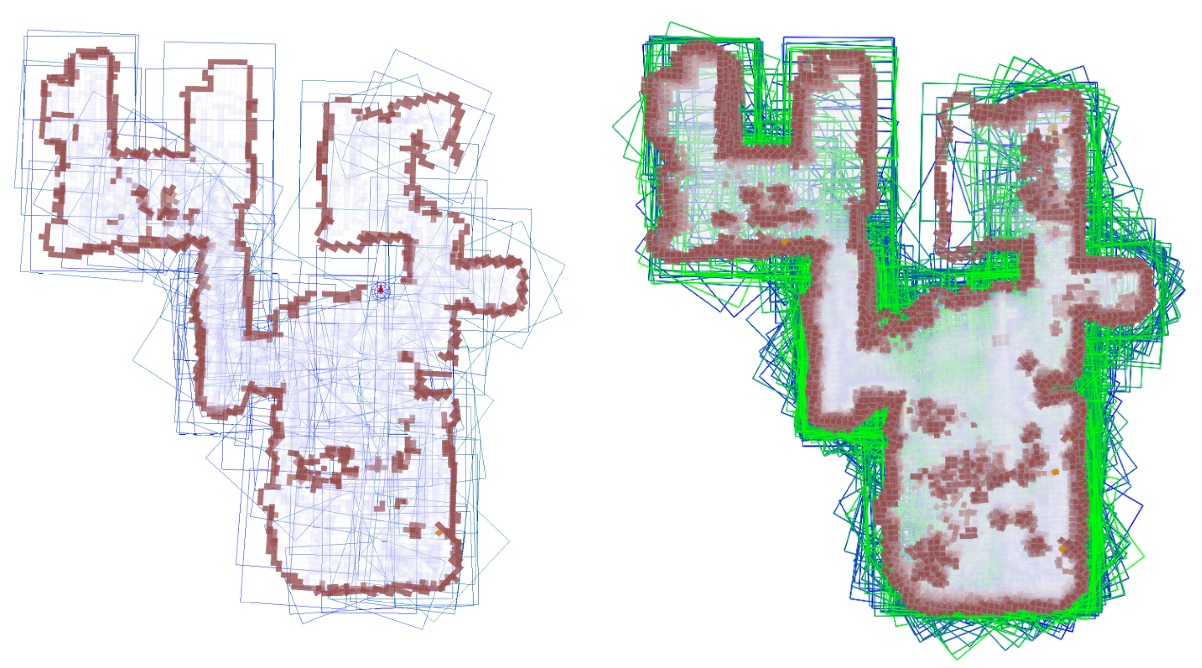
Lifelong Mapping using Adaptive Local Maps

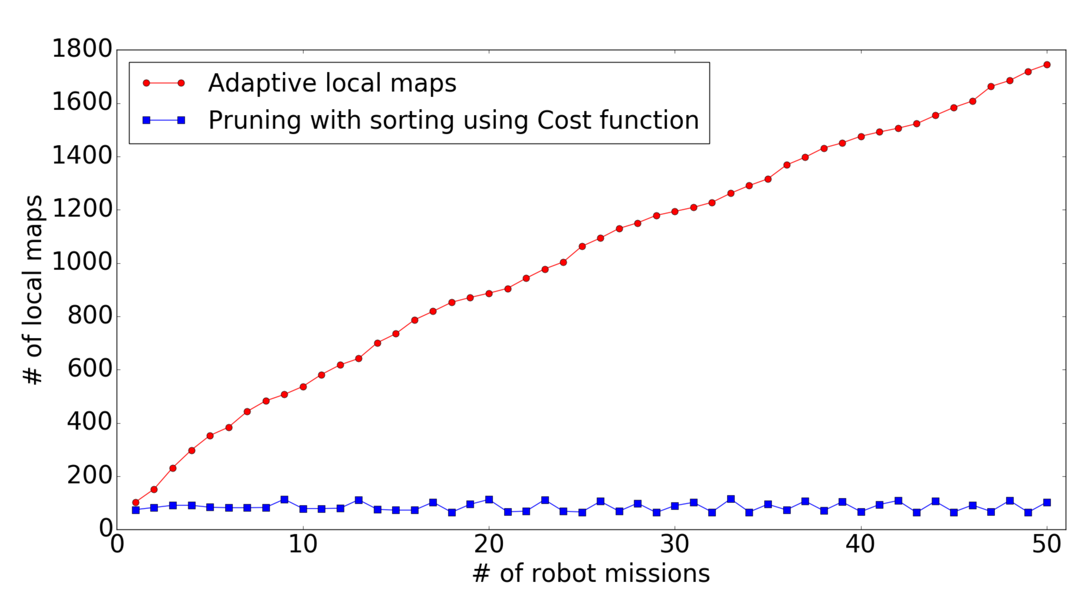
N Banerjee, D Lisin, J Briggs, M Llofriu, M E Munich

Introduction

- Adaptive local maps (1) is an algorithm which represents the occupancy information as a set of overlapping local maps anchored to poses in the robot's trajectory.
- In a graph-based SLAM system (2), an occupancy grid represented as adaptive local maps automatically adjusts to graph optimization.
- Unfortunately, in a life-long mapping scenario, the number of the local maps grows over time, increasing computation and memory consumption of the system.
- We propose a novel approach for pruning redundant local maps, without creating discontinuities in the rendered global map, thus ensuring the robustness and stability required for lifelong mapping.



