Lifelong Mapping using Adaptive Local Maps

Nandan Banerjee*, Dimitri Lisin, Jimmy Briggs, Martin Llofriu, and Mario E. Munich

***Abstract*— Occupancy mapping enables a mobile robot to make intelligent planning decisions to accomplish its tasks. Adaptive local maps is an algorithm which represents the occupancy information as a set of overlapping local maps anchored to poses in the robot's trajectory. At any time, a global occupancy map can be rendered from the local maps to be used for path planning. The advantage of this approach is that the occupancy information stays consistent despite the changes in the pose estimates resulting from loop closures and localization updates. The disadvantage, however, is that the number of local maps grows over time. For long robot runs, or for multiple runs in the same space, this growth will result in redundant occupancy information, which will in turn increase the time it takes to render the global map, as well as the memory footprint of the system. In this paper, we propose a novel approach for the maintenance of an adaptive local maps system, which intelligently prunes redundant local maps, ensuring the robustness and stability required for lifelong mapping.**

I. Introduction

Occupancy grids provide a convenient representation of the environment which allows an autonomous mobile robot to plan a path from its present position to its intended destination. In SLAM systems using views or sparse feature maps, such as monocular visual SLAM [1], localization information is combined with data from other sensors (e.g. IR, ultrasonic, or bumpers) to build an occupancy grid. If the localization information from the SLAM system is time invariant, meaning that the robot's pose estimation does not change over time, then a single occupancy grid solution can be easily implemented to capture the environment.

However, many popular SLAM systems are graph-based and are not time invariant, because they refine previous pose estimates as they acquire new information. These changes to past pose estimates invalidate the single occupancy grid approach as they may produce an inconsistent state. An elegant solution to this problem was proposed by Llofriu et al. [2]. Their method constructs overlapping local sub-maps, which are anchored to nodes of a SLAM graph. These anchored nodes keep the occupancy map consistent with changing trajectory estimates during loop closures and graph optimization operations. At any time, a global occupancy map can be rendered from the local maps and used for path planning. This global map conveniently encodes a snapshot of the robot's knowledge of its environment at a single point in time.

This method works well for individual robot runs where maps are created from scratch. But in the situation of lifelong mapping on a robot, i.e., when the robot continuously updates its map over multiple runs in the same space, the number of local maps will increase over time as shown in in Fig. 1 and Fig. 2. The reason for this growth stems from the way the local maps are created.

*The authors are with the iRobot Corporation, Bedford, MA 01730, USA. {nbanerjee, dlisin, jbriggs, mllofriu, mmunich}@irobot.com

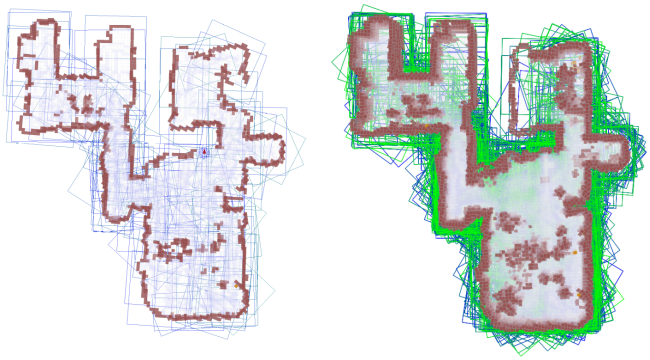

Fig. 1: We propose a method for eliminating redundant local maps without creating discontinuities in the rendered global map (left). Not having this pruning in a lifelong mapping scenario results in an overgrowth of redundant local maps, increasing memory and CPU consumption.

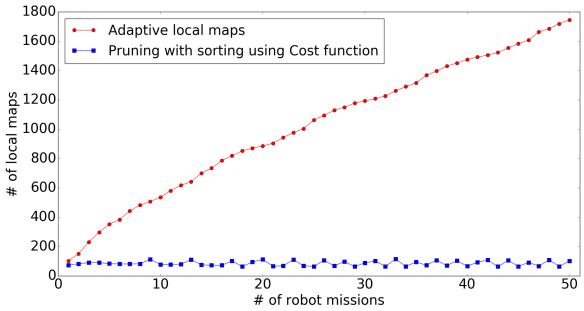

Fig. 2: Growth of the number of local maps with and without pruning across 50 robot missions in the environment from Fig. 1. Without pruning, the total number of local maps increases linearly. With pruning using our method, the total number of local maps stays stable over time.

As the robot travels, it searches for local maps that are *nearby*. We define nearby local maps as those which are anchored to graph nodes whose poses can be calculated relative to the robot's current pose with high certainty. In some cases the robot may not be able to find a nearby local map, either because it has moved into uncharted space, or because the graph contains no recent loop closures – often a result of illumination changes or movement of furniture. In

either case, the robot's pose has become too uncertain with respect to the existing local maps. Unable to find a nearby local map, the robot must generate a new one.

Over the lifetime of a mobile robot, the growth of local maps leads to an increase in the memory requirements and also the rendering time for the global occupancy map. It is clear that we must discard some of the local maps to prevent this growth. However, it is not immediately clear how to efficiently decide which local maps can be removed without losing information and creating discontinuities in the rendered global occupancy grid.

In this paper, we present an algorithm for local map maintenance, which enables the robot to work effectively in lifelong mapping scenarios. We devise a cost function which allows us to decide which local maps can be pruned at the end of each mapping run without sacrificing the completeness and accuracy of the rendered global map. We can thus prevent the number of local maps from growing over time, which in turn stabilizes the memory footprint of the occupancy mapping system, and also the amount of time it takes to render the global map.

