Poisson-Driven Dirt Maps for Efficient Robot Cleaning

Jürgen Hess Maximilian Beinhofer Daniel Kuhner Philipp Ruchti Wolfram Burgard

Abstract— **Being able to estimate the dirt distribution in an environment makes it possible to compute efficient cleaning paths for robotic cleaners. In this paper, we present a novel approach for modeling and estimating the dynamics of the dirt generation in an environment. Our model uses cell-wise Poisson processes on a regular grid to represent the dirt in the environment, which allows for an effective estimation of the dynamics of the dirt generation and for making predictions about the absolute dirt values. We propose two efficient cleaning policies which are based on the estimated dirt distributions and can easily be adapted to different needs of potential users. In extensive experiments carried out in simulation and with a modified iRobot Roomba vacuum cleaning robot, we demonstrate the effectiveness of our approach.**

I. Introduction

Cleaning is one of the most important applications of nowadays and also future household robots. The very first cleaning robots such as the iRobot Roomba, however, did not systematically clean the environment but rather performed predefined motion patterns mixed with random movements [14]. Accordingly, such systems in general cannot guarantee the success of the cleaning process, especially when their operation time is limited. Recently, vacuum cleaning robots that are able to systematically clean an environment, like the Neato XV, the Samsung Navibot, and the Evolution Robotics Mint, have been developed. These robots apply simultaneous localization and mapping (SLAM) procedures to estimate the pose of the vehicle which enables them to clean the floor in a more systematic fashion. However, cleaning the entire environment is often not required as some parts might quickly become dirty while others stay relatively clean for longer periods of time. For example, in the entrance area of a home or in the kitchen, there might be more dirt than in other areas such as the living room. Obviously, knowledge about the distribution of dirt in the environment can enable cleaning robots to generate much more efficient cleaning paths, as only the dirty parts of the environment would have to be worked upon.

In this paper, we consider the problem of estimating the dirt distribution on the floor of an environment and of planning efficient cleaning paths given this distribution to be used by a robot with appropriate sensing capabilities. To model the dirt distribution, we divide the environment into regular grid cells and apply cell-wise Poisson processes for estimating the amount of dirt in every cell. The corresponding dirt values grow over time and are reset by the cleaning operation. We show that this model allows for the effective estimation of the dynamics of dirt generation and the prediction of the absolute dirt values.

All authors are with the Autonomous Intelligent Systems Lab, Computer Science Department, University of Freiburg, Germany. This work has partly been supported by the German Research Foundation (DFG) within the Research Training Group 1103. hess@informatik.uni-freiburg.de

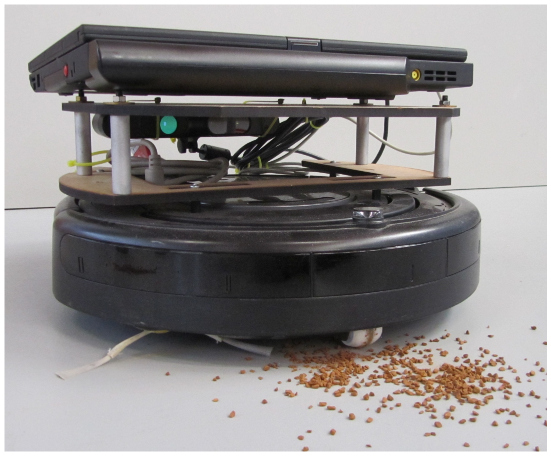

Fig. 1. The Roomba 560 robot we used in the experiments. The vacuum cleaning unit of the robot is equipped with a dirt detection sensor. The mounted Asus Xtion Pro Live depth camera is used for localization.

To exploit the estimated dirt distribution maps, we develop two efficient cleaning policies. The first policy aims at minimizing the cleaning time while guaranteeing with high confidence a user-defined bound on the maximum dirt level in the environment. The second policy minimizes the maximum dirt value in the environment within a given, limited cleaning time.

