Lifelong Localization in Changing Environments

Gian Diego Tipaldi* Daniel Meyer-Delius† Wolfram Burgard*
*Department of Computer Science, University of Freiburg, D-79110 Freiburg, Germany.
†KUKA Laboratories GmbH, D-86165 Augsburg, Germany.

***Abstract*—Robot localization systems typically assume that the environment is static, ignoring the dynamics inherent in most real-world settings. Corresponding scenarios include households, offices, warehouses and parking lots, where the location of certain objects such as goods, furniture or cars can change over time. These changes typically lead to inconsistent observations with respect to previously learned maps and thus decrease the localization accuracy or even prevent the robot from globally localizing itself. In this paper we present a sound probabilistic approach to lifelong localization in changing environments using a combination of a Rao-Blackwellized particle filter with a hidden Markov model. By exploiting several properties of this model, we obtain a highly efficient map management approach for dynamic environments, which makes it feasible to run our algorithm online. Extensive experiments with a real robot in a dynamically changing environment demonstrate that our algorithm reliably adapts to changes in the environment and also outperforms the popular Monte-Carlo localization approach.**

I. Introduction

Long-term operations of mobile robots in changing environments is a highly relevant research topic as this ability is required for truly autonomous robots navigating in the real world. One of the most challenging tasks in this context is that of dealing with the dynamic aspects of the environment. One popular approach to robot navigation in dynamic environments is to treat dynamic objects as outliers [1, 2, 3]. For highly dynamic objects like moving people or cars, such methods typically work quite well, but they are less effective for *semi-static* objects. By semi-static objects we mean objects that change their location slowly or seldom like doors, furniture, pallets in warehouses or parked cars. In many realistic scenarios (see Figure 1), in which robots operate for extended periods of time, semi-static objects are ubiquitous and we believe that appropriately dealing with them in a localization approach can substantially increase the overall navigation performance.

In this paper, we present a novel approach to lifelong localization in changing environments, which explicitly takes into account the dynamics of the environment. Our approach is able to distinguish between objects that exhibit high dynamic behaviors, e.g., cars and people, objects that can be moved around and change configuration, e.g., boxes, shelves, or doors, and objects that are static and do not move around, e.g., walls.

To represent the environment, we use a *dynamic occupancy grid* [4], which employs hidden Markov models on a two-dimensional grid to represent the occupancy and its dynamics. We learn the parameters of this representation using a variant of the expectation maximization (EM) algorithm and then employ this information to jointly estimate the pose of the robot and the state of the environment during global localization. We furthermore apply a Rao-Blackwellized particle filter (RBPF), in which the robot pose is represented by the sampled part of the filter and the occupancy probability of a cell is represented in the analytical part of the factorization. In addition, we propose a map management method that is based on a local map representation and that is able to minimize memory requirements as well as to forget changes in a sound probabilistic way. We achieve this by considering the mixing times of the associated Markov chain.

Fig. 1. A mobile robot with a horizontally scanning laser range finder navigating in a parking lot at noon (top) and at 6 pm (bottom). Note that despite being at the same spot in both cases, the perceived scans will be substantially different due to the changed number of parked cars.

Compared to previous approaches, our algorithm has several desirable advantages. First, it improves the robustness and accuracy of the pose estimates. Second, our method is able to provide an up-to-date map of the environment around the current robot location. Finally, our map management method considerably reduces the runtime of the process whilst minimizing the memory requirements. As a result, our approach

allows a robot to localize itself and to simultaneously estimate a local configuration of the environment in an online fashion.

This paper is organized as follows. After discussing related work in Section II, we provide a more precise formulation of the problem in Section III. We then present an overview about the dynamic occupancy grid representation in Section IV. In Section V, we explain the algorithm for the joint estimation and the map management. Finally, in Section VI, we present extensive experiments carried out with a real robot, showing that our approach significantly outperforms state-of-the-art localization methods in changing environments in terms of the accuracy and robustness of the localization process and the consistency of the generated maps.

II. Related Work

Localization in dynamic environments has been an active topic in robotics research in the last decade. Many proposed approaches treat dynamic objects as outliers and hence filter out observations of dynamic objects. The observations of the static part of the environment are then used to perform map building, localization and navigation. For example, Fox *et al.* [1] used an entropy gain filter and a distance filter based on the expected distance of a measurement, while Schultz *et al.* [2] considered local minima of range measurement as observations from dynamic objects if they do not match an already available map. Montemerlo *et al.* [5] used observations of humans for people tracking while localizing the robot. The tracking of people simplifies the rejection of readings due to dynamic objects and increases reliability in populated environments. They employed a conditional particle filter to estimate the position of people conditioned to the pose of the robot, thus also enabling tracking in the presence of global uncertainty of the robot pose.

The main limitation of those approaches, however, is that they rely on the static world assumption for the underlying navigation system. In environments that change configurations over time or where the dynamics are low, e.g., parking lots, warehouses, apartments and cluttered environments, the changes may persist over long periods of time and could be useful to localize the robot. In extreme situations, the parts of the static environments that are visible are not informative enough for a reliable navigation and reasoning about changes is of utmost importance. In this paper we address those limitations and propose a localization system able to reason about changes and use that information to improve the localization performances.

Other approaches specifically focus on separating the static and dynamic aspects of the environment by building two maps. Wolf and Sukhatme [6] proposed a model that maintains two separate occupancy grids, one for the static parts of the environment and the other for the dynamic parts. Wang *et al.* [7] formulated a general framework for mapping and dynamic object detection by employing a system to detect if a measurement is caused by a dynamic object. Montesano *et al.* [8] extended the previous approach by jointly considering the problem of dynamic object detection with the one of mapping and including the error estimation of the robot in the classifier. Similarly, Gallagher *et al.* [9] built maps for individual objects that can then be overlaid to represent the current configuration of the environment. Hähnel *et al.* [3], on the other hand, combined the EM algorithm and a sensor model that considers dynamic objects to obtain accurate maps. The approach of Anguelov *et al.* [10] computes shape models of non-stationary objects. In their approach, the authors created maps at different points in time and compared those maps using an EM-based algorithm to identify the parts of the environment that change over time.

Although those approaches do not simply consider observations of dynamic objects as outliers, they still rely on a static representation of the environment to perform navigation. Their main advantage over filtering dynamic observations is that they are able to provide a better static map of the environment and the detection of dynamic observations can be done in a more reliable way. However, they still share the same limitations of the static world assumptions when deployed in changing environments or where the dynamics are low.

In order to overcome the limitations due to the static world assumptions, some authors worked on how to model the dynamics of the environment in a single unified representation. Chen *et al.* [11] and lately Brechtel *et al.* [12] extended the occupancy grid paradigm to include moving objects. Their approach, the Bayesian occupancy filter, is based on the idea that since occupancy is caused by objects, when those objects move, the corresponding occupied cell of the map should move accordingly. From this point of view, They propose an *object-centered* representation of the dynamics, and every cell in the environment need to be tracked over time. Moreover, the state transitions are assumed to be given a priori and no algorithm to learn them from data is presented. On the contrary, we follow a *map-centric* approach and model how the environment changes in an agnostic way with respect to the cause of the change.

Biber and Duckett [13] proposed a model that represents the environment on multiple timescales simultaneously. For each timescale a separate sample-based representation is maintained and updated using the observations of the robot according to an associated timescale parameter. Our approach differs from theirs in the sense that we fuse all the different maps into a unified representation and provide tools to estimate the optimal timescale parameter for each cell. Moreover their approach has higher memory and computational requirements than ours. In global localization settings, multiple hypotheses over the state of the environment are needed, thus memory requirements are an important aspect to be considered.

Yang and Wang [14] proposed the *feasibility* grids to facilitate the representation of both the static scene and the moving objects. A dual sensor model is used to discriminate between stationary and moving objects in mobile robot localization. Their work, however, assumes that the position of the robot is known with a certain accuracy to compute and update the maps and therefore is not suited to be used for global localization problems.

Recently, Saarinen *et al.* [15] proposed to model the environment as a set of independent Markov chains, one per grid cell, each with two states. The state transition parameters are learned on-line and modeled as two Poisson processes. A strategy based on recency weighting is used to deal with non-stationary cell dynamics. Their representation is very similar to the one used in our paper but differs from the way the occupancy probabilities are computed and the transitions learned. The focus of the presented work, however, is to show that Markov chain based representations can be effectively used for localization. Both representation could be used and we believe would produce similar results.

Some other works have been introduced in the past with the aim to address global localization problems in dynamic environments. However, to the best of our knowledge, none of them is general enough to work with different dynamics and objects or in real-world scenarios. Murphy *et al.* [16] proposed to apply a Rao-Blackwellized particle filter solution to the SLAM problem and showed that it could also deal with dynamic maps in a theoretical way. Their approach, however, assumes that the probabilities of changing state is independent from the current state of the environment and is given a priori. Moreover, they only presented results in small scale environments and with known initial position. In this paper we extend their representation and show how it can be used for global localization introducing a novel memory management strategy.

To handle the complexity of the Rao-Blackwellized particle filter solution in practical applications, some authors propose to focus on only some dynamic aspects or restrict the dynamics to a set of static configurations. Avots *et al.* [17], used the Rao-Blackwellized particle filter to estimate the pose of the robot and the state of doors in the environment. They represented the environment using a reference occupancy grid where the location of the doors is known, but not their state (i.e., opened or closed). Petrovskaya and Ng [18] proposed a similar approach where instead of a binary model, a parametrized model (i.e., opening angle) of the doors is used. Stachniss and Burgard [19] clustered local grid maps to identify a set of possible configurations of the environment. The Rao-Blackwellized particle filter is then used to localize the robot and estimate the configuration of the environment from the set. In contrast to their methods, we estimate the state of the complete environment, and not only of a small, specific area or element. Additionally, we also learn the model parameters from data and we are able to generalize over unforeseen environment configurations.

Meyer-Delius *et al.* [20] kept track of the observations caused by unexpected objects in the environment using temporary local maps. The robot pose is then estimated using a particle filter that relies both on these temporary local maps and on a reference map of the environment. The work, however, still relies on a static map for global localization and temporary maps are only created when a failure in position tracking occurs.

An interesting approach to lifelong mapping in dynamic environments has been presented by Konolige *et al.* [21]. The approach focuses mainly on visual maps, and presents a framework where local maps (views) can be updated over time and new local maps are added/deleted when the configuration of the environment changes. Using similar ideas, Kretzschmar *et al.* [22], applied graph compression using an efficient information-theoretic graph pruning strategy. The approach can be used with a bias on more recent observations to obtain a similar behavior of the work of Konolige *et al.* [21]. Those two approaches, however, mainly focus on the scalability problems arising in long-term operations and not on the dynamical aspect of environments changing over time.