II. Related Work

Mobile robots invariably require information about their surroundings for use in path planning, manipulation, and human interaction. Consequently, there is a staggering quantity of research on the topic of simultaneous localization and mapping (SLAM) for mobile robots. We discuss the existing research most relevant to our approach in this section, focusing on methods applicable to visual SLAM and occupancy grid mapping.

Environmental data is most commonly organized into geometric maps for use in mobile robotics. Konolige and Bowman proposed the term *lifelong maps* to describe maps which can be updated to represent a changing environment and which can recover from localization failure [3]. The lifelong mapping literature often focuses on the vision side of the SLAM problem, aiming to control the number of views in a given map. Strategies are myriad, but include view-clustering algorithms [3], [4], [5], estimates of view quality [5], [6], and summarization of map data from individual mapping runs [7], [8], [9]. These techniques, while effective for view management, do not extend naturally to the occupancy data for which our approach is designed.

Occupancy grids, proposed by Moravec and Elfes [10], refer to a number of structures which store dense occupancy estimates of a mobile robot's environment. The density of this information makes it especially appealing for path planning, where unexpected obstacles can lead to unrecoverable failure. Early occupancy grids employed rasterized arrays estimating purely static environments [11]. More complex techniques have reduced the redundancy of rasterized information. Octree-based Octomaps [12] have gained great traction both for their memory efficiency and their natural optimizations for use with laser scan data [13], [14]. Additionally, compressed maps which simulate optical permeability [15], or sparsify occupancy data [16], [17], [18] have been developed. These methods reduce the total memory usage and runtime of occupancy mapping, but do not provide a mechanism for lifelong updates to the occupancy grids.

Additional work in occupancy grid mapping drops the assumptions of a single agent in a purely static environment. Several methods have been suggested for handling dynamic occupancy data. One method employs fuzzy logic to classify occupied cells as static, quasi-static, or moving [19], while another removes nonstationary objects with a spatially aware binary classifier [20]. The merging of data from multiple agents has been achieved more ways than can be enumerated, with highlights featuring dynamic programming [21], advanced uncertainty modeling [22], and numerical optimization [23]. These methods are powerful, but ill-suited to an independent, resource-constrained platform, whereas our algorithm runs in real time on an embedded system.

One of the most successful fusions of lifelong mapping and occupancy grids has come in the form of *local maps*, a scheme in which the global environmental map is segmented into many independent sub-maps [24]. The idea of local maps has been improved by many researchers. The ATLAS framework, for instance, introduced modular components for the many sub-tasks involved in managing a local maps system [25]. The relaxation of the independence criterion of local maps allows sub-maps to share data [26]. This observation allows multiple sub-maps to be combined into one when there is a large overlap in features between them [27]. Hybrid maps, another popular extension of local maps, feature a global topological graph with local maps superimposed over it [28]. This scheme allows for fine detail on the local maps used for path planning, but requires less computation than a fully global map.

Our work is most closely related to the system proposed in [2], a hybrid mapping system which adaptively incorporates changing trajectory estimates into occupancy information. Our algorithm is also compatible with solutions like VOG-maps [29] which track 3D free-space data in octree-based local maps. While these systems make important strides toward lifelong mapping, their memory footprints increase without bound over the course of many runs. By contrast, our algorithm scales in space rather than time while preserving other desirable properties of adaptive local maps for lifelong mapping.

III. Method

The primary contribution of this paper is an algorithm which manages an adaptive local mapping system by intelligently pruning sub-maps. In this section, we present the approach used to select local maps that can be pruned without altering the occupancy information in the rendered global map.

A. SLAM System

The adaptive local maps algorithm relies on a graph-based SLAM system, such as one proposed by Eade et al. [1]. The SLAM system anchors each local map to a pose from

the robot's estimated trajectory. Thus, when pose estimates are updated by loop closures and graph optimization, the positions and orientations of local maps are updated as well. The SLAM system must also be able to estimate the relative uncertainty of the estimated transformation between any two poses. This uncertainty data determines when a new local map must be created.

B. Occupancy Mapping System

Our occupancy maps are based on the adaptive local mapping system. Let a *trajectory* be given by a set of poses p_t, and corresponding sensory data s_t indexed by time $t \in \{1,\ldots,T\}$. Then we can define a family of occupancy functions f_t which represent an estimate of the occupancy information of a given location (x,y) given information available at time t.

$$f_t : \mathbb{R} \times \mathbb{R} \to \{\text{free}, \text{occupied}, \text{unknown}\} \tag{1}$$

Occupancy information can be modeled probabilistically. However, for simplicity, we assume discrete cell occupancy values, *free*, *occupied*, and *unknown*. Let I be an inference function that takes a robot pose estimate, robot sensory information, and an x- and y-coordinate as inputs and estimates an occupancy value. f_1 can be trivially defined in terms of I.

$$f_1(x,y) = I(p_1, s_1, x, y) \tag{2}$$

Let a combination function c be defined as a function which takes a sequence of occupancy estimates of one cell for various t and outputs a synthesized occupancy estimate. Now depending on our choice of I and c, we can define an arbitrary occupancy function at time T as in (3).

$$f_T(x,y) = c(I(p_1, s_1, x, y), \ldots, I(p_T, s_T, x, y)) \tag{3}$$

Such a mapping function can be used effectively with a single occupancy grid, if the pose estimates p_t do not change after they are first computed. In graph-based SLAM systems, however, pose estimates do change over time as new observations are made. To account for that, we store occupancy information in collection of local maps, which we denote by $\left\{\ell_t^i \mid i \in \mathbb{N} \leq L\right\}$. The local maps are anchored to nodes in the pose graph, and move together with the pose estimates as the graph is optimized.

