sensing omni-directionally with an even radius of its sensor and bound in range. If the measurement graph resulting from the robot exploration can be separated by a subgraph of size smaller than a constant* c_S *there exist constants* α, β *with* $\alpha < 1, \beta > 0$ *such that if* G *is any* $n-$*vertex graph* S*, the vertices of* G *can be partitioned into three sets* A, B, C *such that no edge joins a vertex in* A *with a vertex in* B*, neither* A *nor* B *contains more than* αn *vertices and* C *contains no more than* $\beta f(n)$ *vertices.*

CONSTANT MAXIMAL SEPARATOR BOUND: *Let* G *be the measurement graph of a robot sensing omni-directionally with an even radius of its sensor and bound in range. Given that* α, β *can be found in the sense of the Constant Maximal Separator Theorem the* $total\ fill - in$ *associated with an ordering produced Alg. 4 on* G *is at most* $c_1 n\, log_2 n + O(n)$ *and the multiplication count is at most* $c_2 n^{\frac{3}{2}} + O(n\,(\log n)^2)$ *where*

$$c_1 = \beta^2(\frac{1}{2} + 2\sqrt{\alpha}/(1 - \sqrt{\alpha}))/log_2(1/\alpha)$$

and

$$c_2 = \beta^2(\frac{1}{6} + \beta\sqrt{\alpha}(2 + \sqrt{\alpha}/(1 + \sqrt{\alpha}) + 4\alpha/(1 - \alpha))/(1 - \sqrt{\alpha}))/(1 - \delta)$$

with

$$\delta = \alpha^{\frac{3}{2}} + (1-\alpha)^{\frac{3}{2}}$$

The last two statements show that for example under the assumptions made in this section for a k-way walk in a hallway we can obtain a

$$\beta^{\frac{1}{2}} = maximal\ separator\ size = (2d+1)k)^{\frac{1}{2}}. \tag{3}$$

For $\alpha = \frac{2}{3}n$ and $f(n) = n^{\frac{1}{2}}$ we than obtain the same bounds except for the constants $c_1$and c_2 as if the graph was planar. The change in the constants is of course dramatic for small n.

In this section we have shown that under reasonable assumptions we can obtain tighter complexity bounds for the factorization of a measurement matrix that adheres to typical measurement patterns. These results of course only hold for the measurement patterns presented here. In the following section we use a typical indoor scenario to demonstrate the partitioning process and show the impact of the changed constants.

4.3 Indoor Scenario

To show the performance of the Nested Dissection we apply it in this section to an indoor grid world. This problem is challenging as there are no "natural" partitions like rooms to the sides of a hallway. These might save cuts or keep the cuts small.

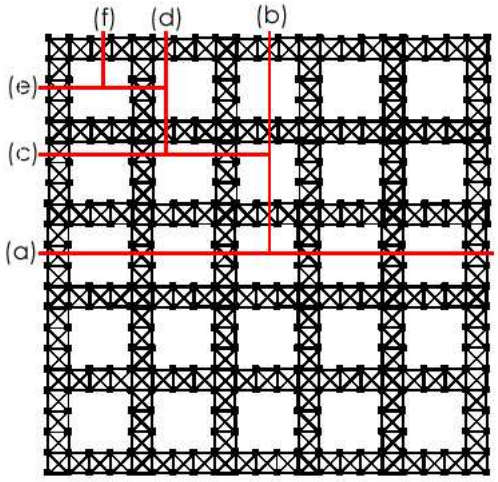


Figure 14: Grid World Example

Our grid world and possible cuts in case of a valid the even-degree-assumption are shown in Fig. 14. The cuts correspond to the cuts of the first Depth-First recursion. For a

	n	$\sqrt{n}$	$\lvert A\rvert, \lvert B\rvert$	$\lvert C\rvert$	$\beta\sqrt{n}$
(a)	25,000	158.1	12446	108	1643.1
(b)	12446	111.5	6196	54	1159.3
(c)	6196	78.7	3071	54	818.0
(d)	3071	55.4	1518	36	575.9
(e)	1518	38.9	741	36	404.8
(f)	741	27.2	362	18	282.8
(g)	362	19.0	172	18	197.5
(h)	172	13.1	82	9	136.2
(i)	82	9.0	36	9	94.1
(j)	36	6	14	9	62.3
(k)	14	3.6	3	9	38.1
(l)	3	1.7	stop	9	18

