Location Utility-based Map Reduction

Ted J. Steiner, Guoquan Huang, and John J. Leonard

***Abstract*— Maps used for navigation often include a database of location descriptions for place recognition (loop closing), which permits bounded-error performance. A standard pose-graph SLAM system adds a new entry for every new pose into the location database, which grows linearly and unbounded in time and thus becomes unsustainable. To address this issue, in this paper we propose a new map-reduction approach that pre-constructs a fixed-size place-recognition database amenable to the limited storage and processing resources of the vehicle by exploiting the high-level structure of the environment as well as the vehicle motion. In particular, we introduce the concept of *location utility* – which encapsulates the visitation probability of a location and its spatial distribution relative to nearby locations in the database – as a measure of the value of potential loop-closure events to occur at that location. While finding the optimal reduced location database is NP-hard, we develop an efficient greedy algorithm to sort all the locations in a map based on their relative utility *without* access to sensor measurements or the vehicle trajectory. This enables pre-determination of a generic, limited-size place-recognition database containing the N best locations in the environment. To validate the proposed approach, we develop an open-source street-map simulator using real city-map data and show that an accurate map (pose-graph) can be attained even when using a place-recognition database with only 1% of the entries of the corresponding full database.**

I. Introduction

In the popular pose-graph formulation of simultaneous localization and mapping (SLAM) [1], the map is represented by a graph of vehicle poses as nodes and measurements as edges. Consecutive poses are connected by (inferred) odometry measurements, and nonconsecutive poses whose appearances (i.e., the sensed information describing the locations) overlap are connected by loop-closure measurements that represent location revisits. Location appearances are stored in a database, and typically each pose contains a reference to an entry in this database. If a new pose's appearance matches an appearance already in the database, a loop-closure measurement will be added between the associated poses and subsequently be utilized to reduce the uncertainty of the map estimates. In the canonical pose-graph SLAM system, a new pose and location appearance are added at every time step. This causes the map to grow unbounded over time, which clearly is unsustainable. All location appearances are stored permanently on the chance that the vehicle returns to the location at some point in the unknown future. However, it is important to note that not all of these locations, and thus the associated loop-closure measurements, are equally informative for a navigational task. In practice, a vehicle often has stringent resources available, including data storage and computational power. Thus, it is desirable to know which locations will be the most useful in the future so that they can be given priority in a size-limited database.

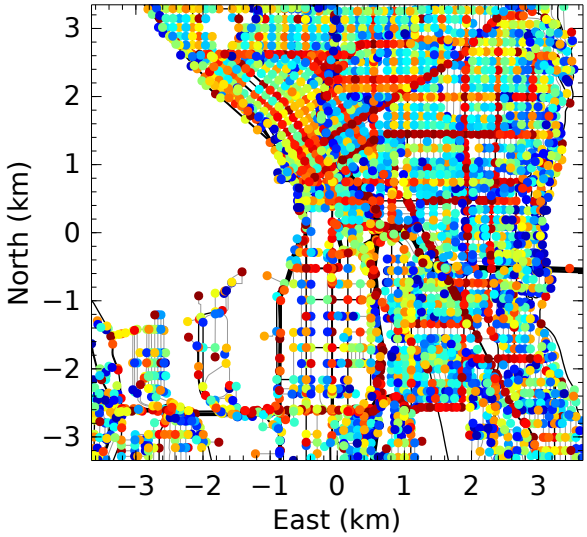


Fig. 1: Street intersections in *Seattle, Washington*, which are sorted based on their location utility as proposed in this paper. Such utility is a combination of the probability that a location is visited on any point-to-point shortest-time route through the environment and a spatial distribution term (see Section III), and is used to select a reduced location database for a storage/computation-constrained SLAM system, seeking to minimize map uncertainty without *a priori* knowledge of the vehicle trajectory. Note that in this graph, dark red denotes the maximum utility while dark blue is the minimum.

To better understand the need for reducing location databases in practice, consider a "Google Maps-like" system from which a vehicle can query routes and a minimal location database for a specified region. This database must be computed *a priori* in order to facilitate loop closure when navigating an unknown route within the region. This means that the database can not be delivered pre-associated to poses, so instead the first pose with an appearance match is associated to that database entry.

To that end, in this paper, we first introduce the notion of *location utility*, which is a measure of a location's informativeness based on its potential loop-closure measurements

T. Steiner is supported by a research fellowship from the Charles Stark Draper Laboratory in Cambridge, Massachusetts. All map data presented in this work is copyright OpenStreetMap contributors and available under the Open Database License.

T. Steiner and J. Leonard are with the Computer Science and Artificial Intelligence Laboratory (CSAIL), MIT, Cambridge, MA 02139, USA. Email: {tsteiner,jleonard}@mit.edu

G. Huang is with the Department of Mechanical Engineering, University of Delaware, Newark, DE 19716, USA. Email: ghuang@udel.edu

and is defined independently of any particular vehicle trajectory. The location utility encapsulates the structure of the environment and the vehicle motion provided by the knowledge of the vehicle's path planner and a high-level network model of potential paths in the environment. We show how limited, but commonly available, prior information can be used to significantly reduce the amount of data required for vehicle navigation in path-based environments, such as street networks (e.g., see Fig. 1). We further show how entries in a limited-size databased of location appearance can be selected to maximize the total utility of the database. Moreover, we provide an efficient approximation algorithm to sort all locations in a map according to their utilities, which allows for an optimal reduction in database size according to the memory limitations of a system. In particular, the main contributions of this work are the following:

- We, for the first time, introduce the concept of location utility, which is a measure of location's informativeness and is used to facilitate location selection, *independently* of vehicle trajectory and sensor measurements. Furthermore, based on this utility, we formulate map reduction as an integer programming (IP) problem.
- We develop an efficient greedy algorithm for (approximately) solving the formulated IP and thus the map reduction problem, which is, in general, NP-hard.
- We develop an open-source street-map pose-graph simulator using real city-map data. To validate the proposed approach, we perform extensive tests using this simulator with many different city maps.