Local maps are updated by combining the cell value at (x,y) stored in the local map with the occupancy value generated by the inference function I. We can denote this update with another combination function indexed by i and t.

$$\ell_{t+1}^i(x,y) = c_t^i(\ell_t^i(x,y), I(p_t, s_t, x, y)) \tag{4}$$

The mapping function f_t at (x,y) using local maps now becomes a combination of all the local maps that contribute to the occupancy information of (x,y), that is

$$f_t(x,y) = c_t(\ell_t^1(x,y), \ldots, \ell_t^L(x,y)). \tag{5}$$

Let p_t be the robot's current pose estimate. Let p_g be the pose of the graph node associated with a local map. Let $\Sigma(p_t, p_g)$ be the relative uncertainty between these two estimates. Let $\text{tr}(M)$ denote the sum of the diagonal elements of matrix M. Then we will consider the local map to be *nearby* if $\text{tr}(\Sigma(p_t, p_g)) < \sigma_{min}$, where σ_{min} is a threshold.

At each point in time, the robot tries to find a nearby local map if one exists. If a nearby local map cannot be found, a new local map is created and anchored to the robot's latest pose node. Once a local map is found or created, the robot marks the corresponding cell in that map as *free* or *occupied* based on the inference function I. In [2], a distance threshold is also used along with the uncertainty threshold, we omit it from our method, as our experiments did not show it to be useful.

The final part of the system consists of a rendering function that combines all or a subset of the local maps as required and renders occupancy information onto a complete or a partial rendered map. This map can be used for robot path planning, room segmentation, human interaction, and a host of other applications. Conveniently, the occupancy function f_t is a rendering function by construction. Thus we can create global occupancy map G from our local maps by the rule in (6).

$$G_{x,y} = f_t(x,y) \tag{6}$$

The evaluation of $G_{x,y}$ depends linearly upon the number of maps L. The increase in render time as the number of local maps increase in the system is shown in Fig. 3. In order to bound rendering times, it is vital to keep L small.

C. Growth of local maps over time

Sometimes, localization systems cannot find the transform between current and past poses with high certainty when revisiting the same area. A common situation that could lead to failure occurs when a robot running a visual SLAM system is unable to observe visual landmarks in an area because of poor lighting conditions. In such cases, the robot is forced to create new local maps which occupy the same space as existing local maps. Since the robot cannot guarantee

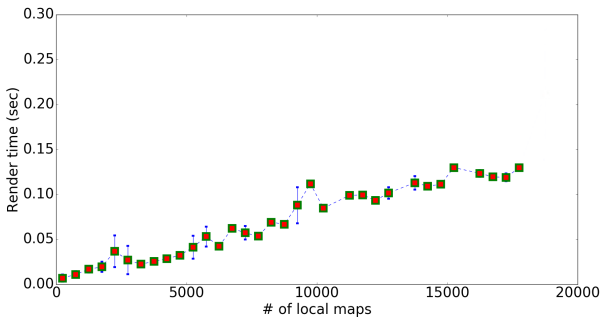

Fig. 3: The time it takes to render a global occupancy grid $G_{x,y}$ grows as a function of the number of local maps. The times were averaged over bins of 500 local maps.

consistent observations in these cases, the number of local maps can and will grow without bound.

We use $u(\ell_t^i)$ to denote the relative uncertainty of the local map ℓ_t^i. We define *path uncertainty* as the product of uncertainties of each edge along a path in the SLAM graph. Then $u(\ell_t^i)$ is equal to the minimum path uncertainty for any path between ℓ_i's SLAM graph node and any landmark's SLAM graph node. In our system, uncertainty is represented with covariance matrices. If ${}^a\Sigma^b$ denotes the shortest path covariance between nodes a and b in the SLAM graph, and $lm*$ denotes the landmark corresponding to ℓ_t^i, we arrive at (7).

$$\begin{aligned} u(\ell_t^i) &= \min_{lm \in \mathbb{LM}} \text{tr}(\text{shortest\_path\_uncertainty}(\ell_t^i, lm)) \\ &= \text{tr}({}^{lm*}\Sigma^{l_t^i}) \end{aligned} \tag{7}$$

To be able to successfully perform lifelong mapping over hundreds of robot runs in an environment with adaptive local maps, there needs to be a check on the growth of local maps. We discuss our approach to pruning of these local maps in the following sub-section.

