2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
Daejeon Convention Center
October 9-14, 2016, Daejeon, Korea

# Towards Effective Localization in Dynamic Environments

Dali Sun, Florian Geißer, Bernhard Nebel

*Abstract*— **Localization in dynamic environments is still a challenging problem in robotics – especially if rapid and large changes occur irregularly. Inspired by SLAM algorithms, our Bayesian approach to this so-called dynamic localization problem divides it into a localization problem and a mapping problem, respectively. To tackle the localization problem we use a particle filter, coupled with a distance filter and a scan matching method, which achieves a more robust localization against dynamic obstacles. For the mapping problem we use an extended sensor model which results in an effective and precise map update effect. We compare our approach against other localization methods and evaluate the impact the map update effect has on the localization in dynamic environments.**

## I. INTRODUCTION

Localization is a fundamental problem for autonomous mobile robot systems. For static environments, where the map does not change, one can use existing solutions to the so-called simultaneous localization and mapping (SLAM) problem [7], [10], by building a map which is used for the remainder of the localization. While common approaches to SLAM are successfully applied in static environments, they may not be well-suited for highly dynamic and complex environments, such as intralogistic centers or production halls. The challenge in these environments is that rapid and large changes occur irregularly and change the map for a longer period of time. Examples of such changes are people moving storage boxes or the rearrangement of shelves.

In this work, we introduce an effective and robust dynamic localization method, which separates localization from the mapping process. Thus, the complexity of localization is unaffected, while the map update is only required when changes in the environment occur. Dynamic mapping results in more accurate maps, which leads to a robust localization. We extend the normal sensor model [9], so that it takes dynamic influences into account, as well as the state transition probability of the map. Our work integrates several previously introduced concepts and state of the art algorithms, to achieve an accurate localization in dynamic real world environments.

The remainder of this paper is organized as follows: in section II we present related work. Section III formally defines the problem of dynamic localization, and gives necessary background for localization with a particle filter and scan matching. In section IV we describe how we can use a distance filter [4] to detect dynamic changes in the environment. Section V describes the dynamic mapping process and the extended sensor model in detail. Finally, we evaluate our method for multiple real world environments in section VI.

D. Sun, F. Geißer and B. Nebel are with the Department of Computer Science, University of Freiburg, Germany

## II. RELATED WORK

A common approach to deal with dynamic obstacles that only occur for short periods of time is to treat them as outliers. Fox et al. [4] apply, among other things, a *distance filter* to remove readings of a scan which are shorter than the expected value and are therefore caused by an unmodeled object. During localization we also filter outliers, but additionally use the distance filter to detect changes in the environment and update the map accordingly.

When changes occur irregularly and change the map for a longer period of time a distance filter is not sufficient. Tipaldi et al. [17] show that even if their SLAM algorithm is applied permanently and loop closure techniques join the different trajectories together, the generated map gets highly inconsistent and the trajectory path is wrong. Instead, they treat the lifelong localization problem as a multi-session localization and mapping problem. Their map is based on a *dynamic occupancy grid* [9], which consists of a collection of individual cells, modeled using a hidden Markov model. The grid requires a state transition probability, corresponding to the (static) probability that a cell changes its state. Tipaldi et al. separate the estimation of the robot trajectory from the map estimation and use a particle filter to compute the robot trajectory, where each particle is associated to its own map. They reduce the size of these local maps by exploiting properties of the hidden Markov model, but since the number of particles required for global localization lies in the thousands, memory management still plays a key role.

Another way to achieve global localization is to use scan matching instead of a particle filter. Recently, Olson [12] presented a multi-resolution scan matching method, which generates the same result as brute-force full resolution methods, but runs an order of magnitude faster than previous approaches. This technique is well-suited if the initial map of the environment is known.

Our approach also separates mapping from localization. We use a particle filter localization method and apply scan matching afterwards, to improve the accuracy of the estimated pose. However, we don't associate a map to each particle. Instead, we initiate a dynamic mapping process, to update the map if uncertainty of the estimated robot pose is low. This prevents map updates which are based on wrong estimates. The updated map will then be used for upcoming localization steps.

Other work on localization in dynamic environments represents dynamic objects as explicit object models in the map. Anguelov et al. [1] extract dynamic object snapshots in office-type environments from static occupancy grid maps acquired at different times. They use an EM-algorithm [8]

978-1-5090-3762-9/16/$31.00 ©2016 IEEE

to learn two-level hierarchical shape models, which link these object snapshots to generic shape templates of several object classes. Similarly, Biswas et al. [2] also apply static occupancy grid maps to extract object snapshots and use an EM-algorithm to learn object models represented as local occupancy grid maps. Both of these methods have shown significant accuracy at modeling dynamic objects in environments. However, they suffer from disadvantages of the off-line algorithms and the limited complexity of objects.

