Active Localization with Dynamic Obstacles

Alberto Quattrini Li[1], Marios Xanthidis[1], Jason M. O'Kane[1], and Ioannis Rekleitis[1]

***Abstract*— This paper addresses the problem of robot global localization in a known environment, in the presence of many dynamic obstacles. Deploying a robot in crowded spaces such as museums, shopping malls, department stores, or university campuses is especially challenging because the moving people occlude the static parts of the environment, such as walls and doorways, making the robot essentially blind. A new weighting function is proposed for a particle filter state estimation algorithm that accounts for the presence of dynamic obstacles and avoids population depletion. An active localization strategy is employed which guides the robot to locations that resolve ambiguities and eliminate hypotheses in a systematic manner. Experimental results from multiple simulations and from real robot deployments validate the localization improvements achieved by the proposed method.**

I. INTRODUCTION

As autonomous robots integrate more deeply into everyday human activities, it will be increasingly important for them to operate reliably in public spaces. Crowded spaces such as museums and shopping malls [1], [2], [3] have proven to be particularly challenging for robot localization, because co-located humans often occlude the robot's sensors' view of the environment, leading to loss of positional accuracy. In some cases, this has constrained the deployment of robots to times outside the venue's normal operational hours [4].

In this paper, we attack the problem of mobile robot global localization in arbitrary crowded environments on two fronts. First, we consider the *passive* problem of inferring some partial information about the robot's pose, based on range sensor data, in environments with a high density of dynamic obstacles. In this scenario, the particular challenge is that each range measurement may arise from either the known static obstacles or from the unknown dynamic obstacles, and there is no obvious way to distinguish between these two cases. Second, we address the question of *active* localization, in which the robot can choose motions intended specifically to reduce its uncertainty about its own pose.

This paper makes three specific contributions.

- We introduce a new particle filter designed to account for the presence of large numbers of unpredictably-moving dynamic obstacles. The key idea is to 'punish' particles that are closer to static obstacles than the sensor range indicates, by assigning them smaller weights. Simply put, the range sensor cannot see through a wall. At the same time, this approach does not interfere with particles that see something, probably a dynamic obstacle, closer than expected. This asymmetric weight function allows the particle filter to make partial estimates of the robot's state, without being confused by the presence of dynamic obstacles.
- We describe a planning algorithm for selecting motions that decrease the pose uncertainty encoded by this particle filter. The underlying idea is to construct a *compound map*, expressed in the robot's body frame, and to drive the robot toward locations that allow the robot to disambiguate between two or more competing hypotheses about its current position.
- We describe an implementation of these techniques, along with a series of experiments—both in simulation and on a Turtlebot 2, with a fleet of other robots acting as dynamic obstacles—that demonstrate their effectiveness compared to baseline approaches.

Fig. 1. The robot, a TurtleBot 2 equipped with a laser range finder surrounded by several dynamic obstacles (other robots).

The remainder of this paper is structured as follows: the next section provides an overview of related work. Section III describes the problem in detail. Section IV discusses the particle filter formulation employed and provides a detailed description of the active localization strategy proposed. The following section presents experimental results from numerous simulations and from experiments with real robots. The paper concludes with lessons learned and a discussion of future directions.

II. RELATED WORK

From the memorable videos of children climbing on robotic museum tour guides [5], the problem of localizing in the presence of dynamic obstacles has attracted significant attention. Early work proposed to navigate close to the static boundaries, termed coastal navigation [6], in order to improve the localization quality. This approach enabled accurate localization even in the presence of many dynamic obstacles. Alternatively, dynamic obstacles were treated as outliers [7] assuming a static map of the environment.

[1]Alberto Quattrini Li, Marios Xanthidis, Jason M. O'Kane, and Ioannis Rekleitis are with the Computer Science & Engineering Department, University of South Carolina, 315 Main St. Columbia, SC, 29208, USA, [albertoq,jokane,yiannisr]@cse.sc.edu, mariosx@email.sc.edu

Stachniss and Burgard [8] also treated dynamic obstacles as outliers and maintained a list of alternative hypotheses.

Monte Carlo localization [9], generally in the form of a particle filter, has been used extensively for passive localization. Milstein, Sánchez, and Williamson [10] proposed a variant that improves robustness by clustering the particle population around the major hypotheses. More recently, Li, Sun, and Duan [11] proposed grouping the particles along their spatial concentration. In the work of Liu, Shi, and Zhao [12], the number of particles was dynamically modified for each cluster in order to improve computational efficiency. The accuracy of particle filter localization, in conjunction with scan matching, was shown by Röwekämper *et al.* [13].

Mapping dynamic environments requires the robot to differentiate between static and dynamic features. Wolf and Sukhatme [14] maintained two occupancy grid maps, requiring the ability to differentiate between dynamic and static obstacles. In an industrial setting, Valencia *et al.* [15] used multiple maps at different time scales in order to distinguish between dynamic and static obstacles.

The problem of global localization, even under the ideal conditions of a noiseless, infinite range, visibility sensor, has been proven to be NP-complete. Dudek, Romanik, and Whitesides [16] proposed an approximation algorithm, based on the visibility cell decomposition, for selecting specific destination points that will eliminate alternative hypotheses and eventually localize the robot. We extend their approach in a setting of noisy sensors, with occlusions from dynamic obstacles.

In the past, there has been little work to actively plan localization positions that will counteract the effect of dynamic obstacles. In this paper we propose a new planning method to actively reduce the localization uncertainty by eliminating alternative hypotheses.

