# Evaluating the Performance of Map Optimization Algorithms

Edwin Olson, University of Michigan
Michael Kaess, MIT

***Abstract*— Localization and mapping are essential capabilities of virtually all mobile robots. These topics have been the focus of a great deal of research, but it is not always easy to tell which methods are best. This paper discusses performance evaluation for an important sub-problem of robot mapping, map optimization. We explore aspects underlying the evaluation of map optimization such as the quality of the result and computational complexity. For each aspect we discuss evaluation metrics and provide specific recommendations.**

## I. Introduction

Virtually all mobile robots require some form of mapping or localization, whether for obstacle avoidance, path planning, or building models of the environment. The central importance of this problem has attracted a great deal of research (see [1], [2], [3] for good surveys). However, it is often difficult to determine which methods are better than others. In turn, this makes it difficult for consumers of these methods to make good choices.

Evaluating mapping methods is difficult because the mapping problem is not really a single problem, but really a number of sub-problems, each of which has a significant effect on the quality of the final result. To produce a final map from low-level sensor data, for example, a researcher might first extract features from LIDAR data, match those features from different poses, reject outliers, and finally compute a posterior map. Each sub-problem is itself the subject of active research, and researchers may approach each sub-problem differently. Making the problem worse, different researchers may decompose the problem differently, for example replacing feature-matching and explicit data association with direct pose-to-pose scan matching.

The consequence of these issues is that comparing the end-to-end performance of two systems is not particularly illuminating: differences in end-to-end performance could be the result of any one of the processing stages, and there is generally little indication as to which.

An ideal evaluation mechanism would allow direct comparison between different approaches. Given the challenges outlined above, it seems unlikely that a perfect benchmark exists. Still, good (though perhaps imperfect) metrics are worth pursuing.

One reasonable approach is to develop evaluation methods for each of the sub-problems individually. A single module can then be tested in isolation. Ideally, other researchers could similarly test their own modules, allowing apples-to-apples comparisons. The challenge is that designing good performance metrics—even for just a single sub-problem—is difficult.

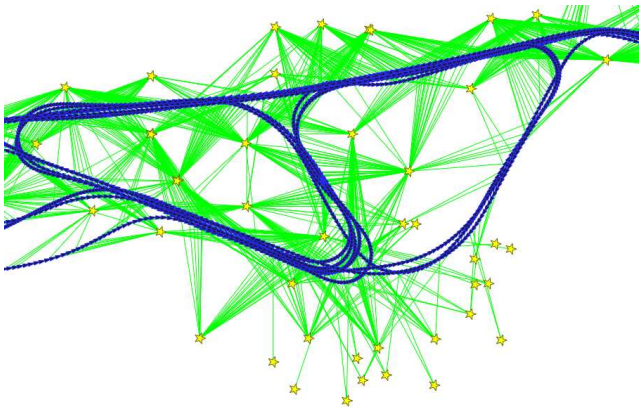

Fig. 1. Example pose/feature graph. The map optimization problem can be represented as a graph whose nodes represent robot poses and landmarks. Edges encode uncertain geometrical relationships between nodes. The figure shows a portion of the Victoria Park dataset. Landmarks, depicted in the figure as stars, are trees extracted from a LIDAR dataset.

In this paper, we focus on the map optimization sub-problem. Even within this limited scope, there is no single accepted set of evaluation metrics. Should algorithms be compared based on the quality of the map they produce, the time it takes them to produce a reasonable map, or their robustness to poor initial estimates? Methods can be categorically different as well: some methods might be designed for batch operation, while others are intended for online use.

We describe multiple aspects of map optimization performance in this paper and discuss evaluation metrics for each one. We also provide recommendations for how to proceed with performance evaluation. For completeness, after a discussion of related work, we start with a short introduction of the map optimization problem.

## II. Related Work

While there are few widely-used benchmarks in mapping, other disciplines have succeeded in establishing common benchmarking standards and datasets. These have enabled direct performance comparisons between different researchers' approaches and have generally contributed towards methodical and systematic improvements. While it can be argued that these benchmarks encourage trivial and incremental improvements (or even tuning for the benchmarks themselves), there

is considerable value in having an objective evaluation metric, even if it is imperfect.