Table 1: The level of the recursion, number of nodes n in the each partition, $\sqrt{n}$,the size of the next partitions and the separator.

grid with 25,000 nodes the partitions in Tab. 1 would be achieved. Where possible we divide the graph into two equal pieces. If this is not possible we choose the bigger partition in the recursion. Note that the measurement graph contains only parts where the robot passed along two times at maximum. A feature is seen from 4 poses on average. For $\sigma = \frac{1}{2}, \alpha = \frac{2}{3}$ we determine β following Eq. 3 as $\beta = \sqrt{(2d+1)k(m+1)} = 10.3$. The results in Tab. 1 show that it is clearly possible to dissect a grid graph recursively into small pieces which can be easily solved. Note that after the eighth partition all nodes become one subgraph and the recursion stops.

4.4 Constants in the Complexity Bounds

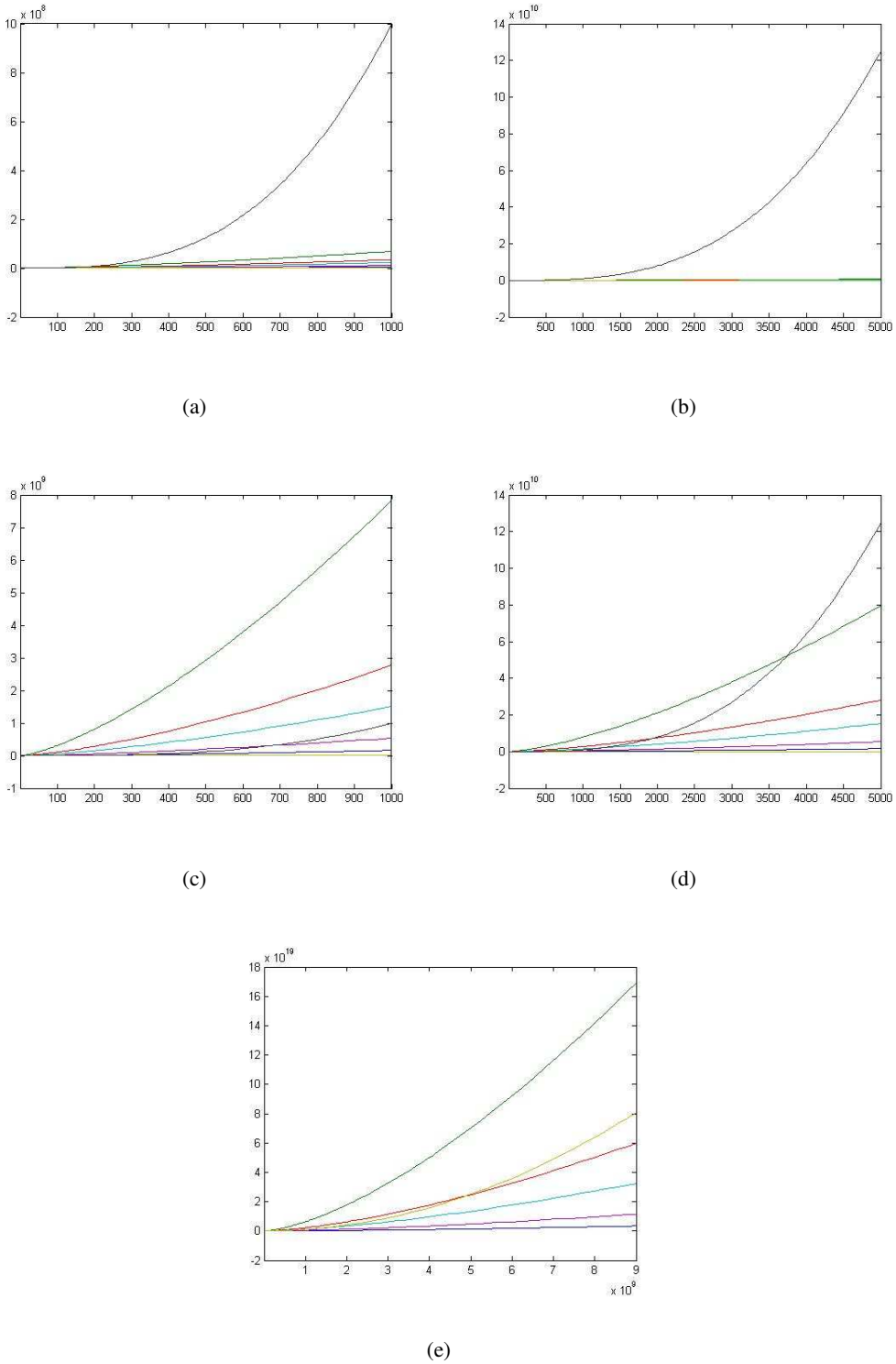
In this section we want to highlight that the constants in the *Constant Maximal Separator Bound* given in Section 4.2 are not negligible.

