Large-Scale Map-Making

Kurt Konolige

SRI International
333 Ravenswood Avenue
Menlo Park, CA 94025
konolige@ai.sri.com

Abstract

Current mapping algorithms using Consistent Pose Estimation (CPE) algorithms can successfully map areas of 10^4 square meters, using thousands of poses. However, the computation to construct the map grows as $O(n \log n)$, so larger maps get increasingly difficult to build. We present an abstraction method for postponing the growth in computation. This method solves a much smaller problem in the space of the connection graph of the map.

Introduction

In this paper we address one of the major goals of mobile robot research, the creation of a map from local sensor data collected as the robot moves around an unknown environment. Our emphasis is on the asymptotic behavior of the map creation algorithm, as the environments get very large. Current methods [Gutmann and Konolige 1999] are already capable of creating accurate, dense metric maps of environments of 10^4 m^2, but have computational problems as the scale gets larger.

Current metric mapping methods are distinguished by whether they use features (usually called *landmarks*) or rely on dense surface information that does not distinguish features. In this paper, we remain agnostic about this issue; the only requirement is that the map is represented as a set of robot poses with *constraints* among them. The reason for this choice is that such a representation scales well as the area of the map grows, since it generally represents only local constraints among poses.

Another issue in map building is the difference between incremental and batch methods. During exploration, new information is added to the map in an incremental fashion, and the cost should be low (ideally, constant time), as only a small portion of the map would be modified. However, when the robot recognizes a previously-visited area, it may have to "close the loop," and change a large section of the map. Closing a large loop, and registering all the poses of a map, are the hard problems in map construction.

In this paper, we analyze the computational properties of the CPE method for loop closure and full map registration, by simulating very large scale maps, on the order of 10^5 poses. These maps are several orders of magnitude larger than current mapping techniques can handle. The main results are:

- We provide an $O(n)$ algorithm for single loop closure, where n is the number of poses along the loop.
- Multiple loop maps are harder to solve, but by noting the sparse matrix structure of CPE, we derive an efficient $O(n \log n)$ algorithm.
- We develop a refined algorithm that takes advantage of the structure of connections in a map to substantially postpone the $n \log n$ asymptote.

Previous Work

Map making from a mobile robot is usually referred to as the SLAM (Simultaneous Localization and Mapping) problem. CPE is a SLAM method that uses constraints among local poses. As an alternative, Extended Kalman Filter (EKF) methods construct a large covariance matrix whose computational properties are poor at larger scales [Moutarlier and Chatila, 1989].

Recently there have been several efforts to develop EKF SLAM methods with good scaling properties. In [Newman et al. 2002], features and geometric constraints are used to merge partial maps created with SLAM techniques, using interesting geometric search techniques.

[Thrun et al. 2002] have developed a sparse matrix method for the information filter. They claim that the method scales linearly with the size of the map; however, it is difficult to determine if this is correct, since the limiting operation is sparse matrix inversion, which does not scale linearly in the general case.

[Leonard and Feder 2000] have a variation of SLAM that divides the world into submaps, so that the size of an update remains constant as the map grows.

[Guivant and Nebot 2001] also use a local update scheme, with local updates taking $O(n^2)$ time in the number of local features. Incorporation of local constraints into the global map, however, requires a complete calculation of the global EKF.

[Duckett et al. 2000] describe a system of linear constraints that has a network structure similar to the one described here; however, the convergence of the relaxation algorithm they propose is slow and inferior to the methods presented in this paper.

Map Construction via Pose Constraints

In this section we review the CPE method of Lu and Milios, and investigate its computational properties in preparation for determining large-scale behavior.

Lu-Milios Registration

We base our technique on the consistent pose registration paradigm of Lu and Milios [Lu and Milios 1997, Gutmann and Konolige 1999]. Figure 1 shows a set of poses from a mobile robot run, along with their scans from a laser range finder. The poses X_i are the free variables of the system. For each pose, a set of *constraints* connect it to other poses. These constraints can arise from odometry between successive poses, from matching scans between poses, and from any other information relating the poses, including *a priori* information.