Pruning redundant local maps (left) vs not pruning the maps (right).



Growth of the number of local maps with and without pruning across 50 robot missions. 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.

References

[1] 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.

[2] 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.

Local Map Pruning Algorithm

Objective

- Let $\mathscr{L}$ be the set of all local maps ℓ_T^i,
- Let S be the subset of local maps which should be pruned.
- For any subset of local maps $X \subseteq \mathscr{L}$, let $q(X)$ be a measure of the quality of the rendered global grid constructed from X.
- Define $q(X)$ to be the difference in number of cells between the global grid rendered from L and the global grid rendered from X.

Our goal is to find the maximal subset of local maps that can be pruned:

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

The Algorithm

- Define the cost of keeping a local map. If the cost is high, the contribution of the local map to the rendered global map is low, and vice versa.
- Let $s(x,y)$ denote the total number of local maps that make a contribution to the final rendered cell at (x,y). When $s(x,y) = 1$, removing the corresponding local map will leave a hole.
- Define the local map's uncertainty as the minimum uncertainty between the grid node and a landmark node:

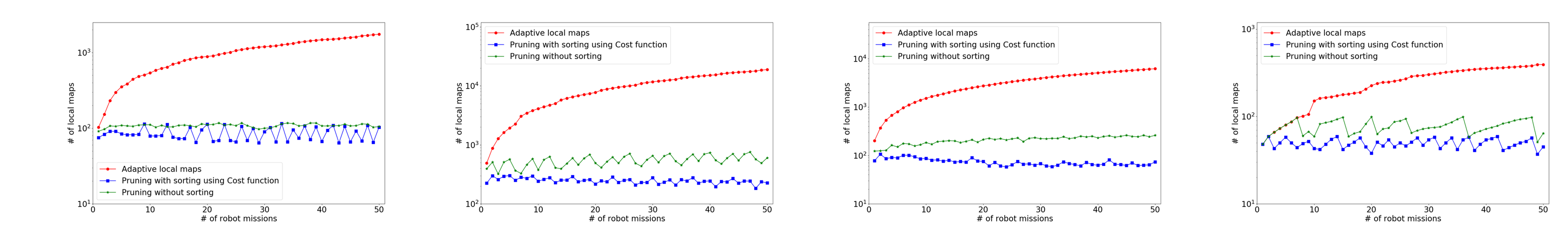
$$\begin{aligned} u(\ell_t^i) &= \min_{lm \in \mathbb{LM}} \operatorname{tr}(\text{shortest\_path\_uncertainty}(\ell_t^i, lm)) \\ &= \operatorname{tr}(^{lm*}\Sigma^{\ell_t^i}) \end{aligned}$$

Algorithm 1 Local map pruning

Require: List $\mathscr{L}$ of all local maps ℓ_t^i, $p(x)$, $s(x)$, λ
1: Local maps to be pruned: $S \leftarrow \{\varnothing\}$
// Calculate the cost of all the local maps in S
2: **for** $\ell_t^i \in \mathscr{L}$ **do**
3: $\text{cost}[\ell_i^t] \leftarrow \left\{ \sum_{(x,y)\in\ell_t^i} p(s(x,y)) \times s(x,y) \right\} + \lambda u(\ell_t^i)$
// Sort $\mathscr{L}$ in a descending order based on local map cost
4: $\mathscr{L}_{sorted} \leftarrow \text{sort}(S, \text{cost})$
// Check constraint, add local map to S if within ε_q
5: **for** $(i = 0; i < len(\mathscr{L}_{sorted}); i = i+1)$ **do**
6: $\ell \leftarrow L_{sorted}[i]$
7: $S_t \leftarrow S \cup \{\ell\}$
8: **if** $|q(\mathscr{L}) - q(\mathscr{L} \setminus S_t)| \leq \varepsilon_q$ **then**
9: $S \leftarrow S_t$
10: **return** S

