Experience-Based Map Building Re-Exploration of Quasi-Dynamic Indoor Environments

Davide Bertalero[1], Mauro Tellaroli[1], Pierre Bordeau[1,2], Michele Antonazzi[1], Matteo Luperto[1], Nicola Basilico[1]
[1] :University of Milan, Milan, Italy
[2] :ENSTA, Institut Polytechnique de Paris, France
{nicola.basilico, matteo.luperto}@unimi.it

***Abstract*—Robots operating in indoor environments often face dynamic changes caused by movable objects, clutter, and varying accessibility conditions. In such scenarios, navigation strategies based on a single map may lead the robot through regions that are frequently obstructed, reducing efficiency and robustness. This work proposes an approach that leverages occupancy grid maps collected from repeated explorations of the same environment to build a unified representation capturing both stable structures and recurring environmental variations. The aggregated map is used to identify regions that are less likely to be occupied by obstacles, enabling safer and more efficient navigation across the entire environment, with the goal of obtaining a new map of the environment. Preliminary experiments show that the proposed method reduces traveled distance, execution time, and variability across runs compared to a greedy frontier-based exploration baseline, while maintaining comparable environment coverage.**

I. Introduction

Autonomous service robots operating in real-world indoor environments typically rely on 2D maps, obtained via exploration and SLAM, to plan for navigation and task execution in different locations of the environment. A fairly common assumption in current SLAM approaches, the *static world assumption*, is that the world remains unchanged as the robot moves in it. When maps are used over long time scales and in large environments, change is inevitable. However, indoor environments are rarely static: objects are displaced, doors open and close, and furniture is rearranged over time. A map built during a previous exploration session might therefore become outdated. The mainstream approach to deal with this is to repeat the exploration phase to acquire a new map and replace the previous one [1].

When SLAM is performed, a common practice is to prep the physical spaces the robot will visit so that the obtained map will represent the structure of the environment in its entirety, hence reflecting some sort of *nominal condition*. This is customarily achieved by optimizing the space for navigation and sensor visibility, for example opening all the doors that give access to the areas that have to be mapped, removing obstacles or clutter that might obstruct or disturb the robot's perceptions or force it into complex maneuvers [2].

In long-term deployments, the robot may face scenarios where the gap between the map's nominal conditions and the actual space configurations differ greatly. For example, some rooms might become temporarily inaccessible or new objects might block specific paths, preventing the robot from accessing portions of the environment that could be of significant size.

As the robot is using a map of the *nominal* environment to plan its actions, this may result in it trying to actuate some trajectories that, although feasible and safe in a clutter-free map, are actually difficult or risky to perform in the current situation, eventually leading to frequent replans or requests for human interventions. As an example, consider a robot that can navigate seamlessly around a table when all chairs are in their *tidy* positions as reflected in the nominal map. However, chairs are frequently displaced from their expected locations, causing unexpected obstacles and potential collisions.

To cope with this limitation, we propose a method for robot to autonomously and safely obtain via SLAM a new map that represents the current and unknown conditions of the environment. This map is designed to represent not the entirety of the environment, but only those areas where the robot can navigate safely in the current configuration of a quasi-dynamic environment, that is an environment that appears static over short time spans but may undergo recurring structural changes over longer periods (e.g., hours, days, or weeks). Such a map can be later used for planning and executing tasks or for navigation, thus fulfilling the *static world assumption*. To do so, we leverage the fact that the robot performs the same tasks repeatedly in the same environment, accumulating a history of repetitive observations.

Specifically, we integrate information gathered across multiple observations in order to capture both persistent structures and recurring changes due to quasi-dynamic components. This representation enables the identification of regions that are consistently free, as well as areas that are frequently occupied, prone to clutter, or often difficult to navigate. An example of this is the area surrounding a table, where chairs are often in different positions and can present several challenges for the navigation of small robots such as vacuum cleaners. Leveraging such knowledge is particularly relevant for coverage tasks, where the objective is not only to visit all relevant areas but also to do so efficiently and safely.

We tackle this gap with an *experience-based* Weighted Coverage Path Planning (WCPP) strategy for obtaining a new map of the environment, given historical mapping data. The key insight is that, although a dynamic environment changes over time, its dynamics exhibit statistical regularity: chairs are near tables, tables don't move much, and corridors usually remain

traversable. By collecting occupancy maps across multiple exploration sessions, we estimate how likely each cell is to be occupied, and use this likelihood as a cost signal for allowing the robot to safely perform a mapping run in the environment that can obtain a full map of the current working area of the environment, without performing challenging or complex trajectories leading to a mapping, localization, or navigation failure.