β	c_1	c_2
$\sqrt{18}$	289.2	13500.9
$\sqrt{36}$	578.4	38177.0
$\sqrt{54}$	867.5	70127.9
$\sqrt{108}$	1753.3	198323.0

Table 2: Values for $c_1$and c_2 of the *Constant Maximal Separator Bound* for varying values of β.

Note that these numbers are quite realistic. For example for the indoor scenario of the last section $\beta = \sqrt{(2d+1)k(m+1)} = 10.3 = \sqrt{108}$, thus $c_1 = 1753.3$ and $c_2 = 198323$.

Fig. 15 shows the extent to which the constants influence the complexity bounds. The



(a) (b)

(c) (d)

(e)

Figure 15: The top two figures show the fill-in for the planar case, using a β^2 of 18, 36, 54 and 108 as well as $n^3$and n^2. The middle two figures show the multiplication count for the same β values. The top and middle figures show $0 \leq n \leq 1000$ versus $0 \leq n \leq 5000$. The bottom figure shows the multiplication count for the same $beta$ values but for $0 \leq n \leq 9000,000,000$.

fill-in and the multiplication count roughly behave alike. We want to focus on the middle and bottom figures, the figures of the multiplication count. In all figures we have additionally plotted the $n^3$and n^2. It is obvious that except for $n \leq 3500$ the complexity bounds derived with the *Constant Maximal Separator Bound* are tighter than n^3. Nevertheless Fig. 15 (e) shows that it will take quite either large n or even tighter approximations of the separator size to achieve a complexity lower than n^2 for the sparse matrix factorization with the given bounds.

5 Submap Motivation

The application of SRIS is a fairly new approach to the mapping problem. The most widely used approach so far is based on using Extended-Kalman-Filters (EKF) or its dual the Extended-Information-Filter (EIF). Most of the EKF literature is based on [33] and [34]. Whereas [2] can be seen as the fundamental work on EIFs. In this section we will show the fundamental problems of both approaches and how extensions exploit the locality inherent in the mapping problem. In addition we will show how Nested Dissection orderings can be understood in terms of submaps.

An EKF represents the environment in form of a covariance matrix which becomes denser each time new observations are made. A problem of the EKF is that it describes new features through their relation to other already sensed features. Over time the number of unknowns used to express a new unknown becomes large. Thus an update of the matrix becomes computationally infeasible [8]. The EIF which is the EKFs dual system represents the environment by an information matrix which is the inverse of the covariance matrix. The information matrix has the nice property that any non-zero value in the matrix indicates a strong dependency between two features and thus this matrix is relatively sparse. Whereas the matrix will still become dense over time.

There have been a variety of approaches aimed at reducing the cost of an update in the EKF by exploiting the locality in the mapping problem through dividing the environment or map into several disjoint submaps during the filtering process [23, 31]. The aim is to calculate an update of the environment covariance matrix with constant computational cost as one only updates a small, neighboring piece of the environment. The submaps approach has been well studied and is very promising [3]. An interesting constant time implementation is described in [24]. In the rest of this section we will disregard all the problems that arise when working with submaps such as how to detect a loop when re-entering an already existing submap, whether it is better to keep a global frame of reference or several interconnected frames, how to fuse local maps into a global map or how an information increase / uncertainty decrease in one submap will be propagated to adjacent submaps thus improving the quality of the overall map [3, 4, 12, 24]. We focus on how to create the submap partitions of the environment and what properties these submaps have in contrast to a Nested Dissection partitioning.