For dense stereo algorithms, for example, the Middlebury stereo dataset [4] at `http://vision.middlebury.edu/stereo/` provides several left and right camera images; the task is to compute the disparity map. Ground truth is provided, along with specific instructions on how performance should be measured. Having several datasets conforming to the same input format, along with a strict prohibition on per-dataset tuning, limits the effectiveness of optimizing for the benchmark. In contrast, the mapping community does not have a similar set of coordinated datasets, and per-dataset tuning is generally considered inevitable[1].

The Middlebury dataset goes further, adding an automated evaluation system. Authors of new algorithms upload their results, and the system automatically evaluates and ranks them against other known algorithms. For one of the test problems, ground truth is known only by the evaluation system: this further combats tuning.

The Caltech 101 computer vision dataset [5] is another example of an performance benchmark. This dataset contains 9,144 images belonging to 101 object classes (such as "crocodile", "airplane", etc.) The task is to identify the category of each image. Ground truth, including hand-annotated object outlines, is provided.

The robotics community has been working towards better performance evaluation, primarily through the sharing of datasets. However, while commonly-available datasets (e.g. Radish [6], or the recently proposed "data papers" [7]) are very useful, they generally are *not*, in themselves, benchmarks. These datasets do not specify a particular task, much less a specific evaluation criterion. These datasets often lack ground truth (generally because there's no obvious way to obtain ground truth), which makes it difficult to devise good metrics. Still, these are a step in the right direction.

For many sub-problems in the robotics mapping domain, there is no general consensus regarding which performance metrics to use. While this paper attempts to address this issue for map optimization, other aspects of the mapping problem pose challenges as well. With data association, for example, there is no general consensus on which statistics to report, or how much an algorithm should be penalized for a false positive. Within the last year, however, Udo Frese has made available an excellent dataset for data association that contains ground truth.

## III. Problem Formulation

In this paper, we consider the specific sub-problem of map optimization. That is, we assume that earlier stages of sensor processing have already completed correctly, including feature detection, matching (loop closing), and outlier rejection. The task is to compute a posterior map given this data.

Computing a posterior map can be naturally cast as a non-linear optimization problem. The parameters being optimized are the positions of the robot and/or features in the environment; these parameters are optimized so that they are in the best possible agreement with the robot's observations.

An observation is information about the relative position of two state variables. For example, the robot might measure that the distance between it and a nearby landmark is approximately 5 meters with a variance of 1 meter. The robot's position and the position of the landmark are the state variables, and the observation is the uncertain distance between them.

For the $i^{th}$ observation, let $z_i$ represent the observed quantity, $f_i$ represent the observation function (which predicts the value of the observation given the current state estimate), and $\Sigma_i$ represent the uncertainty of the observation. The $\chi^2$ error for a single observation is a measure of how well the current state agrees with the observation, and can be written in terms of the state estimate $x$. The $\chi^2$ error is the squared error times the inverse of the uncertainty:

$$\chi_i^2 = (z_i - f_i(x))^T \Sigma_i^{-1} (z_i - f_i(x)) \tag{1}$$

Given a number of observations, the total cost is simply the sum of the $\chi^2$ errors of each observation:

$$\chi^2 = \sum_i (z_i - f_i(x))^T \Sigma_i^{-1} (z_i - f_i(x)) \tag{2}$$

Minimizing the $\chi^2$ error is explicitly the objective of the optimization: smaller values indicate that the state vector is in better agreement with the observations.

The structure of this optimization problem is well represented as a pose/feature graph. In this graph, nodes represent disjoint subsets of the state vector corresponding to positions in space (the locations of the robot and the landmarks), and edges represent observations relating the positions of pairs of locations. The pose/feature graph interpretation is quite general[2] and lends itself to a natural visualization of the problem (see Fig. 1).

This pose/feature graph formulation, while arising out of more recent work in the field, is applicable to virtually all map optimization algorithms. Even those methods that are not based on this formulation can be viewed as optimization algorithms for pose/feature graphs. Take, for example, the Extended Kalman Filter (EKF). It can be viewed as a graph optimization algorithm that incrementally processes one observation at a time, minimizing the $\chi^2$ error for those observations it has seen. While the EKF traditionally marginalizes out

[1] Since any two robotics datasets may use different robots, different sensors, use different coordinate system conventions, and/or operate in dramatically different environments, it is reasonable to allow essential parameters to be adjusted. A coordinated dataset in which these essential parameters were provided in a consistent manner would allow a prohibition on tuning.