To put all the constraints into a common framework, we represent them in terms of local differences D_{ij} between X_i and X_j. The term D_{ij} is a random variable with mean $\overline{D_{ij}}$ and covariance C_{ij}. Each constraint is a local spring, with a stiffness determined by C_{ij}. The optimal value for the poses is at the minimum energy of the constraint system. Lu and Milios phrase this condition in terms of a maximum likelihood:

$$\arg\max_{X_i} P(D_{ij} \mid \overline{D_{ij}}) \tag{1}$$

In the case of Gaussian covariance and mutually independent D_{ij}, this condition is equivalent to minimizing the Mahalanobis distance:

$$\sum_{i,j} \left(D_{ij} - \overline{D_{ij}}\right)^T C_{ij}^{-1} \left(D_{ij} - \overline{D_{ij}}\right) \tag{2}$$

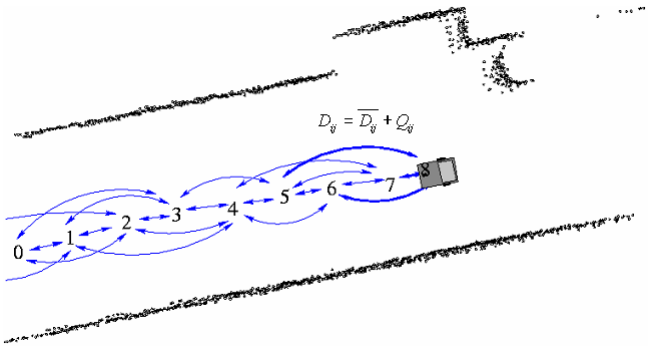

Figure 1 A set of poses and scan matches, with their relationships.

Frequently we will refer to this term as the *energy* of the system. Because the constraints are soft, the poses X_i can be moved to accommodate new information. For example, suppose we know the relationship of two poses in Figure 2, along the sides of the long corridors. By adding this relationship as a constraint, the minimum energy system is distorted to conform to the relationship.

Local Constraints

The energy equation (2) demands that constraints be expressed as differences between poses. Typically, these constraints arise from two sources:

1. *Odometry* between two successive poses of the robot.
2. *Matching* between two poses. If a sensor can see the same features from the two poses, then it is possible to derive an estimate of the position of the poses relative to each other. This estimate can take into account the error of the sensor in deriving the uncertainty between the two poses.

Global constraints can also be incorporated by establishing a fictional pose O at the origin. If a pose X has a global constraint, for example from GPS, then it can be expressed as the difference between X and O.

Pose differences D_{ij} are expressed in the origin of the pose X_i. Typically the difference is derived from an underlying measurement equation, which relates the measurement variables to pose coordinates. The measurement equation for a measurement Z is represented as a function:

$$X_j = f(X_i, Z) \tag{3}$$

The mean of the result is computed directly from the equation:

$$\hat{X}_j = f(\hat{X}_i, \hat{Z}), \tag{4}$$

while the covariance is derived using the Jacobian $J(Z,X)$. Let C_Z be the covariance of the measurement Z; then the covariance of the pose difference estimate is

$$C_{ij} = JC_Z J^T \tag{5}$$

The most common measurement is angle and distance – for example, odometry and matching with range sensors can

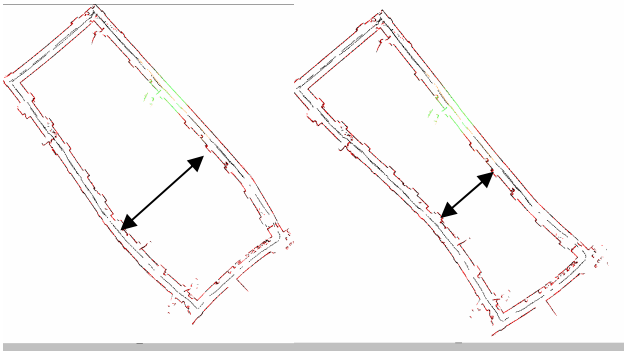

Figure 2 Two deformations of a set of poses with minimum values of Equation (2), based on added constraints between two poses on the sides of the map. On the left the poses are pushed apart, on the right they are drawn together.