The approach of Walcott-Bryant *et al.* [23] follows a similar idea as well. They introduce a novel representation, the Dynamic Pose Graph (DPG), to perform SLAM in environments with low dynamics over long periods of time. Their approach, DPG-SLAM, assumes that data association is solved using scan matching and that the starting position of the robot is known and coincides with the last pose of the previous run. On the contrary, our approach can deal with different dynamics (low, medium and high) and does not require the knowledge of the initial position of the robot.

Churchill and Newman [24] presented a different point of view to the problem of lifelong mapping. They reason that a global reference frame is not needed in navigation and introduce *experiences*, i.e., robot paths with relative metrical information. Experiences can be connected together using appearance-based data association methods and places that change over time are represented by a set of different experiences.

Their work is similar in spirit and generalizes both Stachniss and Burgard [19] and Meyer-Delius *et al.* [20]. In our paper we take another perspective, where we propose a *model-based* approach in contrast to the *data-driven* one of Churchill and Newman [24]. We believe that having a unified model of the environments is important for human robot interaction. Navigation tasks, e.g., go to a specific position or follow a specific path, only need to be defined once in model-based approaches. Using the experience approach, on the contrary, requires the user to define the same task on every experience the robot recorded. This may increase the complexity of human-robot interaction and reduce the reliability of the system, since maintaining the task consistence across different experiences is not trivial. Moreover, with our approach, we are also able to predict how part of the environment will look in the future, by directly modeling their rate of change.

III. Problem Statement

Imagine a robot that continuously performs tasks during a session, i.e., until its battery runs out. It then gets back to its charging station and, when charged, continues to perform the previous tasks in the next session. To perform its tasks, the robot knows the positions of several locations of interest and plans paths to reach them avoiding obstacles. These locations need to be consistent between several sessions, to reduce the teaching effort of the user. One way to ensure this is to use a

single map and express those locations in a global reference frame. Localizing the robot in the global frame allows then to know the displacement between the current robot pose and the locations of interest.

In static environments, the map does not change between different sessions. This simplifies the problem and a readily available solution is to build a map of the environment during the first session using a SLAM algorithm [25] and then use the computed map for localization in the remaining sessions. This solution is very mature, effective, and robust to minor changes in the environment and dynamic objects [26]. Nevertheless, if the amount of changes increases and low dynamic phenomena appear (i.e., boxes that stay for long, parked cars, moved furniture, etc ...) such a solution loses reliability and precision. The robot starts to get de-localized and invalid paths are planned, since the map does not represent the current state of the environment.

In those cases, since the environment changes over time and the map needs to be updated, a straightforward solution would be to always use a SLAM algorithm and rely on loop closure techniques to join the trajectories of subsequent sessions together. This approach has several shortcomings, which we will show in more details in Section VI. A first problem consists in scalability issues. In case a graph-based approach for SLAM is used, the amount of data that needs to be processed grows linearly with the length of the traveled path. Data compression techniques have been developed to discard measurements according to the entropy of the map distribution or some decay value. Those techniques represent a step towards the solution although the more general exploration vs. exploitation aspect – when to stop mapping and start localizing – still remains. Filtering based approaches for SLAM may be better fitted in this case, since the trajectories are marginalized out and their complexity scales with the size of the environment. However, filtering approaches do not scale well with respect to the size of the environment and do not offer the flexibility of graph-based approaches to select which observations to use at the moment to have an up-to-date map.

A second problem regards data association issues. Algorithms designed for static environments may fail in dynamic ones since unexplained measurements can be caused by either incorrect localization or environment changes. Optimization solutions to SLAM cannot be used effectively, since a single hypothesis is not sufficient to disambiguate those situations. Moreover, no assumption can be made on the prior location of the robot between multiple sessions of the lifelong operation, since the robot starting position may have been changed as well. One way of addressing this point is to use multiple-hypotheses filters for SLAM, where each hypothesis carries its own map. In the case of medium-sized environments and unknown initial location, this leads to high memory consumption and scalability issues. Finally, static representations of the environment are not suited for lifelong navigation in changing environments. The reason is that they are slow to be updated when a change happens, since they have a *memory* effect and they need to observe an occupied cell as free as many times as it was observed as occupied to change its state. An experimental evaluation of these shortcomings with additional insights is presented in Section VI.

In this paper, we aim at presenting another point of view for the lifelong localization problem. We argue that addressing it as a pure SLAM or localization problem is suboptimal and that is because this problem is neither an instance of a SLAM problem nor of a pure localization problem. It is, in our opinion, a hybrid problem that lies in-between them and needs to be tackled in a particular way. In the remainder of the paper, we will describe our solution to the problem by first describing a map representation tailored for dynamic environments and then an efficient and effective algorithm to jointly estimate the pose of the robot and the environment configuration.

IV. Occupancy Grids for Changing Environments

Occupancy grids [27] are one of the most popular representations of a mobile robot's environment. They partition the space into rectangular cells and store, for each cell, a probability value indicating whether the underlying area of the environment is occupied by an obstacle or not.

One of the main disadvantages of occupancy grids for our problem is that they assume the environment to be static. To be able to deal with changes in the environment we utilize a *dynamic occupancy grid* [4], a generalization of an occupancy grid that overcomes the static-world assumption by explicitly accounting for changes in the environment. In the remainder of this section, we give a short overview on the representation and in the next section we will describe how it can be used for global localization.

The map consists of a collection of individual cells, $m_t = \{c_t^{(i)}\}$, where each cell is modeled using an hidden Markov model (HMM) and requires the specification of a state transition probability, an observation model, and an initial state distribution. We adopt the same assumptions of the occupancy grid model, namely the independence of the individual observations, given the map, and the independence among the cells. Those assumptions are needed for having an efficient inference during localization. We believe those are reasonable assumptions and, during our experiments, we did not observe any critical issues either in localization or in the accuracy of the map.

Let c_t be a discrete random variable that represents the occupancy state of a cell c at time t. The initial state distribution $p(c_0)$ specifies the occupancy probability of a cell at the initial time step $t = 0$ prior to any observation.

The state transition model $p(c_t \mid c_{t-1})$ describes the evolution of the cell between consecutive time steps. In their paper, the authors assume that changes in the environment are caused by a stationary process, that is, the state transition probabilities are the same for all time steps t. Since a cell is either free (*free*) or occupied (*occ*), the state transition model can be specified using only two transition probabilities, namely

$$p_c^{o|f} = p(c_t = occ \mid c_{t-1} = free) \tag{1}$$

and

$$p_c^{f|o} = p(c_t = \textit{free} \mid c_{t-1} = \textit{occ})\,. \tag{2}$$

Note that, by assuming a stationary process, these probabilities do not depend on the absolute value of t. Therefore, the dynamics of a cell at any time can be captured by its transition matrix

$$A_c = \begin{bmatrix} 1 - p_c^{f|o} & p_c^{f|o} \\ p_c^{o|f} & 1 - p_c^{o|f} \end{bmatrix}. \tag{3}$$

This transition model generalizes the one utilized by Murphy *et al.* [16] and Chen *et al.* [11] where $p_c^{o|f} = p_c^{f|o}$.

The observation model $p(z \mid c)$ represents the likelihood of the observation z given the state of the cell c. In our case, it corresponds to the sensor model of a laser range finder. Given the limited field of view of the sensor, we additionally consider the case where no direct measurement is observed by the sensor. Explicitly considering this *no-observation* case allows us to update and estimate the parameters of the model using the HMM framework directly without having to artificially distinguish between cells that are observed and cells that are not. The probabilities can be specified by three matrices

$$B_z = \begin{bmatrix} p(z \mid c = \textit{occ}) & 0 \\ 0 & p(z \mid c = \textit{free}) \end{bmatrix}, \tag{4}$$

where $z \in \{\textit{hit}, \textit{miss}, \textit{no-observation}\}$.

The update of the occupancy state of the cells follows a Bayesian approach. The goal is to estimate the posterior distribution $p(c_t \mid z_{1:t})$ over the current occupancy state c_t of a cell given all the available evidence $z_{1:t}$ up to time t. The update formula is:

$$p(c_t \mid z_{1:t}) = \eta\, p(z_t \mid c_t) \sum_{c_{t-1}} p(c_t \mid c_{t-1})\, p(c_{t-1} \mid z_{1:t-1})\,, \tag{5}$$

where η is a normalization constant. The structure of HMMs allows for a simple and efficient implementation of this recursive approach. Utilizing the matrix notation and defining the posterior at time t as the vector

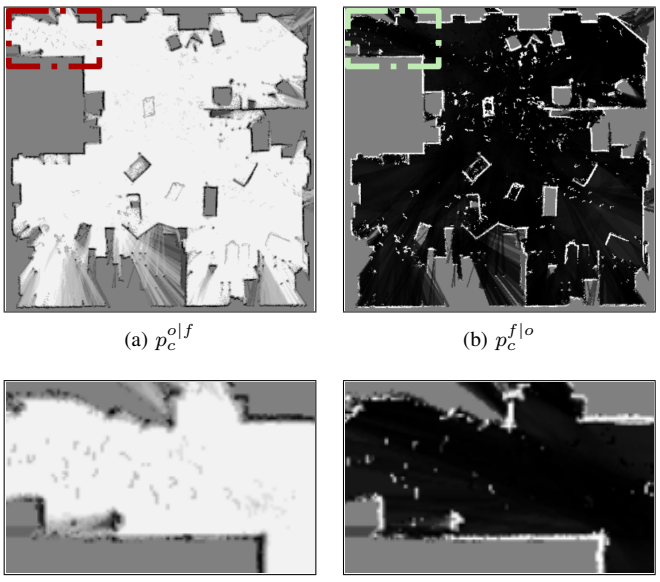
$$Q_t = \begin{bmatrix} p(c_t = \textit{occ} \mid z_{1:t}) & p(c_t = \textit{free} \mid z_{1:t}) \end{bmatrix}, \tag{6}$$

one can compute the posterior at time $t+1$ as

$$Q_{t+1} = Q_t\, A_c\, B_{z_{t+1}}\, \eta\,. \tag{7}$$

Note that the map update for occupancy grids is a special case of our update equations, where the sum in (5) is replaced with the posterior $p(c_t \mid z_{1:t-1})$, or equivalently, the matrix A_c in (7) is replaced with the identity matrix I.