The contribution of this paper is two-fold: First, we present an approach for modeling and estimating the current dirt distribution as well as its dynamics in a given environment. The resulting dirt distribution map can help the user to find typical dirt hot spots and maybe change his behavior accordingly. Also, it can be used to generate efficient cleaning policies for cleaning robots. The second contribution is an approach to generate these policies, resulting in shorter operation times of the robot while still restricting the maximum dirt level in the environment. The policies can be adjusted to the specific preferences of the user in terms of cleanliness, cleaning time, and even different importance of different areas in the environment. The user benefits from this cleaning behavior, as he is less disturbed by the noise generated by the moving robot and the robot's energy consumption is reduced.

This paper is organized as follows. After discussing related work in the next section, we introduce the dirt map model, the estimation, and the prediction of dirt in Sec. III. In Sec. IV, we present the efficient cleaning policies. Finally, we provide extensive experiments that evaluate our approach both in simulation and with the real robot shown in Fig. 1.

II. Related Work

One of the earliest navigation systems for autonomous cleaning robots, targeted at chain stores, is the SINAS system which started operation as early as 1996 [10]. It is one example for the specific task of cleaning or rather covering a known surface which is referred to as coverage path planning. There is a large body of literature regarding the general problem of coverage path planning (see Choset [1] for an overview). Many approaches seek for the shortest path that covers the entire, a priori known environment [3], [9], [12]. In contrast to these complete coverage approaches, we aim at covering only a part of the environment such that according to our policies, either the time needed to reach a certain level of cleanliness or the weighted maximum dirtiness after cleaning is minimized. Note that only without any prior information about the dirt distribution, the problem considered in this paper corresponds to the complete coverage problem.

Covering only specific parts of an environment has also been addressed in multi-robot navigation [8], [17]. Zlot et al. [17] for example generate possible exploration goals and similarly to our approach frame the problem of visiting all goals as a traveling salesman problem (TSP).

Cleaning applications that aim at covering specific parts or the entire environment require that the robot can accurately map the environment and localize itself. In the area of floor cleaning, Gutmann et al. [4] propose an algorithm for simultaneously estimating a vector field induced by stationary signal sources in the environment as well as the robot pose in this vector field. Jeong and Lee [6] use a single camera pointing towards the ceiling and apply a landmark-based SLAM algorithm. The approaches of Zhang et al. [16] and Erickson et al. [2] show that SLAM can also be solved with very limited range sensors like bumpers. In our work we use a low-cost range sensor for mapping and localization.

The ideas presented in this paper are related to the work of Luber et al. [11] from the area of people tracking. In that paper, the authors propose spatial affordance maps to represent space-dependent occurrences of people activity events to improve the people tracking. Similar to our approach, these maps use cell-wise Poisson processes. An earlier work of Kruse et al. [7] proposes a statistical grid, a grid map in which each cell models the probability of the appearance of a dynamic obstacle with a Poisson process. In this paper, we apply Poisson processes to model and estimate the dynamics of the generation of dirt in an environment and utilize this dirt map to generate efficient cleaning policies.

III. Poisson Processes for Modeling Dirt

To selectively clean an environment, a cleaning robot needs to know where the dirt is. One approach for achieving this if the state of the entire environment is not observable by the robot, is to learn how quickly the individual parts of the environment typically get dirty.

Motivated by the well-known occupancy grid maps [13], our approach makes use of a regular tessellation of the environment into grid cells. In the occupancy grid mapping approach, it is typically assumed that the environment is static and that therefore the occupancy of a cell does not change over time. This assumption, however, is not reasonable in the context of the distribution of dirt, as the dirt in a cell grows over time because of polluting events like crumbling or chipping and gets typically reset to zero by cleaning. To model this behavior, we apply for every cell c a homogeneous Poisson process $N^c(t)$ over time t.

A. Properties

A homogeneous Poisson process $\{N^c(t), t \geq 0\}$ is a continuous-time counting process whose increments are stationary, independent, and Poisson distributed. The probability of observing k polluting events during time interval $(s,t]$ is given by

$$p(N^c(t) - N^c(s) = k) = \frac{e^{-\lambda^c(t-s)}(\lambda^c(t-s))^k}{k!}, \qquad (1)$$

where the parameter λ^c is called intensity. Additionally, at time steps t_v at which cell c is cleaned, we fix its level of dirt to $N^c(t_v) = 0$.