II. Related Work

Mobile robots still struggle to reach long-term autonomy in quasi-dynamic environments [3]. Some methods focus on maintaining map consistency without requiring full re-mapping through several advanced strategies. One prominent approach is the Dynamic Pose Graph [4], which labels laser points as static, added, or removed to manage low dynamics specifically. Building on this, Lázaro et al. [5] developed an efficient SLAM system that utilizes ICP-based alignment and local map merging to incrementally update point-cloud representations. Other methodologies leverage probabilistic modeling to predict environmental states; for instance, in [6] Hidden Markov Models are employed to estimate the retention time of occupied cells, allowing the robot to infer the likelihood of encountering parked cars or similar semi-static objects. More recently, the integration of Deep Reinforcement Learning has enhanced the adaptability of navigation policies [7], enabling robots to handle real-time interactions and environmental uncertainties in these non-stationary settings more effectively than traditional planners.

III. Method

A. Obtaining experience maps

We envisage the following scenario: a service robot is tasked to perform an activity in an environment, which is repeated several times over a long time span. An example of this is a robot vacuum cleaner that has to perform a full coverage of the environment every day. To do so, it relies on a map of the environment; as the environment is quasi-dynamic, the robot, before starting its mission, creates a new map of the environment by doing SLAM. When it uses this map, the static map assumption holds, and the map represents the part of the environment that is available to the robot given the current configuration of the environment.

When the robot is initially deployed, it has no experience; thus, it performs an exploratory run of the environment while doing SLAM, using an exploration strategy (in our case a nearest-frontier exploration strategy [8]). After this step is repeated n times, n independent exploration runs have been performed, and n maps are available. Our method then uses this knowledge to optimise the process of obtaining a new map of the environment; we call this process *remap*ping.

B. Problem definition

Given a dataset of n occupancy grid maps $\{M_1, \ldots, M_n\}$ that represent the same quasi-dynamic environment at different times, we want to identify a set of viewpoints G that the robot can visit sequentially to obtain a new map covering the current available parts of the environment. An occupancy grid map M is defined as a matrix of size $h \times w$, where each cell value is the probability of occupancy. In the absence of observations for a given cell, we assign it the value corresponding to maximum occupancy. We use experience to identify a set of viewpoints that are likely to be safe to visit in all the environment's configurations. The robot, by going to the subset of viewpoints that are reachable in the current configuration of the environment, can observe, via its sensors, the entire environment, and build, via SLAM, a full map of the portion of the environment where it can safely operate, given the current configuration of the obstacles.

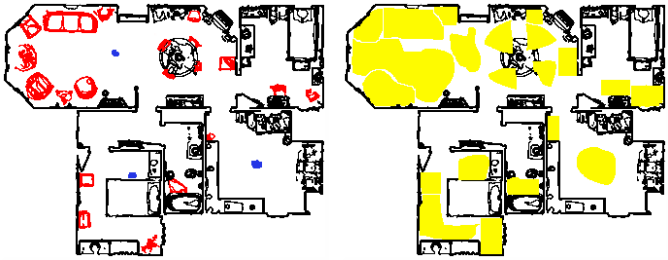


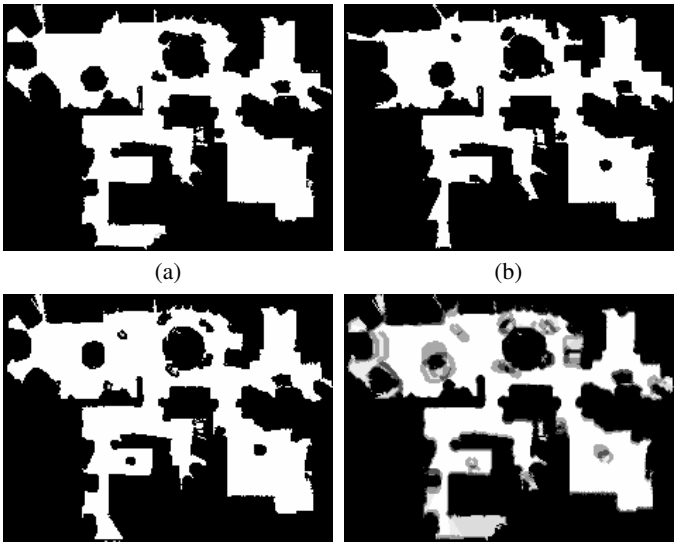
Fig. 1: Left: environment map highlighting obstacle categories, with semi-static obstacles shown in red and transient semi-static obstacles in blue. Right: corresponding predefined motion regions, depicted in yellow.