Figure 2 depicts the transition probabilities of a small hall (2a, 2b), and an enlarged portion of it (2c, 2d), where darker colors correspond to higher probabilities. During the data acquisition, some desks have been moved around and people have been walking in the room. Subfigures 2a and 2c show $p^{f|o}$ and Subfigures 2b and 2d show $p^{o|f}$ of the whole environment and an enlarged portion, respectively. Comparing 2a and 2b we see some areas with high probabilities for $p^{f|o}$ and



(a) $p_c^{o|f}$ (b) $p_c^{f|o}$

(c) $p_c^{o|f}$ (d) $p_c^{f|o}$

Fig. 2. State transition probabilities in a small hall where people usually go from the office in the top left corner to one of the two exits on the right. The left and right images correspond to the distributions $p_c^{o|f}$ and $p_c^{f|o}$ respectively: the darker the color, the larger the state change probability. Subfigures (c) and (d) shows an enlarged portion of the top left corner, to better highlight the effect of fast moving objects.

low probabilities for $p^{o|f}$ in the middle of the room. They correspond to furniture being moved around and left there. The high transition probabilities from free to occupied reflect the possibility of the cell to change state quickly (something may appear there), while the low transition probabilities from occupied to free reflect the long persistence of the object.

On the other hand, if we compare 2a and 2b we see a different behavior: some cells have a high probability for both $p^{o|f}$ and $p^{f|o}$. This means that as soon as objects are observed there, their state switches to occupied in the next step (an object appeared) and then quickly returns to free (the object quickly moved away).

A. Parameter Estimation

As mentioned above, an HMM is characterized by the state transition probabilities, the observation model, and the initial state distribution. We assume that the observation model only depends on the sensor used and not on the location. Therefore it can be specified beforehand and is the same for each HMM. We also assume the same uniform initial state distribution for each cell. Thus, the parameters to be learned are the state transition probabilities of the cell.

One of the most popular approaches for estimating the parameters of an HMM from observed data is an instance of the expectation-maximization (EM) algorithm. The basic idea is to iteratively estimate the model parameters using the observations and the parameters estimated in the previous iteration until the values converge. Let $\hat{\theta}^{(n)}$ represent the parameters estimated in the n-th iteration. The EM algorithm

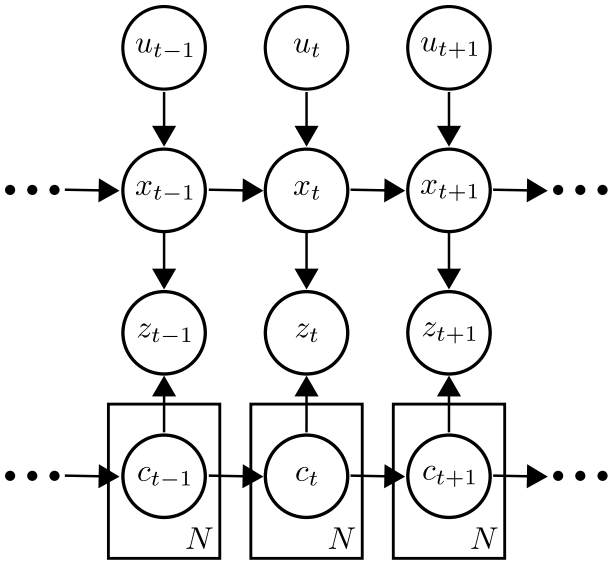


Fig. 3. Dynamic Bayesian network describing the localization problem addressed in the paper. Note the temporal connections among the individual cells of different steps.

results in the following re-estimation formula for the transition model of cell c:

$$\hat{p}(c_t = i \mid c_{t-1} = j)^{(n+1)} = \frac{\sum_{\tau=1}^{T} p(c_{\tau-1} = i, c_\tau = j \mid z_{1:T}, \hat{\theta}^{(n)})}{\sum_{\tau=1}^{T} p(c_{\tau-1} = i \mid z_{1:T}, \hat{\theta}^{(n)})}, \quad (8)$$

where $i, j \in \{\textit{free}, \textit{occ}\}$ and T is the length of the observation sequence. Note that the probabilities on the right-hand side are conditioned on the observation sequence $z_{1:T}$ and the previous parameter estimates $\hat{\theta}^{(n)}$. The probabilities in (8) can be efficiently computed using a forward-backward procedure [28].

V. Lifelong Localization Using Dynamic Maps

In this paper, we consider, without loss of generality, the lifelong localization problem as a multi-session localization and mapping problem. In each session, the robot needs to estimate the distribution of its pose, x_t, and the current map configuration, m_t, given the prior on the robot pose, x_0, the *lifelong map*, m_0, the current observations, $z_{1:t}$, and commands, $u_{1:t-1}$

$$p(x_{1:t}, m_t \mid z_{1:t}, u_{1:t-1}, m_0, x_0). \quad (9)$$

Its lifelong aspect lies in the fact that the pose of the robot and the state of environment is not deleted between sessions but carried on in the next one. This formulation naturally extends to the case of continuous navigation, where each session can be seen as one individual measurement.

We explicitly do not put any constraints on x_0 and m_0. Any representation and any distribution can be used. For example, the pose prior x_0 can range from being a Gaussian with a small covariance matrix or a uniform distribution. In the first case, this may indicate, for instance, that the robot always starts from and ends in its *charging station*. The second case, on the contrary, indicates that the robot has been moved between sessions and needs to perform global localization. In a similar way, the lifelong map m_0 represents our prior knowledge over the environment and may be updated at the end of each session. This map can be represented in a data-driven fashion, as a distribution over previously observed configurations [24, 19], or in a model-based fashion, e.g., employing the hidden Markov model maps [4]. A graphical model representation of the problem is illustrated in Figure 3.

Our problem shares the same state space of SLAM, i.e., the robot pose and the state of the map, and one can exploit the same factorization used in Rao-Blackwellized particle filters [16, 29]. The key idea is to separate the estimation of the robot trajectory from the map estimation process,

$$p(x_{1:t}, m_t \mid z_{1:t}, u_{1:t-1}, m_0, x_0) = p(m_t \mid x_{1:t}, z_{1:t}, m_0, x_0) p(x_{1:t} \mid z_{1:t}, u_{1:t-1}, m_0, x_0). \quad (10)$$

This can be done efficiently, since the posterior over maps $p(m_t \mid x_{1:t}, z_{1:t}, m_0, x_0)$ can be computed analytically given the knowledge of $x_{1:t}$, $z_{1:t}$ and m_0 and using the forward algorithm for the HMM. The remaining posterior, $p(x_{1:t} \mid z_{1:t}, u_{1:t-1}, m_0, x_0)$, is estimated using a particle filter that incrementally processes the observations and the odometry readings. Following Doucet *et al.* [30], we can use the motion model $x_t^{(i)} \sim p(x_t | x_{t-1}^{(i)}, u_{t-1})$ as proposal distribution $\pi(x_t)$ to obtain a sample of the robot pose. This recursive sampling schema allows us to recursively compute the importance weights using the following equation

$$\begin{aligned} w_t^{(i)} &= w_{t-1}^{(i)} \frac{p(z_t \mid x_{1:t}^{(i)}, z_{1:t-1}, m_0, x_0) p(x_t^{(i)} \mid x_{t-1}^{(i)}, u_{t-1})}{\pi(x_t^{(i)})} \\ &= w_{t-1}^{(i)} \frac{p(z_t \mid x_{1:t}^{(i)}, z_{1:t-1}, m_0, x_0) p(x_t^{(i)} \mid x_{t-1}^{(i)}, u_{t-1})}{p(x_t^{(i)} \mid x_{t-1}^{(i)}, u_{t-1})} \\ &= w_{t-1}^{(i)} p(z_t \mid x_{1:t}^{(i)}, z_{1:t-1}, m_0, x_0). \end{aligned} \quad (11)$$

The observation likelihood is then computed by marginalization over the predicted state of the map leading to

$$\begin{aligned} p(z_t \mid x_{1:t}^{(i)}, z_{1:t-1}, m_0, x_0) &= \int p(z_t \mid x_t^{(i)}, m_t) p(m_t \mid x_{1:t-1}^{(i)}, z_{1:t-1}, m_0, x_0) dm_t \\ &= \prod_j \mathcal{N}(z_t^j; \hat{z}_t^j, \sigma^2), \end{aligned} \quad (12)$$

where z_t^j is an individual range reading, $\hat{z}_t^j$ is the range to its closest cell in the map with an occupancy probability above a certain threshold and we have that

$$\begin{aligned} p(m_t \mid x_{1:t-1}^{(i)}, z_{1:t-1}, m_0, x_0) &= \int p(m_t \mid m_{t-1}) p(m_{t-1} \mid x_{1:t-1}^{(i)}, z_{1:t-1}, m_0, x_0) dm_{t-1} \\ &= p(m_t \mid m_{t-1}^{(i)}). \end{aligned} \quad (13)$$

The term $m_{t-1}^{(i)}$ represents the map associated with particle i and is computed in the previous time step. The map motion

Algorithm 1: RBPF-HMM for Changing Environments

In: The previous sample set $\mathcal{S}_{t-1}$, the lifelong map m_0, the current observation z_t and odometry u_t

Out: The new sample set $\mathcal{S}_t$

$\mathcal{S}_t = \{\}$

foreach $s_{t-1}^{(i)}$ *in* $\mathcal{S}_{t-1}$ **do**

  $< x_{1:t-1}^{(i)}, m_{t-1}^{(i)}, w_{t-1}^{(i)} >= s_{t-1}^{(i)}$

  `// State prediction`

  $x_t^{(i)} \sim p(x_t | x_{t-1}^{(i)}, u_{t-1})$

  $x_{1:t}^{(i)} = x_t^{(i)} \cup x_{1:t-1}^{(i)}$

  foreach c_{t-1} *in* $m_{t-1}^{(i)}$ **do**

    $$p(c_t \mid z_{1:t-1}) = \sum_{c_{t-1} \in \{f,o\}} p(c_t \mid c_{t-1}) p(c_{t-1} \mid z_{1:t-1})$$

  end

  `// Weight computation`

  $$w_t^{(i)} = w_{t-1}^{(i)} \prod_j \mathcal{N}(z_t^j; \hat{z}_t^j, \sigma^2)$$

  `// Map update`

  foreach c_t *in* $m_t^{(i)}$ **do**

    $p(c_t \mid z_{1:t}) = \eta p(z_t \mid c_t) p(c_t \mid z_{1:t-1})$

  end

  $\mathcal{S}_t = \mathcal{S}_t \cup \{< x_{1:t}^{(i)}, m_t^{(i)}, w_t^{(i)} >\}$

end

`// Normalize the weights`

normalize($\mathcal{S}_t$)

`// Resample`

$$N_{\mathit{eff}} = \frac{1}{\sum_i (w_t^{(i)})^2}$$

if $N_{\mathit{eff}} < T$ **then**

  $\mathcal{S}_t$ = resample($\mathcal{S}_t$)

end

model $p(m_t \mid m_{t-1}^{(i)})$ is computed using the HMM as described in Section IV. Note that the *disappearance* of the integral is not an approximation but a direct consequence of using the likelihood field model [31] and the Dirac distribution for the particle.