B. Dirt Prediction

For deciding which cell to clean next, the cleaning robot needs to be able to make predictions about the amount of dirt in the individual cells of the dirt map. For predicting $N^c(t)$, we use its expectation. The expected value of the amount of dirt k produced during the interval $(s,t]$ in cell c according to the distribution of Eq. (1) is given by

$$\mathbb{E}[N^c(t) - N^c(s)] = \lambda^c(t-s). \qquad (2)$$

If the latest cleaning operation happened at time s, the dirt level at time $t > s$ can be predicted as

$$\begin{aligned} &\mathbb{E}[N^c(t)] \\ =&\mathbb{E}[N^c(t) - N^c(s)] + \mathbb{E}[N^c(s)] \\ =&\lambda^c(t-s) + 0. \end{aligned} \qquad (3)$$

C. Parameter Estimation

Before the robot can use the dirt map for efficient cleaning, it needs to estimate its parameters λ^c for every cell c. During operation, a cleaning robot equipped with a dirt detection sensor receives for every grid-cell c a series of dirt readings $k_0^c, \ldots, k_n^c$ at time steps $t_0, \ldots, t_n$. At the moment of sensing, the cleaning unit with the dirt detection sensor coevally cleans the observed cell. Therefore, every reading k_i except the first k_0 is a sample from the Poisson distribution defined in Eq. (1). If the robot observes the cell in unit intervals, i.e., $t_i - t_{i-1} = 1$ for all i, the maximum likelihood estimator for λ^c is given by $\hat{\lambda}^c_{\text{unit}} = \frac{1}{n}\sum_{i=1}^{n} k_i^c$. In our applications, the observation intervals of the vacuum cleaning robot typically do not have unit length. However, we can still construct a maximum likelihood estimator for λ^c by considering its log-likelihood

$$\begin{aligned} &\log \mathscr{L}(\lambda^c \mid k_1, \ldots, k_n, t_0, \ldots, t_n) \\ =&\sum_{i=1}^{n} \left(k_i \log \lambda^c - \lambda^c(t_i - t_{i-1}) + \log \frac{(t_i - t_{i-1})^{k_i}}{k_i!}\right). \end{aligned} \qquad (4)$$

Calculating the derivative of Eq. (4) and setting it to zero yields the maximum likelihood estimator

$$\hat{\lambda}^c = \frac{1}{\sum(t_i - t_{i-1})} \sum_{i=1}^{n} k_i^c \tag{5}$$

for the non-unit interval case.

In an initial learning phase, the robot cleans the whole environment several times and applies the estimator $\hat{\lambda}^c$ to estimate the dirt map. After that, the learned map can be used for efficient cleaning, while its parameters can still be updated according to Eq. (5) to refine the estimate. For cells c for which all $k_i^c = 0$, we set λ^c to $\varepsilon > 0$, ensuring that all cells are considered in the cleaning procedure.

IV. Efficiently Cleaning using a Dirt Map

In this section, we show how the learned dirt map can be utilized for efficient cleaning. We propose two policies. The first policy aims at minimizing the cleaning time while guaranteeing that after cleaning, the dirtiest cell in the map is less dirty than a user defined threshold. The second one allows a user defined maximum duration of the cleaning cycle and aims at minimizing the maximum dirt value in the map during this time. A cleaning policy takes into account the predicted state of the dirt map and the user preferences. We define the predicted state of the dirt map at time t as

$$m(t) := (\mathbb{E}[N^{c_1}(t)], \ldots, \mathbb{E}[N^{c_M}(t)]), \tag{6}$$

where M is the maximum number of cells in the map. The user preferences include a weight w_c for every cell c, describing how important the cleanliness of this cell is for the user. The weights default to 100% and can be increased in areas important for the user and decreased in unimportant areas. Utilizing these definitions, we define a cleaning policy

$$\pi_t(m(t), w_{c_1}, \ldots, w_{c_M}) := \{c_{i_1}, \ldots, c_{i_{n(\pi_t)}}\} \tag{7}$$

as the set of cells that the robot has to clean during the cleaning cycle at time t.