Wang et al. [19] present two approaches to solve localization, mapping and moving object tracking (SLAMMOT) simultaneously. In their first approach, all objects (including robot pose) are modeled with a hybrid state model consisting of a state and a motion mode. They compute a joint posterior over all objects, similar to other SLAM methods [10]. But due to the required motion modeling of all objects, this approach is computationally more complex and thus generally not suited for real time applications. In their second approach the estimation problem is decomposed into two separate estimators, for stationary objects and moving objects, respectively. They demonstrate that this satisfies navigation and safety requirements in high speed autonomous driving applications in urban areas.

Gallagher et al. [5] present an online approach to deal with dynamic changes. Their map consists of two parts, a static occupancy grid map and several lists of dynamic objects. Dynamic objects will be detected and classified via laser observations and recognized objects are saved in lists with positions and orientations. These lists will be updated whenever the map is changed. They have shown that their resulting map improves both localization and navigation.

In contrast to these explicit modeling approaches we integrate dynamic object detection directly into the occupancy grid mapping, which leads to independence from the shape and type of dynamic objects.

## III. DYNAMIC LOCALIZATION

The problem of dynamic localization is to calculate the posterior distribution of map configurations $m_{1:t}$ and robot poses $x_{1:t}$ up to time step $t$, given observations $z_{1:t}$, odometries $u_{1:t}$, initial map configuration $m_0$ and initial robot pose $x_0$, i.e. to estimate $p(m_{1:t}, x_{1:t} \mid z_{1:t}, u_{1:t}, m_0, x_0)$. The main difference to the SLAM problem is that $m_0$ and $x_0$ are prior distributions known in advance and that the map configuration changes over time, since the environment is dynamic. Similar to SLAM, we can separate the estimation of the robot pose from the map configuration, by using Bayes' rule and the Markov assumption:

$$\begin{aligned} &p(m_{1:t}, x_{1:t} \mid z_{1:t}, u_{1:t}, m_0, x_0) = \\ &p(m_{1:t} \mid x_{1:t}, z_{1:t}, m_0, x_0) p(x_{1:t} \mid z_{1:t}, u_{1:t}, m_0, x_0). \end{aligned} \tag{1}$$

Figure 1 describes the Bayesian network that models the factorization.

Afterwards, we have two separated problems, where $p(m_{1:t} \mid x_{1:t}, z_{1:t}, m_0, x_0)$ corresponds to the problem of mapping with known poses and $p(x_{1:t} \mid z_{1:t}, u_{1:t}, m_0, x_0)$ corresponds to the localization problem. We will discuss the mapping problem in section V and first concentrate on localization.

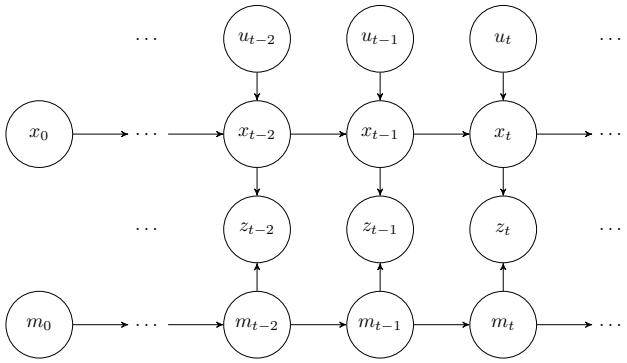



Fig. 1: Bayesian network of the dynamic localization problem.

We use a particle filter approach to solve the basic localization problem, and apply a scan matching method to improve accuracy of the estimated position. For scan matching, we apply the multi-layer searching technique by Olson [12] to find the local maximum of the position within a limited region. Since this method is extremely fast for small regions, we can substantially increase the accuracy of the localization with only requiring few computational resources. Additionally, we also get a matching score, which describes the degree of the matching. We can use this score as a parameter to prevent updates of the map, based on wrong estimates. Only if the score exceeds a specific threshold, the map update step will be evoked. For our experiments, this threshold was determined empirically; we will discuss other ways to determine it in section VII.

To account for dynamic obstacles, a distance filter marks observations that are shorter or longer than the expected observation as outliers. Both types of outliers will be used to update the map, but only shorter outliers will be rejected during localization.

## IV. DISTANCE FILTER

Originally, the distance filter by Fox et al. was used to filter those readings of the laser scan, that are shorter than the distance expected from the map. Let $d_1, \ldots, d_n$ be a set of possible distances and $p_m(d_i|x_l)$ the probability of measuring distance $d_i$ if the robot is at position $x_l$. The probability that distance $d_i$ is shorter than expected is defined by

$$p_{short}(d_i|x_l) = \sum_{d_j > d_i} p_m(d_j|x_l),$$

based on the fact that this is equivalent to the probability that the expected measurement is longer than $d_i$. Since the position $x_l$ is based on the current belief, we have to average over all possible positions of the robot:

$$p_{short}(d_i) = \sum_l p_{short}(d_i|x_l) \cdot p(x_l),$$

where $p(x_l)$ is given by our particle filter. Given $p_{short}(d_i)$ we can now mark those measurements $d_i$, where $p_{short}(d_i)$

exceeds some threshold $\gamma$. Similarly, we mark those measurements that are longer than expected.