First of all it dividing an environment into submaps is a local on-line process whereas the Nested Dissection partition is a global offline process. This means that in the optimal case submaps can be understood as an on-line Nested Dissection partitioning of the map. Optimal means that the submaps partition the measurements evenly and that the separator structure of the cuts between submaps is minimal in the graph partitioning sense. Note

that if all measurements are considered equally likely we thereby minimize the conditional independence of the submaps amongst themselves. Of course these two conditions can be relaxed by allowing a deviation by a small constant in the number of nodes on the cut and in the submaps. Note that it is a prerequisite for achieving a constant time EKF algorithm that the submaps are of roughly equal size [24]. Nested Dissection partitions provide balanced partitions with small separating structure due to their construction [26, 25]. Reciprocally to our derivation above we can understand that a Nested Dissection ordering as an ordering in which we work on the lowest level on submaps and then merge the local results hierarchically into one global map. Nevertheless there is a difference between the lowest level of partitioning in Nested Dissection and the submaps as such. In the Nested Dissection algorithm the size of the smallest partition is determined by the computation cost of factorizing this partition [25]. Only in theory every subgraph down to the single vertex level would be partitioned. In contrast the submaps are designed to have exactly equal size. We want to highlight that it is known that small submaps are preferable over large submaps not only due to computational issues but also as the error in the odometry measurements of the robot grows over time starting a new submap resets the error to zero or gives it less influence on later measurements. In addition there are results that lead to the conclusion that submaps might provide better maps if they were aligned and cut according to environment they work on [19, 15]. As proposed in [19, 15] for indoor environments it may make sense to let whole rooms be one submap so that only the hallway needs to be partitioned. This definitely would have the advantage that for these rooms no submaps would need to be connected and no information would need to be propagated through several submaps. This would mean that the solution for a room will be independent of the rest of the map given the separator. This would imply that we can solve smaller submaps and then fuse them on higher level. Hierarchically computing results is well studied in the scientific computation community [18, 30] as it is a fundamental problem for parallelizing calculations and to the SLAM community [16, 9]. The locality inherent in the mapping problem is the basis for every hierarchical mapping algorithm.

As explained earlier in this paragraph the EIF is dual to the EKF and represents the world in the form of an inverse of the covariance matrix, the information matrix [35]. The non-zeroes in the information matrix represent the information of the features relative to each other. Therefore the topological relations between the features is quite obvious from the entries in the information matrix. There is a strong relation between features if they are geometrically close. This influence decays rapidly with increasing distance. In [29, 10, 11] it has been shown that a lot of entries with a value that tends to be zero are negligible and hardly contribute to the quality of the map. Thus it is claimed that the position of a feature or the robot can be calculated solely by the elements directly connected with it which form the so called Markov blanket. Following the approach in [29, 10, 11] newly added features are made independent from already existing features in the map that are not contained in the Markov blanket of the new feature, thus the information matrix becomes sparser. This of course greatly reduces the run time, creating what is called a Sparse EIF (SEIF). Again, the sparse matrix can be interpreted as graph. Note that if all the *inter* feature edges in this information graph are removed and the filtered relations between the poses and the features were added the information graph would be isomorph to the measurement graph introduced in [8]. We understand that the sparsification process can be seen as a sliding submap.

We think that the Nested Dissection partitioning could be improved by using one of the ideas behind the SEIF. We might obtain better separator structures if we not only use the

separator size as a criterion but find a cut that will maximize the *intra* submap information and minimize the *inter* submap information. Whether this separator refinement is applicable will of course depend on the size of the partitions.