To execute a policy π_t, we compute a path from the parking position of the robot through all cells comprised in the policy and back to the parking position. We frame this problem as a traveling salesman problem (TSP) with $c_{i_1}, \ldots, c_{i_{n(\pi_t)}}$ as vertices in a fully connected graph and apply a state-of-the-art TSP solver [5] to this graph to find the shortest collision-free path through all cells $c_{i_1}, \ldots, c_{i_{n(\pi_t)}}$. As edge costs, we apply the Euclidean distances of the shortest collision-free paths between the vertices, thereby ignoring the rotational cost of the robot. They could however easily be inserted into the calculation by extending the TSP to a generalized TSP in which a state is comprised of several orientations in the same position. Assuming that the robot executes this path with constant velocity, we can calculate the duration $\tau(\pi_t)$ of the policy execution.

A. Bounded Dirt Time Minimization Cleaning

The first efficient cleaning policy aims at reducing the weighted maximum dirt value in the map after cleaning below a user defined threshold $d_{\max}$ with confidence $1-\delta$, while minimizing the execution time. More formally, at the beginning t of each cleaning cycle, we select the *bounded dirt policy*

$$\pi_t(d_{\max}) = \underset{\pi; P_\pi(d_{\max}) < \delta}{\operatorname{argmin}} \; (\tau(\pi)), \tag{8}$$

where $P_\pi(d_{\max}) = P(\max_c[w_c N^c(t+\tau(\pi))] > d_{\max} \mid \pi_t = \pi)$ is the probability that after executing policy π at time t the dirtiest cell in the map has a weighted dirt value higher than $d_{\max}$. We can calculate $P_\pi(d_{\max})$ from the estimated λ^c values, the predicted dirt map $m(t)$ and the policy π. To do so, we set $N^c(t) = 0$ for all cells c in π and calculate the $(1-\delta)$-quantiles of the dirt distributions of all cells in the map according to their learned λ^c values. Note that the confidence level $1-\delta$ holds true assuming that the estimated dirt map is correct. To calculate the policy, we select all cells whose maximum value of the $(1-\delta)$-quantiles exceeds $d_{\max}$ and calculate a TSP path through these cells.

B. Bounded Time Maximum Dirt Minimization Cleaning

For the second efficient cleaning policy we consider, the user specifies a maximum allowed execution time $\tau_{\max}$ for every cleaning cycle. Given this maximum execution time, we aim at finding the policy that minimizes the weighted maximum dirt value in the map after cleaning. More formally, at the beginning t of each cleaning cycle, we select the *bounded time policy*

$$\pi_t(\tau_{\max}) = \underset{\pi; \tau(\pi) < \tau_{\max}}{\operatorname{argmin}} \left(\max_c \mathbb{E}[w_c N^c(t+\tau(\pi)) \mid \pi_t = \pi] \right). \tag{9}$$

To calculate this policy, we do not need to consider quantiles, as the cell with the highest expected dirt value is also the cell with the highest $(1-\delta)$-quantile for every $\delta < 0.5$. For this policy, other than for the first one, the TSP solver has to be applied iteratively. In every iteration, we add the remaining cell with the highest expected dirt value to the policy and apply the TSP solver. We repeat this process until the specified $\tau_{\max}$ value is reached.

Note that for both cleaning policies the set of cells selected for cleaning changes in consecutive runs of the same policy. If a cell with low dirt production value λ^c does not get selected for cleaning several times, its absolute dirt value $N^c(t)$ keeps growing until its expected dirt value is high enough such that the cleaning policy incorporates this cell in the next cleaning cycle.

V. Experiments

To evaluate the dirt map estimation and cleaning path generation, we performed extensive experiments both in simulation and with a real robot. In all experiments, we set the confidence level $1-\delta$ for the bounded dirt policy to 95%.

A. Learning a Dirt Map form Real Data

To analyze the dirt map learning in practice, we equipped an iRobot Roomba 560 vacuum cleaning robot with an Asus Xtion Pro Live Sensor and a notebook with a 2.26 GHz Intel Core2 Duo processor (see Fig. 1). We found the Roomba