D. Pruning of local maps

Pruning of local maps is essential to enable a lifelong mapping system using adaptive local maps. This step can usually be performed at the end of a mapping run. Let $\mathscr{L}$ denote the set of all local maps ℓ_T^i, and let S be the subset of local maps which should be pruned. For any collection of local maps $X \subseteq L$, let $q(X)$ be a measure of the quality of the rendered global grid constructed from X. In our system, $q(X)$ is defined as the difference in number of occupied cells between the global grid rendered from L and the global grid rendered from X. Finally, let ε_q be a maximum quality loss threshold specified by the user. The objective of the pruning algorithm is to maximize the total number of local maps to be pruned while being constrained by the quality of the rendered map that is produced by the remaining set of local maps. Equation 8 formulates this objective as an optimization problem.

$$\begin{aligned} &\underset{S}{\text{maximize}} \quad |S| \\ &\text{subject to} \quad q(\mathscr{L}) - q(\mathscr{L} \setminus S) \leq \varepsilon_q \end{aligned} \tag{8}$$

Since the number of ways the pruning set S can be constructed increases exponentially with the number of local maps in $\mathscr{L}$, it is not feasible to search exhaustively for an optimal S. Instead, we begin with S empty and then repeatedly add local maps from $\mathscr{L}$ whose addition to S does not violate the quality constraint. After a local map is added to S, it is removed from $\mathscr{L}$. While straightforward, this naive pruning algorithm may not produce the optimal S, as this way of constructing S is dependent on the ordering of the checking and addition of local maps from $\mathscr{L}$. Therefore, we must find the ordering of local maps which will expose the best pruning set.

We define a cost function for a local map that describes the amount of information provided by the local map to the final rendered grid. If the cost of a local map is high, the contribution of that local map to the rendered grid is low, and if the cost is low, the map is important, with a meaningful contribution to the rendered grid.

Let $s(x,y)$ denote the total number of local maps that make a contribution to the final rendered cell at (x,y). Note that $s(x,y)$ is similar to the mapping function in (5) except that instead of combining all the information available, it simply counts the number of local maps involved. We also define a function $p(x)$ that gives us a multiplicative factor to calculate the cost.

$$p(x) = \begin{cases} gain, & \text{if } x > 1 \\ penalty, & \text{if } x = 1 \end{cases} \tag{9}$$

The product of $s(x,y)$ and p for a particular cell $s(x,y)$ of a local map ℓ_t^i is summed over all (x,y) cells in ℓ_t^i, this gives the cost for the local map ℓ_t^i as in (10). If the contribution $s(x,y)$ of a rendered cell (x,y) is from a single local map $(x=1)$, a negative penalty is applied, and a positive gain is applied otherwise. The negative penalty makes sure the cost stays low for local maps that have less overlaps with other local maps, making them less of a candidate for pruning. The relative uncertainty of the local map is also incorporated in the cost with a weight λ.

$$Cost(\ell_t^i) = \left\{ \sum_{(x,y) \in \ell_t^i} p(s(x,y)) \times s(x,y) \right\} + \lambda u(\ell_t^i) \tag{10}$$

Algorithm 1 Local map pruning

Require: List $\mathscr{L}$ of all local maps ℓ_t^i, $p(x)$, $s(x)$, λ

Local maps to be pruned: $S \leftarrow \{\varnothing\}$

// Calculate the cost of all the local maps in S

for $\ell_t^i \in \mathscr{L}$ **do**

  $\text{cost}[\ell_i^t] \leftarrow \left\{ \sum_{(x,y) \in \ell_t^i} p(s(x,y)) \times s(x,y) \right\} + \lambda u(\ell_t^i)$

// Sort $\mathscr{L}$ in a descending order based on local map cost

$\mathscr{L}_{sorted} \leftarrow \text{sort}(S, \text{cost})$

// Check constraint, add local map to S if within ε_q

for $(i = 0;\ i < len(\mathscr{L}_{sorted});\ i = i+1)$ **do**

  $\ell \leftarrow L_{sorted}[i]$

  $S_t \leftarrow S \cup \{\ell\}$

  if $|q(\mathscr{L}) - q(\mathscr{L} \setminus S_t)| \leq \varepsilon_q$ **then**

    $S \leftarrow S_t$

return S

Algorithm 1 details the local map pruning process. The cost of every local map in $\mathscr{L}$ is calculated, and $\mathscr{L}$ is then sorted in a descending order based on the cost and stored in $\mathscr{L}_{sorted}$. Local maps are then selected from $\mathscr{L}_{sorted}$ with the local map with the highest cost selected first, all the way to the local map with the least cost at the very end. At every