After exploring the similarities between EKF submaps and Nested Dissection it is obvious to ask whether Nested Dissection or graph partitioning in general are able to show us how to construct optimal computational subdivisions for the mapping problem. We think that this is possible only to a limited degree. The reason is that the recursion of the partitioning process in Nested Dissection will always stop at a certain level or will continue until there is only one vertex in every subgraph. Thus either the partition is determined by the complexity of the factorization or there is no coarser granularity of the computational units than the vertex level. Nevertheless both exploit the locality in the same manor. Whereas the SEIF exploits the locality not by partitioning but by restricting the mutually influence of vertices locally but never splits the measurements up into divisions. Whether the information theoretic background of the SEIF can be used to improve the Nested Dissection partitioning algorithm remains an open question.

6 Conclusion

In this paper we have shown that the smoothing and mapping problem when solved by a SRIS can be understood as a factorization measurement matrix or a manipulation of a graph of measurements. On the dual representations we have shown how an ordering of variables changes the behavior of factorization or respectively the elimination process. Furthermore we have described the properties and complexity bounds on the Nested Dissection algorithm. In brief the contributions of this paper can be summarized in the following enumeration

- For the SAM problem we have shown that the two most prominent reordering techniques yield comparable results. We have explained how the Nested Dissection algorithm exploits the locality inherent in the SAM problem.

- We give tighter computational complexity bounds for factorizing the measurement matrix. We have shown that with certain bounds on the separator sizes a $\sqrt{n} - separator\,theorem$ can be derived. This means that in contrast to the complexity of $O(n^3)$ for a dense matrix factorization we can obtain a complexity bound of $O(n\,log_2 n)$ for the total fill-in and $O(n^{\frac{3}{2}})$ for the multiplication count of the factorization of the pre-ordered matrix . Nevertheless these bounds are linked to high constant factors that need to be taken into consideration when applying the algorithm and give an example for Nested Dissection in a typical indoor scenario.

- We have exposed the relationship between submaps as used with an EKF and Nested Dissection. We believe Nested Dissection algorithm might help understand how efficient computational units can be locally determined. We understand that [15] hints in the same direction.

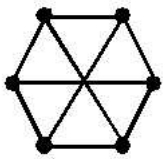
Although the way sparsification is performed in the SEIF we think that a further occupation with this technique might be beneficial in the development of separator refinement techniques. As we have described in Section 5, the covariance matrix represents features by their relation to other features already sensed. It remains to be exposed how this filtering process is related to the Gaussian Elimination process as described in Section 3 and whether Gaussian Elimination can be understood as a subset of filtering in the Markov Random Field or vice versa.

Despite our work there is still only very little known about Nested Dissection of measurement graphs. The aim must be a unified complexity bound for the factorization of general measurement graphs given certain assumptions. We think that such a theorem must be thoroughly examined in regard to how the theoretical bounds reflect the calculation needed for real world scenarios. Several aspects lend themselves to further investigation such as what would be the optimal partition size in regard to performance of the cache hierarchy or how distributed Nested Dissection for swarms of robots exploring an environment can possibly be developed.

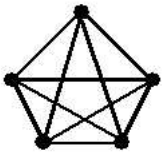
7 Appendix

This is an additional comprehensive summary of the theorems used in this paper and taken from [25, 26].

KURATOWSKI THEOREM: *A graph is planar if and only if it contains neither a complete bipartite graph on two sets of three vertices, Fig.16(a), nor a complete graph on five vertices, Fig.16(b).*



(a) A complete bipartite graph of two sets of three vertices



(b) Kuratowski graph

Figure 16: It is sufficient for a graph to have one of the above graphs as a subgraph or being reducible to one of these, to show non-planarity and vice versa.

SEPARATOR THEOREM: *Let G be any $n-$ planar vertex graph. The vertices of G can be partitioned into three sets, A, B, C such that no edge joins a vertex in A with a vertex in B, neither do A nor B contain more than $\frac{2}{3}n$ vertices and C contains no more than $\sqrt{2}\sqrt{n}$ vertices. This partition can be found in $O(n)$ time.*

$f(n)$-SEPARATOR THEOREM: *A graph is $f(n)$- separable if there exist constants α, β with $\alpha < 1, \beta > 0$ such that if G is any $n-$vertex graph S, the vertices of G can be partitioned into three sets A, B, C such that no edge joins a vertex in A with a vertex in*

B, neither A nor B contains more than αn vertices and C contains no more than $\beta f(n)$ vertices.