III. Problem Statement

A single autonomous mobile robot moves in a known, bounded, planar environment $E \subset \mathbb{R}^2$, which needs not be simply-connected. The map is represented as a two-dimensional occupancy grid, in which each cell is marked as either free or a static obstacle. This map of static obstacles can be built beforehand by using any SLAM algorithm, like the one proposed by Grisetti, Stachniss, and Burgard [17]. Within this environment, dynamic obstacles of unknown number and shape move along unpredictable trajectories. The robot's initial pose, $q_0 \in E \times [-\pi, \pi)$, is also unknown.

The robot moves via differential drive, equipped with a laser range finder that is able to perceive the distances to obstacles within a range r and its field of view. Our experiments use the TurtleBot 2[1] platform, equipped with a laser range scanner with a field of view of 240°, an angular resolution of about half a degree, and 6 m range. In general, the proposed method could be extended to other robotic platforms.

As the robot moves, it selects a series of destination poses, relative to its current pose, and continually updates its belief about its pose relative to the map through the acquired sensor data. This process leads to the two related problems addressed in this paper.

a) Passive localization: Observing a sequence of laser scans and movement commands, maintain a probability distribution of the possible poses of the robot in the presence of dynamic obstacles.

b) Active localization: Select a sequence of local poses $Q = \langle q_0, q_1, \ldots, q_n \rangle$ the robot should reach so that the belief on its pose is a consistent estimate of the actual pose of the robot in the environment. This may be a closed-loop strategy, so that the next destination pose q_{i+1} can be selected based on the robot's belief upon reaching destination q_i. While performing the active localization, both cost—i.e., traveled distance—and belief updates are considered, the latter having a priority over the former.

IV. Information-based global localization

The general idea of the proposed algorithm is to use a particle filter, modified to account gracefully for the presence of dynamic obstacles. From the particle set, a set of clusters is extracted, modeling different hypotheses about the robot's true pose. The centroids of each cluster are then utilized to form motion plans that can disambiguate between the clusters, eventually localizing the robot.

A. Particle Filter based global localization

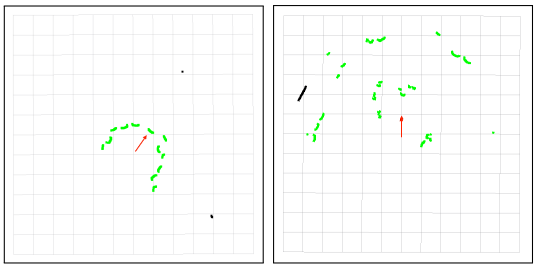
We employ a particle filter [18], [19], [20] to solve the passive global localization problem. At the initial step, the particles are distributed uniformly over the free space of the known map. Consequently, as the robot moves around collecting range data, the propagation, update, and occasional resampling steps are performed. The propagation step follows the standard equations of motion with linear velocity v_t and angular velocity ω_t; see Equation 1 for particle $\mathbf{x^i_{t+1}}$:

$$\mathbf{x^i_{t+1}} = \begin{bmatrix} x^i_{t+1} \\ y^i_{t+1} \\ \theta^i_{t+1} \end{bmatrix} = \begin{bmatrix} x^i_t + v_{t+1} dt \cos\theta^i_t \\ y^i_t + v_{t+1} dt \sin\theta^i_t \\ \theta^i_t + \omega_{t+1} dt \end{bmatrix} \quad (1)$$

For the update step, the standard weight functions are inappropriate. The presence of dynamic obstacles means that, even for particles that are at or near the correct pose, the distances measured by the laser sensor may be significantly less than the distance to the nearest static obstacle in that direction. Figure 2 illustrates the idea with representative laser scans from two experiments. In Figure 2a, the robot is surrounded by twelve other Turtlebots that operate as dynamic obstacles; in Figure 2b, the robot is surrounded by humans moving through the environment. In both cases, close range measurements from the dynamic obstacles dominate and they would have eliminated most particles when using a traditional weighting scheme for the updates. In our preliminary experiments, a standard weighting function—i.e., a Gaussian weighting—used for update in such a scenario led the particle filter to divergence very quickly, as some particles were depleted by shorter measurements due to occlusions from dynamic obstacles. In addition, the

[1] http://www.turtlebot.com/

AMCL localization ROS package,[2] a Monte Carlo localization approach, when used with many dynamic obstacles was unable to localize. This can be explained by the fact that, although the observation model includes a term for probabilistically accounting for dynamic obstacles, particles eventually deplete if they obstruct the view of the robot for an extended period of time.



(a) (b)

Fig. 2. Laser scans with robots (left) and people (right); the red arrow represents the robot pose, the scans are post-colored: green represents dynamic obstacles, while black represents static obstacles.

To resolve this limitation of the standard approach, to model the sensor, we propose the weighting function in Equations 2 and 3, using the range measurements ρ_r, for $r = 1, \ldots, l$:

$$\mathbf{w}_{t+1}^i = \mathbf{w}_t^i \prod_{r=1}^{l} w(i,r) \tag{2}$$

in which

$$w(i,r) = \begin{cases} \frac{1}{\sigma_{laser}\sqrt{2\pi}} e^{-\frac{(\rho_r - \hat{\rho}_r^i)^2}{2\sigma_{laser}^2}} & \text{if } \rho_r - \hat{\rho}_r^i \geq -\delta \\ 1 & \text{if } \rho_r - \hat{\rho}_r^i < -\delta \end{cases} \tag{3}$$

$\hat{\rho}_r^i$ is the estimated range measurement r to the wall from the pose of particle i; σ_{laser} represents the laser sensor's noise; and δ is a small positive constant to account for noise of the sensor. Reasonable values for δ are within an interval between zero and σ_{laser}. The parameter δ is suggested to be close to zero so that the weighting function is more conservative and more scan readings, possibly deriving from dynamic obstacles, are filtered out.