We should mention here that Fox et al. precompute the distance to the closest obstacle in the map for each possible robot location. However, since our environment is not static we have to compute this distance on the fly. While this is the main bottleneck of the distance filter, this approach still works well in practice.

## V. DYNAMIC MAPPING

Recall that the second problem to tackle was to compute the full posterior distribution of $p(m_{1:t} \mid x_{1:t}, z_{1:t}, m_0, x_0)$. This computation is not only intractable, but is also unnecessary in practice, because we are only interested in the last configuration of the map. Therefore, we use a Bayesian filter to recursively compute the marginal distribution $p(m_t \mid x_{0:t}, z_{1:t})$ for the current map $m_t$. A common approach is to decompose the problem into several one-dimensional estimation problems (cf. [11], [15]), with the underlying assumption that grid cells are conditionally independent from one another. Therefore, we approximate $p(m_t \mid x_{0:t}, z_{1:t})$ with $\prod_{i=0}^{N} P(c_t^i | z_{1:t}^i)$, where $z^i \in \{hit, miss\}$, $c \in \{occ, free\}$ and $i = 1, \ldots, N$, $N$ being the total number of cells. The meaning of the observation and cell values will become clear in a moment, when we introduce the sensor model.

To represent dynamic changes we use dynamic occupancy grid maps, introduced by Meyer-Delius et al. [9], which apply the theory of hidden Markov models (HMM) [14]. In the original work, the state transition model $p(c_t^i \mid c_{t-1}^i)$ is assumed to be stationary (i.e., the state transition probabilities do not change over time) and can be learned offline or online. We make the same assumption and learn it offline. We also adopt the sensor model introduced along with dynamic occupancy grids, which considers two cases: a measurement ends up in a cell ($hit$) or goes through it ($miss$). However, this sensor model does not consider dynamics in the environment. The probability for a measurement $miss$, given that a cell is occupied (i.e. cell state $occ$), is usually very low, due to the good accuracy of today's laser range finders. But this probability should increase when we know that some objects may disappear in the environment. To achieve this, we incorporate information about the transition dynamics of each cell: $st^i$ is a boolean, time-independent flag that specifies whether cell $c^i$ is a static cell, or whether objects may appear or disappear. For example, in a production hall we might flag walls and machinery equipment as static, while hallways, movable shelves and stock in a storeroom are flagged as being dynamic. We may learn such information offline, but often we can also manually flag the map by studying a layout plan of the facility. The second information we incorporate are the results of the distance filter. We not only consider whether a measurement hits a cell $c^i$, but also if this measurement is considered being an outlier: $\hat{z}_t^i \in \{hit, miss, hit_o, miss_o\}$. Figure 2 represents the underlying HMM of our model.

As a result, we have to compute $\prod_{i=0}^{N} P(c_t^i | \hat{z}_{1:t}^i, st^i)$. By applying the definition of conditional probability and the forward algorithm we get

$$P(c_t^i | \hat{z}_{1:t}^i, st^i) = \mu \sum_{c_{t-1}} \cdot P(\hat{z}_t^i, c_t^i, c_{t-1}^i, \hat{z}_{1:t-1}^i, st^i),$$

where $\mu$ is a normalization constant. Applying the chain rule results in

$$\begin{aligned} P(c_t^i | \hat{z}_{1:t}^i, st^i) = \mu \sum_{c_{t-1}} & P(\hat{z}_t^i | c_t^i, c_{t-1}^i, \hat{z}_{1:t-1}^i, st^i) \\ & \cdot P(c_t^i | c_{t-1}^i, \hat{z}_{1:t-1}^i, st^i) \\ & \cdot P(c_{t-1}^i | \hat{z}_{1:t-1}^i, st^i) \cdot P(\hat{z}_{1:t-1}^i, st^i). \end{aligned}$$

Due to conditional independence we get

$$\begin{aligned} P(\hat{z}_t^i | c_t^i, c_{t-1}^i, \hat{z}_{1:t-1}^i, st^i) &= P(\hat{z}_t^i | c_t^i, c_{t-1}^i, st^i), \\ P(c_t^i | c_{t-1}^i, \hat{z}_{1:t-1}^i, st^i) &= P(c_t^i | c_{t-1}^i). \end{aligned}$$