Note that the problem of location database reduction treated in this paper is related to, but distinct from, three well-known problems: measurement selection, bounded mapping, and sensor placement. In particular, our approach formulates a sensor placement problem to maximize information content of loop-closure measurements of a subset of locations in the environment. Because our "sensor" locations are determined prior to navigation, we must place them according to *expected* measurements, different from other map reduction techniques. Because the loop-closure locations are pre-selected, the location database does not grow during operation, resulting in a partially bounded-size map. If desired, poses without loop-closure measurements can be marginalized out without consequence to achieve a truly bounded-storage map.

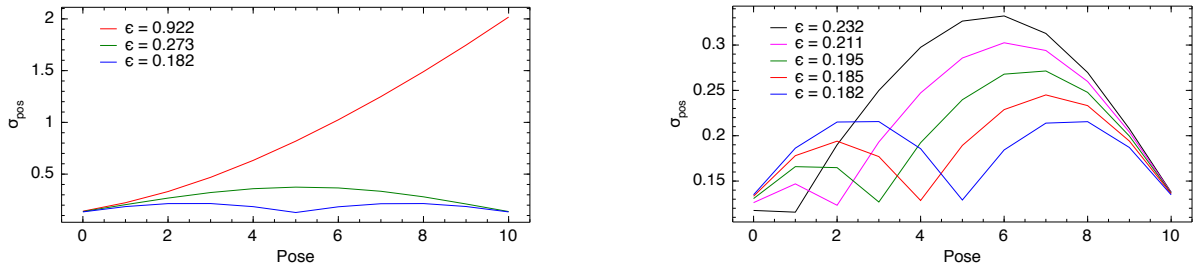
II. Related Work

Pose-graph SLAM entails solving a large least-squares problem over all poses. To do so efficiently, most solvers judiciously exploit sparsity in the problem structure [2]–[6]. This sparsity results from odometry constraints affecting only consecutive poses and loop-closures being occasional and limited. In some cases, such as long-duration, multi-session systems, the graphs grow too large to solve in real-time. To mitigate this issue, pose-graph sparsification algorithms can be used to reduce the number of poses and constraints in a map, thus improving efficiency. These algorithms work exclusively in the pose-graph solver, or "back-end." Carlevaris-Bianco et al. [7]–[10] consolidate densely connected regions of a pose-graph into Generic Linear Constraints (GLCs), while Huang et al. [11] use ℓ_1-optimization to consistently remove weak edges in the graph. Sparsification approaches have been used in conjunction with location saliency metrics [12] in order to weight the optimization towards retaining the database entries that are most likely to be recognized [13], but do not explicitly incorporate the likelihood of revisit. Note that all graph sparsification approaches have involved removing or replacing constraints after they have already been added to the graph. In contrast, in our database selection, we do not have access to the full graph when determining which *predicted* constraints to enable through the limited database.

Constraint selection has been applied in the SLAM front-end, which generates the odometry and loop-closure constraints. Keyframe-based approaches reduce the number of constraints in visual SLAM by keeping only a subset of measurement frames [14]. Kretzschmar et al. [15] estimated the mutual information of laser-scan measurements with regard to an occupancy grid, only incorporating new measurements when they are sufficiently informative. They marginalize out old poses to bound map growth to a fixed memory size. Ila et al. [16] only add measurements when they are sufficiently distant in information space. Wang et al. [17] use Kullbach-Leibler divergence to decide whether to incorporate a pose and its associated feature observations in feature-based SLAM. While effective at map *constraint* compression, all of these approaches first compute the actual constraint before determining whether or not to incorporate it into the map, whereas we do not have access to the actual constraints, and instead must evaluate the value of *potential* constraints. Furthermore, in order to enable future loop-closure constraints, these constraint selection approaches maintain full location databases even when constraints are not added to the graph, thus the databases still grow continuously in time.

Rather than focusing explicitly on map sparsification, other approaches have sought to bound map growth. The reduced pose-graph [18] bounds the number of poses (and thus the size of the location database) by area rather than time. Whenever a location is revisited, the system adds a constraint to the existing pose rather than establishing a new pose. RTAB-Map [19] bounds the map by computation required for search. They divide the location database into active and inactive locations, with only the active ones being searched for loop closure, and the inactive being promoted to the active when a nearby active location is visited. However, like the above methods, these approaches require access to the actual measurements in order to reduce the map.

In determining which locations to close loops, we introduce a new optimal sensor placement problem in the context of vehicle localization. Krause et al. formulate an optimal sensor problem by placing sensors to maximize the mutual information of sensed and unsensed locations using learned Gaussian processes [20]. Beinhofer et al. [21] and Allen et al.