both be represented this way. For this case, with distance l and angle α, we have:

$$f(X_i, Z) = \begin{bmatrix} x_i + l\cos(\theta_i + \alpha) \\ y_i + l\sin(\theta_i + \alpha) \end{bmatrix}$$

(6)

$$J_Z = \begin{bmatrix} \cos(\theta_i + \alpha) & -l\sin(\theta_i + \alpha) \\ \sin(\theta_i + \alpha) & l\cos(\theta_i + \alpha) \end{bmatrix}$$

Note that the expression for J_Z and hence C_{ij} is nonlinear, and the total energy function becomes nonlinear. In this case, there is no closed-form solution, and in general we will be able to find only local minima of the energy, using iterative linear methods.

Iterative Linear Method

In the linear case, the energy function (2) constitutes an unconstrained quadratic optimization problem, which can be solved by direct methods [Lu and Milios 1997]. Let **X** be the vector of size $3n$ formed by stacking all the X_i. Then we can solve (2) by solving the following linear system:

(7) $\mathbf{GX} = \mathbf{B}$,

where **G** is a $3n \times 3n$ matrix, and **B** is a constant vector of size $3n$.

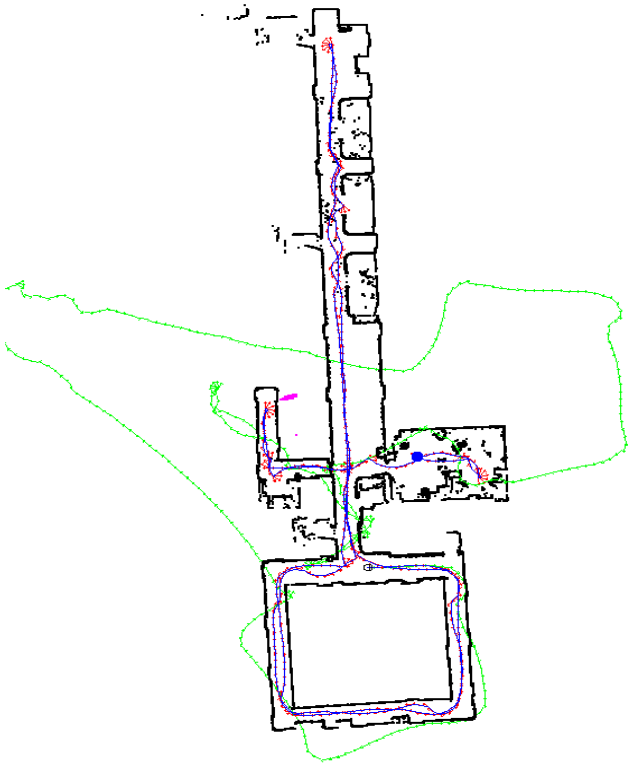

Figure 3 Sample map built from 419 poses with 1109 links. The green line shows the robot position based on odometry.

The elements of **G** are partitioned into 3x3 blocks G_{ij}, as follows:

$$G_{ii} = \sum_{j=0}^{n} C_{ij}^{-1}$$

(8)

$$G_{ij} = -C_{ij}^{-1}, \quad i \neq j$$

The vector **B** consists of a stacked set of 3-vectors B_i, defined as:

$$B_i = \sum_{j=0, j\neq i}^{n} C_{ij}\overline{D}_{ij} \quad (9)$$

If **G** is invertible, then the solution of the linear system gives the minimum energy solution of (2), when the covariances are linear. There is no proof yet that **G** is always invertible; however, in practice over many real and simulated mapping runs, **G** has always been invertible.