PLANAR NESTED DISSECTION THEOREM: *Let G be any planar graph. Then G has an elimination ordering which produces a fill-in of size $c_1 n \log n + O(n)$ and a multiplication count of $c_2 n^{3/2} + O(n (\log n)^2)$, where $c_1 \leq 129$ and $c_2 \leq 4002$. Such an ordering can be found in $O(n \log n)$ time.*

RELAXED NESTED DISSECTION THEOREM 1: *Let S be any class of graphs closed under the subgraph criterion on which an n^σ separator theorem holds for $\sigma > \frac{1}{2}$. Then for any $n-$vertex graph G in S, there is an elimination ordering with $O(n^{2\sigma})$ fill-in size and $O(n^{3\sigma})$ multiplication count.*

TIGHTER NESTED DISSECTION THEOREM 1: *Let S be any class of graphs closed under the subgraph criterion on which an n^σ separator theorem holds for $\frac{1}{3} < \sigma < \frac{1}{2}$. Then for any $n-$vertex graph G in S, there is an elimination ordering with $O(n)$ fill-in size and $O(n^{3\sigma})$ multiplication count.*

TIGHTER NESTED DISSECTION THEOREM 2: *Let S be any class of graphs closed under the subgraph criterion on which a $\sqrt[3]{n}-$separator theorem holds. Then for any $n-$vertex graph G in S, there is an elimination ordering with $O(n)$ fill-in size and $O(n \log_2 n)$ multiplication count.*

TIGHTER NESTED DISSECTION THEOREM 3: *Let S be any class of graphs closed under the subgraph criterion on which a n^σ separator theorem holds for $\sigma < \frac{1}{3}$. Then for any $n-$vertex graph G in S, there is an elimination ordering with $O(n)$ fill-in size and multiplication count.*

References

[1] P. R. Amestoy, T. Davis, and I. S. Duff. An approximate minimum degree ordering algorithm. *SIAM Journal on Matrix Analysis and Applications*, 17(4):886–905, 1996.

[2] G.J. Bierman. *Factorization methods for discrete sequential estimation*, volume 128 of *Mathematics in Science and Engineering*. Academic Press, New York, 1977.

[3] M. Bosse, P. Newman, J. Leonard, M. Soika, W. Feiten, and S. Teller. An Atlas framework for scalable mapping. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2003.

[4] M. C. Bosse, P. M. Newman, J. J. Leonard, and S. Teller. SLAM in large-scale cyclic environments using the Atlas framework. Accepted for publication in the International Journal of Robotics Research, 2005.

[5] T. A. Davis. Algorithm 8xx: a concise sparse Cholesky factorization package. Technical Report TR-04-001, Univ. of Florida, January 2004. Submitted to ACM Trans. Math. Software.

[6] T.A. Davis, J.R. Gilbert, S.I. Larimore, and E.G. Ng. A column approximate minimum degree ordering algorithm. *ACM Trans. Math. Softw.*, 30(3):353–376, 2004.

[7] F. Dellaert. Square Root SAM: Simultaneous localization and mapping via square root information smoothing. Technical Report GIT-GVU-05-11, Georgia Institute of Technology, 2005.

[8] F. Dellaert. Square Root SAM: Simultaneous location and mapping via square root information smoothing. In *Robotics: Science and Systems (RSS)*, 2005.

[9] F. Dellaert, A. Kipp, and P. Krauthausen. A multifrontal QR factorization approach to distributed inference applied to multi-robot localization and mapping. In *AAAI Nat. Conf. on Artificial Intelligence*, 2005.

[10] R. Eustice, H. Singh, and J. Leonard. Exactly sparse delayed-state filters. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 2428–2435, Barcelona, Spain, April 2005.

[11] R. Eustice, M. Walter, and J. Leonard. Sparse extended information filters: Insights into sparsification. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, August 2005.