[2] We note that the pose/feature graph is not perfectly general. A single observation that simultaneously relates three or more nodes does not have a pose/feature graph representation, even though such an observation could be represented by Eqn. 2. "Unary" constraints, such as those arising from GPS observations, can be handled in a pose/feature graph representation by adding an additional node that represents a fixed reference point.

historical robot poses, these poses can be recovered. As a result, $\chi^2$ values can be computed for the EKF that are directly comparable to that of other algorithms.

The generality of the pose/feature graph formulation motivates the use of it as the canonical formulation when comparing optimization methods.

## IV. Map quality

Perhaps the most obvious performance metric is the quality of the posterior map. Since the problem is defined in terms of $\chi^2$ error (as in Eqn. 2), it is natural to compare two optimization methods based on the $\chi^2$ error of the maps that they produce. In this section, we describe some of the limitations of $\chi^2$ error as a performance metric, and detail a complementary metric that is often helpful. We also illustrate that the map optimization problem is subject to overfitting, and discuss how this affects performance evaluation.

### A. The limitations of $\chi^2$

By definition, the optimal map is the map whose $\chi^2$ is a global minimum. But suppose that we are given two sub-optimal maps: how do we know which of those maps is better?

Why do we care about comparing the quality of sub-optimal maps? First, many map optimization algorithms produce sub-optimal estimates by design, such as the Extended Kalman Filter and its information-form variants: they are sub-optimal due to linearization approximations. Sub-optimal estimates are also computed by a number of non-linear optimization methods that cut edges (e.g., TreeMap [8] and Thin-Junction Trees [9]) and those that are based on stochastic gradient descent [10]. Second, even those algorithms that can produce optimal maps take time to do so. Along the way, they produce intermediate results that we want to evaluate.

*The critical limitation of $\chi^2$ error is that a map with small $\chi^2$ error is not necessarily better than a map with a larger $\chi^2$ error.* At first, this may seem odd. In the end, our goal is to produce the optimal map, or a map that is as similar to it as possible. The key issue is what we mean by "similar". The goal of mapping is to produce maps in which the relative positions of poses and landmarks is as close as possible to the optimal values. Thus, we must measure the similarity of two maps by comparing the positions of the poses and features, not by comparing the $\chi^2$ error. For example, it is possible for a map with a small $\chi^2$ error to be severely distorted, while a map with a larger $\chi^2$ is in better agreement with the optimal map.

In Fig. 2 we see a simplified scenario where $\chi^2$ does not tell the whole story. In this figure, we are estimating a single value (the position of a one-dimensional robot along the $x$ axis, perhaps). The figure shows two different estimates, both of which have the same $\chi^2$ error (as plotted on the $y$ axis). However, estimate number one is significantly closer to the optimal value. Clearly, estimate one would be a better answer than estimate two.

Even in this simple example, we see that this sort of behavior arises from optimization surfaces with complex shapes. In the map optimization problem domain, the optimization surfaces can be very complex due to the highly non-linear nature of the underlying observations. Many optimization surfaces arising from mapping problems have long shallow valleys where it is possible for the map to change dramatically with very little effect on $\chi^2$.

This unintuitive behavior is further illustrated in Fig. 3. In the figure, a map with low $\chi^2$ error is severely deformed in comparison to a map with much larger $\chi^2$ error. If we compare the positions of the individual poses in the map, we find that the map with a small $\chi^2$ error has a very large mean squared error and that the map with a larger $\chi^2$ error has a small mean squared error relative to the ground truth.

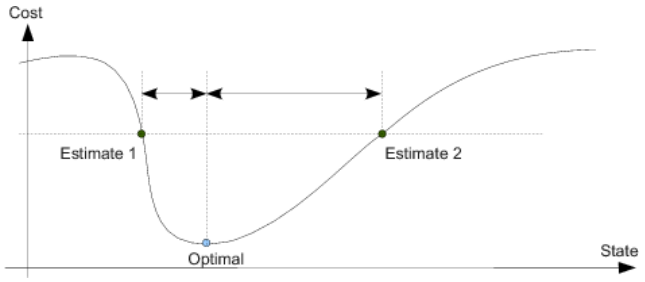


Fig. 2. $\chi^2$ error is not indicative of map quality. The $\chi^2$ error of a state estimate does not always indicate how close it is to the optimal value. In this figure, two estimates have equal $\chi^2$ error, but estimate 1 is much closer to the optimal solution.