When the covariances are nonlinear, there is no direct reduction to the linear system (7). Instead, an approximate solution is found by linearizing the covariances using a Taylor series, solving the resulting linear system, and iterating the process. There are no guarantees that this method will converge, or that if it does, that the result is a global minimum. In fact it is easy to construct examples in which the system converges to a local minimum that is not globally optimal.

In building maps, the nonlinear system is typically well-behaved. Figure 3 shows an example mapping run of 418 poses and 1108 links; the poses are approximately 0.5 meters apart. The nonlinear system converges to a minimum after 3 steps of the linear solver; from inspection, this also appears to be the global minimum. Almost always the starting point for the nonlinear iteration is near to the solution, as forward matching aligns new poses before they are incorporated into the map [Gutmann and Konolige 1999]. The green line shows the unrestricted poses based on odometry, which would be harder to solve.

Computational Properties

The key computation to solve the linear system is inverting the matrix **G**. In a straightforward algorithm this would take $O(n^3)$ time and storage, where n is the number of variables (poses). But we can take advantage of the structure of **G** to improve on this result.

From inspection of Equation (8), **G** has the following form:

$$\begin{bmatrix} G_{00} & & G_{02} & \cdots \\ & G_{11} & & \\ G_{02} & & G_{22} & \\ \vdots & & & \ddots \end{bmatrix}, \quad (10)$$

where each of the submatrices G_{ij} is 3x3. All of the diagonal blocks G_{ii} are filled, but the off-diagonal elements

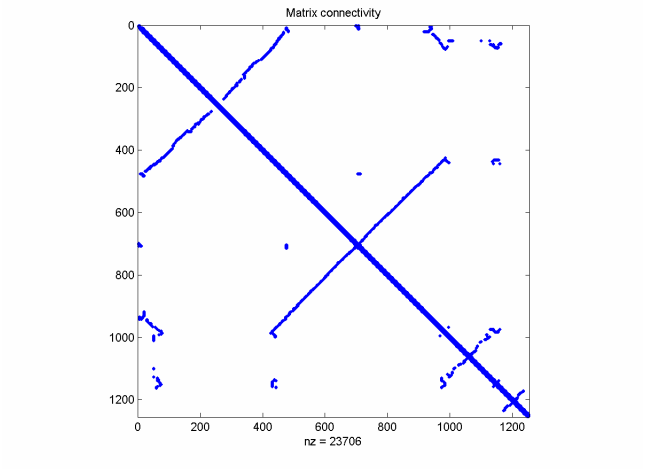

Figure 4 Connectivity of the matrix **G** *for the sample map.*

are sparsely populated – they occur only where there are links between poses that are not successors. Because of the locality of sensing, these links are limited to a neighborhood of a pose. For global constraints, there could also be a number of links from the origin pose *O* to other poses. We will assume for the remainder of the paper that the number of links is at most a constant factor times the number of poses. Figure 4 shows the connectivity of matrix **G** for the example map of Figure 3. The off-diagonal elements are where the robot backtracked over previously-traversed areas. Only 1.5% of the matrix entries are non-zero.

The matrix **G** is positive, symmetric, and definite. The solution of choice for sparse systems of this type is the conjugate gradient (CG) method. CG is an iterative method that converges in at least n iterations, where n is the rank of **G**. With sparse matrices, CG requires storing only the sparse elements, and performs just one matrix-vector multiplication per iteration, so that the number of operations in the parse case is $O(n)$ per iteration. If there are m iterations to convergence, then the whole process takes $O(nm)$ operations.

In the worst case, with $m = n$, CG is an $O(n^2)$ algorithm. To improve this result, it is important to reduce the number of iterations necessary for convergence. The *condition number* [Golub and Van Loan 1996] of **G** is a measure of how quickly CG will converge – numbers near 1 show a well-conditioned matrix, whose eigenvalues are similar. Unfortunately, **G** tends to have a small condition number for the mapping problem, since there can be large variations in the covariance matrix values for different links. For example, the condition number for the matrix of Figure 4 is 10^{13}. The matrix is so poorly conditioned that roundoff error prevents it from converging without additional help.