(a) (b)

Fig. 2: Effects on error covariance for a sequence of 11 poses due to: (a) adding additional loop-closures, and (b) changing their spatial distribution. The average pose uncertainty ε (7b), which we aim to minimize, is provided in the legend. Note that in these plots, we have assumed that loop-closure constraints are added to already-well-estimated poses in the map, thus modeling them as location priors.

[22] formulate sensor placement problems to place artificial navigation landmarks and sonar beacons, respectively, along pre-defined trajectories for localization. Vitus and Tomlin [23] formulated an optimal sensor placement problem in which a vehicle deploys a limited set of sensor beacons in the environment to minimize its navigation estimation error. Our location database selection involves choosing a limited-size subset of the locations in the environment prior to navigation at which potential loop-closure measurements will be most beneficial to *any* trajectory through the environment.

III. Design of Utility-Based Location Selection

A SLAM solution seeks to minimize the errors of map estimates using odometry and loop-closure measurements. It is known that if only using odometry measurements, the SLAM estimation errors will grow unbounded, while this can be compensated by using loop-closure constraints to bound errors as well as reduce uncertainty. Fig. 2 shows an example of possible error covariances of a segment of a pose-graph containing 11 pose nodes, given various combinations of loop-closure constraints. If we assume that all loop-closure measurements are equally precise, the two sources of map covariance variation are the number and spatial distribution of loop-closure measurements. The average pose covariance is reduced as additional measurements are added (see Fig. 2a), and, for a fixed number of constraints, the average pose covariance is minimized when the constraints are maximally distributed (see Fig. 2b). This motivates us to introduce the following location utility metrics that will be used to sort and reduce the locations in the map.

A. Location Utility

The utility of a location is defined as being proportional to the covariance reduction for *all* poses in the map resulting from all predicted loop-closure measurements that involve that location. Therefore, location utility is composed of the following two factors:

- The probability of a location being visited and recognized for a given map.
- The covariance reduction due to loop closing at that location (which is approximately proportional to the distance traveled since the most recent loop closure).

Note that without knowledge of a specific trajectory, these two factors become independent, and reflect the total number and spatial distribution of the loop-closure measurements in the map. That is, the probability of a vehicle traversing a location in the map is independent of the spatial distribution of the location database as long as the location database is not used by the path planner.

Linear combination of the above two utility measures yields the total utility, u of a database location, ℓ, i.e.,

$$\mathrm{u}_\ell = \frac{v_\ell}{v_{max}} + \lambda \frac{d_\ell}{d_{max,\ell}} \tag{1}$$

where v_ℓ is the joint probability of a route visiting and recognizing the location, normalized by the maximum such probability for the region, v_{max}, d_ℓ is the minimum distance to another database location or the region boundary, normalized by the maximum distance to any database location, $d_{max,\ell}$, and λ is a tunable weighting parameter. Note that, while this does form the core definition of location utility, this definition is not necessarily complete. For example, it may make sense in some application-specific situations to include an additional term in (1) to represent the complexity at that location of the task to be performed relative to position uncertainty. In what follows, we explain these two utilities in detail.

1) Probability of visit: This probability encodes the environment and routing structure, and is the probability that a vehicle route within the environment passes through and recognizes a given location. Specifically, using Bayes rule, we define it as follows:

$$v = P(\text{visit})P(\text{recognize}|\text{visit}) = P(\text{visit}, \text{recognize}) \tag{2}$$

$P(visit)$, is obtained using a modified form of the betweenness centrality [24] of the network of traversable routes in the environment, in which the specified location node, ℓ, is additionally allowed to be the start, s, or end, t, of a network path. Specifically, for all routes from every location in the world to every other location, we count the number of times

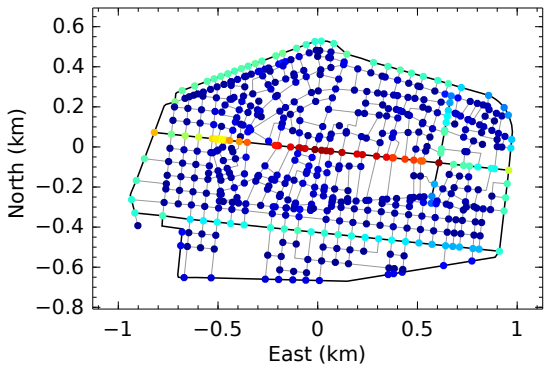


Fig. 3: Street map of Island of Malé (also see Fig. 7a), whose intersections are colored to show probability of visit (3), ranging from dark red (maximum) to dark blue (minimum).

each location is traversed, $n_{st,\ell}$, and divide this by the total number of routes in the region, n_{st}:

$$P(visit) = \sum_{s \neq t} \frac{n_{st,\ell}}{n_{st}} \tag{3}$$

These routes should be computed using the same method as the one used by the vehicle's path planner. If sample sensor measurements are available for locations in the environment, the location recognition probability can be estimated from a saliency score such as [12], otherwise we assume $P(\text{recognize}|\text{visit}) = 1$. An example of the probability of location visit in a street network is shown in Fig. 3.