We believe that map optimization algorithms should report the Mean Squared Error (MSE) of their maps with respect to the optimal map, *in addition* to the $\chi^2$ error. Let $x^*$ be the state vector of the optimal map, and $x$ be some estimate of that map. Naively, the mean squared error is simply $(x-x^*)^2$. To be clear, these state vectors contain the positions of poses and features in the global coordinate frame. Of course, the $\chi^2$ error should also be reported: it is still highly relevant, since minimizing it is the explicit optimization objective.

In fact, $\chi^2$ and MSE error describe orthogonal components of solution quality. Consider the simple example in Fig. 2: the MSE is a measure of the distance along the $x$ axis, while $\chi^2$ is a measure along the $y$ axis. This general idea extends to higher-dimensional optimization problems.

Computing the MSE literally, as suggested above, is problematic in that it mixes units: position and rotational quantities are summed together, which makes it more difficult to intuitively understand the result. Instead, we propose separately reporting the mean squared error for position and rotation. For our application, this produces two quantities, $MSE_{xy}$ and $MSE_\theta$.

In many mapping problems, the posterior map is under-determined: if the whole map is subjected to a global rotation and translation, the $\chi^2$ error is unaffected. This occurs when the pose/feature graph contains only relative constraints; if GPS data is available, this degeneracy does not exist.

If the map is under-determined, we cannot simply compute the mean squared error metrics: we could end up with arbi-

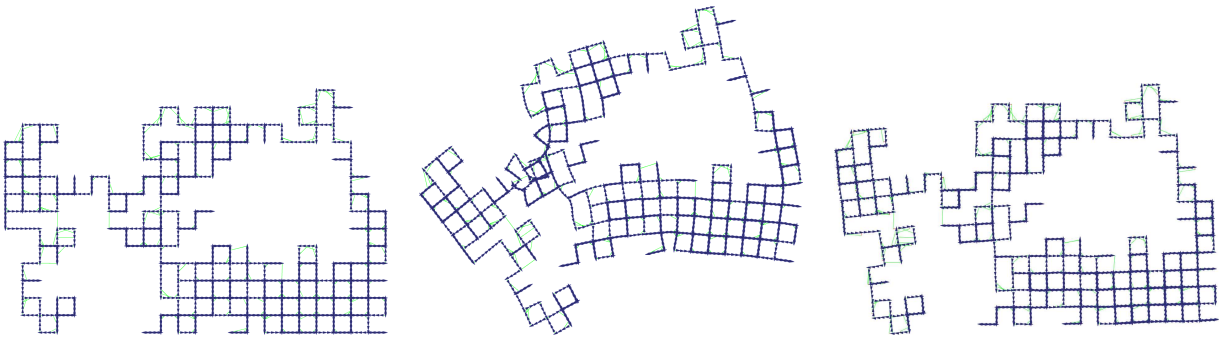


Fig. 3. $\chi^2$ anomaly. $\chi^2$ error is not always a good indicator of map quality. From a practical perspective, the right map is superior to the middle map, despite the fact that the middle map has one-tenth the $\chi^2$ error. The MSE metrics, in contrast, clearly identify the map on the right as being better than the middle map.

trarily inflated values due to the unknown global translation and rotation.

Our solution to this problem is to return the minimum possible MSE: if we freely translate and rotate the whole posterior map, what is the minimum attainable MSE? This is equivalent to finding the rigid-body transformation that best aligns the posterior map with the optimal map which has an efficient solution [11] in closed form. Code to perform this alignment is available from the authors' website.

### B. Overfitting

In many problems, the average node degree of the graph is relatively low. In these cases, the numerical global minimum often has a substantially lower $\chi^2$ error than the ground truth. In other words, the optimization algorithm can perturb some nodes in the graph in order to better fit the *noise* in the edges. This overfitting is evident in Fig 4.