The intuition is to follow a conservative policy for the updates. Equation 3 scales down the weight for particles that appear closer to an obstacle than the sensor reading. Otherwise, Equation 3 leaves the weight unchanged, because a shorter range measurement is possibly the result of a dynamic obstacle that obscures the view of the laser and thus no additional information is available. We apply a threshold ε for leaving the weight unchanged, given the sensor model. See Figure 3 for an instance of the weighting function.

Due to the large number of particles required to effectively cover all the available free space initially, only a limited number of laser ranges are used in the update. In the experiments shown in the following section, with 10000 particles, about 60 scans uniformly distributed across the sensor's field of view are considered in the filter.

[2] http://wiki.ros.org/amcl

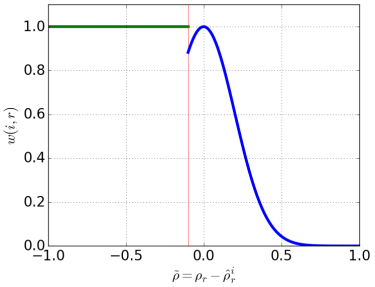


Fig. 3. An example of $w(i,r)$ (Equation 3) with $\sigma_{laser} = 0.2$ and $\varepsilon = 0.1$.

The proposed weighting function does not decrease its applicability to different scenarios. It is also general enough to work when, for example, there is a change in the environment and no dynamic obstacles are present. The reason is that particles are not punished if the expected sensor reading is longer than the actual measurement, as, for example, in a scenario in which the robot has a wall on one side and open space on the other side. In such a case, even if possibly taking more time because of the non-reliability assumption of some readings, the algorithm will still converge.

As done in state-of-the-art particle filters [20], the effective sample size ESS [21] is estimated after every update and, when it drops below 80% of the particle population, resampling [22] probabilistically eliminates particles with negligible weights and multiplies the ones with significant weights.

B. Active Localization strategy

Starting from the particle filter, we derive the following method to actively decide where to send the robot for improving the localization accuracy; see Fig. 4 for an outline of the proposed algorithm.

Our experiments with the passive algorithm from Section IV-A show that, after a few iterations of movement, the particles concentrate in several clusters, each representing a possible area where the robot could be. For example, Figure 5a depicts particles in an artificial environment, particularly challenging for localization because of the self-similarities, where clusters formed after the robot traveled less than 10 m. In general, the way that clusters form

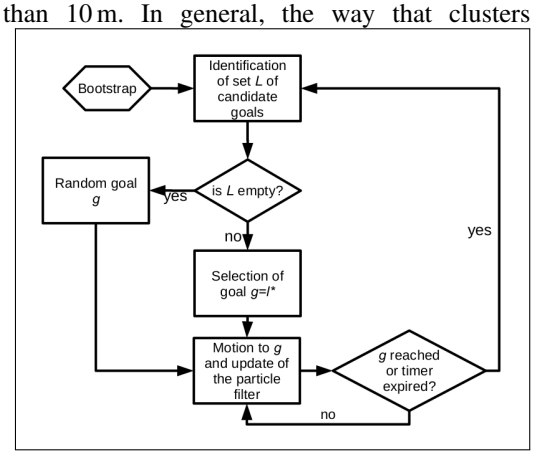


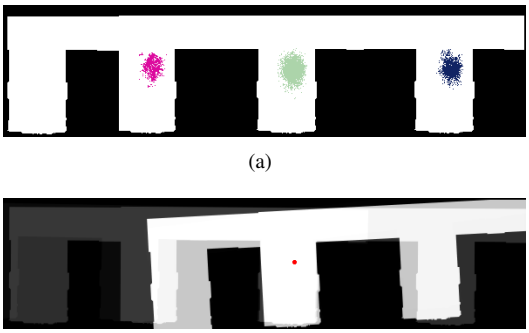
Fig. 4. Block diagram representing active localization.

depends on the initial pose of the robot and the number of self-similarities present in the environment. As the robot continues to move, some of the clusters of particles may remain, some others might decrease in size or completely disappear. In fact, even if the robot has not been able to sense any of the static obstacles in the map, the motion and the limited sensing range may provide adequate information to eliminate particles from areas of the map.

Adequate coverage of the free space requires a large number of particles, however, reasoning for thousands of particles is impractical. Therefore, a clustering algorithm is applied on the particles to concentrate the control strategy on a select few candidate locations.

The Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm [23] was selected among the several clustering algorithms, mainly because, unlike other classical clustering algorithms (e.g., k-means [24]), it does not require the number of clusters to be given as input. The algorithm, which takes two input parameters: ε and *MinPts*, is outlined next. The first parameter defines an ε-neighborhood: $N_\varepsilon(x) = \{y \in X | \text{dist}(x,y) \leq \varepsilon\}$, where, in our case, x and y are two particles and dist() is the distance function between the two particles in the three-dimensional state space. In this paper, the Euclidean distance is computed with a weighting factor to combine linear distance and angular distance. Its value should be set according to the measure unit used by the robot and should account for the noise in the robot odometry. Smaller values increase the number of clusters or outliers that can be found, while bigger values reduce that number. The second input parameter defines the *core object*, namely, whether a new cluster should be formed. More formally, the algorithm requires that $|N_\varepsilon(x)| \geq MinPts$ before forming a new cluster at a given particle x. Setting the value of this parameter has similar effect as ε for the number of clusters and outliers. Given these input parameters, the algorithm starts with an arbitrary point which has not been checked yet. A query is performed to retrieve the ε-neighbors and if there are at least *MinPts* points, then a new cluster is added. Otherwise, the point is marked as noise. These steps are iteratively performed for each non-visited point. If the set returned by an ε-neighborhood query contains a point that is already in another cluster, then the two clusters are merged together; this property is also known as *density reachability*. Its time complexity is $O(n \log n)$ when using an appropriate indexing structure to store and query the points. In order to improve its efficiency for a real-time deployment, particles to be considered in the clustering are sampled randomly. For example, with 10000 particles and a sampling of 30% of the particles, our preliminary experiments showed a decrease of computation time from about 30 seconds to 1 second with very little change to the clustering output.

We treat each of these clusters of particles as a plausible hypothesis about the robot's current pose. The objective of the active localization problem is to drive the robot to eliminate such hypotheses until only one remains. Taking inspiration from the approach of Dudek, Romanik, and Whitesides [16] briefly described in Section II, we propose a method to find the destination locations that will drive the particle filter to discard at least one of the hypotheses on its position.



(a)

(b)

Fig. 5. Result of the clustering and the related compound map. In (a), each color represents a different cluster of particles. In (b), which is cut on the right side to fit in the paper, the fixed reference frame is enlarged and displayed in red; the white and black areas are the common area (free and occupied) between the maps from different clusters' centroids; while the grey area represents spaces that look different from different candidate poses.

Starting from the clusters found by the DBSCAN algorithm, we identify a representative pose for each of them (e.g., the centroid), we transform the map associated with each of the representatives to a common frame (without loss of generality, we arbitrarily take one of them as the fixed reference frame), and overlay each of the transformed maps on top of each other. This forms a compound map that shows differences in the static obstacles that the robot might expect to observe for each of the cluster hypotheses, in a common coordinate frame.

More formally, given one pose (x_f, y_f, θ_f) that will be the fixed reference frame for the compound map and another pose (x, y, θ), the following transformation matrix is applied to each point of the map of the latter:

$$T = \begin{pmatrix} \cos(\theta - \theta_f) & \sin(\theta - \theta_f) & -(x - x_f) \\ -\sin(\theta - \theta_f) & \cos(\theta - \theta_f) & -(y - y_f) \\ 0 & 0 & 1 \end{pmatrix}. \quad (4)$$

The resulting compound map is a grid of integers, similar to a standard occupancy grid, but in which each cell stores the number of clusters whose transformed maps have obstacles overlapping that cell. See Figure 5b.

In our setting, recall that the particles can be discarded only if, during propagation, they fall on an obstacle. Particle weights can change if the robot can acquire a new sensor reading whose ranges are longer than the last ones. Thus, to maximize the number of particles that can be discarded—thereby eliminating hypotheses and making progress toward global localization— the robot should move to a location in the compound map that is free for some, but not all, of the candidate poses. To avoid unnecessarily long movements, consider only cells with this property that are reachable by traversing cells that are clear in all layers of the compound map. We let L denote the set of such cells in the compound map, and evaluate each candidate $l \in L$ according to two criteria:

- the number of particles expected to be discarded $DP(l)$: this information can be derived by simulating a motion with respect to the robot frame for each particle;
- the traveled distance $d(l)$, namely the distance between the fixed reference frame of the compound map and the candidate location.

Theoretically, the next candidate location can be chosen as $l^* = \arg\min_{l \in \{L | DP(l) = n/2\}} d(l)$, in which n denotes the number of particles. In such a way, the theoretical complexity in the worst case is $O(n)$. However, sometimes it could happen that no location allows the robot to have an estimated number of particles removed equal to $n/2$. Thus, we combine the two criteria in a utility function, resulting in the following optimization function:

$$l^* = \arg\max_{l \in L} \left[\alpha \tilde{DP}(l) + (1-\alpha) \hat{d}(l) \right], \tag{5}$$

where α is a weight that accounts for the importance of the criterion DP and $\tilde{DP}()$ and $\hat{d}()$ are normalized values of $DP()$ and $d()$, respectively, to a common scale between 0 (worst) and 1 (best), by using a linear relative normalization.