To improve the robustness against objects with high dynamics, we applied a robust likelihood function when computing the weights. This approach is similar in spirit to the works in which high dynamic objects are treated as outliers [1, 2].

The overall process for global localization proposed in this paper is summarized in Algorithm 1.

Note that our representation resembles the one used for SLAM and for Monte Carlo localization approaches. The presented problem, however, differs from them with respect to the assumptions made. In the common SLAM formulation, one assumes to have no information on the map (uniform distribution) and to know the initial pose with certainty (Dirac distribution). In global localization, on the contrary, the map is assumed to be static and known (Dirac distribution) and a weak prior on the initial pose is available (uniform distribution on the free space or Gaussian). Our problem is somewhat in between, since we assume a weak prior on the initial pose, like in global localization, and a weak prior on the map (the dynamic occupancy grid in our case).

A. Map Management

As already mentioned above, the distribution of the initial pose of the robot is generally unknown and it may be uniformly distributed over the environment. This forces us to use a high number of particles, generally in thousands, to accurately represent the initial distribution. Since every particle needs to have its own estimate of the map, memory management is a key aspect of the whole algorithm. It is worth noticing that even if memory is becoming cheap and available in large quantity, the amount needed is still beyond what is currently available. As an example, in an environment of $200\text{x}200\,\text{m}^2$, stored at a resolution of $0.1\,\text{m}$, and using $10,000$ particles we need about $50\,\text{GB}$ of memory. In order to save memory, we want to only store the cells in the map that have been considerably changed from the lifelong map m_0, which is shared among the different particles. This is done by exploiting two important aspects of the Markov chain associated to the HMM: the *stationary distribution* and the *mixing time*.

The occupancy value of a cell converges to a unique stationary distribution π as the number of time steps for that no observation is available tends to infinity [32]. This stationary distribution represents the case where the environment has not been observed for a long time and is represented by the limit distribution of our lifelong map m_0. In the case of a binary HMM as the one used in this paper, this distribution is computed using the transition probabilities

$$\begin{bmatrix} \pi_f \\ \pi_o \end{bmatrix} = \frac{1}{p_c^{o|f} + p_c^{f|o}} \begin{bmatrix} p_c^{f|o} \\ p_c^{o|f} \end{bmatrix}. \tag{14}$$

Every time an individual particle observes the state of a cell for the first time, the state distribution of that particular cell changes from the stationary one and the particle needs to store the new state of the cell. In order to reduce memory requirements, only a limited number of cells should be stored and a *forgetting* mechanism should be implemented. This can be done in a sound probabilistic way, by exploiting the mixing time of the associated Markov chain. The mixing time is defined as the time needed to converge from a particular state to the stationary distribution. The concrete definition depends on the measure used to compute the difference between distributions. In this paper we use the total variation distance as defined by Levin *et al.* [32]. Since our HMMs have only two states, the total variation distance Δ_t between the stationary distribution π and the occupancy distribution p_t at time t can be specified as

$$\Delta_t = |1 - p_c^{o|f} - p_c^{f|o}|^t \Delta_0\,, \tag{15}$$

where $\Delta_0 = |p(c_t = \textit{free}) - \pi_f| = |p(c_t = \textit{occ}) - \pi_o|$ is the difference between the current state $p(c_t)$ and the stationary

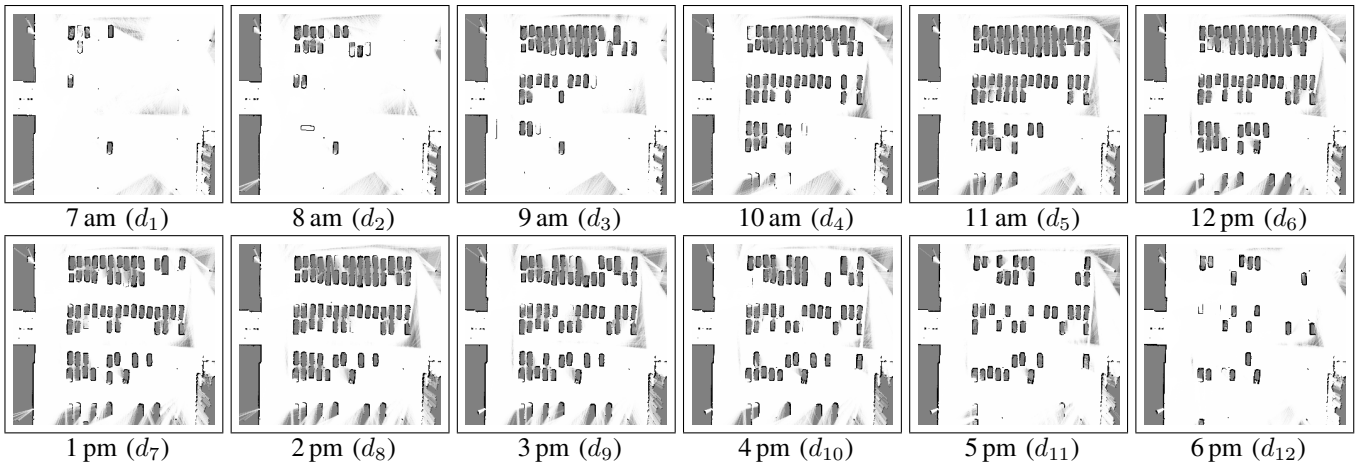


Fig. 4. Maps of the parking lot during the different runs. Each image shows the ground-truth map for the corresponding run.

distribution π. Based on the total variation distance, we can define the mixing time t_m as the smallest t such that the distance Δ_{t_m} is less than a given confidence value ϵ. This leads to

$$t_m = \left\lceil \frac{ln(\epsilon/\Delta_0)}{ln(|1 - p_c^{o|f} - p_c^{f|o}|)} \right\rceil . \tag{16}$$

In other words, the mixing time tells us how many steps are needed for a particular cell to return to its stationary distribution, given an approximation error of ϵ, i.e., how many steps a particle needs to store an unobserved cell before removing it from its local map and relying on the lifelong map m_0.

The approach employed here is orthogonal to the work of Eliazar and Parr [33], where map memory consumption is reduced by exploiting parent-child relationships between resampled particles. Note, further, that when the global localization is initialized, each particle still needs to store an individual map, since no parent particle is present.

Moreover, the map management reduces the computational complexity as well, since in the naive approach every cell has to be updated in the prediction step, while in our case we need to update only the cells belonging to the local maps.

B. Time and Memory Complexity

In this section we describe the time and memory complexity of the presented algorithm in more detail. First of all, the algorithm, as the original MCL one, depends on the number of particles used, which in turns depends on the size of the free space. Although there exist techniques to estimate this number based on statistical divergences with respect to a fixed target distribution [34], this goes beyond the purpose of this paper. We assume, without loss of generality, that the number of particles does not change during the execution of the algorithm.

In a standard RBPF mapping approach [29, 16], each particle needs to store separately a full map of the environment, resulting in a constant time prediction and update for each particle, but at the cost of storing a copy of the map for each particle. Let N be the number of particles and M the size of the map, we have an asymptotic time complexity of $O(N)$ and an asymptotic memory complexity of $O(NM)$. This is also valid when using plastic maps [11, 12, 13].

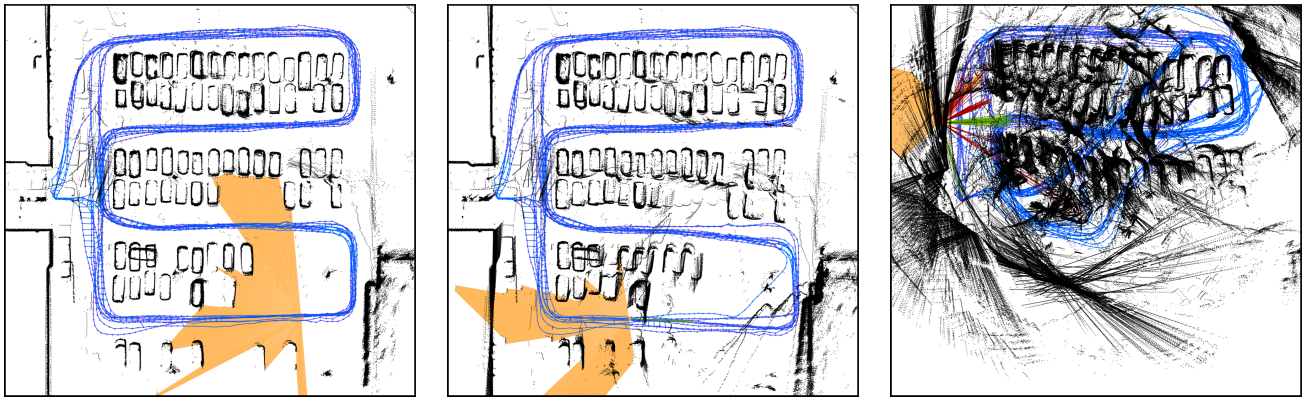
However, if we fully exploit the mixing time and the stable distribution of our model, each particle only need to store a small local map, consisting of the cells that have been observed lately. More formally, let k be the maximum mixing time and A the maximum number of cells observed in each time step, we have that the average number of cells in the local map is bounded by kA. Since those values are constant both with respect to the map size and the number of particles, we have the same asymptotic time complexity as the naive implementation, $O(N)$, but with a lower memory complexity of $O(M+N)$. The cost that we pay is simply a multiplication factor for looking up the cells in the local map, which can be implemented efficiently using kd-trees or hash tables.

Note that this is only achievable thanks to the stationarity of the HMM. If we remove the stationarity assumption from the model, the transition matrix depends on the newly observed cells and would be generally different for each particle. This can be indeed seen as a limitation of the system. However, the experimental results show that it does not have a strong impact in practice.