Results

Capping Growth of Local Maps

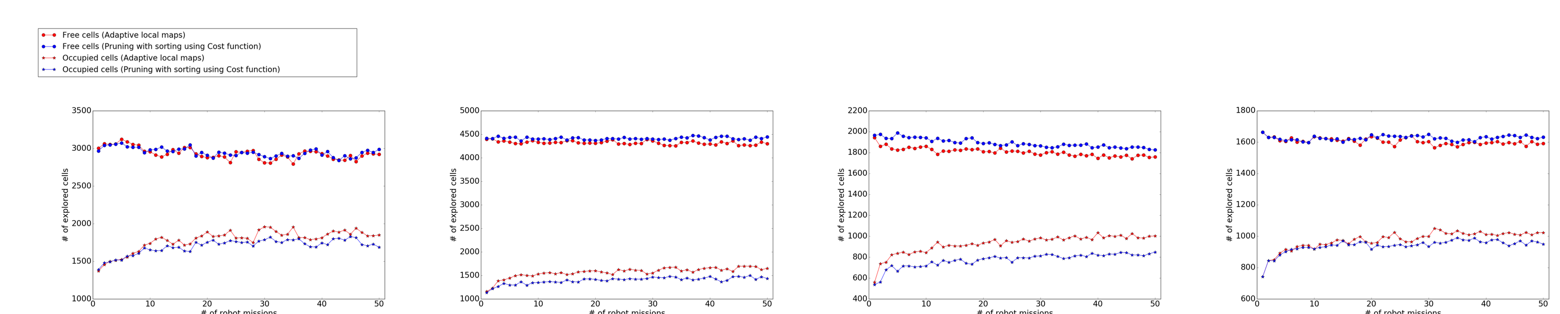


(a) (b) (c) (d)

Log-scale comparison of the numbers of adaptive local maps with and without pruning in 4 different environments. Mapped area - Env. (a): 656 ft^2, Env. (b): 516 ft^2, Env. (c): 297 ft^2, Env. (d): 285 ft^2.

- Red line: no pruning, unbounded growth.
- Green line: pruning without sorting the local maps using the cost function.
- Blue line: our proposed method. Pruning the local maps in the order of the cost.

Preserving Global Map Integrity

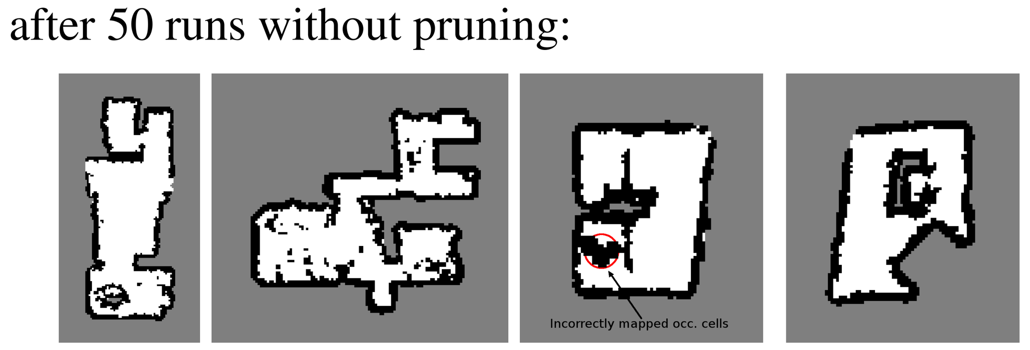


(a) (b) (c) (d)

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.

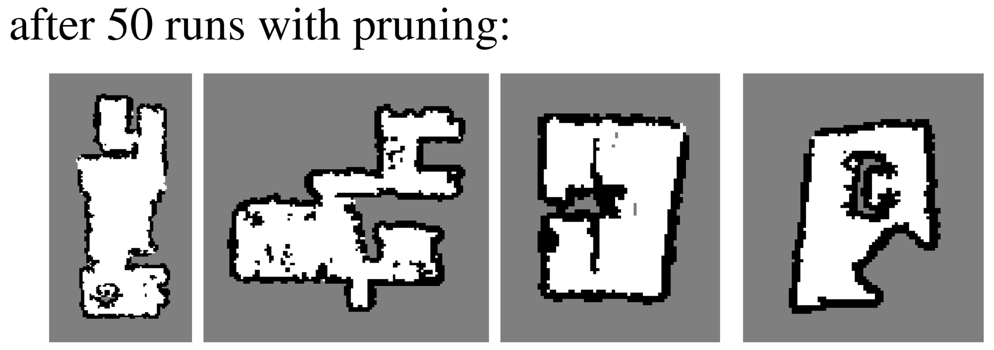
Qualitative Comparison

Global maps after 50 runs without pruning:



(a) (b) (c) (d)

Global maps after 50 runs with pruning:



(a) (b) (c) (d)

Notice that the maps do not look very different, which was our goal. Note also that poorly constraint local maps are sometimes removed, which is beneficial.