Furthermore, $P(\hat{z}_{1:t-1}^i, st^i)$ is independent of the state of the cell so we can substitute it by another normalization constant $\eta$, which leads to

$$\begin{aligned} & P(c_t^i | \hat{z}_{1:t}^i, st^i) = \\ & \mu\eta \sum_{c_{t-1}} P(\hat{z}_t^i | c_t^i, c_{t-1}^i, st^i) \cdot P(c_t^i | c_{t-1}^i) \cdot P(c_{t-1}^i | \hat{z}_{1:t-1}^i, st^i). \end{aligned}$$

This equation can be computed recursively: $P(\hat{z}_t^i | c_t^i, c_{t-1}^i, st^i)$ corresponds to the sensor model, $P(c_t^i | c_{t-1}^i)$ corresponds to the state transition model and $P(c_{t-1}^i | \hat{z}_{1:t-1}^i, st^i)$ is the recursive call, where the base case is given by the initial map $m_0$. Both the sensor model and the state transition model can be learned offline beforehand, as it is done in our experiments (cf. Table I). Note that parameters of the extended sensor model influence the rate at which a cell may switch its state. If the learned parameters are therefore unreasonable, they may lead to inconsistency of the map.

Another advantage of the extended sensor model is that dynamics are not modeled as explicit object models (cf. [1], [2], [7], [19]). Our model is independent of the shape and type of the dynamic objects. This allows to apply the sensor model learned for one environment to another environment, as we will see in our experimental results.

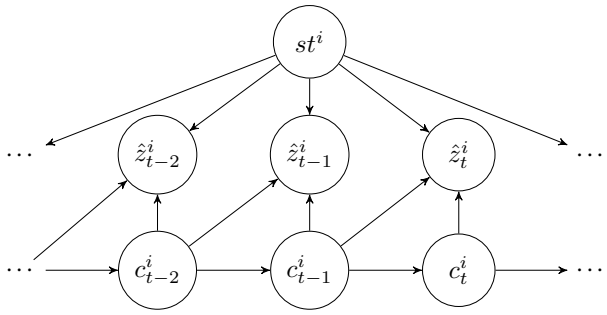



Fig. 2: Hidden Markov Model of the extended sensor model.

## VI. EXPERIMENTAL RESULTS

To evaluate our approach, we conducted a series of experiments in two real production halls (PH1: 104m x 52m, PH2: 47m x 26m) and an intralogistic center (LC: 137m x 99m) from three different companies. Changes in the environments may be caused, for example, by forklifts or people moving storage boxes. We also face more long-term structural changes, like the rearrangement of shelves. In each environment, we collected several data sets with a mobile robot equipped with a SICK S300 laser range finder. Each data set consists of raw odometry and laser data. Since all environments are dynamic, we get different environment settings for each experiment. We examine two data sets with the most changes between the environment and use these to test localization and mapping.

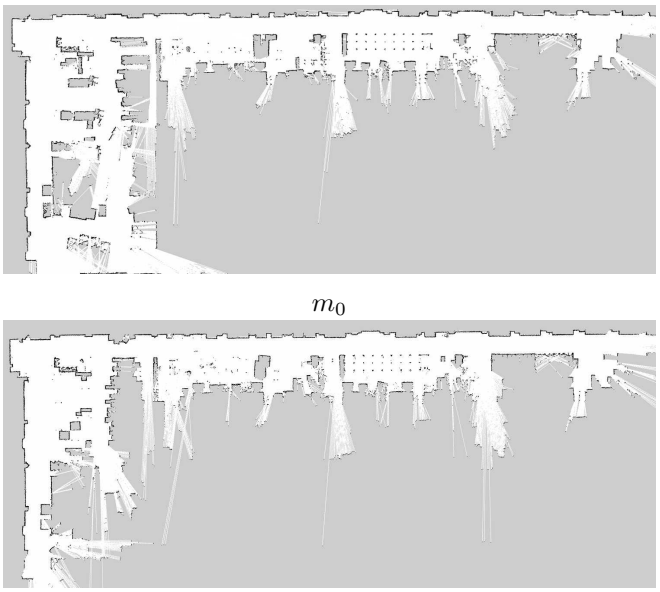
We apply a standard SLAM approach [6], to generate the initial map $m_0$, based on the first data set, and the ground truth (i.e. $x_{1:t}$ and $m_t$), based on the second. Note that this approach is usually not well-suited for dynamic environments (cf. [17]). However, during *one* recording of a data set, only few dynamic changes occur. Therefore the SLAM approach is a reasonable source for ground truth. The state transition model for each map is based on the structure of the environments. All three scenarios can be seen in Figure 3, 4 and 5. We omit the odometry data for clarity; it should be noted that it was generally very unreliable.