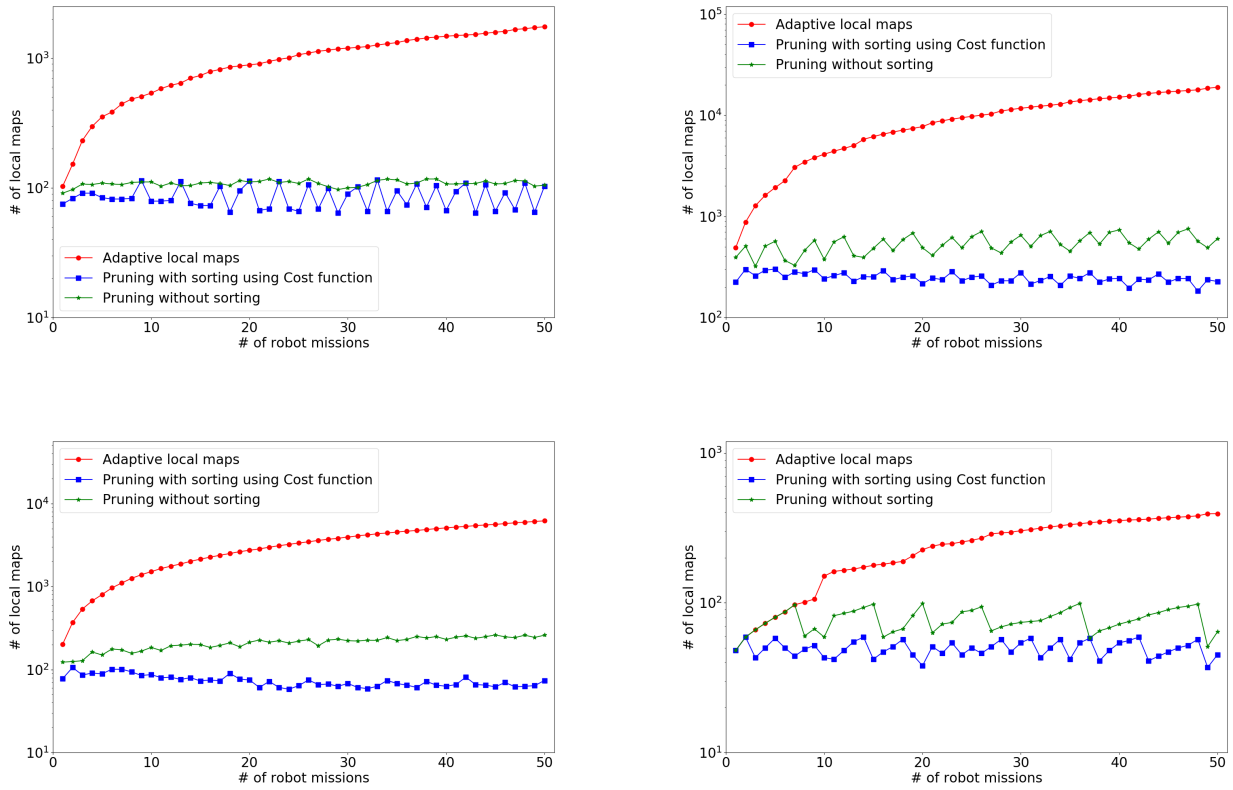

Fig. 4: Log-scale comparison of the numbers of adaptive local maps with and without pruning. The red line shows the unbounded growth of local maps when no pruning is used. A naive pruning approach, i.e., one without using the cost function to sort the local maps, yields a set of local maps stable across runs, shown as the green line. Blue line shows pruning of the local maps with our proposed method of using the cost function to first sort the local maps, and then pruning them in that order. Our method performs better than the naive method as the overall number of local maps are lower than the naive method across all runs. Mapped area - Env. A: 656 ft^2, Env. B: 516 ft^2, Env. C: 297 ft^2, Env. D: 285 ft^2. Clockwise from top (Env. A, B, D, and C).

iteration, a local map from $\mathscr{L}_{sorted}$ is added to the pruning set S which at the beginning is empty. Local maps in $\mathscr{L}$ are rendered, and the render quality is calculated. Then, the local maps in $\mathscr{L} \setminus S$ are rendered, and the render quality is calculated. The difference between the two render qualities is compared with ε_q, and if it is found to be less than ε_q (checking the constraint), the selected local map is removed from $\mathscr{L}$ and put into S permanently, i.e., that local map is marked for pruning.

IV. Experiments and Results

Extensive experiments and analysis have been performed to evaluate the local map pruning algorithm's performance. Run-time logs of raw sensor data were collected using a proprietary robot vacuum cleaning platform in different environments at different times of day with varied lighting conditions. These logs contained timestamped odometry, gyroscope data, and images from the robot's camera. Although most robots started from and finished at the same location (e.g. a dock or a charging station), some logs began at random locations in the environment and ended on a dock. We ran our SLAM system off-line on these logs. After running the system on each log, we saved the resulting map of the environment. These saved maps contained information about the SLAM graph, visual landmarks, local maps, and other supporting data. Maps were not deleted between runs. Thus, even when we ran the SLAM system on the same log multiple times, each run would be different, because it started with a different loaded map.

To simulate a realistic scenario, the robot was run on multiple logs from an environment collected at different times of the day, the map was saved at the end of each run, and remained loaded on the robot at start of the next run. Additionally, we simulated larger sequences of consecutive runs in the environment by sequentially running the SLAM system on repeating clusters of logs. For example, to simulate 15 runs for a particular environment for which only 3 logs were available, the SLAM system was run sequentially on all all three logs (log1, log2, log3), and then on the same set of logs four more times in a randomized order (for