As in other regression domains, overfitting becomes less of a concern as the number of observations (graph edges) grows relative to the number of free variables (graph nodes). Intuitively this makes sense: as the number of edges attached to a node increases, it becomes harder to find ways to artificially reduce the error of some edges without increasing the error of other edges. Since the real-world purpose of map optimization algorithms is to infer the *true* structure of the world, reducing the $\chi^2$ error by overfitting the noise in the graph is not useful.

Overfitting can be a significant issue for graphs with an average node degree of less than about 20 (see Fig. 5). This figure plots the ratio of the numerically-optimized $\chi^2$ versus the $\chi^2$ corresponding to the ground truth as a function of the average node degree. When the average node degree is about 5, for example, it is possible to find a solution whose $\chi^2$ error is only about 40% that of the ground truth.

This data was obtained using synthetic experiments so that ground truth was known; results for each data point were averaged over 15 randomly-generated graphs. Note that the impact of overfitting can be predicted using only the average node degree: factors such as the size of the graph and the noise

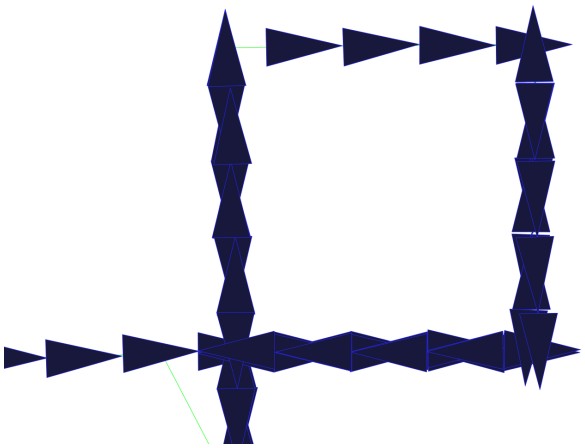

Fig. 4. Overfitting example. In this portion of the CSW synthetic dataset, overfitting is clearly evident. The robot is moving in a grid world, yet misalignment between nodes in the lower right is obvious. This portion of the graph has a relatively small node degree.

level in the edges cancel out in computing the ratio. The figure confirms this: it contains four plots corresponding to different graph sizes and noise levels. The overfitting behavior (when plotted versus average node degree) is consistent across all four experiments.

The overfitting issue must be considered when evaluating map optimization algorithms. When the average node degree is small and two methods' $\chi^2$ values are compared, the lower $\chi^2$ error does not necessarily correspond to the better map. The lower $\chi^2$ value may simply represent over-fitting to the noise in the problem, and may be of little practical significance.

*Our Recommendations*

- Experimental results should explicitly indicate the average node degree, since this quantity conveys significant information about the problem, both in terms of difficulty, and susceptibility to over-fitting.
- When available, the $\chi^2$ error corresponding to the ground truth should be given along side the $\chi^2$ of the map opti-

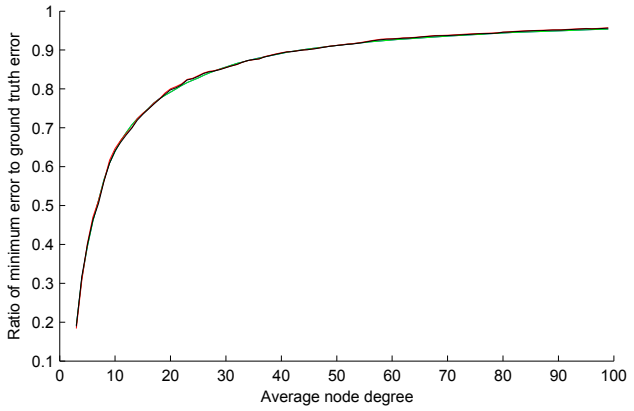


Fig. 5. Overfitting impact. This figure shows the ratio of the minimum $\chi^2$ value (as found by LU decomposition) versus the $\chi^2$ error for the ground truth for a large number of randomly generated synthetic graphs. As the average node degree increases, overfitting becomes less of a factor. Four different trends are plotted for graphs with orders of magnitudes differences in number of nodes and noise levels: agreeing with theoretical prediction, the impact of overfitting is independent of these factors.

mization algorithm. If $\chi^2$ is being plotted (as a function of computation time, for example), we recommend plotting the ground truth $\chi^2$ as a horizontal line on the same plot.