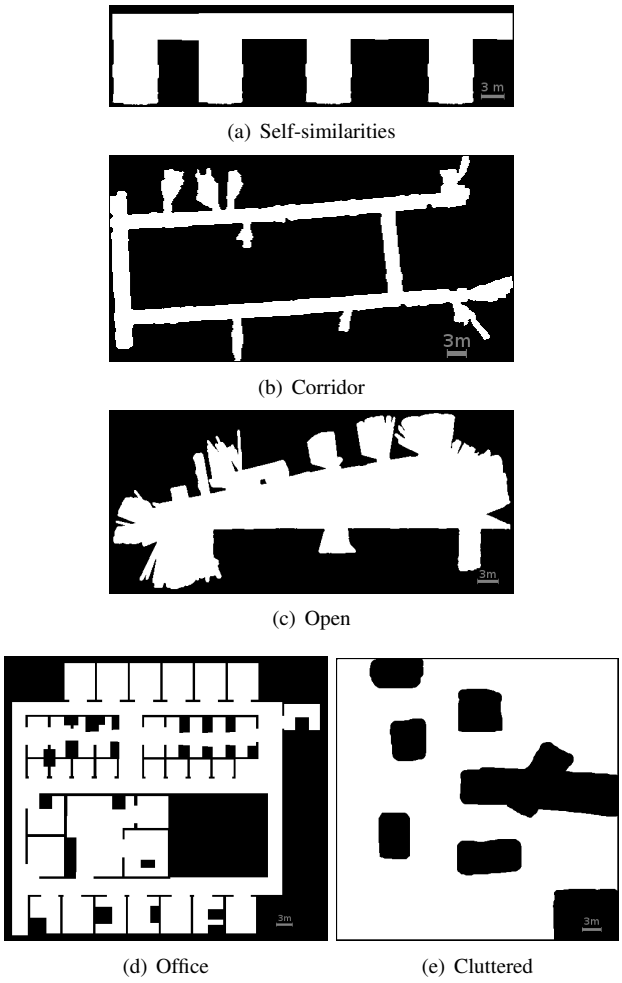
After selecting a destination position, the target orientation of the robot is chosen in such a way that at least one scan could possibly have the maximum reading, namely pointing towards one of the grey areas in the compound map.

This planning process is repeated once the robot reaches the assigned destination location. Note, however, that because of the presence of the dynamic obstacles, and because of the probabilistic nature of the underlying particle filter, the destination location we select might not be reachable in reality. This requires a reactive local path planner with some recovery mechanism, so that the robot does not oscillate over time towards a non-reachable goal location. In particular, when the robot starts moving, it tries a rotation–translation–rotation motion. If an obstacle is encountered it stops and waits until either the obstacle is not present anymore or a timer expired. In the first case, the robot resumes the motion, while in the second case, a Bug-type path planner [25] is activated to try to reach the destination location. If also in this state the robot is not able to reach the destination location, either because the environment is too crowded or because it is oscillating in the same area, then it aborts and computes a new goal location.

In case no compound map, that is clusters, can be found (especially when the robot is just deployed in the environment and thus all the particles are uniformly distributed over the environment), the robot performs a selects one pose within the covered area by the laser sensor.

V. Experimental Results

To validate our method, extensive simulation tests have been performed using the Stage simulator [26] and with a real TurtleBot 2 robot together with other TurtleBot 1, 2, and Create robots, which act as dynamic obstacles. We developed the software within the ROS framework [27]. For the experiments, 10000 to 30000 particles are used for globally localizing the robot according to the size of the environment. For clustering the two parameters are set to $\varepsilon = 1$ and $MinPts = 50$; 30% of the particles are considered for clustering. Changing the parameters values by $\pm 10\%$ did not present any significant change in the results.



(a) Self-similarities (b) Corridor (c) Open (d) Office (e) Cluttered

Fig. 6. Environments used for tests.

A. Simulation tests

The proposed method has been tested on different environments from the Radish repository [28]; here, we show the results for four of them along with one synthetic environment; see Fig. 6. These environments present different characteristics and their sizes approximately range between 25 m and 80 m:

- *self-similarities*: an artificial environment we generated which presents challenges for localization as there are 4 corridors that look similar;
- *corridor* (usc-sal200-021120): a real environment consisting of long corridors forming a loop and some rooms;
- *office* (sdr_site_b): a real environment that has many rooms inside;
- *open* (fr101-explored): a big open space;
- *cluttered* (cave_bitmap): an outdoor-like environment cluttered with several obstacles.

Stage was used for simulating a TurtleBot with a simulated Hokuyo URG04-LX laser range finder. The simulations

also included 8, 24, or 48 additional TurtleBots to act as dynamic obstacles. The dynamic obstacles were initially placed around the robot to cover its field of view. The dynamic obstacles moved using a random walk algorithm. Figure 7 shows an instance of simulation in the structured environment with 48 dynamic obstacles surrounding the robot.

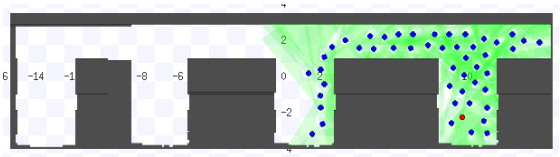


Fig. 7. An instance of a simulation test with Stage (blue: dynamic obstacles; red: localizing robot).

To evaluate our method, we implemented three other strategies:

- *random goal*: the performed steps are as in the proposed method, but the goal location is chosen randomly from the compound map;
- *furthest*: the robot goes along the opposite direction of the furthest range. The rationale is to force particles to be discarded;
- *random walk*: the robot moves randomly through the environment.

For each environment and 5 initial starting poses selected to cover different areas of the environments, we run the four strategies. We measured the computation time to make a decision, the traveled distance, and the weighted average error in the pose of the robot between the particles and the ground truth available from Stage given a 15 minutes timeout. If the weighted average error between the current distribution of particles and the actual position and orientation of the robot is less than 1 m and less than 0.5 rad, respectively, then the robot is considered localized. Note that these thresholds are set considering the hardware available on the Turtlebot 2 and testing it with AMCL in a static scenario, observing how spread the cluster of particles is when the robot is localized. Figure 8 shows the traveled distance, the weighted average error, and the average duration of the experiments for the five environments.