2) Spatial distribution: The covariance reduction resulting from a loop closure in a map is approximately proportional to the distance that the vehicle has traveled since the last loop closure. This is due to the fact that accumulated odometry error is typically proportional to the distance traveled. A lower bound on this distance, d, can be computed as the minimum of either the Euclidean or shortest-path distance from the specified location, ℓ, to the nearest database entry, ℓ_D, and to the region boundary, B:

$$d_\ell = \min\left[\min\left(\text{dist}(\ell, \ell_D)\right),\ \lambda_B\ \text{dist}(\ell, B)\right] \tag{4}$$

where λ_B is used to reduce the impact of the region boundary and can be chosen with *a priori* knowledge.

It should be noted that computing both v and d requires that the potential routes in an environment are available beforehand, which is the case in most operational environments (e.g., streets, buildings, and warehouses). The database only stores the *appearance* of locations, while the *locations* of the specific database entries are estimated online. A prior on the positions of the database locations, if available, can be additionally provided.

B. Location Selection

Once we have determined the utilities for all the locations in the database, we now seek to maximize the database utility for a given number of locations. To this end, by specifying a maximum allowable database size, we can formulate the following nonlinear integer programming (IP) problem:

$$\begin{aligned} \arg\max_{\boldsymbol{\ell}} \quad & J := \boldsymbol{\ell}^T \mathbf{v} + \lambda \boldsymbol{\ell}^T \mathbf{d} \\ \text{subject to} \quad & \boldsymbol{\ell}^T \boldsymbol{\ell} = |D| \\ & \ell_i \in \{0,1\},\ v_i \in [0,1],\ d_i \in [0,1],\ \forall i \end{aligned} \tag{5}$$

where $\boldsymbol{\ell}$ is a vector of binary switching variables representing all possible database locations and $|D|$ is the size of the database. Vectors $\mathbf{v}$ and $\mathbf{d}$ are normalized by their maximum values to remove the unit conflict. λ is a tunable weighting parameter that determines the trade-off between exploration and exploitation, which is commonly encountered in robotics [25]. Specifically, $\lambda > 1$ indicates an emphasis on exploration, or spatial expansion, for database entries, while $\lambda < 1$ emphasizes exploitation of especially "good" locations, which are commonly revisited. This tradeoff decreases in significance as $|D|$ grows large.

IV. A Greedy Algorithm for Location Sorting

Directly solving the constrained optimization problem (5) is an instance of IP with binary switching variables, which is NP-hard [26]. Thus, in order to make location database sorting computationally tractable, we use a greedy approximation algorithm that iteratively adds the remaining location with the highest utility score to the database.

By greedily growing the database, we effectively assume that the optimal database of size N is included in the optimal database of size $N+1$. While this assumption does not generally hold, it does lead to a nice property. The information contained in an accurate loop-closure measurement is non-negative, i.e., adding a constraint to the graph cannot increase map uncertainty. We can extend this statement to say that adding a location to the location database without removal therefore guarantees at worst a null-effect on pose uncertainty, because for any possible trajectory, adding a location to the database can never result in fewer loop-closure detections. This means that the utility of a greedily-selected location database is nondecreasing with database size for *any* trajectory.

The greedy database growing problem is formulated as such: Given a list of all possible locations, L, and a database of locations, $D \subsetneq L$, find the optimal next location ℓ:

$$\begin{aligned} \arg\max_{\ell} \quad & v_\ell + \lambda d_\ell \\ \text{subject to} \quad & \ell \in L,\ \ell \notin D \end{aligned} \tag{6}$$

where $\mathbf{v}$ and $\mathbf{d}$ both contain elements normalized to range $[0,1]$ and are computed using Algorithms 1 and 2, respectively, and λ is the weighting parameter. It is important to point out that since Algorithm 1 runs in $\mathcal{O}(|L|)$, Algorithm 2 in $\mathcal{O}(|L||D|)$, and solving (6) in $\mathcal{O}(|L| - |D|)$, the full algorithm runs in $\mathcal{O}(|L||D|)$ for each iteration. We stress that the proposed map reduction strategy exploits the structure resulting from both the environment (a network of sparse

Algorithm 1 Compute normalized $\mathbf{v}$

Inputs:
Locations L, Location database D,
Visit probabilities $\mathbf{v}_L$
$\mathbf{v} \leftarrow \text{zeros}(|L|)$
for x in L **do**
  if $x \notin D$ **then**
    $\mathbf{v}[x] \leftarrow \mathbf{v}_L[x]$
  end if
end for
$\mathbf{v} \leftarrow \mathbf{v}/\max\ \mathbf{v}$

Algorithm 2 Compute normalized $\mathbf{d}$

Inputs:
Locations L, Location database D, Region boundary B, Boundary repulsion weight λ_B
$d \leftarrow \text{zeros}(|L|)$
for x in L **do**
  if $x \notin D$ **then**
    $\mathbf{d}_D \leftarrow \text{zeros}(|D|)$
    for k in D **do**
      $\mathbf{d}_D[k] \leftarrow \text{dist}(x, k)$ ▷ Euclidean distance
    end for
    $\mathbf{d}_D[|D|+1] \leftarrow \lambda_B\ \text{dist}(x, B)$
    $\mathbf{d}[x] \leftarrow \min\ \mathbf{d}_D$
  end if
end for
$\mathbf{d} \leftarrow \mathbf{d}/\max\ \mathbf{d}$

paths) and optimal route planning, and is especially intended for long-term persistent navigation.