To speed up convergence, **G** can be *preconditioned*, using a symmetric positive definite matrix **H**, to make it better conditioned – the Preconditioned Conjugate Gradient (PCG). One of the most useful preconditioners is the incomplete Cholesky factor [Golub and Van Loan 1996]. The complete Cholesky factor requires $O(n^3)$ computations, and perfectly preconditions **G**, so that it requires only 3 iterations to converge. By trading off completeness, we can get an approximate factor in $O(n)$ that still greatly speeds up convergence. For example, the number of iterations to solve the sparse linear system of Figure 4 can be reduced from 547 to 21, using an appropriate incomplete Cholesky factor.

For most mapping problems of a reasonable size (< 10K poses), the application of PCG is efficient enough to be used without modification. For example, the full map of Figure 3 is solved in under 0.8 seconds, in three PCG steps, with 21 iterations on each step. It should be noted that, because of the structure of the covariance matrices, the number of PCG steps to converge for the nonlinear problem is always 3, and we will consider the difficulty of the problem to be related to the computation of PCG.

With larger n, the number of iterations needed by PCG to converge will grow, and hence the overall computation is not linear in the number of poses. In typical sparse matrix PCG applications, the computation scales as $O(n \log n)$. This is the case for the mapping problem, as we show experimentally in the next section. In some applications, such as large differential modeling over regular grids, the structure of the matrix **G** is further exploited to produce linear scaling [Lackner and Menikoff 1998]. We seek similar results for the mapping problem over larger scales, so that the techniques just presented can be extended to truly large mapping applications.

Linear Loop Closure

In constructing maps incrementally, most poses that are added cause relatively minor changes to the map, as in Figure 1, and can be incorporated without re-optimizing the whole map. When a loop is closed, by matching two poses that may initially be far apart (Figure 5), many poses can be affected, and at a minimum the poses comprising the loop must be optimized.

Loop closure in its simplest form involves a very sparse matrix **G**, with only one off-diagonal element from the matched poses. In this case, the PCG algorithm with incomplete Cholesky preconditioner exhibits only linear

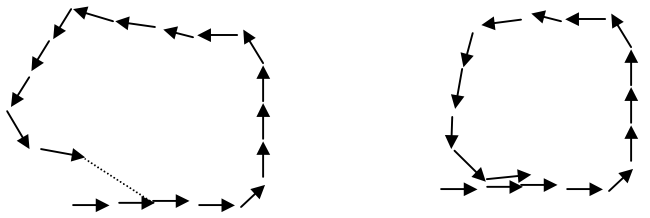

Figure 5 Loop closure adds a constraint to close a loop. The associated matrix is fully diagonal, except for the matched closure poses.

growth in computation as the number of poses increases. Figure 6 shows the results for simple loops with up to 15K nodes. To make sure the preconditioner is not adding more than linear time, we list the actual time spent in the computation as well as the number of PCG iterations. The iterations stayed constant at around 5 per PCG step, through 4 orders of magnitude of the problem size.

Divide and Conquer Algorithm

It isn't surprising that simple loop closure is O(n), since there is a divide-and-conquer algorithm for solving it exactly, in the linear case. Consider a chain of poses as in Figure 5, where the end points of the chain are fixed (they needn't be the same point, just fixed globally). Divide the chain in half, with the midpoint pose being X_m:

$$X_1 \quad X_2 \quad \cdots \quad X_m \quad X_{m+1} \quad \cdots \quad X_n .$$

Instead of solving the full system, we solve a reduced system consisting of just the three poses X_1 X_m X_n. To find the covariance of the pose difference D_{1m}, we use the compounding property of chained covariances [Smith and Cheeseman 1987]. We already have the covariance C_{12}. To find C_{13}, we note that the pose differences add:

$$D_{13} = D_{12} + D_{23} .$$

The covariance of D_{13} can be computed from the measurement equation (6) as:

$$C_{13} = C_{12} + J(X_2, Z_2) C_Z J(X_2, Z_2)^T . \quad (11)$$

Continuing down the chain, the covariance C_{1m} is computed, and similarly for C_{mn}. We solve the reduced system, and fix the pose X_m. Now there are two subproblems, which we solve in the same manner. Continuing recursively, finally we will have solved n subproblems, but the complexity of the subproblems stays constant.