VI. Experiments

We tested our proposed approach in the parking lot of our university and collected a data set with a MobileRobots Powerbot equipped with a SICK LMS laser range finder. The robot performed a run in the test scenario every full hour from 7 am until 6 pm during one day. Figure 4 shows the maps of the parking lot corresponding to each individual run. The range data obtained from the twelve runs (data sets d_1 through d_{12}) corresponds to twelve different configurations of the parked cars, including an almost empty parking lot (data set d_1) and a relatively occupied one (data set d_8).

We chose to use the parking lot as our test bed for several reasons. The parking lot is naturally suited to test algorithms for localization in changing environments and low dynamic



After three sessions (7-9 am) First data association error (9 am) At the end of the day (6 pm)

Fig. 5. SLAM experiment using a graph-based SLAM algorithm. The picture shows that the robot was able to maintain a consistent map up to the third session (left). At that time, a first data association error was introduced by the front-end (middle). At the end of the day, the map is inconsistent (right), due to several data association errors introduced. Blue lines represent the pose graph and the colored area the current laser measurement. Data association hypotheses that passed the consistency test and that were sent to SCGP [35] are shown as green lines whereas the ones that did not are displayed in red.

objects. The cars have not been artificially moved, thus no experimental bias on which kind of dynamics or where to place object has been introduced from us. The environment presents objects at different rate of changes: during the data collection we observed static objects (few buildings), low dynamic objects (parked cars) and high dynamic objects (moving cars and people). The empty parking lot is extremely ambiguous and most of the time a robot will navigate in open space with no measurement available (see Figure 11). Finally, it represents a variety of useful environments where low dynamics are present, such as factory floors, warehouses and environments in logistic and industrial scenarios.

A standard SLAM approach for static environments [29] has been used to correct the odometry of the robot in each data set separately. The resulting maps have been manually aligned to obtain a good estimate of the robot pose to use as ground-truth. The underlying assumption here is that the parking lot did not change considerably during a run. Furthermore, the trajectory and velocity of the robot during each run were approximately the same to avoid a bias in the complete data set.

Four experiments have been performed. In the first experiment, we analyzed what happens if SLAM is used to address this problem. In the second experiment, we performed a systematic comparison between our approach and other localization algorithms on both a global localization and position tracking problems. In the third and the last experiment, we compared our approach with state-of-the-art methods in localization in high dynamic environments and lifelong mapping respectively.

A. Comparison with State-of-the-art SLAM Techniques

The aim of this experiment is to demonstrate that the application of standard SLAM approaches to the lifelong localization problem is suboptimal and that changes in the environment can jeopardize state-of-the-art SLAM algorithms due to false positives in data association. Note that these false positives are systematic and cannot be avoided by tuning the parameters of the SLAM front-end, since they are due to changes in the environment. We will also show that multiple hypothesis SLAM algorithms like that of Grisetti *et al.* [29] lead to inconsistent maps if standard occupancy grids are used. This is mainly due to the lack of plasticity of the representation.

In the experiment we employed both, a graph-based and an RBPF-based SLAM algorithm. The graph-based approach has been chosen for two reasons. The first reason is that graph-based algorithms are considered the state of the art for solving SLAM. The second reason is that this experiment provides an indirect comparison with the Dynamic Pose Graph [23], since it relies on scan matching and graph-based optimization. We chose to use an RBPF-based algorithm in our tests to demonstrate that multi-hypothesis SLAM algorithms perform better than the maximum likelihood ones in the presence of changes. We believe this is due to the fact that they rely on a lazy mechanism for data association, where wrong associations can be rejected after observing more data.

For the graph-based algorithm we chose HOG-Man [36] as SLAM back-end and we follow the approach of Olson [37] for the front-end. While the back-end was chosen for its on-line nature, we selected the front-end because of its robustness to outliers and high precision, since potential data association candidates are first obtained by means of matching the relative scans and then fed into a consistency check using single cluster graph partitioning (SCGP) [35]. They both represent the state of the art for on-line maximum likelihood mapping and data association.

With respect to the RBPF-based algorithm, we chose GMapping [29]. This algorithm represents a robust RBPF SLAM solution and is widely used in several research centers and companies. For the experiment, we concatenated the 12 sessions collected at the parking lot into one session. The final and starting pose of the robot were manually aligned to preserve

continuity in the robot path. Both SLAM algorithms were then tested on the resulting data as it would have been a single robot run.

Figure 5 shows the performance of the graph-based algorithm. The algorithm is able to correctly track the robot for the first three sessions and estimates the correct trajectory and a consistent map of the environment (see the leftmost figure). This is also due to the fact that during the first two sessions, the environment did not change substantially, since few cars were parked there before 9 am. However, around 9 am most of the people came to work and the parking lot configuration changed. In the middle figure, we see that the front-end mistakenly added a loop closing transformation between two robot poses, leading to an inconsistent map. Note that this is not necessarily an error of the front end. The two observations were really almost identical, since a car parked on a spot that was free in the previous sessions and the system matched it with the car parked beside it. Finally, the rightmost figure shows the robot trajectory and the map at the end of the day (6 pm), when all data has been processed. As we can see, the map is highly inconsistent and the robot path wrong.

Please note that this performance violates one of the assumption made in DPG-SLAM [23], i.e., that the trajectory estimation can be done using graph-based SLAM algorithms in combination with scan matching for detecting loop closing edges (more details on this assumption are provided in Walcott-Bryant's PhD thesis [38], Chapter 4, Section 4.1.3).

This behavior of the system convinced us that to better address this data association problem a multi hypotheses approach was needed. To this end, we performed the same experiment using GMapping [29]. By tuning the number of particles, we were able to obtain a consistent map of the environment at the end. However, two limitations are still present. Firstly, the algorithm was slower than the graph-based one and certainly not adapted to on-line navigation. This is mainly due to the number of particles needed and the continuous map updates. Note that this also impose memory limitations, due to each hypothesis carrying its own map. Secondly, the update rate of the map strongly depended on how often a certain cell has been already observed.

This phenomena are however not present when using the dynamic occupancy grid. Figure 6 shows the difference between occupancy grid and the dynamic occupancy grid used in this paper. The top row shows the actual configuration of the environment, the middle row the result using the dynamic occupancy grid and the bottom row using standard occupancy grids. Both maps are computed using the robot trajectory estimated by the RBPF algorithm. In the left column of the figure we see the result on the third session (9 am) and in the right column the results on the last session (6 pm). As one can see already at 9 am, the occupancy grid is not able to correctly represent the environment everywhere, but only in places where the cars were already present in the previous sessions (the parking lot is usually filled starting from the bottom right corner, since it is closest to the entrance). The phenomenon is even greater towards the end of the day, where

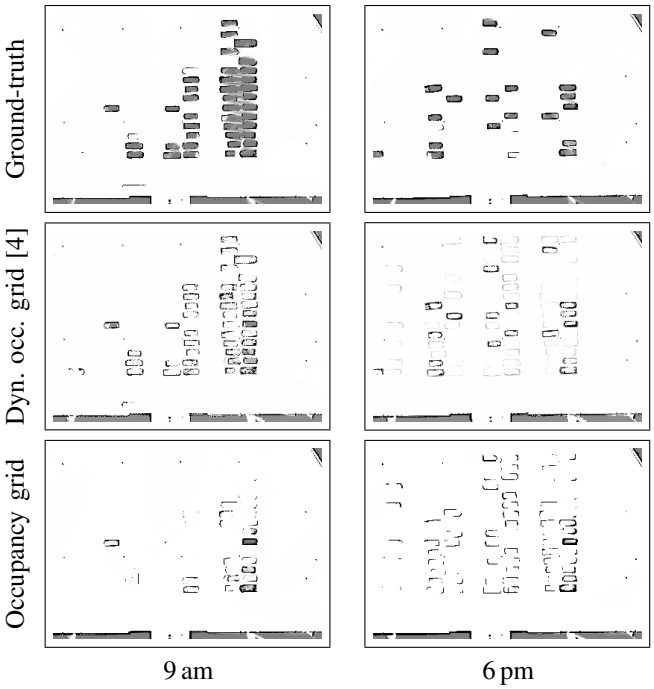


Fig. 6. SLAM experiment using an RBPF-based SLAM algorithm. Shown are the ground-truth maps (top), the dynamic occupancy grids (middle), and standard occupancy grids (bottom). The left column shows the session recorded at 9 am and the right column the one at 6 pm.

many people left the place. There, the occupancy grid was still believing that the parking lot is almost full, without removing the cars that left from the map.

B. Localization Experiment

In order to assess the performance of the localization approach, we compared it to state-of-the-art localization approaches both in a global localization and a position tracking setting. For each data set, we compared our approach (RBPF-HMM), MCL using the standard occupancy grid (MCL-S), MCL using the ground-truth map for that specific data set (MCL-GT), and MCL using the temporary maps [20] (MCL-TM). In global localization, MCL-TM relies on MCL-S before convergence, hence we aggregate the two results in Figure 8 and Table I.

For all the aforementioned approaches, we used the Blake-Zisserman [39] robust function to compute the likelihood. This choice is motivated by robustness to outliers, hence to objects with high dynamics. The Blake-Zisserman function is a combination of Gaussian distribution and a uniform distribution with one parameter that define the crossover point.

We performed 100 runs for each data set, where we randomly sampled the initial pose of the robot. In order to obtain a fair comparison, the same seed has been used to generate the initial pose, as well as to perform all the random sampling processes for each approach. All the approaches have been initialized with $10,000$ particles for global localization and 500 particles for pose tracking. They all used the same set of parameters as well: an occupancy threshold of 0.6 and a

TABLE I
GLOBAL LOCALIZATION EXPERIMENT

Data set	MCL-GT			RBPF-HMM			MCL-S / MCL-TM		
	Success	Error^2	σ^2	Success	Error^2	σ^2	Success	Error^2	σ^2
01	100%	0.21	0.36	50%	0.26	0.36	50%	0.26	0.18
02	100%	0.19	0.29	40%	0.10	0.08	33%	0.13	0.09
03	100%	0.13	0.19	80%	0.10	0.29	52%	0.19	0.17
04	100%	0.04	0.03	60%	0.08	0.14	53%	0.15	0.19
05	100%	0.07	0.18	54%	0.07	0.09	35%	0.15	0.18
06	100%	0.02	0.01	87%	0.02	0.02	45%	0.06	0.02
07	100%	0.06	0.08	59%	0.12	0.22	43%	0.14	0.20
08	100%	0.05	0.10	71%	0.03	0.02	28%	0.03	0.01
09	100%	0.02	0.01	53%	0.12	0.22	31%	0.06	0.02
10	100%	0.14	0.28	62%	0.13	0.31	34%	0.30	1.01
11	100%	0.11	0.11	38%	0.15	0.21	26%	0.24	0.29
12	100%	0.19	0.32	20%	0.16	0.14	22%	0.27	0.38
Total	100%	0.11	0.19	52%	0.11	0.18	36%	0.17	0.22

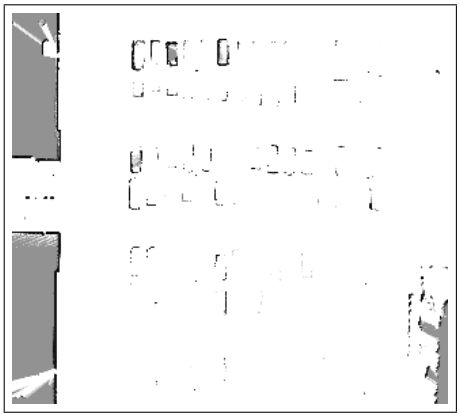


Fig. 7. Prior map used in the localization experiments.