- MSE, along with $\chi^2$ should be used when describing the quality of a partially-converged map.
- Consumers of map optimization algorithms should take overfitting into account when selecting methods.

## V. Computational Performance

Computational performance is one of the most important aspects of robot mapping systems, as it determines if the system can even be used for a particular application, for example under real-time constraints. But comparing computational performance between different algorithms is difficult, not only because of variations in the computers used by researchers, but also because of the various speed/accuracy trade-offs made by different algorithms.

### A. Complexity Bounds

One way to report computational complexity is through analytical complexity bounds, using big O-notation. These principled bounds allow methods to be coarsely sorted into different complexity classes and are independent of particular datasets.

Most real-world data sets have structure that can lead to lower bounds. One good example is the approximate planarity of most pose/feature graphs [12]. It is important for researchers to clearly distinguish between the actual worst-case performance and the performance bounds arising from their assumptions. Similarly, researchers should be careful to indicate if their solutions are approximate.

Asymptotic complexity is a coarse characterization of performance, since it ignores constant factors. It also does not directly address the applicability of a method to real-time use, since complexity bounds may represent amortized complexity.

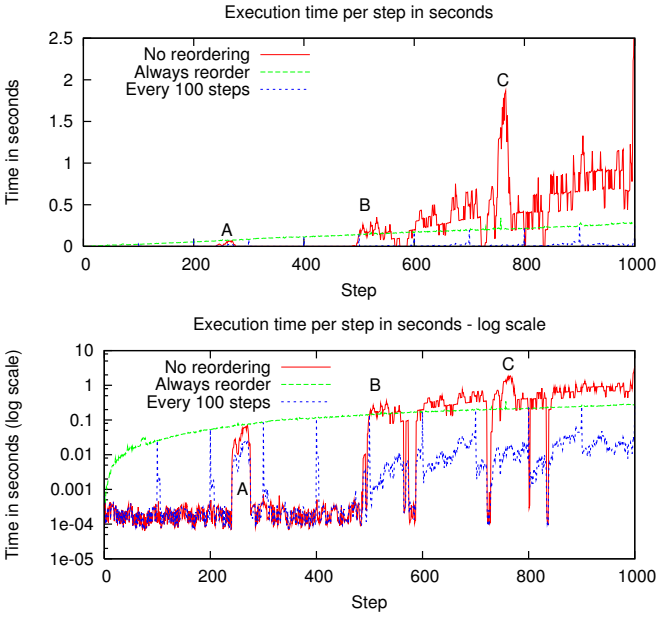


Fig. 6. Example of step-by-step timing and comparison against a reference algorithm, taken from [13]. Both linear (top) and log (bottom) scale are shown, comparing the iSAM algorithm (blue) with the batch SAM algorithm (green) and a naive incremental implementation (red).

### B. Experimental Evaluation

Often an actual measure of the real computational cost of an algorithm on sample data is the only way to directly compare similar algorithms. Batch methods are generally easier to compare: plotting $\chi^2$ and MSE as a function of computational time, is generally very descriptive. Online algorithms pose additional challenges.

Amortized cost is a useful concept for describing complexity. However, the actual variation of runtime at each step can significantly impact the applicability of the method to real-world robots. An algorithm with an amortized runtime of $O(n)$ per step might periodically require $O(n^2)$ computations, for example. From a practical system building perspective, the fact that the algorithm has a low amortized runtime isn't particularly helpful: the CPU sits nearly idle most of the time, but is periodically overwhelmed. Reporting amortized cost is certainly appropriate, but the worst-case costs per step may be of greater practical concern to users of the algorithm.

To illustrate the time-varying CPU costs of an algorithm, we recommend plotting the per-step cost over time, as in Fig 6. However, the temptation to "eyeball" such a plot in order to claim an asymptotic complexity bound should be resisted. First, these plots generally reflect the behavior of the algorithm on a single dataset and thus cannot be relied upon for asymptotic bounds. Second, the actual asymptotic behavior may not be evident over the limited duration of the experiment.