The results show that the proposed method, shown in Figure 8 as 'informed', generally commands the robot to travel less distance and/or allows the robot to localize faster and at the same time to have a comparatively lower average error. Indeed, in all of the experiments, the robot using the informed strategy was able to localize itself. The most benefits appear in the structured environment. The reason can be found in the fact that, being more structured, the compound map is more informative when clusters of particles are formed. If less structured, such as the open environment, there are less spots to travel to at which to deplete the particles. When the number of dynamic obstacles increases, the performance of the methods possibly degrades, because of their interference in the motion, leading the localizing robot to stop or to spend more time in order to avoid them. Nevertheless, the proposed, informed strategy is able to accomplish the global localization by avoiding to keep going back and forth in a small area and trying at every decision step to get to the candidate location that allows to disambiguate between hypotheses; see Figure 9 for an example of paths with different strategies and how the error with respect to the ground truth evolves over time. Also, the entropy of the particles is reported computed with a non-parametric entropy estimator [29]. It is not surprising that the error could increase over time, as cluster of particles can move far away from the actual pose of the robot, depending on the particles' orientation; however, the entropy of the particles decreases as the particles converge to a unique hypothesis.

B. Experiments with real robots

The approach used in the experiments with real robots is the same as the simulated tests: a TurtleBot 2 equipped with a Hokuyo URG04-LX laser sensor and several Turtlebot 1, Turtlebot 2, and iRobot Create robots, each equipped with a cardboard "hat" so that the Turtlebot 2 with the laser sensor can detect them, were placed in the Amoco Hall antechamber at the University of South Carolina; see Figure 1. The size of the environment is about 20 by 20 m. Before the experiments, the TurtleBot 2 with the laser range finder was manually driven (with no dynamic obstacles present) to produce an accurate map of the environment using the ROS gmapping package [17]. The resulting map was used as the map for the particle filter updates. Several experiments were performed using different configurations of the robots as dynamic obstacles. The dynamic obstacles had different behaviors as the Turtlebots were using a random walk, while the Creates were moving using their preprogrammed patterns. We also ran the robot on a few occasions with dozens of people in the space, to verify the performance; see Figure 2b.

Figure 10 represents the distribution of particles during

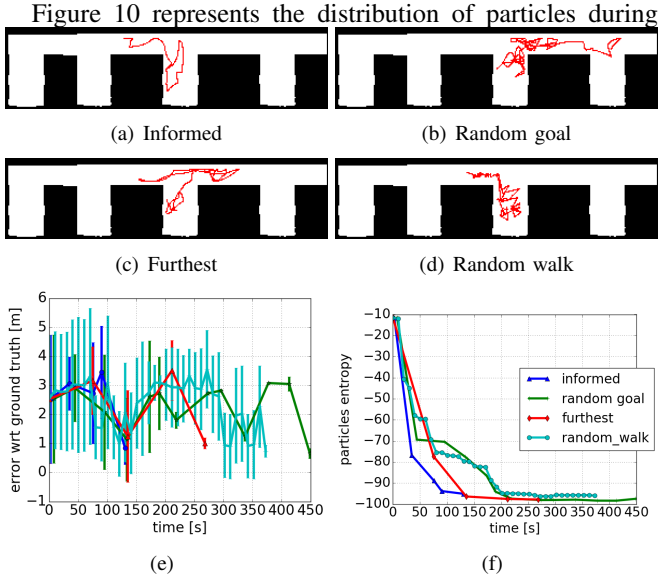


Fig. 9. Paths followed by the localizing robot using the informed (a), random goal (b), furthest (c), and random walk (d) strategies with 24 dynamic obstacles. (e) and (f) shows the average error and entropy over time.