V. Experimental Results

In this section, we demonstrate pose-graph SLAM using limited-size location databases computed by the proposed greedy algorithm presented in the previous section. Since an optimal location database is inherently trajectory-specific (i.e., a truly optimal database would contain only locations traversed by the trajectory), we must evaluate our reduction strategy for many potential trajectories within an environment. Unfortunately, there is no existing localization dataset that has these characteristics, which motivates us to develop a new dynamic simulator that is capable of generating relative-pose measurements for multiple trajectories through a network of traversable paths.

A. Street-Map Simulator

We built a brand-new routing and pose-graph simulator in `Julia` [27] that can simulate a taxi-like vehicle driving in a city through a succession of waypoints.[1] We use publicly-available real map data from `OpenStreetMap.org` to initialize a street network for a given region. Routes through the network are generated by randomly selecting 50 waypoints within the network and computing the fastest-time driving route through them respecting one-way streets with Dijkstra's algorithm using the `Julia OpenStreetMap.jl` package.[2]

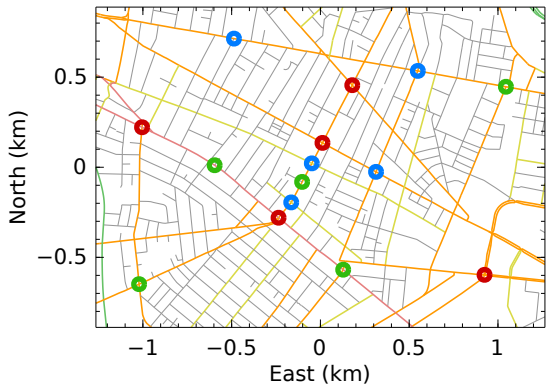


Fig. 4: Best 5 (red), 10 (red + green), and 15 (red + green + blue) locations selected for a minimal database for a region of Cambridge, Massachusetts.

For fastest-time routing, we assign reasonable speed-limits to the six classes of streets that are provided by `OpenStreetMap.org` (residential, freeway, etc.). For simplicity and without loss of generality, we assume that the streets are infinitesimally narrow.

Poses are established every 20 meters along the route and at every street intersection. Relative-pose constraints (odometry) are generated between all successive poses. Loop-closure constraints are generated whenever the vehicle traverses an intersection described by the location database. We assume that loop-closure detection is not directionally dependent and that loop-closures provide full-rank measurements. We limit our location database to only the street intersections within the region in order to reduce computational requirements, and loop-closure constraints are always added between the current pose and the first pose at that location. In order to facilitate direct comparison of location selection methods, our simulator first generates a route and all possible measurements (including Gaussian noise). Multiple pose-graphs can then be created for a common trajectory with identical measurement noise, adding loop-closure constraints according to any desired location database. The iSAM [2] and RISE [28] algorithms are then used to assemble and solve the pose-graph SLAM problem.

As our performance metric, we use the estimation error variance of position and trajectory, given by:

$$\sigma_{pos} = \sqrt{\mathbf{Q}_{pos,xx} + \mathbf{Q}_{pos,yy}} \tag{7a}$$

$$\varepsilon = \sum_{n=1}^{N} \frac{\sigma_{pos,n}}{N} \tag{7b}$$

where $\mathbf{Q}_{pos}$ is the marginal position covariance matrix of a pose. Note that (7a) gives the standard deviation of the

[1] Available at https://github.com/tedsteiner/PoseGraphSimulation.jl.git.

[2] Available at https://github.com/tedsteiner/OpenStreetMap.jl.git.

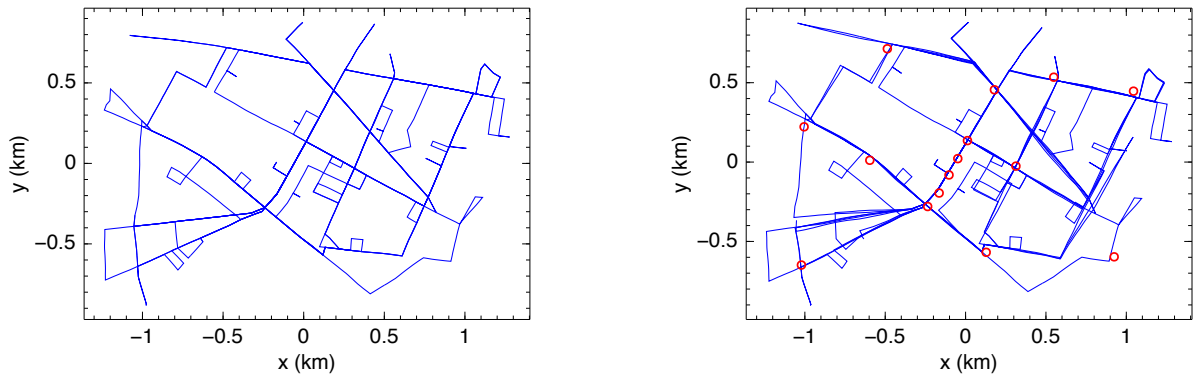


Fig. 5: Results of solving a simulated pose-graph representing driving around Cambridge, Massachusetts for approx. 75 km using: a) a location database containing all 712 street intersections, and b) a dramatically reduced database of only 15 pre-selected intersections (0.5% of the full database of 3,375 poses). The true positions of the selected database entries are shown in red. It is clear that, due to the high quality of the 15 selected locations, the topology of the map is still captured.

position estimate of a single pose, while (7b) is the average standard deviation over the trajectory. Moreover, because the performance of the limited location database is inherently route-dependent, we compute several routes (either 20 or 50, depending on the dataset) and use these routes to generate the associated pose-graphs for each dataset presented. In order to combine and compare these results, we normalize the mean trajectory error, ε, by the error achieved when using the "full" location database containing all street intersections in the map, ε_{full}:

$$\varepsilon_{ratio} = \frac{\varepsilon_{reduced} - \varepsilon_{full}}{\varepsilon_{full}} \tag{8}$$

For visualization, we display ε_{ratio} averaged over many randomly generated routes to eliminate trajectory-specific effects.