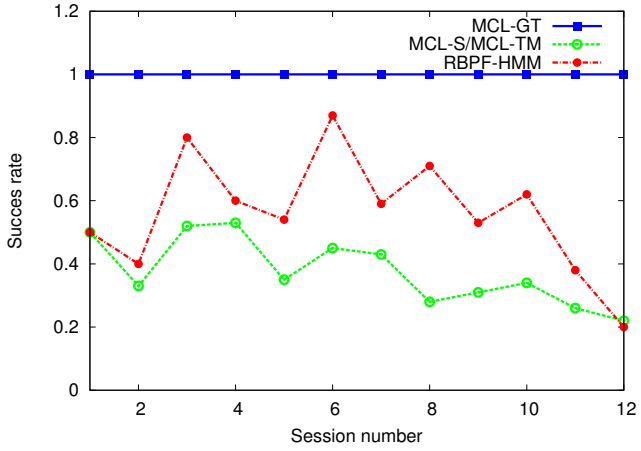


Fig. 8. Success rate for the global localization experiment. The graph shows the success rate of the different algorithms. The MCL-GT algorithm is used as baseline and thus has a 100% success rate. As can be seen our approach (RBPF-HMM) outperforms MCL-S/MCL-TM.

crossover parameter of $1\,m$ for the Blake-Zisserman. To verify convergence, we consider the determinant of the covariance matrix for the translation and rotation part. If the translation determinant is kept below 0.5 and the angle determinant below 0.3 for a distance of at least $0.2\,m$, we assume the filter to have converged to a solution. We then compare the estimated pose with the ground-truth and if this distance is below $1\,m$ and $0.5\,rad$ we consider the run a success, otherwise it is a failure.

The static maps used for MCL-GT have been computed using the corrected log files from the SLAM algorithm and rendering the points into an occupancy grid. The static map for MCL-S has been computed using all the datasets as they would be a unique run of the SLAM algorithm over the whole day. The dynamic map has been estimated using a leave-one-out cross validation over the corrected log files. For each test run, all other runs were used to learn the transitions of the dynamic map. In this way the data on which the RBPF-HMM algorithm was run was never used for the learning part. The RBPF-HMM has been initialized using the static map as the MCL-S as a prior map m_0, for a fair comparison. Figure 7 shows the prior map m_0.

The results of the global localization experiment are shown in Figure 8 and Table I. The figure shows the success rate of the global localization, as the percentage of time the filter converged to the true pose. The table shows the numerical values of the success rate and the residual squared error, with respective variance, after convergence. The success rate is reported relative to the result of MCL on the ground-truth map, in order to have a measure independent of the complexity of the environment. The results show that our approach outperforms the standard MCL on static maps both in terms of convergence rate and accuracy in localization.

Figure 9 and Table II show the results for the position tracking experiment, where the filter is initialized around the true pose and keeps tracking the robot. As before, the figure shows the failure rate, i.e., the percentage of time the robot got lost during tracking, and the table the numerical values of the failure rate as well as the residual squared error in the case the tracking was successful. The results of this experiment show that the performance of our approach in position tracking is almost equivalent to MCL with the ground-truth maps, with a

TABLE II
POSITION TRACKING EXPERIMENT

Data set	MCL-GT			RBPF-HMM			MCL-S			MCL-TM		
	Failure	Error^2	σ^2	Failure	Error^2	σ^2	Failure	Error^2	σ^2	Failure	Error^2	σ^2
01	0%	0.04	0.01	3%	0.09	0.03	5%	0.18	0.07	0%	0.25	0.16
02	0%	0.03	0.01	4%	0.08	0.05	24%	0.18	0.10	0%	0.16	0.04
03	0%	0.04	0.01	2%	0.05	0.04	10%	0.09	0.04	12%	0.63	0.45
04	0%	0.02	0.01	0%	0.04	0.01	10%	0.08	0.02	29%	0.63	0.51
05	0%	0.02	0.01	3%	0.03	0.04	13%	0.06	0.02	1%	0.51	0.31
06	0%	0.02	0.01	2%	0.02	0.01	26%	0.09	0.12	0%	0.21	0.05
07	0%	0.02	0.01	0%	0.03	0.01	34%	0.07	0.01	1%	0.44	0.24
08	0%	0.02	0.01	2%	0.02	0.01	35%	0.09	0.15	35%	0.59	0.56
09	0%	0.02	0.01	4%	0.03	0.01	37%	0.07	0.16	4%	0.49	0.31
10	0%	0.02	0.01	0%	0.03	0.01	36%	0.09	0.10	0%	0.32	0.12
11	0%	0.03	0.01	1%	0.05	0.02	42%	0.10	0.05	1%	0.47	0.28
12	0%	0.03	0.01	5%	0.06	0.01	44%	0.15	0.20	0%	0.23	0.04
Total	0%	0.03	0.01	2%	0.04	0.02	27%	0.10	0.08	7%	0.41	0.25

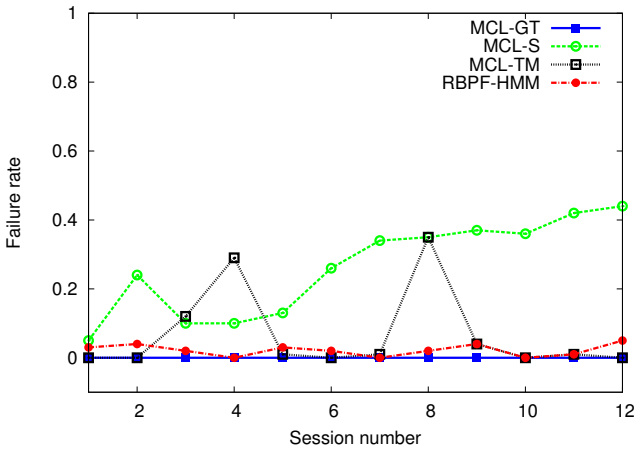


Fig. 9. Failure rate for the position tracking experiment. The graph shows the failure rate of the different algorithms. The MCL-GT algorithm is used as baseline and thus has a 0% failure rate. As can be seen our approach (RBPF-HMM) outperform MCL-S and MCL-TM.

failure rate of only 2%.

The comparison with the temporary map approach [20] reveals two important messages. The first message is that the proposed approach is always more precise in terms of residual error. This come with no surprise, since the estimation of the local surrounding is in some sense constrained by the map geometry and static objects can only appear in places where they have been seen during learning. On the contrary, the temporary maps get initialized with the current pose estimate of the robot and introduce a bias in the estimation that is almost impossible to remove. The second message is that the proposed approach is more robust to the changes and to the initialization. This is evident from the failure rate, where the temporary maps approach is almost always on par with the RBPF-HMM but in three cases, and in one case is even worse than standard MCL. The problem is that if the temporary map is created from a wrong position, there is no possibility to recover, the worst case being when the observations matching the prior map are considered as outliers.

In terms of runtime and size of the local map, we experienced an average mixing time of $k = 10$ and an average size of the local map of about 250 cells. The original map size is 369x456 pixels with a resolution of 0.1 m. A standard RBPF with an occupancy grid map needs about 16GB of memory in the global localization and little less than 1GB for position tracking. Our technique, instead, only uses about 24MB for global localization and 5MB for position tracking, resulting in a memory saving of about three orders of magnitude.

A frame to frame comparison between the proposed approach and standard MCL is shown in Figure 10 and in Extension 1. Both algorithms have the same parameters and the same seed for the random number generator. As it can be seen, MCL converges too fast to a wrong solution, believing that the measurements coming from parked cars (not present in the a priori map) have been generated by the wall on the bottom right (frame 8). The proposed approach, on the other hand, has a slower convergence rate due to the uncertainty in the map estimate. After a few frames (frame 12) it finally converges to the right position. In the last frame one can see the updated map, which better reflects the current configuration of the environment.

Both experiments show two important aspects of the problem and of the solution adopted. The first aspect is that the problem is much more complex than global localization since the search space is bigger and deciding if a measurement is an outlier or is caused by a change of the configuration is not a trivial task. Furthermore, analyzing the performance results in position tracking, we see that if the filter is initialized close to the correct solution, i.e., the search is reduced to the correct basis of attraction, it is able to estimate the correct configuration. The second aspect is how the algorithm scales with different amounts of change in the environment configuration. In the first four sessions, the parking lot is almost empty and it becomes quite full in the last ones. This is evident, when analyzing the results of MCL on the static maps, since the performance gets worse with an increasing amount of change. On the other hand, the performance of our approach is less sensitive to the amount of change in case of