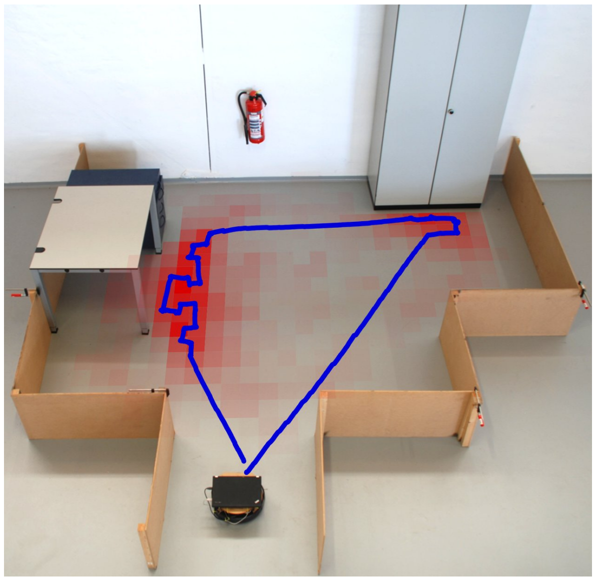
Fig. 2. Experimental setup and learned dirt map (red). The estimated trajectory of the real robot when executing one of the planned paths is shown in blue. The maximum $\hat{\lambda}_c$ value estimated in the experiment is 10.5. The grid size of the dirt map is 0.15 m.

robot particularly suited for our problem, because it is equipped with a dirt sensor, which generates a measurement whenever a dirt particle hits a small metal plate inside the suction unit. In the experiment, we used larger grained flour as dirt. Every dirt measurement corresponded to about 0.03 g of flour. The environment used to perform the robot experiments is shown in Fig. 2.

In order to learn the dirt map, we first built a grid map of the environment. Afterwards, we repeatedly distributed the dirt, mostly in front of the desk and cabinet, and manually steered the robot for to clean the entire environment. We repeated this ten times. Thereby we used Monte Carlo localization [15] for robot pose estimation. In Monte Carlo localization, at every time step t the estimated probability distribution of the pose of the robot is stored as a set of weighted particles $\{x_t^{[1]}, \ldots, x_t^{[n_p]}\}$, each representing a hypothesis about the pose of the robot. After each entire traversal of the environment, we updated the dirt map of the environment according to Eq. (5) using the dirt measurements received. To account for the localization uncertainty, when receiving a dirt measurement k, we considered the particle set after the resampling step in the particle filter, when all particles had the same weight. We divided the dirt measurement evenly on all particles and integrated, for every particle, the resulting value $\frac{k}{n_p}$ as dirt measurement for the cell in which the particle was located. This results in the maximum-likelihood dirt map given the uncertainty in localization. With increasing uncertainty in the localization, the dirt map tends towards a uniform distribution, and with a more accurate localization, the dirt map becomes more peaked. Thus, the maps learned depend on the sensor and actuator noise of the robot used. An example map learned with the vacuum cleaning robot described above is shown in Fig. 2.