B. Location Utility-based Map Reduction

In Fig. 4, we depict the 5, 10, and 15 best loop-closure locations for a vehicle driving within a portion of Cambridge, Massachusetts, computed using the greedy location sorting/selection algorithm presented in Section IV.

In Fig. 5, we show two example pose-graph solutions for a 76.6 km trajectory through Cambridge, Massachusetts containing 3,375 poses. Fig. 5a is computed with all 712 street intersections described in the location database (roughly equivalent to 20% of the poses, though many locations are never actually encountered during the trajectory), and Fig. 5b is computed with only the 15 highest-utility locations (see Fig. 4) in the database. While the smaller database gives a noisier result, the underlying network structure is still distinctly visible, and the vehicle state knowledge is likely sufficient to carry out basic navigational tasks. For example, by inferring path crossings as intersections, uncertainty-cognizant pose-graph-based path planners, such as [29]–[31], could be used to plan routes within such a reduced graph.

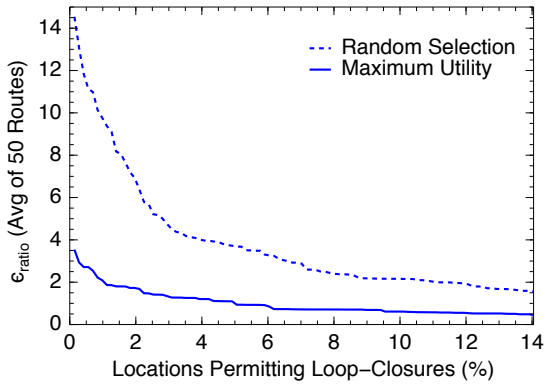
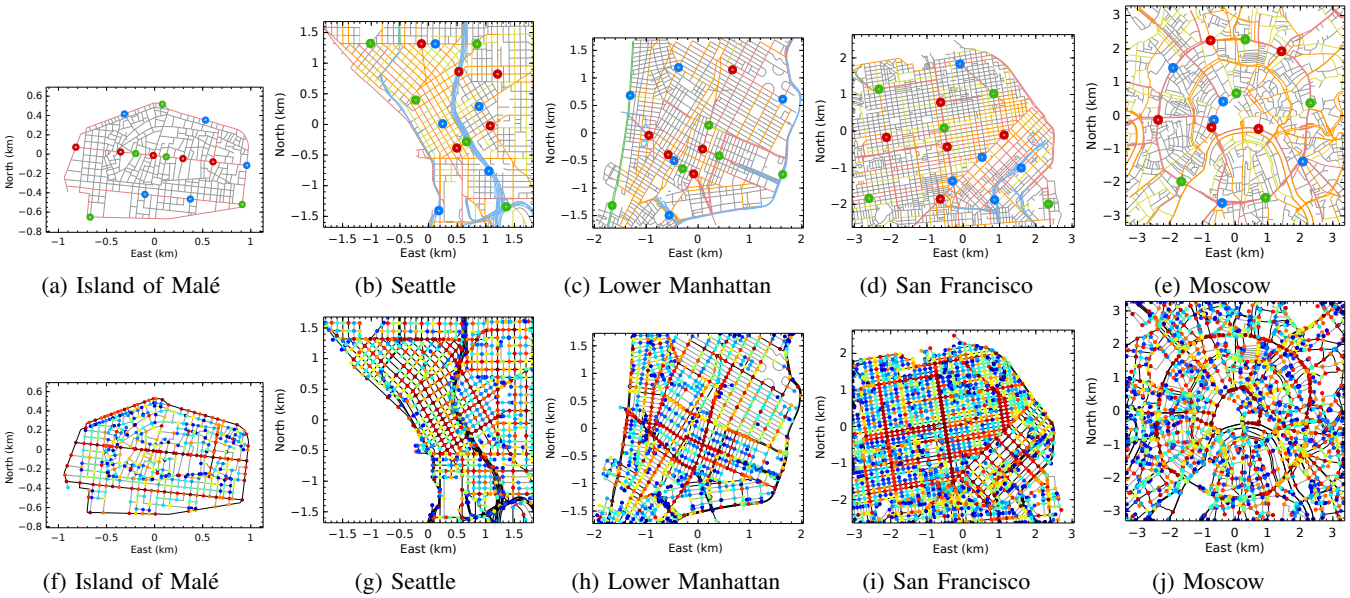


Fig. 6: Comparison of average pose uncertainty when using location databases selected using our method vs. randomly for a vehicle navigating in Cambridge, Massachusetts.