[12] H.-J. S. Feder and J.J. Leonard. Decoupled Stochastic Mapping, Part II: Performance Analysis. Submitted to IEEE Transactions on Robotics and Automation.

[13] C.M. Fiduccia and R.M. Mattheyses. A linear time heuristic for improving network partitions. In *Proc. 19th IEEE Design Automation and Conference*, pages 175–181, 1982.

[14] J. Folkesson and H. I. Christensen. Graphical SLAM - a self-correcting map. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, volume 1, pages 383 – 390, 2004.

[15] J. Folkesson, P.Jensfelt, and H. I. Christensen. Graphical SLAM using vision and the measurement subspace. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2005.

[16] U. Frese, P. Larsson, and T. Duckett. A multilevel relaxation algorithm for simultaneous localisation and mapping. *IEEE Trans. Robototics*, 21(2):196–207, April 2005.

[17] A. Gupta, G. Karypis, and V. Kumar. Highly scalable parallel algorithms for sparse matrix factorization. *IEEE Trans. Parallel and Distributed Systems*, 8(5):502–520, 1997.

[18] P. Heggernes and P. Matstoms. Finding good column orderings for sparse QR factorization. In *Second SIAM Conference on Sparse Matrices*, 1996.

[19] P. Jensfelt, H.I. Christensen, and G. Zunino. Integrated systems for mapping and localization. In J. Leonard and H. Durrant-Whyte, editors, *ICRA-02 SLAM Workshop*. IEEE, May 2002.

[20] G. Karypis and V. Kumar. High performance sparse cholesky factorization algorithm for scalable parallel computers. Technical Report 94-41, U. of Minnesota, 1994.

[21] G. Karypis and V. Kumar. Multilevel algorithms for multi-constraint graph partitioning. In *Supercomputing '98: Proceedings of the 1998 ACM/IEEE conference on Supercomputing (CDROM)*, pages 1–13, Washington, DC, USA, 1998. IEEE Computer Society.

[22] B.W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *The Bell System Technical Journal*, 49(2):291–307, 1970.

[23] J. J. Leonard and H. J. S. Feder. Decoupled stochastic mapping. *IEEE Journal of Oceanic Engineering*, pages 561–571, October 2001.

[24] J. J. Leonard and P. M. Newman. Consistent, convergent, and constant-time SLAM. In *Intl. Joint Conf. on Artificial Intelligence (IJCAI)*, 2003.

[25] R.J. Lipton and R.E. Tarjan. Generalized nested dissection. *SIAM Journal on Applied Mathematics*, 16(2):346–358, 1979.

[26] R.J. Lipton and R.E. Tarjan. A separator theorem for planar graphs. *SIAM Journal on Applied Mathematics*, 36(2):177–189, 1979.

[27] Joseph W. H. Liu. Modification of the minimum-degree algorithm by multiple elimination. *ACM Trans. Math. Softw.*, 11(2):141–153, 1985.

[28] Joseph W. H. Liu. A graph partitioning algorithm by node separators. *ACM Trans. Math. Softw.*, 15(3):198–219, 1989.

[29] Y. Liu and S. Thrun. Results for outdoor-slam using sparse extended information filters, 2002.

[30] P. Matstoms. Sparse QR factorization in MATLAB. *ACM Trans. Math. Softw.*, 20(1):136–159, 1994.

[31] J. Nieto, J. Guivant, and E. Nebot. The hybrid metric maps (hymms): A novel map representation for denseslam. In *IEEE International Conference on Robotics and Automation*, 2004.

[32] Pothen, A., Simon, H., and Wang, L. Spectral nested dissection. Technical Report CS-92-01, 1992.

[33] R. Smith and P. Cheeseman. On the representation and estimation of spatial uncertainty. *Intl. J. of Robotics Research*, 5(4):56–68, 1987.

[34] R. Smith, M. Self, and P. Cheeseman. A stochastic map for uncertain spatial relationships. In *Int. Symp on Robotics Research*, 1987.

[35] S. Thrun and Y. Liu. Multi-robot SLAM with sparse extended information filters. In *Proceedings of the 11th International Symposium of Robotics Research (ISRR'03)*, Sienna, Italy, 2003. Springer.