C. Modeling Quasi-Dynamic Indoor Environments

We provide a generator of different configurations of indoor quasi-dynamic spaces, such as an office or an apartment, where certain objects (e.g., chairs, small furniture, or movable obstacles) may have different locations or appear /disappear over time, while structural elements like walls and built-in fixtures remain unchanged. We model three categories of quasi-dynamic elements, as shown in Figure 1 in different colours. 1) *Static*: structural elements that remain unchanged like walls or pillars (black). 2) *Semi-static*: movable objects that can translate or rotate within the environment like chairs (red). 3) *Transient*: semi-static obstacles that may be present or not over time like bags or items on the ground (blue).

Let (x_i, y_i, θ_i) denote the pose of object i. To randomly place i within its region, three offsets $\Delta x_i, \Delta y_i$ and $\Delta\theta_i$ are sampled from a Gaussian distribution, yielding a candidate pose $(x_i + \Delta x_i, y_i + \Delta y_i, \theta_i + \Delta\theta_i)$. The candidate pose is accepted only if a prescribed fraction of the object's area lies within its assigned motion region and the object does not overlap with any other object; otherwise, the sampling process is repeated until a valid pose satisfying both conditions is obtained. Items not assigned to any motion zone are allowed to occupy any position within the environment's boundaries.

Multiple realistic configurations of the same environment can thus be generated, enabling the simulation of the variability and dynamic behaviour typically observed in real-world settings. Environments are obtained from the dataset of [9], where objects are assigned to different categories according



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

Fig. 2: The figure shows three occupancy grid maps (a,b,c) obtained from independent exploration runs in different environment configurations, while (d) illustrates the resulting average map U_{avg} obtained by applying an arithmetic mean over the corresponding cells of the three input maps.

to their type (e.g., chairs are quasi-dynamic) while motion patterns are manually defined.

D. Experience aggregation

The first step of our method is to obtain a unified map U that aggregates all the environmental information contained in the dataset of n occupancy grid maps $\{M_1, \dots, M_n\}$ of the same size previously acquired in the same environment.

The key step in constructing U consists in applying an arithmetic mean over corresponding cells across all maps in the dataset, thereby aggregating the occupancy information at each location obtaining U_{avg}, where each cell (j, k) is computed as $u_{avg}(j, k) = \frac{1}{n}\sum_{t=1}^{n} m_t(j, k)$ where $m_i(j, k)$ is the occupancy value of the cell (j, k) of the matrix M_i. An example of this process is shown in Figure 2.

The map is then binarized by applying a threshold ρ to the occupancy values: cells with values great or equal (below) ρ are set to the maximum (minimum) value V (v), effectively yielding a binary matrix representation of the environment as shown in Figure 3 (left). The threshold is selected independently for each map by applying the triangle method [10].

To mitigate the impact of poor-quality maps, which can arise from localization failures or incomplete exploration runs (often due to challenging configurations of the environment), the dataset is filtered to retain only maps above a quality threshold before computing.

E. Environment discretization

To compute the set of viewpoints that are used to obtain the new map, the unified map U is discretized so that linear programming techniques can be applied. In particular, two sets

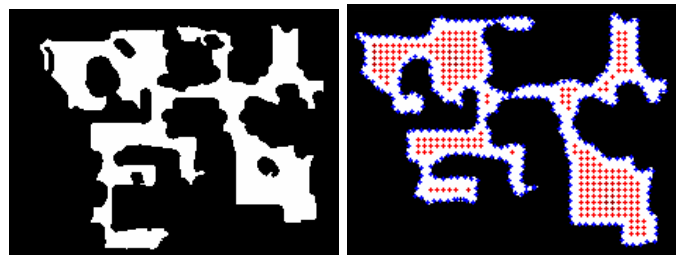


Fig. 3: (left) Binary occupancy map obtained from the averaged map shown in Figure 2d by applying a threshold computed using the triangle method. (right) Discretized map showing the set of candidate guard positions in red and the set of witness points in blue.

are defined: guards and witnesses, denoted respectively with $G = \{g_1, g_2, \dots\}$ and $W = \{w_1, w_2, \dots\}$. Guards represent candidate positions that the robot can reach in order to sense the environment, while witnesses correspond to points in the map that must be covered by the robot's sensors. Guards are generated by overlaying a grid on the map, where the center of each grid cell defines a candidate guard position. A finer grid resolution results in a larger number of guards. Additionally, a minimum distance is enforced between guard positions and obstacle boundaries to avoid selecting points that may be unreachable by the robot. Witnesses are generated by constructing one or more polygons along the boundaries of the obstacles and uniformly sampling points along their contours. The finer the subdivision of these boundaries, the greater the number of witnesses obtained. An example of discretization is shown in Figure 3 (right).