To learn the parameter set $\Theta$ of the extended sensor model we use four separately recorded data sets. As we know the state transition model $st$, the initial map $m_0$ and the ground truth $m_t$ and $x_{1:t}$, we can generate the training set $D = \{d_i \mid d_i = (c^i_{t-1}, c^i_t, st^i, z^i_t), \forall i \in [1, S]\}$ from each data set, $S$ being the size of the data set. The goal is to compute $\Theta$, such that the probability $P(D|\Theta)$ is maximized. This problem can be solved by applying a maximum likelihood estimation (MLE) algorithm, given that there is enough data available. Despite that our data sets consist of millions of data points, we observed that some combinations of measurements, cell states and state transition dynamics occur only rarely. Therefore, we assume a prior probability $P(\Theta)$ and utilize a posteriori probability (MAP) estimation [3] to compute $\Theta$. The resulting estimates are denoted in Table I.

In the following, we report localization and mapping results, based on 10 runs for each scenario. Global scan matching [12] was used to provide the initial position of the robot.

### A. Localization results

To verify the quality of localization we consider maximal translation error and maximal angle error. We consider a run as success, if the translation error is kept below 1.0 meter and the angle error is kept below 25 degree. Otherwise we consider it as failed. First we compared our approach ($DL$) with two static localization methods that do not update the map at run time, $AMCL$ and $AMCL_{DF}$. $AMCL$ is a standard localization method in the ROS package [13] which implements the popular Monte Carlo localization method described by Fox et al. [16]. $AMCL_{DF}$ is the extension



$m_0$

$m_t$

Fig. 3: Ground truth maps of collected data sets of PH1.

$m_0$

$m_t$

Fig. 4: Ground truth maps of collected data sets of PH2.

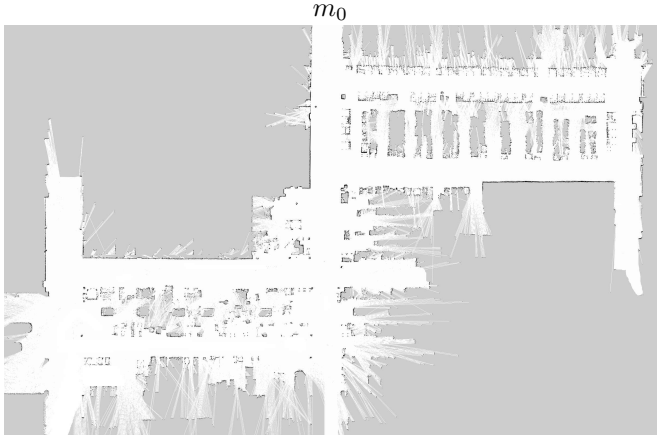
of $AMCL$ with a distance filter [4]. All methods use the same set of parameters for the motion model and the standard sensor model. For a fair comparison we increase the number of particles for $AMCL$ and $AMCL_{DF}$, such that computational time required by every method remains at the same level in each iteration step (cf. Table II). The threshold to allow map updates in our approach is set to 60% for all experiments.

All results are shown in table III. As expected, localization with $DL$ succeeds in all scenarios and all runs, whereas both static methods fail in most cases. Of course, $AMCL$ has no way to incorporate dynamic influences. We can see that

| $st$ | $c_{t-1}$ | $c_t$ | $P(hit\|c_{t-1}, c_t, st)$ | $P(hit_o\|c_{t-1}, c_t, st)$ | $P(miss\|c_{t-1}, c_t, st)$ | $P(miss_o\|c_{t-1}, c_t, st)$ |
|---|---|---|---|---|---|---|
| 1 | $occ$ | $occ$ | 86.365 | 1.289 | 7.126 | 5.220 |
| | | $free$ | 7.955 | 8.721 | 71.494 | 11.830 |
| | $free$ | $occ$ | 14.223 | 10.720 | 72.709 | 2.349 |
| | | $free$ | 0.223 | 1.114 | 95.130 | 3.533 |
| 0 | $occ$ | $occ$ | 69.103 | 2.084 | 24.707 | 4.106 |
| | | $free$ | 34.575 | 0.550 | 7.251 | 57.626 |
| | $free$ | $occ$ | 7.190 | 43.211 | 48.979 | 0.619 |
| | | $free$ | 15.208 | 14.684 | 57.694 | 12.413 |

TABLE I: Learned parameters of the extended sensor model, denoted in %.