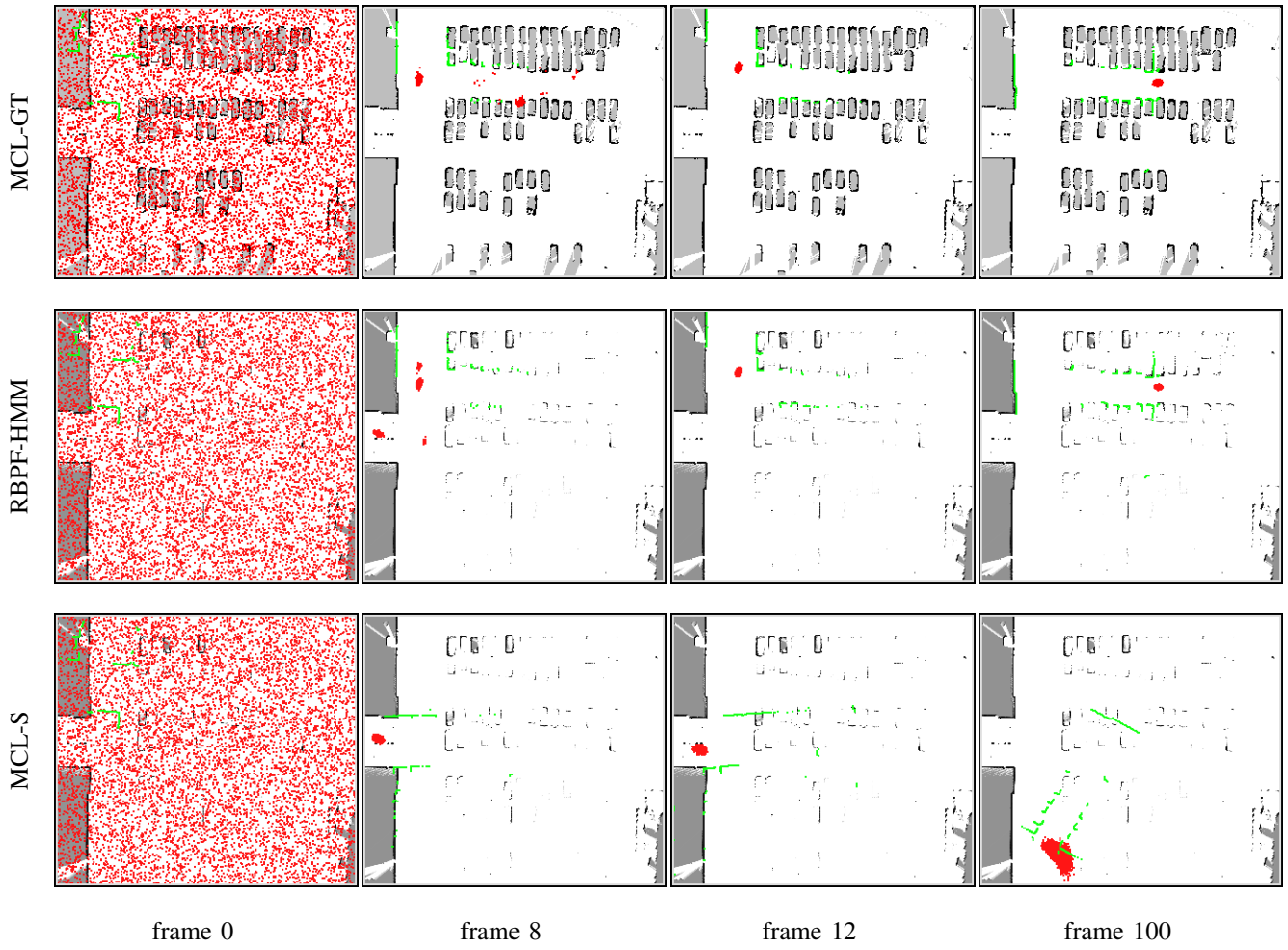


Fig. 10. Comparison between using the ground-truth map (top), the proposed approach (middle) and MCL (bottom) in a global localization setting. The MCL converges too fast and to a wrong position (frame 8), while the proposed approach needs more time to better estimate the current configuration (frame 12). The last frame shows the updated map with the current configuration, notice how it resemble the ground-truth map for the explored portion.

global localization and is even independent of that in case of position tracking, as can be seen from the two tables.

C. Comparison with Outlier Rejection Approaches

In this experiment we show how our approach relates to approaches that rely on a static map of the environment and discard observations of dynamic objects for localization [1, 2, 6, 7]. Four approaches have been compared. The first two approaches, "Static + Rejection" and "Static + Robust", use MCL on a static map of the environment estimated when the parking lot was empty (Figure 11). The difference between them is that Static + Rejection only uses measurements of static objects to localize and Static + Robust uses the Blake-Zisserman robust function instead. The last two approaches correspond to MCL-S and our approach (RBPF-HMM) from the previous experiment.

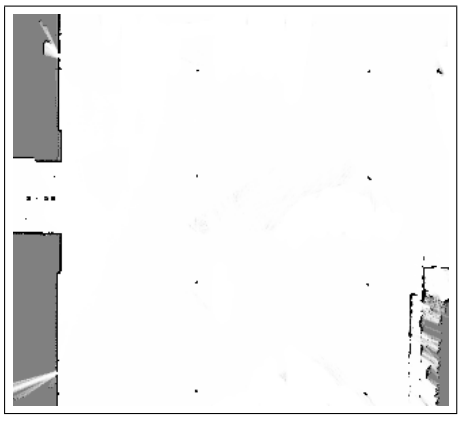


Fig. 11. Static map computed when the paring lot was empty.

We used the same settings of the position tracking experiment, i.e., we performed 100 runs for each data set, where we randomly sampled the initial pose of the robot. All the approaches use the same random seed, have been initialized with 500 particles and an occupancy threshold of 0.6. The particles are initially sampled from a Gaussian distribution centered at the ground-truth position of the robot and with covariance $\Sigma = diag(1, 1, 0.5)$. We let the filter track the position for 100 steps and then compute the mean error with respect to the ground-truth. If this error is below $1\,m$ and

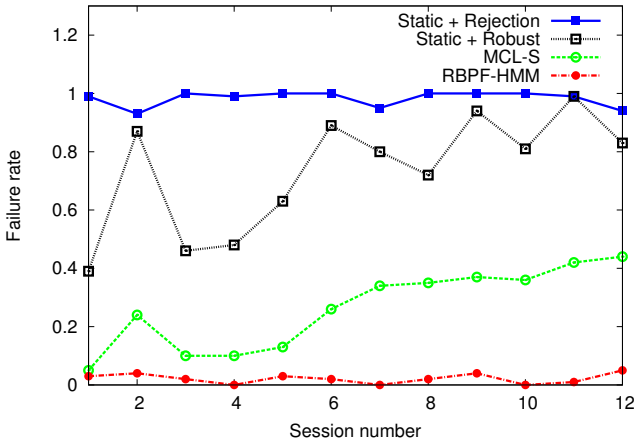


Fig. 12. Failure rate for the outlier rejection experiment. The graph shows the failure rate of the different algorithms. The figure clearly show that outlier rejection mechanisms are more sensitive to changing environments than the robust likelihood function. It also show our approach outperforms techniques based on outlier rejections.

$0.5\,rad$ we consider the run a success, otherwise it is a failure.

Figure 12 depicts the results of the experiment. As in the position tracking experiment, the figure shows the failure rate, i.e., the percentage of time the robot got lost during tracking. From the plot and the static map, we can see two clear messages. The first message is that outlier rejection mechanism are more sensitive to changes in the environment when no static part of the environment is visible. This is clear if one compares the performances of Static + Rejection with Static + Robust.

The second message is that the static portion of the environment does not contain enough information for the robot to localize itself and observations of low dynamic objects improve localization. This is visible in the plot: our approach (RBPF-HMM) has a better performance than MCL-S, which in turn has a better performance than Static + Robust. All the approaches use the same likelihood function and the same algorithm, the only difference is the amount of information stored in the map with respect to objects with low dynamics. For the static map case, no information about the low dynamic objects is present. In the MCL-S case, objects that have been observed most of the time are present in the map, due to the occupancy grid update rule. In our approach, the dynamics of those objects are explicitly modeled in the dynamic occupancy grids. This allows us to infer how often we expect to see a low dynamic object in the environment and for how long.

D. Comparison with Lifelong Mapping using Experiences

The aim of this experiment is to compare the performance of our approach with the experience map of Churchill and Newman [24], a state-of-the-art method for lifelong mapping. In the experience map approach, the environment is represented by a set of *experiences*, where each experience is a sequence of observations connected by visual odometry. During operation, each experience is equipped with a localizer whose task is to

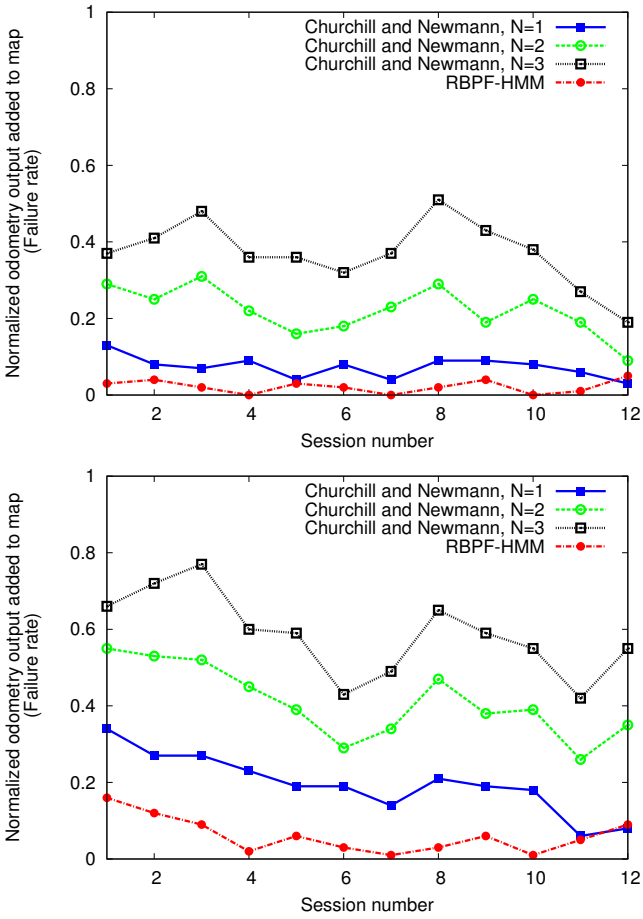


Fig. 13. Results for the lifelong mapping experiment using the generous thresholds (top) and the more restrictive ones (bottom). The plots show the normalized odometry output added to the map for the approach of Churchill and Newman [24], which is equivalent of the failure rate of the position tracking. For the sake of comparison we also plot the performances of our approach. The plot shows that our approach has a better performance even in the case when a single experience is needed for localization.

track the position of the robot in the experience, or declare it "lost" in case of tracking failure. When the system is not able to localize the robot in at least N experiences, the current observation sequence becomes a new experience and is inserted in the map. The approach relies on two main components: the ability to "close a loop", i.e., to globally localize the robot and the ability to navigate locally, i.e., the ability to track the position of the robot. We use the ground-truth position to initialize the localizers of every experience and the MCL algorithm to track the position, since they both work reliably in the case where no changes are present in environment.

For each data set, we consider all other datasets as previous experiences and performed the same 100 runs that were used in the position tracking experiment. For each run, all the localizers for every other experience are initialized with a Gaussian distribution centered at the ground-truth position of the robot and with covariance $\Sigma = diag(1, 1, 0.5)$. We track the position for 100 steps and check how many times the

localizers declared the robot as "lost". For a fair comparison, we used the same thresholds of $1\,m$ and $0.5\,rad$ to check the success of position tracking. To give a better picture about the performances of the two approaches, we also perform an additional experiment using tighter thresholds: $0.2\,m$ and $0.1\,rad$.