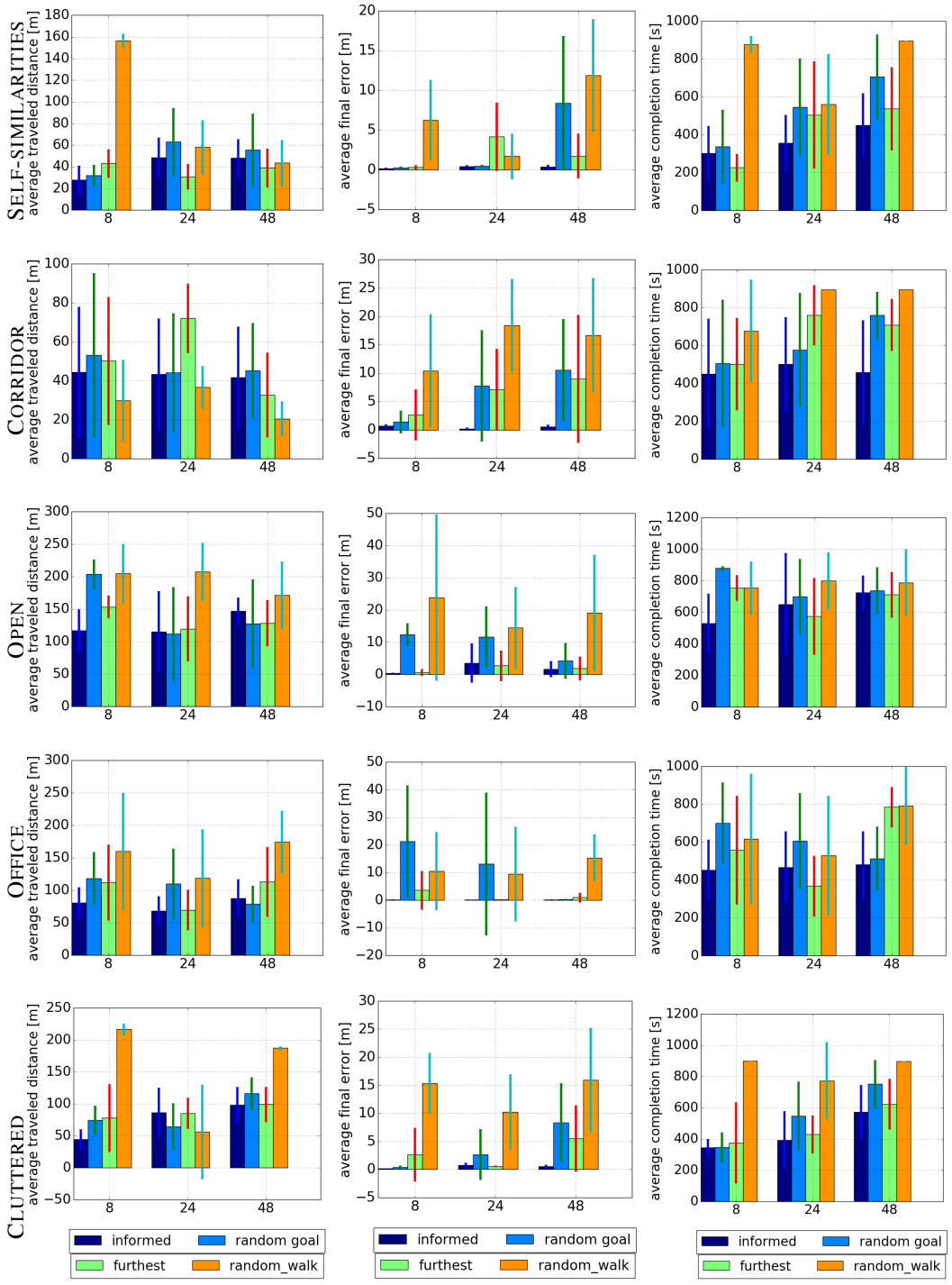


Fig. 8. Results in the five different environments; from left to right: distance traveled, average error, average experiment duration.

a representative run of the real robot outside Amoco hall. The environment has several areas that are quite similar, as such often the localization algorithm results in some symmetrical hypotheses; see Fig. 10b. While no ground truth was available, Figure 10d shows that the particle filter standard deviation decreases over time and the final cluster of particles corresponds to the final position of the robot in the environment; see Fig. 10c. Note that the standard deviation, as the average error in the simulation, is not necessarily monotonically decreasing. The reason is that the standard deviation is computed with respect to the particles' centroid and thus leading to an increase in the standard deviation, as can be seen in Fig. 10b. Although a slight increase can happen also measuring the entropy, e.g., when different clusters of particles are in nearby location, it is a more stable measure to see whether particles are representing a unique hypothesis; see Fig. 10e.

VI. CONCLUSIONS

In this paper, we presented a method for active global localization of a single robot in a known environment in

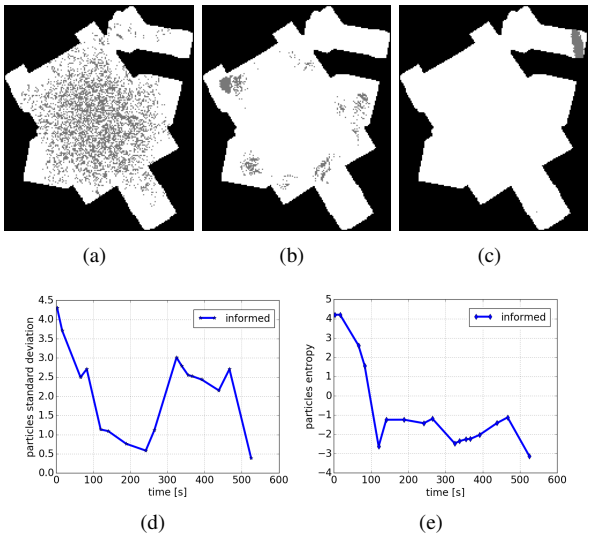


Fig. 10. Different steps of the proposed global localization system, with the robot starting from the center of the environment, surrounded by 12 dynamic obstacles. (a) After a few steps, particles too close to the wall are eliminated; (b) due to symmetries in the environment some clusters form indicating alternative hypotheses; (c) traveling towards the corridor allows to discard all of the hypotheses as they go through the obstacles and thus the robot is localized. (d) and (e) shows the dispersal of particles and the entropy, respectively.

the presence of multiple dynamic obstacles. The well-known particle filter localization algorithm was extended to avoid penalizing particles from sensor data that might have arisen from dynamic obstacles. We also introduced an active localization algorithm which guides the robot efficiently toward areas that will eliminate alternative hypotheses. Extensive experiments both in simulation and using real robots validate the proposed approach in a number of realistic environments.

Extending the approach to combine path planning to task-specific destinations with active localization goals is currently under development. We are currently porting our implementation on a Husky robot in order to verify the proposed approach outdoors. Future experiments will utilize the GPS location of the Husky in order to verify the localization accuracy. Future work will theoretically analyze the observation model for the weighting functions and how to reduce the computational complexity. As robots will coexist with humans in public spaces more frequently, it is crucial to enable them with the capabilities of accurate localization even when most of their sensory information is occluded by humans and/or other robots moving in the same space.

ACKNOWLEDGEMENT

This material is based upon work supported by the National Science Foundation under Grant Numbers 1513203, 1526862, and 0953503.

REFERENCES

[1] W. Burgard, A. Cremers, D. Fox, D. Hahnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun, "Experiences with an interactive museum tour-guide robot," *Artif. Intell.*, vol. 114, no. 1-2, pp. 3–55, 1999.