### C. Accounting for Hardware Differences

Comparing computational cost between different platforms is not an easy task, as it is influenced by a large number of

factors. The most obvious factor is the type and clock speed of CPU used. However, many other factors play a role in the actual computation time, such as the system bus speed, cache size, compiler, and the use of architecture enhancements (such as multimedia instructions).

Even if the algorithm itself appears to be single-threaded, applications may inadvertently benefit from other cores. When applications call external libraries (e.g., BLAS, Boost, IPP, or OpenCV), they may unknowingly be making use of additional cores on their CPU. Graphics processors (GPUs) can also be used to accelerate both robotics algorithms [14] and math libraries in general; as time progresses, libraries may automatically make use of this hardware as well. As a result, the runtime performance of an algorithm may be dependent on both the CPU *and* GPU. Authors should be especially careful to identify these situations and to provide all of the relevant specifications of their computers.

Still, it seems inevitable that no two researchers will be able to exactly replicate each other's performance results due to the large number of factors. One possible approach is for researchers to provide relative performance data, comparing their own method to other "standard" algorithms. This approach is not without its challenges, as we discuss below.

An alternative is for researchers to establish centralized performance testing systems, similar to the Middlebury dataset [4]. But in contrast to the Middlebury dataset that only tests for quality, a computational cost evaluation would require submitting a statically linked executable for a specific platform, or a platform-independent implementation such as Java. This provides a partial solution to the problem, but is also fairly limiting.

### D. Reference Algorithm

Another way to compare different methods is to implement the various algorithms and directly compare them on the same hardware. However, implementing well-optimized versions of other researcher's methods can be challenging and time consuming: experience with the intricacies of the algorithm can be critical in order to produce representative results.

An alternative is to use source code published by other researchers. This doesn't necessarily enable fair comparisons either, as it tends to favor existing methods that have been laboriously optimized.

Some baseline methods are conceptually simple and relatively easy to implement, such as Gauss-Seidel. Sparse permuted Cholesky decomposition is conceptually simple, and depending upon the availability of a well-written sparse matrix library package, could serve as a very good baseline performance metric[3]. See Fig 6 for an example of a graphical comparison to a baseline algorithm.

[3]Sparse permuted Cholesky decomposition is particularly sensitive to the implementation details of the matrix library. Even subtly sub-optimal methods can significantly inflate the runtime of the algorithm. In particular, variable reordering via a method like COLAMD is critical; see [13] for additional information.

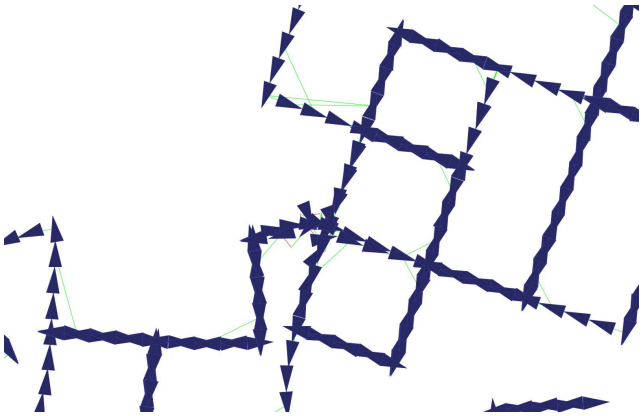

Fig. 7. Local Minimum. On the gridworld CSW dataset, Gauss-Seidel relaxation becomes stuck in a local minimum. Early iterations of the algorithm resulted in large and erratic changes to the state variables, creating a "knot" in the robot trajectory (note the oddly oriented nodes in the center of the figure). Gauss-Seidel will never be able to escape this local minimum.

*Our Recommendations*

- Theoretical complexity bounds should be specified whenever possible. These bounds should reflect actual worst-case performance on the worst possible input. If lower bounds are possible given specific assumptions (e.g. graph planarity, limited observation range), authors should clearly specify these assumptions along side their lowered bounds.
- If the optimization algorithm produces sub-optimal maps, this should be clearly stated along side the complexity bound.
- Experimental results should include overall execution time for batch methods, and amortized and worst-case time per step for online methods, together with all relevant hardware information. At a minimum, this should contain the processor type and frequency.
- Timing results for a simple reference algorithm on the same data and hardware should be provided for comparison.