The compounding operations only contribute O(n) computations, if they are done bottom-up. First the covariances C_{i3}, C_{35}, ... are computed, then they are combined to produce C_{15}, C_{59}, etc. Thus, the complexity of the complete loop closure is O(n).

The solution computed by divide-and-conquer will be exactly the same as solving the full linear system (6), if the constraints are linear. In the nonlinear case, both the linear system and the divide-and-conquer algorithm must be iterated to find a local minimum.

Multi-loop Maps

In actual maps, there are multiple loop closures, depending on the size and structure of the environment, as well as the robot's motion. Multi-loop maps are harder to solve than the single-loop ones. We are interested in the asymptotic performance of a mapping algorithm at very large scales, and it is necessary to generate test cases for these scales. To represent large-scale maps in the abstract, we can generate a random 2D graph structure whose edges are chains of poses, and whose nodes are poses where the chains meet. Figure 7 shows a map of 1900 nodes. The map was created by dropping seeds randomly over the area, then growing the seeds until their borders touched. The distribution of edge lengths follows approximately a Poisson distribution.

We are interested in the the scaling behavior of the PCG algorithm under the following sets of conditions.

1. The node structure stays constant, and the number of poses in the edges grows with the map area.
2. The edges stay the same length, and the number of graph nodes grows proportionally with the map area.

These conditions represent the extremes of possible map structure changes with scaling. Condition (2) is more realistic, in that we expect the loop properties of the environment to remain relatively constant. In both cases, we expect PCG to have O(n log n) performance.

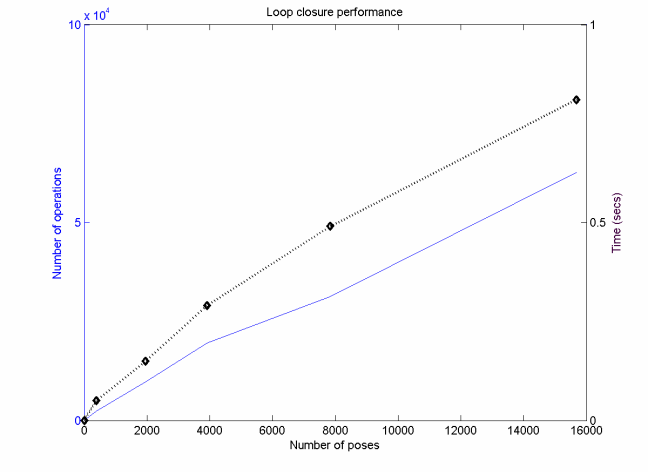

Figure 6 Loop closure performance. CPU time for closure is given by the black line. Number of operations (iterations x nodes) is the blue line.

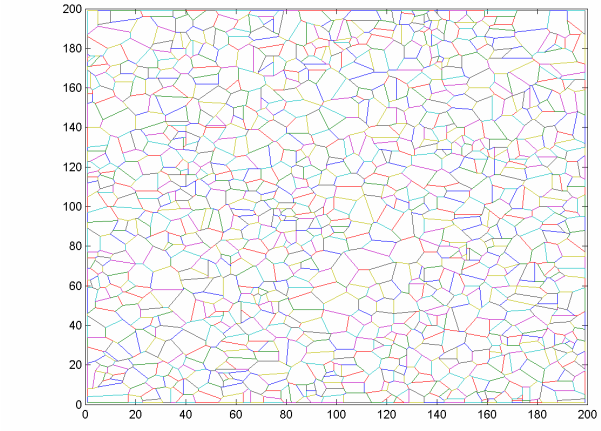

Figure 7 Abstract map consisting of 1900 nodes connected by pose chains.