[2] S. Thrun, M. Beetz, M. Bennewitz, W. Burgard, A. Cremers, F. Dellaert, D. Fox, D. Hahnel, C. Rosenberg, J. S. N. Roy, and D. Schulz, "Probabilistic algorithms and the interactive museum tour-guide robot minerva," *Int. J. of Robot. Res.*, vol. 19, no. 11, pp. 972–999, 2000.

[3] F. Faber, M. Bennewitz, C. Eppner, A. Gorog, C. Gonsior, D. Joho, M. Schreiber, and S. Behnke, "The humanoid museum tour guide robotinho," in *Proc. RO-MAN*, 2009, pp. 891–896.

[4] B.-O. Han, Y.-H. Kim, K. Cho, and H. Yang, "Museum tour guide robot with augmented reality," in *Proc. VSMM*, 2010, pp. 223–229.

[5] J. Schulte, C. Rosenberg, and S. Thrun, "Spontaneous short-term interaction with mobile robots in public places," in *Proc. ICRA*, vol. 1, 1999, pp. 35–40.

[6] N. Roy, W. Burgard, D. Fox, and S. Thrun, "Coastal navigation-mobile robot navigation with uncertainty in dynamic environments," in *Proc. ICRA*, vol. 1, 1999, pp. 35–40.

[7] D. Fox, W. Burgard, and S. Thrun, "Markov localization for mobile robots in dynamic environments," *J. Artif. Intell. Res.*, vol. 11, pp. 391–427, 1999.

[8] C. Stachniss and W. Burgard, "Mobile robot mapping and localization in non-static environments," in *Proc. AAAI*, vol. 20, no. 3, 2005, pp. 1324–1329.

[9] D. Fox, W. Burgard, F. Dellaert, and S. Thrun, "Monte carlo localization: Efficient position estimation for mobile robots," *AAAI/IAAI*, vol. 1999, pp. 343–349, 1999.

[10] A. Milstein, J. N. Sánchez, and E. T. Williamson, "Robust global localization using clustered particle filtering," in *Proc. AAAI*, 2002, pp. 581–586.

[11] T. Li, S. Sun, and J. Duan, "Monte carlo localization for mobile robot using adaptive particle merging and splitting technique," in *Proc. ICIA*, 2010, pp. 1913–1918.

[12] Z. Liu, Z. Shi, M. Zhao, and W. Xu, "Mobile robots global localization using adaptive dynamic clustered particle filters," in *Proc. IROS*, 2007, pp. 1059–1064.

[13] J. Röwekämper, C. Sprunk, G. D. Tipaldi, C. Stachniss, P. Pfaff, and W. Burgard, "On the position accuracy of mobile robot localization based on particle filters combined with scan matching," in *Proc. IROS*, 2012, pp. 3158–3164.

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

[15] R. Valencia, J. Saarinen, H. Andreasson, J. Vallvé, J. Andrade-Cetto, and A. J. Lilienthal, "Localization in highly dynamic environments using dual-timescale ndt-mcl," in *Proc. ICRA*, 2014, pp. 3956–3962.

[16] G. Dudek, K. Romanik, and S. Whitesides, "Localizing a robot with minimum travel," *SIAM J. Comput.*, vol. 27, no. 2, pp. 583–604, 1998.

[17] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with rao-blackwellized particle filters," *IEEE T. Robotics*, vol. 23, no. 1, pp. 34–46, 2007.

[18] F. Dellaert, W. Burgard, D. Fox, and S. Thrun., "Using the condensation algorithm for robust, vision-based mobile robot localization," in *Proc. CVPR*, 1999.

[19] S. Thrun, D. Fox, W. Burgard, and F. Dellaert, "Robust monte carlo localization for mobile robots," *Artificial Intelligence*, vol. 128, no. 1-2, pp. 99–141, 2000.

[20] I. M. Rekleitis, "A particle filter tutorial for mobile robot localization," Centre for Intelligent Machines, McGill University, Montreal, CANADA, Tech. Rep. TR-CIM-04-02, 2004.

[21] J. S. Liu, R. Chen, and T. Logvinenko, "A theoretical framework for sequential importance sampling with resampling," in *Sequential Monte Carlo Methods in Practice*, A. Doucet, N. de Freitas, and N. Gordon, Eds. Springer New York, 2001, pp. 225–246.

[22] J. Carpenter, P. Clifford, and P. Fearnhead, "Improved particle filter for nonlinear problems," *Proc. Radar, Sonar and Navigation*, vol. 146, no. 1, pp. 2–7, 1999.

[23] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *Proc. KDD*, vol. 96, no. 34, 1996, pp. 226–231.

[24] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proc. Berkeley Symposium on Mathematical Statistics and Probability*, vol. 1, no. 14, 1967, pp. 281–297.

[25] V. Lumelsky and A. Stepanov, "Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape," *Algorithmica*, vol. 2, no. 1-4, pp. 403–430, 1987.

[26] R. Vaughan, "Massively multiple robot simulations in stage," *Swarm Intelligence*, vol. 2, no. 2-4, pp. 189–208, 2008.

[27] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "ROS: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, 2009.

[28] A. Howard and N. Roy, "The robotics data set repository (Radish)," http://radish.sourceforge.net/, 2003.

[29] G. V. Steeg, "Non-parametric entropy estimation toolbox (NPEET)," http://www.isi.edu/~gregv/npeet.html, 2013.