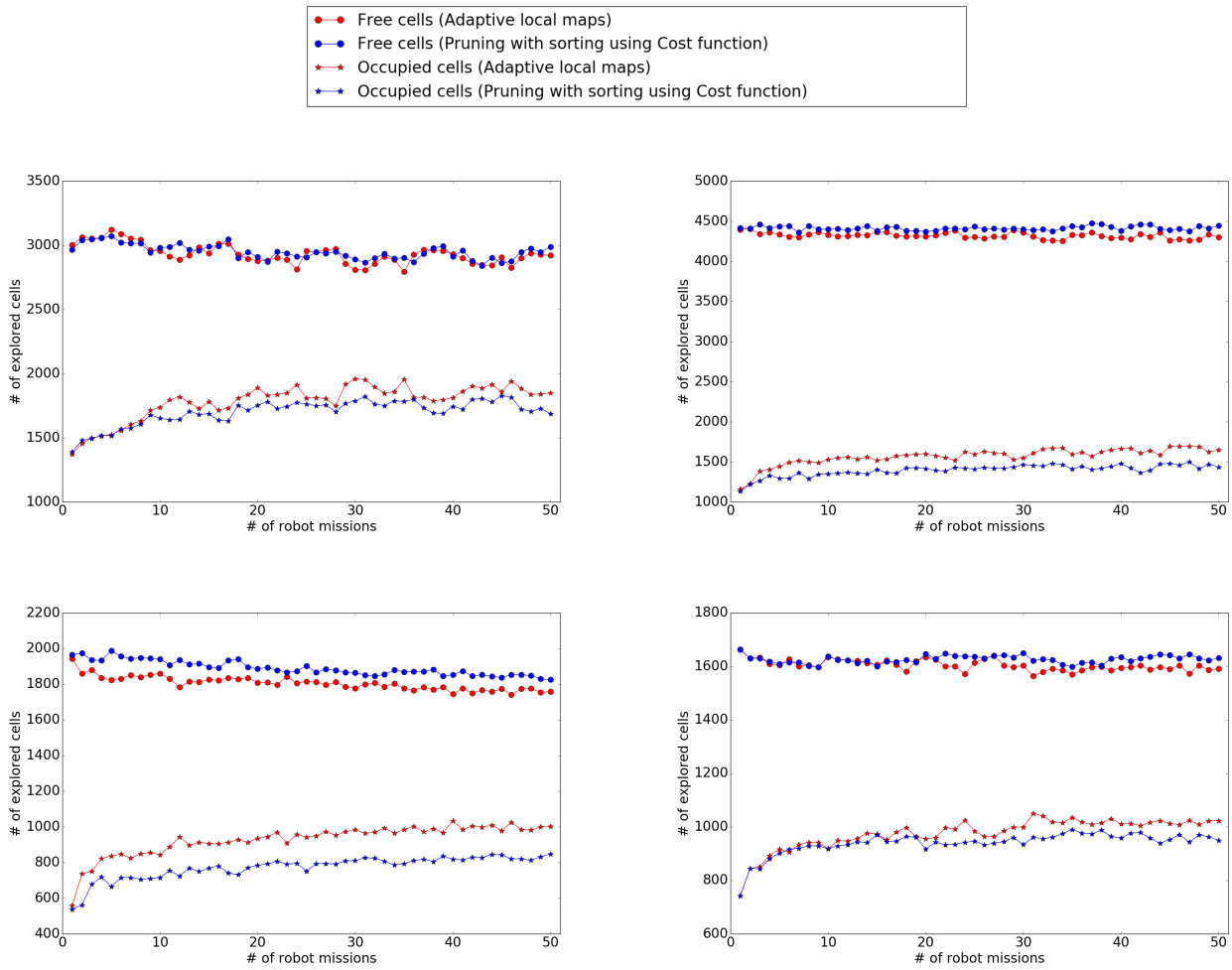

Fig. 5: Comparison of the free and occupied cells from the rendered maps with and without pruning for 50 sequential runs in four environments. Since the environment doesn't change over the runs, the number of occupied and free cells ideally shouldn't deviate too much from previous runs. With our method, it can be seen that the count of free and occupied cells for the four environments after every run are very close for both our method and adaptive local maps without pruning. This shows that our method does not have an adverse impact on the quality of the generated maps. Clockwise from top (Env. A, B, D, and C).

example: log2, log1, log3). The loaded map at the start of every sequential run was different, thereby making each run unique.

We have performed sequential mission analysis over 50 missions in four different environments with a *gain* of 1 and a *penalty* of -10 for $p(x)$ (Eqn. 9). We chose a small λ value for the relative uncertainty weight as that is very dependent on the SLAM system and robot motion and observation models. We chose the $q(\mathscr{L})$ function to be the total number of occupancy cells in the rendered map from all local maps in $\mathscr{L}$. We chose an ε_q value of 0 so that the total number of occupancy cells in the rendered grid map remain the same before and after pruning. Environments A and B are similar in size, Env A has a higher odometry uncertainty whereas Env B has a lower odometry uncertainty due to different floor surfaces. Env C and Env D are smaller spaces, but all the logs that were collected from Env D had lesser variation in lighting conditions, leading to better localization with features from older maps.

Fig. 4 shows the growth of local adaptive maps without pruning and how our method caps the growth of local maps across multiple missions. The effect of grid management can be directly observed in the periodic oscillations in the blue and green lines (corresponding to our method with and without sorting, respectively). The total number of local maps stabilize after a few missions leading to no increase in memory footprint or render times. In environment B, the robot creates more local maps because of a higher uncertainty in its odometry because of the floor surface. This results in the steeper growth of local maps than in some of the other environments. In Env D, the growth rate is lower than in the other environments because of better localization leading to local maps that were created in earlier missions being picked for mapping. Large fluctuations in the total number of

Fig. 6: Occupancy maps of different environments A, B, C, and D (from left to right) with no local map pruning after 50 missions. In env C., there are some incorrectly mapped occ. cells (inside red circle) because of movement of some of the local maps from previous runs. With our method as shown below, this is not seen as those local maps that were poorly constrained were most likely removed. Although pruning of poorly constrained local maps wasn't the problem that we accounted for, our method does help sometimes in these cases.