Structured PCG

For both conditions, we propose an algorithm that does sustantially better than PCG, by taking advantage of the structure of the connection graph, and exploiting the divide-and-conquer algorithm.

Consider the map M represented in Figure 7, and note that the chain between two nodes X_i and X_j induces a constraint between these nodes. Since there are no outside links to the interior poses of the chain, the effect of the chain X_i and X_j can be represented by a single constraint, whose covariance is computed by compounding of the covariances of the individual links of the chain. The reduced map M_R is formed by removing all chains and replacing them with the equivalent pose differences between the map nodes.

Algorithm Structured PCG

Form the reduced map $M \Rightarrow M_R$.
Solve M_R using PCG until convergence.
Fix the coordinates of the poses of M_R.
For each edge chain in M, solve the chain using PCG with fixed end poses.

Structured PCG performs the $O(n \log n)$ step on the reduced graph M_R, and thus we expect its performance to be superior to the standard PCG algorithm. The additional work to optimize the chains is linear in the total size of the chains, and hence the total number of poses.

Figure 8 shows how the two algorithms scale with increasing chain size, on the sample map with 1900 nodes. This is a log-log graph, so care must be taken in interpreting the shapes of the graphed results.

The upper line is the standard PCG algorithm. Even though it is an asymptotically straight line, its asymptotic behavior is $n \log n$.

The structured algorithm (bottom line) does much better, because it is just solving the reduced map. At the left-most point, the algorithms solve the same problem (chain size is 1). As the problem size grows, structured PCG spends less time solving the $O(n \log n)$ step, and just has to solve chains of poses, which is a linear operation. Hence, its asymptotic behavior is linear, and much more efficient than standard PCG.

For comparison, the middle line shows how PCG would perform if it *were* linear in the problem size, using the same number of iterations in PCG as needed for the initial problem. This clearly shows the efficiency of structured PCG, which factors out the solution of the interconnected nodes from the chains of links.

A more realistic assessment of performance comes from considering condition (2), where the map connectivity grows with map area, and the chains stay the same size (Figure 9). Here we picked a mean chain size of 12 poses. Most real-world environments would tend to have multiple loops, and the larger the area, the more loops they would have. For example, city streets have this characteristic.

The standard PCG algorithm (top line) exhibits the same $O(n \log n)$ growth, as the PCG iterations increase with problem size. Note that the PCG algorithm has essentially the same performance as it did in the previous fixed-structure case. The algorithm doesn't take advantage of any structural differences between the two types of problems. Again, for comparison the center line shows the PCG algorithm with fixed iterations (linear growth).

Unlike in the fixed structure case, the structured PCG algorithm now increases more rapidly as the problem size grows, because the reduced graph M_R is growing, and this is the hard part of the solution. For any given problem size, though, structured PCG performs less work than the standard PCG algorithm, because it only solves the reduced

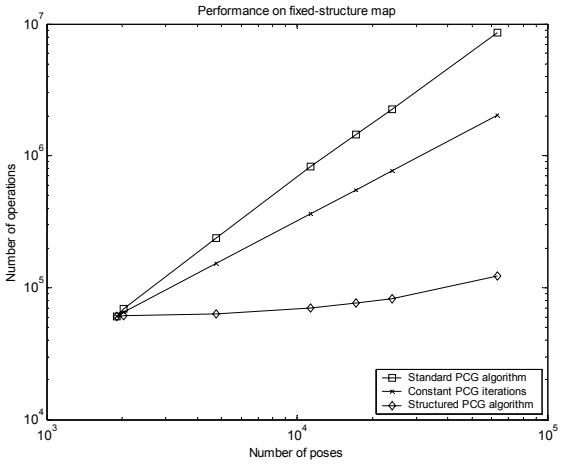

Figure 8 Scaling performance of the standard and structured PCG algorithm. The middle line is the standard PCG algorithm with a fixed number of iterations (linear in node size).