$m_0$



$m_t$

Fig. 5: Ground truth maps of collected data sets of LC.

| Method | #Particles | Distance Filter | Scan Matching | Map Update |
|---|---|---|---|---|
| $AMCL$ | 1000 | no | no | no |
| $AMCL_{DF}$ | 1000 | yes | no | no |
| $DL_{simple}$ | 1000 | no | no | yes |
| $DL$ | 500 | yes | yes | yes |

TABLE II: Description of static and dynamic methods

| Scenarios | Method | Error (m) | | Error (degree) | | Success |
|---|---|---|---|---|---|---|
| | | Mean | Var | Mean | Var | |
| PH1 | $AMCL$ | 1.791 | 3.868 | 10.212 | 21.245 | 0/10 |
| | $AMCL_{DF}$ | 0.138 | 0.051 | 0.638 | 0.523 | 10/10 |
| | $DL_{simple}$ | 0.415 | 0.756 | 1.184 | 1.570 | 7/10 |
| | $DL$ | 0.126 | 0.059 | 0.687 | 0.629 | 10/10 |
| PH2 | $AMCL$ | 0.750 | 0.957 | 6.337 | 9.286 | 1/10 |
| | $AMCL_{DF}$ | 0.691 | 0.857 | 5.018 | 6.288 | 1/10 |
| | $DL_{simple}$ | 0.160 | 0.080 | 1.381 | 1.307 | 10/10 |
| | $DL$ | 0.165 | 0.054 | 0.615 | 0.528 | 10/10 |
| LC | $AMCL$ | 10.107 | 23.054 | 22.442 | 48.222 | 0/10 |
| | $AMCL_{DF}$ | 6.891 | 18.160 | 10.966 | 29.145 | 0/10 |
| | $DL_{simple}$ | 3.144 | 4.282 | 3.356 | 7.112 | 0/10 |
| | $DL$ | 0.110 | 0.096 | 0.602 | 0.883 | 10/10 |

TABLE III: Comparison results of static and dynamic methods

the distance filter helps, as $AMCL_{DF}$ can deal with the dynamics in scenario PH1. However, as the scenarios get more complex, the success rate also decreases. The reason is the well-known particle deprivation problem [18], which leads to divergence of the particles in the second and third scenario. To the contrary, our approach responds very quickly to changes and updates the map in time, so that the robot stays localized with the updated map.

We also evaluated an implementation of our approach which updates the map at run time, but has distance filter and scan matching disabled ($DL_{simple}$). We double the number of particles to achieve a fair comparison regarding computational time (cf. Table II). Note that without post-processing of scan matching, the map will be updated in every iteration step. This can lead to a divergence of the map. To overcome this problem, each particle should update and maintain its own map [17]. However, this will increase the load and usage of CPU and memory. As we can see from the results in Table III, this is not necessary if we activate distance filter and scan matching.

### *B. Mapping results*

To evaluate the mapping results we saved the updated map $m_t$ at the end of each run. Accurate mapping should lead to accurate trajectories, especially in highly dynamic parts of the environment. Figures 6, 7 and 8 show the resulting maps with corresponding trajectories over all 10 runs for all approaches. As can be seen, the updated maps and trajectories have both converged to the ground truth in our approach, whereas other methods diverged in most

runs (recall that $AMCL$ does not update the map). The results of scenario PH2 also show that the resulting maps and trajectories of $DL$ are more consistent and robust than the results of $DL_{simple}$, despite that it also succeeds. This is also the case for the other scenarios. Altogether, these results suggest that our approach is very robust and consistent for different dynamic environments. Note that if we would use separated map updates for each particle, as described by Tipaldi et al. [17], we could expect similar robust and consistent results for $DL_{simple}$. However, as this requires more resources, it may be more suitable to apply $DL$.

We attached a video to illustrate our approach. Laser readings are colored according to the results of the distance filter. Red readings refer to outliers that are too short, blue readings to outliers that are too long and green readings refer to regular readings.

## VII. CONCLUSION

In this work we presented a dynamic localization approach, which incorporates several previously introduced concepts. As others, we separate the problem and treat localization and mapping individually. By incorporating recent advancements on scan matching we can give precise localization updates without consuming too many computational resources. Furthermore, we use a distance filter to incorporate dynamics of the environment into the sensor model, which leads to quicker convergence of the map update.

The empirical evaluation shows the advantages of our approach. We learn the parameters of the sensor model and achieve a robust localization in several different real-world environments, all including dynamic influences. At the same time the mapping stays consistent with the ground truth.

An important parameter for our approach is the threshold for map updates, which we determined empirically. Setting that threshold too high prevents an accurate representation of the real map and influences localization; setting it too low can lead to distorted maps, due to localization inaccuracies. It seems reasonable to aim for a threshold that results in a good average matching score. Therefore we could also learn the threshold parameter from data, with the average matching score as an evaluation function. Furthermore, all of our scenarios consisted of environments with a clearly defined static structure, like machinery and walls. While we exploit this information, there may be production halls which are completely dynamic, i.e. a very basic wall structure, no hallways and dynamic machinery equipment. The amount of static objects this approach requires remains an open question.