C. Comparison to Alternative Selection Methods

We additionally compare our approach to three alternative selection methods: (i) random selection, (ii) total loop-closure maximization, and (iii) maximal spatial distribution. Fig. 6 compares the error ratio of pose-graphs using our location-utility-based greedy database selection method to selecting random location databases for 50 routes, for databases containing up to 100 locations in Cambridge, Massachusetts (Fig. 4). Our approach results in significantly better accuracy than randomly selecting locations.

We further extend this comparison to five additional cities with various regular and irregular street networks. Figs. 7a-7e show the street maps and top 15 highest-utility locations for these cities, and Figs. 7f-7j show all street intersections in the map color-coded by their greedily-computed utility scores. Fig. 8 shows that our greedy selection algorithm performs well for all of these cases in comparison to random database selection, especially in the cases of the largest maps. Furthermore, we see that our algorithm favors both major intersections and locations spatially distributed between them



(a) Island of Malé (b) Seattle (c) Lower Manhattan (d) San Francisco (e) Moscow

(f) Island of Malé (g) Seattle (h) Lower Manhattan (i) San Francisco (j) Moscow

Fig. 7: The best 1-5 (red), 6-10 (green), and 11-15 (blue) street intersections (top) and all street intersections sorted based on loop-closure utility (bottom) for five major cities around the world, with maximum database sizes ranging from 535 to 2811 locations, increasing from left to right. In all cases, our algorithm favors both locations on major highways and locations spatially distributed evenly between them.

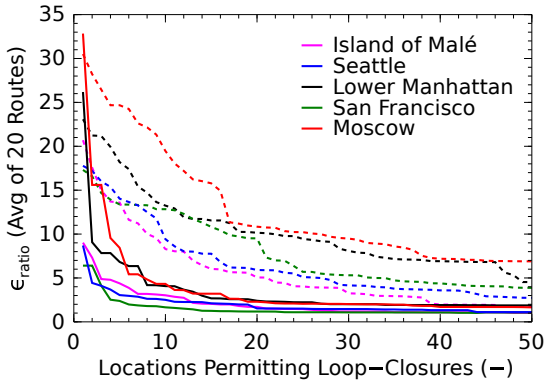


Fig. 8: Comparison of average pose uncertainty using locations selected by our method (solid lines) vs. randomly (dashed lines) for a vehicle navigating in five cities.

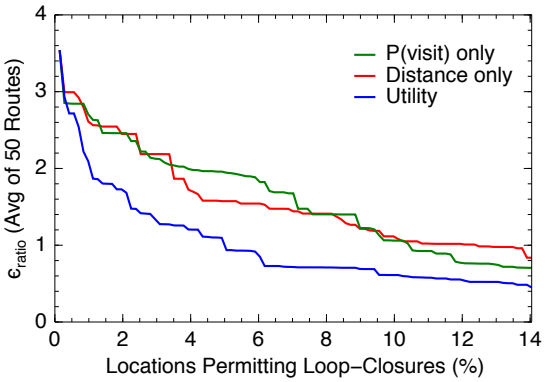


Fig. 9: Comparison of the effects of the two objectives that form the "location utility" on pose smoothing covariance. The green line determines location utility using only visit probability (v), the red line uses only distance (d), and the blue line uses an equal combination of the two terms. Note that all three approaches are initialized using the location with maximum $\mathrm{P(visit)}$ for $|D| = 1$.

for all city styles tested.

We also compare our results to two alternative database selection methods: (i) maximizing the predicted number of loop closures, and (ii) maximizing spatial distribution between database entries. These are the two extreme cases [see (6)]: $\lambda = 0$ (pure exploitation) and $\lambda = \infty$ (pure exploration). Our algorithm offers a balance between these two effects, $\lambda = 1$ (equilibrium). Fig. 9 shows ε_{ratio} for each of these cases, averaged over 50 fastest-path routes with identical measurement noise. We see that both of these terms are independently valuable, but our balanced combination outperforms both approaches. In general, we found the results to be fairly insensitive to the specific value of λ chosen.

VI. Conclusions and Future Work

Our results have shown that the locations at which loop-closure events occur can be sorted according to their expected reduction in map uncertainty for a general, unknown route, even without access to the pose marginal covariances or actual measurements. In particular, the proposed location utility consists of the visit probability and spatial distribution of database locations. We have shown that the structure of a network environment and vehicle motion can be encoded in our measure of location utility, which in turn improves reduced-size location database performance. Using location

utility to construct limited-size location databases, an accurate pose-graph can be constructed even when using a location database less than 1% the size of that of a typical pose-graph SLAM system. The proposed greedy-selection algorithm can additionally be used as a design tool to estimate the minimum data storage required to navigate in an environment within a specified position uncertainty.

While this work is limited to *a priori* computation of location databases, which requires vehicle routing information and the network of potential paths, it opens the door to future work such as learning the environment structure online and managing a reduced-size location database in real-time (i.e., when to replace locations in a fixed-size database).

REFERENCES

[1] G. Grisetti, R. Kummerle, C. Stachniss, and W. Burgard, "A tutorial on graph-based SLAM," *IEEE Intelligent Transportation Systems Magazine*, vol. 2, no. 4, pp. 31–43, 2010.

[2] M. Kaess, A. Ranganathan, and F. Dellaert, "iSAM: Incremental smoothing and mapping," *IEEE Transactions on Robotics*, vol. 24, no. 6, pp. 1365–78, 2008.

[3] R. Kuemmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "g2o: A general framework for graph optimization," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2011.