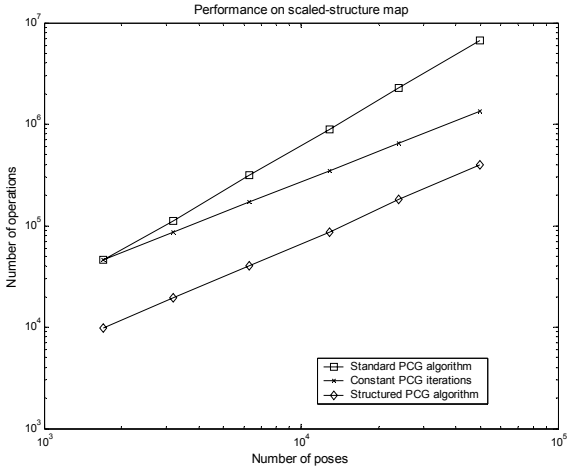

Figure 9 Scaling performance of the algorithms as the chain size stays constant, and the map structure grows with map area. Chain size was fixed at 12 poses.

graph. It also has better asymptotic behavior. Instead of n log n, the reduced nature of the graph means that the asymptote is (n/k) log (n/k), where n/k is the number of nodes in the reduced graph M_R.

We checked the computational behavior for both conditions by looking at the CPU time used. CPU time tracked the number of operations determined from PCG iterations. Interestingly, the CPU resources were quite modest – the largest problem, some 60K poses, used only 6.5 minutes of time on a standard PC.

Conclusions

With the success of small-scale indoor and outdoor mapping, researchers are starting to turn their attention to the scaling behavior of their algorithms. In this paper, we determine this behavior for Consistent Pose Estimation methods, and show experimentally that O(n log n) can be expected by using PCG to solve the pose registration problem. A more sophisticated algorithm, structured PCG, can take advantage of the particular form of loops in mapping, to do better by several orders of magnitude. With suitable data sets, we can now hope to map areas comprising 10^5-10^6 poses, and reasonably expect to cover a million square meters with such maps.

References

T. Duckett, S. Marsland, and J. Shapiro. Learning globally consistent maps by relaxation. In *Proc. ICRA'2000*, pages 3841 - 3846, 2000.

G. H. Golub and C. F. Van Loan. *Matrix Computations*. John Hopkins University Press, 1996.

J. Guivant and E. Nebot. Compressed Filter for Real Time Implementation of Simultaneous Localization and Map Building. *Field and Service Robots 2001* June 2001, Finland.

Gutmann, J. S. and K. Konolige. Incremental Mapping of Large Cyclic Environments. In *CIRA 99*, Monterey, California, 1999.

K. Lackner and R. Menikoff. Multi-scale linear solvers for very large systems derived from PDES. In *Fifth Copper Mountain Conference on Iterative Methods*, April, 1998.

J.J. Leonard and H.J.S. Feder. A computationally efficient method for large-scale concurrent mapping and localization. In *Proceedings of the Ninth ISRR*, pages 169–176. Springer-Verlag, 2000.

Lu, F. and E. E. Milios. Globally consistent range scan alignment for environment mapping. *Autonomous Robots*, **4**(4), 1997.

P. Moutarlier and R. Chatila. An experimental system for incremental environment modeling by an autonomous mobile robot. *Proc. ISER-89*.

P.M. Newman, J.J. Leonard, J. Neira and J.D. Tardós: "Explore and Return: Experimental Validation of Real Time Concurrent Mapping and Localization". ICRA, May, 2002.

R. C. Smith and P. Cheeseman, *On the Representation and Estimation of Spatial Uncertainty.* The International Journal of Robotics Research **5**(4), pp. 56-68.

S. Thrun, D. Koller, Z. Ghahramani, H. Durrant-Whyte, and A. Y. Ng. Simultaneous Mapping and Localization with Sparse Extended Information Filters. *Proc. Algorithmic Foundations of Robotics*, Nice, 2002.