For further future work it would be interesting to perform a direct comparison with the approach of Tipaldi et al. in several more environments. However, currently their algorithm is not openly available. An interesting consideration is to merge their approach with ours. When the matching score drops below a certain threshold, we could associate each particle to its own map, until the matching score improves again. Another topic is dynamic localization for multi-robot scenarios. By sharing information about the environment between multiple robots we may be able to further improve stability of localization and consistency of the mapping. We also plan to publish our data sets so that they can be used by other researchers concerned with long-term autonomy.

## ACKNOWLEDGMENT

This work has been funded by BMBF grant 02PJ2667 as part of the KARIS PRO project.

## REFERENCES

[1] Dragomir Anguelov, Rahul Biswas, Daphne Koller, Benson Limketkai, Scott Sanner, and Sebastian Thrun. Learning hierarchical object maps of non-stationary environments with mobile robots. In *In Proc. of the Conf. on Uncertainty in Artificial Intelligence (UAI)*, 2002.

[2] R. Biswas, B. Limketkai, S. Sanner, and S. Thrun. Towards object mapping in non-stationary environments with mobile robots. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, volume 1, pages 1014–1019 vol.1, 2002.

[3] Morris H DeGroot. *Optimal statistical decisions*, volume 82. John Wiley & Sons, 2005.

[4] Dieter Fox, Wolfram Burgard, and Sebastian Thrun. Markov localization for mobile robots in dynamic environments. *Journal of Artificial Intelligence Research*, 11:391–427, 1999.

[5] G. Gallagher, S.S. Srinivasa, J.A. Bagnell, and Dave Ferguson. Gatmo: A generalized approach to tracking movable objects. In *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pages 2043–2048, May 2009.

[6] G. Grisetti, C. Stachniss, and W. Burgard. GMapping – OpenSLAM.org. http://www.openslam.org/gmapping.html.

[7] Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard. Improved techniques for grid mapping with Rao-Blackwellized particle filters. *IEEE Transactions on Robotics*, 23(1):34–46, 2007.

[8] Geoffrey J. McLachlan and Thriyambakam Krishnan. *The EM algorithm and extensions*. Wiley series in probability and statistics. Wiley, Hoboken, NJ, 2. ed edition, 2008.

[9] 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.

[10] Michael Montemerlo, Sebastian Thrun, Daphne Koller, and Ben Wegbreit. FastSlam: A factored solution to the simultaneous localization and mapping problem. In *In Proceedings of the AAAI National Conference on Artificial Intelligence*, pages 593–598. AAAI, 2002.

[11] H.P. Moravec and A. Elfes. High resolution maps from wide angle sonar. In *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*, volume 2, pages 116–121, Mar 1985.

[12] Edwin Olson. M3RSM: Many-to-many multi-resolution scan matching. In *IEEE International Conference on Robotics and Automation, ICRA 2015, Seattle, WA, USA, 26-30 May, 2015*, pages 5815–5821, 2015.

[13] Morgan Quigley, Ken Conley, Brian P. Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y. Ng. ROS: an open-source robot operating system. In *ICRA Workshop on Open Source Software*, 2009.

[14] Lawrence R Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

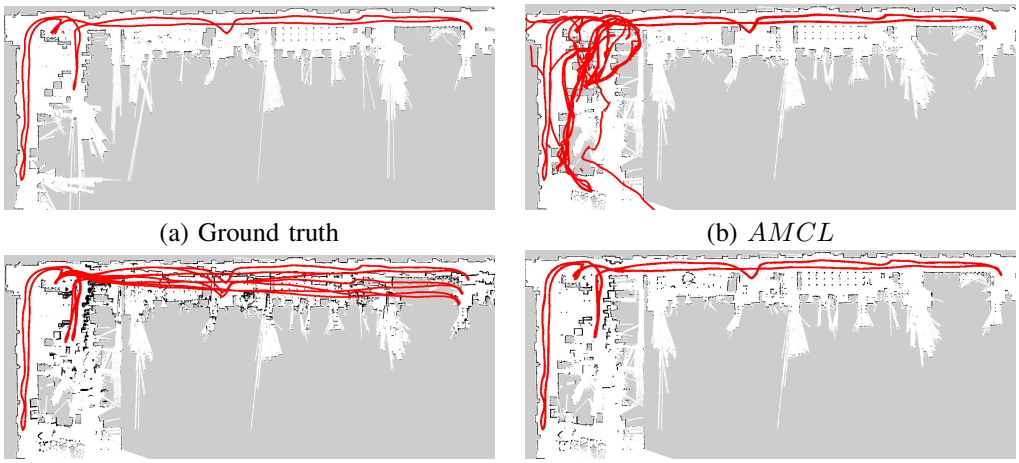
[15] S. Thrun. Learning occupancy grids with forward models. In *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, volume 3, pages 1676–1681 vol.3, 2001.