Another aspect concerns the computation of costs associated with the guards. A higher cost reflects a guard that is more difficult for the robot to reach. Starting from the dataset $\{M_1, \dots, M_n\}$, a cost is assigned to each cell of the occupancy grid, proportional to the number $q(j, k) \leq n$ of times that cell (j, k) is observed as occupied. More precisely, the cost of a cell (j, k) is defined as the $q(j, k)/n$. The cost of a guard g_i is the cost of the cell in which it is located and it is denoted with $C(g_i)$.

To determine whether a witness is covered by a guard, two conditions must be satisfied: there must be a clear line of sight between the guard and the witness, and their distance must not exceed a threshold defined by the sensing range of the robot. This procedure results in a collection of witness subsets, each associated with a guard that provides coverage, together with the corresponding cost assigned to that guard.

F. Experience-based Remapping

The problem can thus be formulated as a variant of the Weighted Set Cover Problem (WSCP) [11], where each subset consists of the witnesses covered by a given guard, and the objective is to select a collection of guards that ensures full coverage of all witnesses while minimizing the total associated cost. Building upon this formulation, we now define the total cost. The cost associated with a selected set of guards is

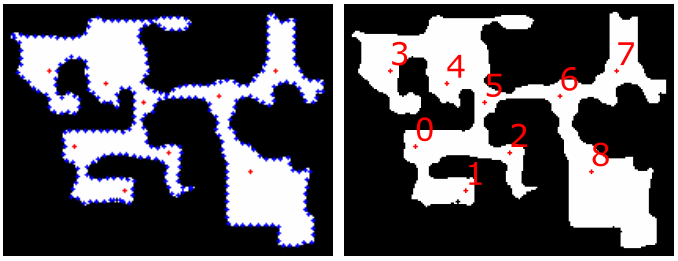


Fig. 4: Left: selected guards as solution of the WSCP in Figure 3. (left). Right: optimal path over the selected guards obtained by solving the TSP.

designed to account both for the individual guard costs $C(\cdot)$ previously defined and for the distances between them. This combined cost encourages the selection of guards that are not only less difficult to reach individually, but also spatially well positioned with respect to each other, thereby reducing the overall effort required by the robot to traverse them.

Formally, let $X \subseteq G$ be the guards choice of the solution, and let $d : G \times G \to \mathbb{R}$ be the distance between guards. The cost of the solution X is:

$$f(X) = \alpha \sum_{g \in X} C(g) + \beta \frac{1}{2} \sum_{g \in X} \sum_{t \in X} d(g, t)$$

where α and β are coefficients that balance the contribution of the guard costs (first term) and the pairwise distances between guards (second term). The factor $1/2$ accounts for the symmetry of the distance function d.

To introduce additional flexibility, a coverage threshold is defined, requiring only a specified percentage of witnesses to be covered; for instance, the constraint may enforce coverage of 80% of the witnesses rather than the entire set.

The solution of this WSCP formulation, which can be obtained using standard mathematical programming techniques, yields a subset of selected guards. In the context of the coverage problem, these guards correspond to the positions that the robot must reach.

Once the set of target points has been determined, a minimum-cost path must be computed to allow the robot to visit all of them. This is formulated as a Traveling Salesman Problem (TSP), which is solved using mathematical programming techniques. A fixed starting point is imposed, and the return to the starting location is not considered. Figure 4 illustrates an example of the solution of the two problems: on the left, the WSCP solution with the selected guards highlighted, and on the right, the resulting TSP solution showing the order in which the selected guards are visited.

IV. Experimental Results

We present preliminary results that were conducted in a simulated apartment environment, shown in Figure 1, using ROS and Stage as the simulator and a Turtlebot3 Burger as a robot. Given a set of generated $n = 20$ environment configurations as described in Section III-C, we acquire the initial experience needed by the robot by exploring the environment using a nearest frontier exploration strategy [8], producing, through SLAM, a 2D occupancy grid map using `slam_toolbox`.

	Time (s)		*Distance* (m)		*Area* (m^2)	
Method	σ	μ	σ	μ	σ	μ
Remap	94.1	302.78	6.99	27.46	97.34	649.59
Frontier	218.12	596.12	21.87	48.37	130.36	649.55

TABLE I: Experimental results for the evaluated methods. For each performance metric, both the mean μ and standard deviation σ are reported.