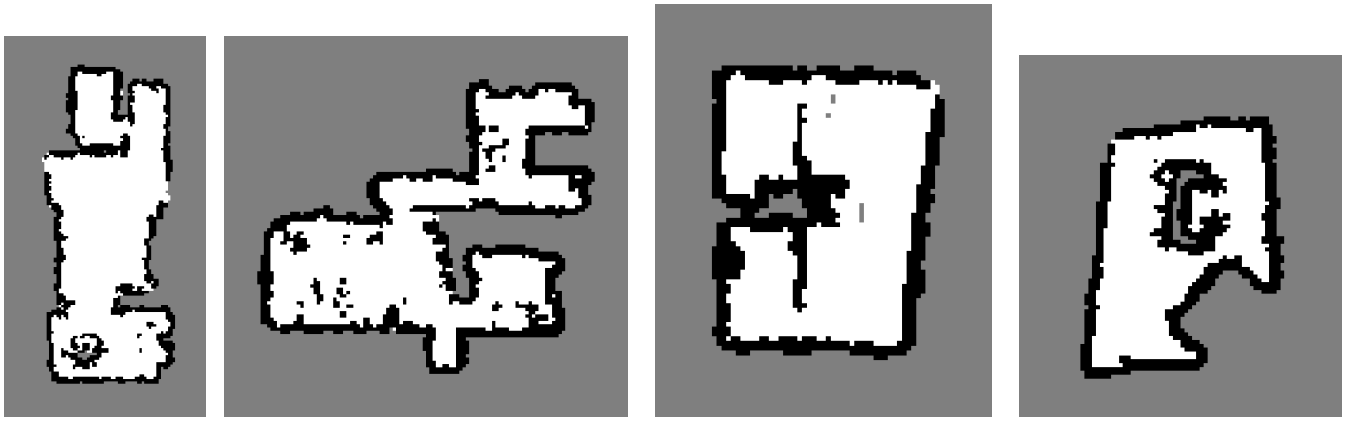

Fig. 7: Occupancy maps of different environments A, B, C, and D (from left to right) with our local map pruning algorithm after 50 missions.

local maps across robot missions are observed when no cost function based sorting is used, whereas our method reduces the amount of fluctuation in the total number of maps across runs. Fig. 8 plots the number of local maps left after pruning in one environment to compare different ε_q values.

For a fair evaluation of the quality of the maps that are generated after pruning, we compare the rendered occupancy grid maps by looking at the number of occupied and free cells in them. The occupied and free cells from the maps generated after every mission without any pruning is compared against maps generated with our method. Fig. 5 shows the plots of occupied and free cells after every run for the same four environments. It can be seen that the occupied and free cell count in the rendered maps with and without pruning are very close to each other in all four environments. This goes to show that even though our method is lossy, since we don't incorporate the information from the local maps to be pruned into other local maps, our method doesn't affect the quality of the map when compared to maps that would otherwise have been generated without pruning.

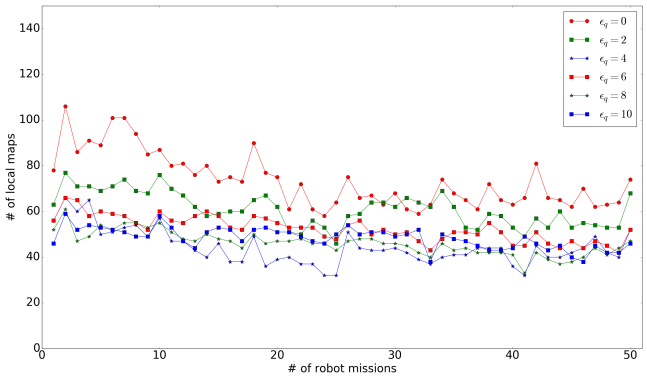

Fig. 8: Multiple plots of the total number of local maps that are left after pruning has happened with different values of ε_q. Our $q(\mathscr{L})$ function that looks at the total number of occ. cells is used here.

Fig. 6 shows the rendered occupancy maps from our four testing environments after 50 missions without performing any local map pruning. Fig. 7 shows the rendered maps after 50 missions with our local pruning pruning algorithm applied after each mission. While the corresponding maps in the two figures are not identical, it is clear that our pruning has not caused any drastic changes or discontinuities.

V. Conclusions and Future Work

In this paper we have presented a novel practical method of pruning redundant local maps, which allows a mobile robot to maintain the accuracy and consistency of its occupancy grid during lifelong mapping, without exceeding its memory and computational capacity. We have defined a cost function for each local map, representing its contribution to the global rendered grid. We then used this function to decide which local maps can be removed without compromising the global grid's integrity, i.e. without creating discontinuities. Thus we have achieved our objective of curtailing the growth of the number of local maps, without which lifelong mapping using adaptive local maps would have been impossible.

In this paper we have assumed discrete cell occupancy values. On the other hand, modeling the occupancy information with probabilities opens the possibility of preventing the increase in the number of local maps by merging some of them, i.e. distributing the information of the to-be-pruned local map to other local maps and then pruning it. Further research is needed to explore the possible advantages of merging over pruning the local maps, and to devise efficient and practical algorithms for it.

References

[1] E. Eade, P. Fong, and M. E. Munich, "Monocular graph SLAM with complexity reduction," *IEEE/RSJ 2010 International Conference on Intelligent Robots and Systems, IROS 2010 - Conference Proceedings*, pp. 3017–3024, 2010.

[2] M. Llofriu, P. Fong, V. Karapetyan, and M. Munich, "Mapping Under Changing Trajectory Estimates," *IEEE/RSJ 2017 International Conference on Intelligent Robots and Systems, IROS 2017 - Conference Proceedings*, pp. 1403–1410, 2017.

[3] K. Konolige and J. Bowman, "Towards lifelong visual maps," in *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*. IEEE, 2009, pp. 1156–1163.

[4] M. Muja and D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration," in *In VISAPP International Conference on Computer Vision Theory and Applications*, 2009, pp. 331–340.