## VI. Robustness Evaluation

It hardly matters how fast a map optimization method is if it does not find the correct answer. Methods can become trapped in local minima, and these local minima can reflect significant deformations of the correct map. Of course, when the global minimum is known, it is easy to determine when an algorithm has become stuck in a local minimum: the $\chi^2$ error will be too high. In general, algorithms are more likely to become stuck in local minima when there is significant noise in the observations and/or the initial estimate is poor.

Evaluating robustness requires implementers to try their methods on a large number of inputs: success on a single input, even a "difficult" input, is at best anecdotal. The behavior of the algorithm must also be characterized over a range of noise levels.

Using a synthetic problem generator, a large number of datasets can be generated. A new method can be compared to

an existing baseline method by plotting, as a function of noise level, the fraction of the datasets in which the method found the global (and not a local) minimum. In such an experiment, each algorithm should be allowed to run to convergence, and some $\chi^2$ or MSE tolerance should be allowed so as to not penalize methods that do not generally find the numerical minimum, even though they may find the basin of the global minimum.

*Our Recommendations*

- The robustness (or tendency to find the global minimum) of optimization algorithms is an important performance criterion and should be evaluated for new methods.
- A large number of synthetic data problems (thousands) should be generated, spanning a range of noise levels. The fraction of problems solved at each noise level should be plotted.
- Results for a baseline algorithm should also be provided.

## VII. Conclusion

We have described performance evaluation metrics for map optimization, identifying shortcomings in traditional metrics like $\chi^2$, and suggesting effective ways of presenting results.

In the future, we plan to assemble a benchmark dataset for map optimization, modeling it (in part) on the successful benchmarks found in related fields.

Of course, map optimization is just one part of the mapping problem. Other parts, like feature detection and loop closing, would also benefit from the development of better evaluation metrics and benchmark datasets.

## References


[1] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. The MIT press, Cambridge, MA, 2005.

[2] H. Durrant-Whyte and T. Bailey, "Simultaneous localisation and mapping (SLAM): Part I the essential algorithms," *Robotics & Automation Magazine*, Jun 2006.

[3] T. Bailey and H. Durrant-Whyte, "Simultaneous localisation and mapping (SLAM): Part II state of the art," *Robotics & Automation Magazine*, Sep 2006.

[4] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *International Journal of Computer Vision*, vol. 47, no. 1, pp. 7–42, May 2002.

[5] L. Fei-Fei, R. Fergus, and P. Perona, "One-shot learning of object categories," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 4, pp. 594–611, April 2006.

[6] A. Howard and N. Roy, "The robotics data set repository (Radish)," 2003. [Online]. Available: http://radish.sourceforge.net/

[7] P. Newman and P. Corke, "Editorial: Data papers – peer reviewed publication of high quality data sets," *International Journal of Robotics Research*, vol. 28, no. 5, p. 587, May 2009.

[8] U. Frese, "Treemap: An $\mathcal{O}(\log n)$ algorithm for simultaneous localization and mapping," in *Spatial Cognition IV*, C. Freksa, Ed. Springer–Verlag, 2005, pp. 455–476.

[9] M. Paskin, "Thin junction tree filters for simultaneous localization and mapping," University of California, Berkeley, Tech. Rep. UCB/CSD-02-1198, September 2002.

[10] E. Olson, J. Leonard, and S. Teller, "Fast iterative optimization of pose graphs with poor initial estimates," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2006, pp. 2262–2269.

[11] B. K. P. Horn, "Closed-form solution of absolute orientation using unit quaternions," *Journal of the Optical Society of America. A*, vol. 4, no. 4, pp. 629–642, Apr 1987.

[12] P. Krauthausen, F. Dellaert, and A. Kipp, "Exploiting locality by nested dissection for square root smoothing and mapping," in *Proceedings of Robotics: Science and Systems (RSS)*, 2006.

[13] M. Kaess, A. Ranganathan, and F. Dellaert, "iSAM: Incremental smoothing and mapping," *IEEE Transactions on Robotics"*, vol. 24, no. 6, pp. 1365–1378, Dec 2008.

[14] E. Olson, "Real-time correlative scan matching," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, May 2009.