The results obtained with the proposed method, called *remap*, are also compared against a nearest-frontier exploration strategy as a baseline [8], called *frontier*. In the baseline configuration, we assume that the robot does not consider past experience and maps the environment every time as if the environment is unknown, thus performing an exploration of the environment. For each of the 20 environment configurations, 10 independent simulation runs were conducted per method, resulting in a total of 200 simulations for each method.To evaluate the methods, we collected the travel *distance* (m), the elapsed *time* (s) and the mapped *area* (m^2).

Table I summarizes the results of the simulations. All three evaluated metrics exhibit lower variability in the *remap* approach compared to *frontier*, reflecting its more stable and planned behavior. Despite achieving similar mapped areas, coverage significantly reduces both execution time and traveled distance, while generally maintaining equal or better performance.

V. Conclusion and Future Works

This work focused on improving robot navigation in dynamic indoor environments by leveraging maps collected from previous observations. By aggregating multiple occupancy grid maps into a unified representation, the approach captures both stable structures and recurring environmental changes, enabling the identification of safer regions for navigation. Experimental results show that exploiting prior maps leads to more efficient and consistent navigation compared to a greedy frontier-based approach, reducing execution time, traveled distance, and variability across runs. These findings highlight the effectiveness of using historical information to enhance navigation performance in dynamic settings.

Future work will focus on several directions. First, the environment model will be extended to include additional types of obstacles, such as doors with varying states (open, partially closed, closed), which directly affect the robot's reachable space. Second, the experimental evaluation will be expanded to include a larger number of environments and simulation runs. We also want to identify what is the extent of experience (namely, the number n of maps of the environment in different configurations) that is needed to bootstrap our approach by to obtaining an aggregated map that effectively improves navigation. Finally, the impact of the proposed approach will be assessed on downstream applications, such as coverage path planning or robotic vacuum cleaning.

REFERENCES

[1] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Transactions on robotics*, vol. 32, no. 6, pp. 1309–1332, 2017.

[2] M. Luperto, F. Amadelli, M. Di Berardino, and F. Amigoni, "Mapping beyond what you can see: Predicting the layout of rooms behind closed doors," *Robotics and Autonomous Systems*, vol. 159, p. 104282, 2023.

[3] L. Kunze, N. Hawes, T. Duckett, M. Hanheide, and T. Krajník, "Artificial intelligence for long-term robot autonomy: A survey," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 4023–4030, 2018.

[4] A. Walcott-Bryant, M. Kaess, H. Johannsson, and J. J. Leonard, "Dynamic pose graph slam: Long-term mapping in low dynamic environments," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 1871–1878.

[5] M. T. Lázaro, R. Capobianco, and G. Grisetti, "Efficient long-term mapping in dynamic environments," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 153–160.

[6] G. D. Tipaldi, D. Meyer-Delius, and W. Burgard, "Lifelong localization in changing environments," *The International Journal of Robotics Research*, vol. 32, no. 14, pp. 1662–1678, 2013.

[7] Y. Zhu, W. Z. Wan Hasan, H. R. Harun Ramli, N. M. H. Norsahperi, M. S. Mohd Kassim, and Y. Yao, "Deep reinforcement learning of mobile robot navigation in dynamic environment: A review," *Sensors*, vol. 25, no. 11, p. 3394, 2025.

[8] B. Yamauchi, "Frontier-based exploration using multiple robots," in *Proceedings of the Second International Conference on Autonomous Agents*, ser. AGENTS '98. New York, NY, USA: Association for Computing Machinery, 1998, p. 47–53. [Online]. Available: https://doi.org/10.1145/280765.280773

[9] S. K. Ramakrishnan, A. Gokaslan, E. Wijmans, O. Maksymets, A. Clegg, J. Turner, E. Undersander, W. Galuba, A. Westbury, A. X. Chang *et al.*, "Habitat-matterport 3d dataset (hm3d): 1000 large-scale 3d environments for embodied ai," *arXiv preprint arXiv:2109.08238*, 2021.

[10] G. W. Zack, W. E. Rogers, and S. A. Latt, "Automatic measurement of sister chromatid exchange frequency." *Journal of Histochemistry & Cytochemistry*, vol. 25, no. 7, pp. 741–753, 1977.

[11] E. Balas and M. W. Padberg, "Set partitioning: A survey," *SIAM review*, vol. 18, no. 4, pp. 710–760, 1976.