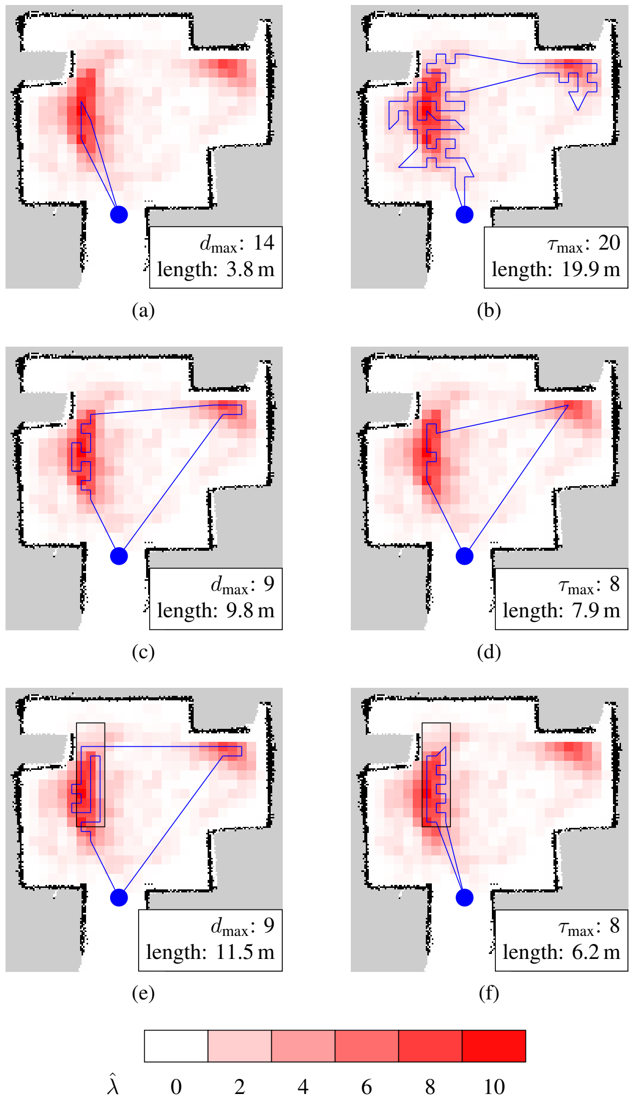

Fig. 3. Paths planned with our policies for the first cleaning cycle on the dirt map learned from real data. The paths in (a) and (c) result from the bounded dirt policy $\pi(d_{\max})$, the ones in (b) and (d) from the bounded time policy $\pi(\tau_{\max})$. The cleaning paths in (e) and (f) for the dirt and bounded time policy result from increasing the weights w_c of the cells in the rectangle to 300%. To allow a comparison with the path length, the unit of the τ values stated in the figure is scaled such that the robot traverses one meter per time step. In (a) and (c), the estimated values $\hat{\lambda}$ for which the dirt stays below a level of 9 and 14 with 95% confidence correspond to 5 and 9, respectively.

B. Cleaning Policies with User Preferences

Having learned the dirt map, we created cleaning paths according to the bounded dirt and the bounded time policies. Fig. 3 shows two results for each policy with different values for the maximum dirt $d_{\max}$ and the maximum time $\tau_{\max}$ allowed by the user. The cleaning paths were planned for the first cleaning cycle after learning the dirt map. For additional cleaning cycles, the paths change as the previously cleaned cells get reset and other cells may get selected.

The time for computing the policies ranged from about one second for the path from Fig. 3a to about four minutes for the path from Fig. 3b. The large difference in computation time is due to the fact that the bounded dirt policy (Fig. 3a)

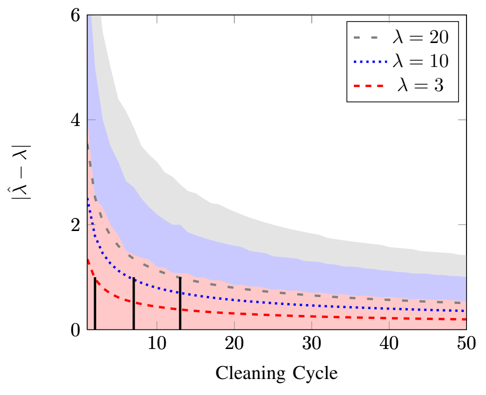

Fig. 4. Difference between the estimated dirt value of a cell $\hat{\lambda}$ and the ground truth λ over a number of consecutive cleaning cycles. This figure shows the average deviations as well as the corresponding empirical 95% confidence intervals.

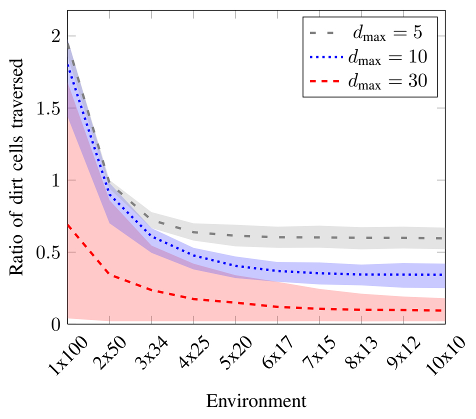

Fig. 5. Evaluation of the bounded dirt policy. The figure shows the average ratio of cells traversed to the total number of cells given different values for $d_{\max}$ and differently structured environments. Also shown are the empirical 95% confidence intervals.

needs to call the TSP solver only once, while the bounded time policy (Fig. 3b) applies the TSP solver iteratively. The computation time, of course, depends on the number of dirty cells in the environment as well as the TSP solver used. The solver used in our experiments runs in approximately $O(n^{2.2})$ [5], but a faster one could be applied if less accuracy is required. As can be seen in the figure, if the user specifies a low maximum dirt value, more cells are cleaned by the robot. The same holds for the maximum time allowed. For a higher maximum time value, more cleaning steps are performed and thus the environment is cleaned more thoroughly.