[16] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.

[17] Gian Diego Tipaldi, Daniel Meyer-Delius, and Wolfram Burgard. Lifelong localization in changing environments. *The International Journal of Robotics Research*, 2013.

[18] Rudolph van der Merwe, Arnaud Doucet, Nando de Freitas, and Eric A. Wan. The unscented particle filter. In *Advances in Neural Information Processing Systems 13, Papers from Neural Information Processing Systems (NIPS) 2000, Denver, CO, USA*, pages 584–590, 2000.

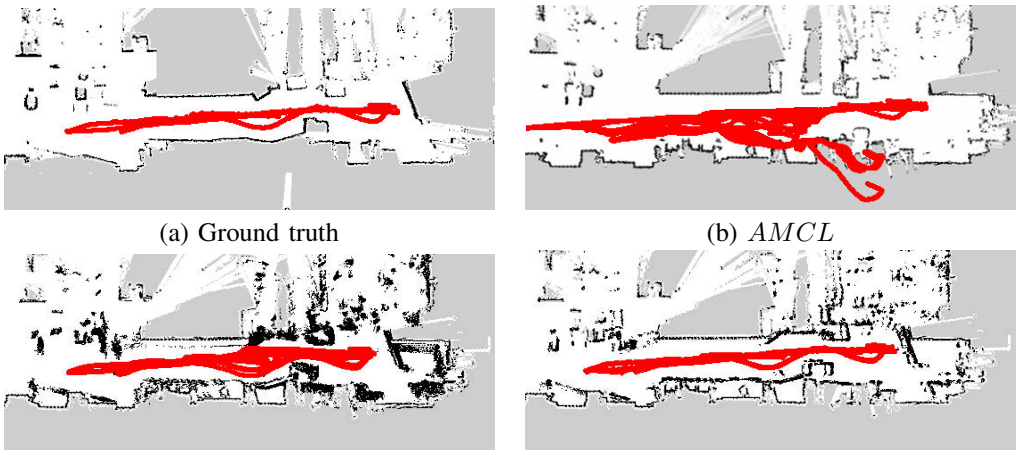
[19] Chieh-Chih Wang, Charles Thorpe, Sebastian Thrun, Martial Hebert, and Hugh Durrant-Whyte. Simultaneous localization, mapping and moving object tracking. *Int. J. Rob. Res.*, 26(9):889–916, September 2007.



(a) Ground truth (b) $AMCL$

(c) $DL_{simple}$ (d) $DL$

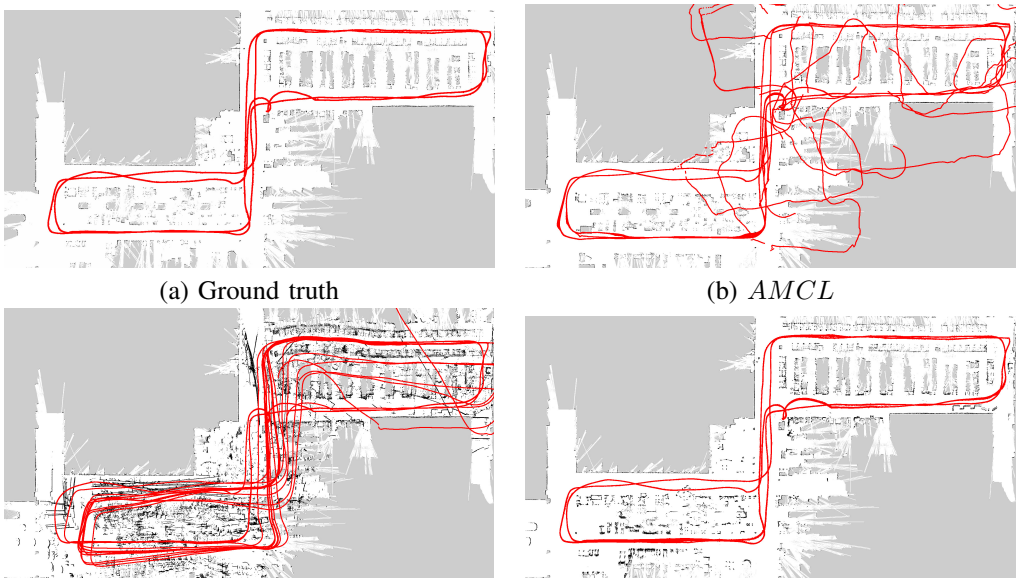
Fig. 6: Mapping and localization results for PH1



(a) Ground truth (b) $AMCL$

(c) $DL_{simple}$ (d) $DL$

Fig. 7: Mapping and localization results for PH2



(a) Ground truth (b) $AMCL$

(c) $DL_{simple}$ (d) $DL$

Fig. 8: Mapping and localization results for LC