[5] S. Hochdorfer, M. Lutz, and C. Schlegel, "Lifelong localization of a mobile service-robot in everyday indoor environments using omnidirectional vision," in *IEEE International Conference on Technologies for Practical Robot Applications (TEPRA)*, November 2009.

[6] M. Dymczyk, T. Schneider, I. Gilitschenski, R. Siegwart, and E. Stumm, "Erasing bad memories: agent-side summarization for long-term mapping," in *IEEE International Conference on Intelligent Robots and SYstems (IROS)*, October 2016.

[7] P. Mühlfellner, M. Bürki, M. Bosse, W. Derendarz, R. Philippsen, and P. Furgale, "Summary maps for lifelong visual localization," *Journal of Field Robotics*, vol. 33, no. 5, pp. 561–590, 2015.

[8] M. Brki, I. Gilitschenski, E. Stumm, R. Siegwart, and J. Nieto, "Appearance-based landmark selection for efficient long-term visual localization," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2016, pp. 4137–4143.

[9] M. Dymczyk, S. Lynen, M. Bosse, and R. Siegwart, "Keep it brief: Scalable creation of compressed localization maps," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2015, pp. 2536–2542.

[10] H. Moravec and A. E. Elfes, "High resolution maps from wide angle sonar," in *Proceedings of the 1985 IEEE International Conference on Robotics and Automation*, March 1985, pp. 116 – 121.

[11] A. E. Elfes, "Occupancy grids: A probabilistic framework for robot perception and navigation," Ph.D. dissertation, Carnegie-Mellon University, May 1989.

[12] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, 2013, software available at http://octomap.github.com. [Online]. Available: http://octomap.github.com

[13] Y. Kwon, D. Kim, I. An, and S. Yoon, "Super rays and culling region for real-time updates on grid-based occupancy maps," *IEEE Transactions on Robotics*, vol. 35, no. 2, pp. 482–497, April 2019.

[14] L. Garrote, J. Rosa, J. Paulo, C. Premebida, P. Peixoto, and U. Nunes, "3d point cloud downsampling for 2d indoor scene modelling in mobile robotics," 04 2017, pp. 228–233.

[15] A. Schaefer, L. Luft, and W. Burgard, "Dct maps: Compact differentiable lidar maps based on the cosine transform," *IEEE Robotics and Automation Letters*, vol. PP, pp. 1–1, 01 2018.

[16] A. Caccavale and M. Schwager, "Wireframe mapping for resource-constrained robots," 10 2018, pp. 1–9.

[17] A. Schiotka, B. Suger, and W. Burgard, "Robot localization with sparse scan-based maps," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2017, pp. 642–647.

[18] J. Saarinen, R. Mazl, M. Kulich, J. Suomela, L. Preucil, and A. Halme, "Methods for personal localisation and mapping," *IFAC Proceedings Volumes*, vol. 37, no. 8, pp. 388 – 393, 2004, iFAC/EURON Symposium on Intelligent Autonomous Vehicles, Lisbon, Portugal, 5-7 July 2004. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1474667017320074

[19] A. Heinemann, J. Velten, and A. Kummert, "Map building using occupancy grids with differentiated occupancy states," in *2015 IEEE 9th International Workshop on Multidimensional (nD) Systems (nDS)*, Sep. 2015, pp. 1–6.

[20] J. Schauer and A. Nchter, "The peopleremoverremoving dynamic objects from 3-d point cloud data by traversing a voxel occupancy grid," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1679–1686, July 2018.

[21] D. Kakuma, S. Tsuichihara, G. A. G. Ricardez, J. Takamatsu, and T. Ogasawara, "Alignment of occupancy grid and floor maps using graph matching," in *2017 IEEE 11th International Conference on Semantic Computing (ICSC)*, Jan 2017, pp. 57–60.

[22] Y. Yue, P. G. C. N. Senarathne, C. Yang, J. Zhang, M. Wen, and D. Wang, "Hierarchical probabilistic fusion framework for matching and merging of 3-d occupancy maps," *IEEE Sensors Journal*, vol. 18, pp. 8933–8949, 2018.

[23] H. Li, M. Tsukada, F. Nashashibi, and M. Parent, "Multivehicle cooperative local mapping: A methodology based on occupancy grid map merging," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 5, pp. 2089–2100, Oct 2014.

[24] J. J. Leonard and H. J. S. Feder, "Decoupled stochastic mapping [for mobile robot amp; auv navigation]," *IEEE Journal of Oceanic Engineering*, vol. 26, no. 4, pp. 561–571, Oct 2001.

[25] M. C. Bosse, "Atlas: a framework for large scale automated mapping and localization," Ph.D. dissertation, Massachusetts Institute of Technology, 2004.

[26] P. Pinis and J. D. Tards, "Large-scale slam building conditionally independent local maps: Application to monocular vision," *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 1094–1106, Oct 2008.

[27] J. Aulinas, J. Salvi, X. Llado, and Y. Petillot, "Local map update for large scale slam," *Electronics Letters*, vol. 46, pp. 564 – 566, 05 2010.

[28] K. Konolige, E. Marder-Eppstein, and B. Marthi, "Navigation in hybrid metric-topological maps," in *2011 IEEE International Conference on Robotics and Automation*, May 2011, pp. 3041–3047.

[29] B.-J. Ho, P. Sodhi, P. Teixeira, M. Hsiao, T. Kusnur, and M. Kaess, "Virtual occupancy grid map for submap-based pose graph slam and planning in 3d environments," 10 2018, pp. 2175–2182.