Figure 13 shows the normalized odometry output added to the map for different values of N, the minimum number of successful localizers. Note that this number is equivalent to the failure rate of the system to localize the robot, since new experiences are added in case of localization failures. For the sake of comparison, we also included the failure rate of our approach in the same settings. Note that we used on purpose the same runs and the same algorithm for position tracking to have a fair comparison. We also used the same amount of information about the environment, i.e., all the datasets/experiences not used for the testing run. The only difference in the two approaches is the environment representation and the way inference over the environment is performed.

The plot shows that our approach has the best performance, with the experience map relying only on a single localizer having a similar performance. Increasing the number of required localizers, drastically decreases the performance of the system. We believe this is due to the local nature of the changes in the parking lot. Note that the same phenomena have been reported in the original paper, where the car park was in one of the regions with high variation. If we analyze the bottom plot with the tighter thresholds, we see that the gap in performances between our approach and the experience map increases. We believe this is due to the fact that the stored maps do not fully represent the current configuration of the environment, resulting in higher error in localization. Our approach, instead, is able to generalize better to unseen environments and can still achieve high localization accuracy.

The computational complexity of the experience map is much higher than in our approach. We only require one localizer and the local update of the map, where the experience map requires one localizer *for each* experience. Hence, our approach scales with the environment size while their approach scales with the environment size and the number of different configurations.

The experiment also provides an indirect comparison with the approach of Stachniss and Burgard [19], since it is a special case of the experience map, in case a particle filter is used as localizer and only one experience is needed for localization.

VII. Conclusions

In this paper, we presented a probabilistic localization framework for robots operating in dynamic environments. Our approach recursively estimates not only the pose of the robot, but also the state of the environment. It employs a hidden Markov model to represent the dynamics of the environment and a Rao-Blackwellized particle filter to efficiently estimate the joint state. In addition, it exploits the properties of Markov chains to reduce the memory requirements so that the algorithm can be run online on a real robot. Our approach has two advantages. First, it allows for accurate and robust localization even in changing environments and, second, it provides up-to-date maps of them. We evaluated our algorithm extensively using real-world data. The results demonstrate that our model substantially outperforms the popular Monte-Carlo localization algorithm. This makes our method more suitable for long-term operation of mobile robots in changing environments.

In future, we would like to extend our model to reason about objects and not only about individual cells. We will furthermore investigate alternative models to encode the changes (e.g., Dynamic Bayesian Networks and second order hidden Markov models). This will provide a novel perspective on how to reason about correlations in a grid map. In addition, we plan to look further into the detection of moving object and motion segmentation.

We also plan on releasing the software package implementing the approach described in this article as open source and on making the datasets used available at publication time.

Acknowledgment

This work has been partially supported by the European Commission under contract numbers FP7-248258-FirstMM, FP7-260026-TAPAS and ERC-267686-LifeNav.

References

[1] D. Fox, W. Burgard, and S. Thrun, "Markov localization for mobile robots in dynamic environments," *Journal of Artificial Intelligence Research*, vol. 11, 1999.

[2] D. Schulz, D. Fox, and J. Hightower, "People tracking with anonymous and id-sensors using rao-blackwellised particle filters," in *Proc. of the Int. Conf. on Artificial Intelligence (IJCAI)*, 2003. [Online]. Available: http://dl.acm.org/citation.cfm?id=1630659.1630792

[3] D. Hähnel, R. Triebel, W. Burgard, and S. Thrun, "Map building with mobile robots in dynamic environments," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2003.

[4] D. Meyer-Delius, M. Beinhofer, and W. Burgard, "Occupancy grid models for robot mapping in changing environments," in *Proc. of the AAAI Conf. on Artificial Intelligence (AAAI)*, Toronto, Canada, July 2012.

[5] M. Montemerlo, S. Thrun, and W. Whittaker, "Conditional particle filters for simultaneous mobile robot localization and people-tracking," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2002.

[6] D. F. Wolf and G. S. Sukhatme, "Mobile robot simultaneous localization and mapping in dynamic environments," *Autonomous Robots*, vol. 19, no. 1, pp. 53–65, 2005.

[7] C.-C. Wang, C. Thorpe, S. Thrun, M. Hebert, and H. Durrant-Whyte, "Simultaneous localization, mapping and moving object tracking," *International Journal of Robotics Research (IJRR)*, 2007.

[8] L. Montesano, J. Minguez, and L. Montano, "Modeling the static and the dynamic parts of the environment to improve sensor-based navigation," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2005.

[9] G. Gallagher, S. S. Srinivasa, J. A. Bagnell, and D. Ferguson, "GATMO: A generalized approach to tracking movable objects," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2009.

[10] D. Anguelov, R. Biswas, D. Koller, B. Limketkai, S. Sanner, and S. Thrun, "Learning hierarchical object maps of non-stationary environments with mobile robots," in *Proc. of the Conference on Uncertainty in AI (UAI)*, 2002.

[11] C. Chen, C. Tay, C. Laugier, and K. Mekhnacha, "Dynamic environment modeling with gridmap: A multiple-object tracking application," in *Proc. of the IEEE Int. Conf. on Control, Automation, Robotics and Vision (ICARCV)*, 2006.

[12] S. Brechtel, T. Gindele, and R. Dillmann, "Recursive importance sampling for efficient grid-based occupancy filtering in dynamic environments," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2010.

[13] P. Biber and T. Duckett, "Dynamic maps for long-term operation of mobile service robots," in *Proc. of Robotics: Science and Systems (RSS)*, 2005.

[14] S.-W. Yang and C.-C. Wang, "Feasibility grids for localization and mapping in crowded urban scenes," in *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 2011.

[15] J. Saarinen, H. Andreasson, and A. J. Lilienthal, "Independent markov chain occupancy grid maps for representation of dynamic environments," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2012.

[16] K. Murphy, "Bayesian map learning in dynamic environments," in *Proc. of the Conf. on Neural Information Processing Systems (NIPS)*, Denver, CO, USA, 1999, pp. 1015–1021.

[17] D. Avots, E. Lim, R. Thibaux, and S. Thrun, "A probabilistic technique for simultaneous localization and door state estimation with mobile robots in dynamic environments," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2002.

[18] A. Petrovskaya and A. Y. Ng, "Probabilistic mobile manipulation in dynamic environments, with application to opening doors," in *Proc. of the Int. Conf. on Artificial Intelligence (IJCAI)*, 2007.

[19] C. Stachniss and W. Burgard, "Mobile robot mapping and localization in non-static environments," in *Proc. of the Nat. Conf. on Artificial Intelligence (AAAI)*, 2005.

[20] D. Meyer-Delius, J. Hess, G. Grisetti, and W. Burgard, "Temporary maps for robust localization in semi-static environments," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Taipei, Taiwan, 2010.

[21] K. Konolige and J. Bowman, "Towards lifelong visual maps," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2009.

[22] H. Kretzschmar and C. Stachniss, "Information-theoretic compression of pose graphs for laser-based slam," *International Journal of Robotics Research (IJRR)*, vol. 31, 2012.

[23] A. Walcott-Bryant, M. Kaess, H. Johannsson, and J. Leonard, "Dynamic pose graph SLAM: Long-term mapping in low dynamic environments," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, Vilamoura, Portugal, 2012.

[24] W. Churchill and P. Newman, "Practice makes perfect? managing and leveraging visual experiences for lifelong navigation," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2012, pp. 4525 –4532.

[25] G. Grisetti, R. Kümmerle, C. Stachniss, and W. Burgard, "A tutorial on graph-based SLAM," *IEEE Transactions on Intelligent Transportation Systems Magazine*, vol. 2, pp. 31–43, 2010.

[26] J. Röwekämper, C. Sprunk, G. D. Tipaldi, S. Cyrill, P. Pfaff, and W. Burgard, "On the position accuracy of mobile robot localization based on particle filters combined with scan matching," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Villamoura, Protugal, 2012.

[27] H. Moravec and A. Elfes, "High resolution maps from wide angle sonar," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 1985.

[28] L. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition." in *Proceedings of the IEEE*, vol. 77 (2), 1989, pp. 257–286.

[29] G. Grisetti, C. Stachniss, and W. Burgard, "Improved Techniques for Grid Mapping with Rao-Blackwellized Particle Filters," *IEEE Transactions on Robotics*, vol. 23, no. 1, pp. 34–46, 2007.

[30] A. Doucet, J. de Freitas, K. Murphy, and S. Russel, "Rao-Blackwellized partcile filtering for dynamic bayesian networks," in *Proc. of the Conf. on Uncertainty in Artificial Intelligence (UAI)*, Stanford, CA, USA, 2000, pp. 176–183.

[31] S. Thrun, "A probabilistic online mapping algorithm for teams of mobile robots," *International Journal of Robotics Research (IJRR)*, vol. 20, no. 5, pp. 335–363, 2001.

[32] D. A. Levin, Y. Peres, and E. L. Wilmer, *Markov Chains and Mixing Times*. American Mathematical Society, 2008.

[33] A. I. Eliazar and R. Parr, "Dp-slam 2.0," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2004.

[34] D. Fox, "Adapting the sample size in particle filters through kld-sampling," *International Journal of Robotics Research (IJRR)*, vol. 22, 2003.

[35] E. Olson, M. Walter, J. Leonard, and S. Teller, "Single cluster graph partitioning for robotics applications," in *Proceedings of Robotics Science and Systems*, 2005.

[36] G. Grisetti, R. Kümmerle, C. Stachniss, U. Frese, and

C. Hertzberg, "Hierarchical optimization on manifolds for online 2d and 3d mapping," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2010.

[37] E. Olson, "Robust and efficient robotic mapping," Ph.D. dissertation, Massachusetts Institute of Technology, Cambridge, MA, USA, June 2008.

[38] A. Walcott, "Long-term mobile robot mapping in dynamic environments," Ph.D. dissertation, Massachusetts Institute of Technology, Cambridge, MA, USA, May 2011.

[39] A. Blake and A. Zisserman, *Visual reconstruction*. Cambridge, MA, USA: MIT Press, 1987.

APPENDIX A
INDEX TO MULTIMEDIA EXTENSION

TABLE III
LIST OF MULTIMEDIA EXTENSIONS

Extension	Media type	Description
1	Video	Comparison between RBPF-HMM and MCL-S