[4] S. Thrun, Y. Liu, D. Koller, A. Y. Ng, Z. Ghahramani, and H. Durrant-Whyte, "Simultaneous localization and mapping with sparse extended information filters," *The International Journal of Robotics Research*, vol. 23, no. 7-8, pp. 693–716, 2004.

[5] R. Eustice, H. Singh, and J. Leonard, "Exactly sparse delayed-state filters for view-based SLAM," *IEEE Transactions on Robotics*, vol. 22, no. 6, pp. 1100–14, 12 2006.

[6] J. Vial, H. Durrant-Whyte, and T. Bailey, "Conservative sparsification for efficient and consistent approximate estimation," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011, pp. 886–93.

[7] N. Carlevaris-Bianco and R. M. Eustice, "Generic factor-based node marginalization and edge sparsification for pose-graph SLAM," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2013, pp. 5748–55.

[8] ——, "Long-term simultaneous localization and mapping with generic linear constraint node removal," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 1034–41.

[9] N. Carlevaris-Bianco, M. Kaess, and R. M. Eustice, "Generic node removal for factor-graph SLAM," *IEEE Transactions on Robotics*, 2014.

[10] N. Carlevaris-Bianco and R. M. Eustice, "Conservative edge sparsification for graph SLAM node removal," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Hong Kong, China, 6 2014, pp. 854–860.

[11] G. Huang, M. Kaess, and J. J. Leonard, "Consistent sparsification for graph optimization," in *Proceedings of the European Conference on Mobile Robots*, 2013, pp. 150–7.

[12] A. Kim and R. M. Eustice, "Combined visually and geometrically informative link hypothesis for pose-graph visual SLAM using bag-of-words," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011, pp. 1647–54.

[13] P. Ozog and R. M. Eustice, "Toward long-term, automated ship hull inspection with visual SLAM, explicit surface optimization, and generic graph-sparsification," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Hong Kong, China, 6 2014, pp. 3832–9.

[14] K. Konolige and M. Agrawal, "FrameSLAM: From bundle adjustment to real-time visual mapping," *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 1066–77, 2008.

[15] H. Kretzschmar and C. Stachniss, "Information-theoretic compression of pose graphs for laser-based SLAM," *The International Journal of Robotics Research*, vol. 31, no. 11, pp. 1219–30, 2012.

[16] V. Ila, J. M. Porta, and J. Andrade-Cetto, "Information-based compact pose SLAM," *IEEE Transactions on Robotics*, vol. 26, no. 1, pp. 78–93, 2010.

[17] Y. Wang, R. Xiong, Q. Li, and S. Huang, "Kullback-Leibler divergence based graph pruning in robotic feature mapping," in *Proceedings of the European Conference on Mobile Robots*, 2013, pp. 32–7.

[18] H. Johannsson, M. Kaess, M. Fallon, and J. J. Leonard, "Temporally scalable visual SLAM using a reduced pose graph," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2013, pp. 54–61.

[19] M. Labbe and F. Michaud, "Appearance-based loop closure detection for online large-scale and long-term operation," *IEEE Transactions on Robotics*, vol. 29, no. 3, pp. 734–45, 2013.

[20] A. Krause, A. Singh, and C. Guestrin, "Near-optimal sensor placements in Gaussian processes: Theory, efficient algorithms and empirical studies," *The Journal of Machine Learning Research*, vol. 9, pp. 235–84, 2008.

[21] M. Beinhofer, J. Müller, and W. Burgard, "Effective landmark placement for accurate and reliable mobile robot navigation," *Robotics and Autonomous Systems*, vol. 61, no. 10, pp. 1060–9, 2013.

[22] R. Allen, N. MacMillan, D. Marinakis, R. I. Nishat, R. Rahman, and S. Whitesides, "The range beacon placement problem for robot navigation," in *2014 Canadian Conference on Computer and Robot Vision*, 2014, pp. 151–8.

[23] M. P. Vitus and C. J. Tomlin, "Sensor placement for improved robotic navigation." in *Robotics: Science and Systems*, 2010.

[24] L. C. Freeman, "A set of measures of centrality based on betweenness," *Sociometry*, pp. 35–41, 1977.

[25] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. MIT press, 2005.

[26] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein *et al.*, *Introduction to algorithms*. MIT Press Cambridge, 2001, vol. 2.

[27] J. Bezanson, S. Karpinski, V. B. Shah, and A. Edelman, "Julia: A fast dynamic language for technical computing," *arXiv Computing Research Repository*, vol. abs/1209.5145, 2012.

[28] D. Rosen, M. Kaess, and J. Leonard, "RISE: An incremental trust-region method for robust online sparse least-squares estimation," *IEEE Transactions on Robotics*, vol. PP, no. 99, 6 2014.

[29] R. Valencia, J. Andrade-Cetto, and J. M. Porta, "Path planning in belief space with pose SLAM," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2011, pp. 78–83.

[30] R. Valencia, M. Morta, J. Andrade-Cetto, and J. M. Porta, "Planning reliable paths with pose SLAM," *IEEE Transactions on Robotics*, vol. 29, no. 4, pp. 1050–9, 8 2013.

[31] H. Carrillo, Y. Latif, J. Neira, and J. A. Castellanos, "Fast minimum uncertainty search on a graph map representation," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 2504–11.