We also let the real robot execute the cleaning path from Fig. 3c. The executed path estimated by the localization system of the robot is shown in blue in Fig. 2. The path execution was performed by an off-the-shelf navigation system. Evaluating the accuracy of this system and adapting the path execution accordingly could improve the cleaning performance and will be addressed in future work.

Fig. 3e and Fig. 3f show the results of increasing the weights of some cells in the dirt map. The rectangle in the figure, located in the working area in front of the table, specifies an area where the user increased the weight of the cells to 300%. The changed cleaning path of the corresponding bounded dirt policy is shown in Fig. 3e. Due to the increased weights, more cells in the specified area reach the dirt threshold and are thus visited additionally compared to Fig. 3c. For the bounded time policy (Fig. 3f), the cleaning path is changed such that more cells are visited in the specified area. The restricted time, however, does not allow the robot to pass all previously covered dirty cells, e.g., those in front of the cabinet.

These experiments show that the dirt map estimation performs well and that the policies yield reasonable paths. Additionally, they show that the user adapted weights naturally integrate with the proposed policies.

But the results of the experiments also raise questions for the practical application, e.g., how long does it take until the estimation of the dirt map has converged? How efficient are the proposed policies compared to a full coverage of the environment and how does the structure of the environment influence this efficiency? To answer these questions, we performed a number of simulation experiments.

C. Map Parameter Estimation

To efficiently clean the environment, the robot needs a converged estimate of the parameter λ^c for each cell in the dirt map. This raises the question of how many times the entire area has to be covered until the learning of the dirt map has converged and the difference between the estimated $\hat{\lambda}^c$ for each cell and the real λ^c is acceptable. For this experiment, we simulated the cleaning of a single cell. We repeatedly sampled polluting events from Poisson distributions given different values of λ, cleaned the cell and updated the estimate $\hat{\lambda}$. Fig. 4 visualizes the result for λ values of 3, 10, and 20. For all values, the difference between the real and the estimated value decreases significantly after only a few update steps. As an example, a difference of one between the estimated and the real values is noted as a black line in the figure. Note that in the experiment with the real robot, a difference of one corresponds to a difference of 0.03 g of dirt generation per cycle. As one can see, the number of cycles needed for reaching this value increases with the value to be estimated. However, even for a relatively large value of 20, only 14 steps are needed until the required distance of the estimated to the real λ is reached.

D. Evaluation of Cleaning Policies

To evaluate the cleaning policies and their dependency on the structure of the environment, we simulated different maps with approximately 100 cells. The map size ranges from an environment with 1×100 cells to a quadratic environment with 10×10 cells. The 1×100 environment corresponds to a long narrow corridor and is the worst case for our approach as a single dirty cell can make it necessary to traverse twice the number of cells in the environment. We repeated the experiment 1,000 times for each environment and different maximum dirt levels $d_{\max}$ as well as maximum durations $\tau_{\max}$. For each episode, we sampled a new dirt map, i.e., we sampled a new ground-truth lambda value from an exponential distribution with mean $\mu_e = 3$ independently

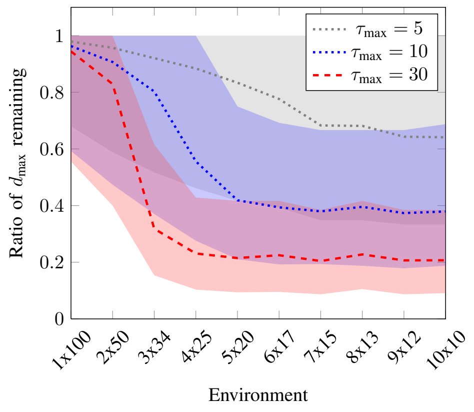

Fig. 6. Evaluation of the bounded time policy. The figure shows the average maximum dirt value remaining in the map after cleaning for different values of $d_{\max}$ and differently structured environments. Also shown are the empirical 95% confidence intervals.

for each cell. We applied the exponential distribution for its similarity to the distributions in real environments. In real environments, it is common that few parts are very dirty more often, e.g., around the breakfast table, while most other areas do not get dirty as regularly. The possible spatial dependency of dirt values in the environment was not considered in the simulation. This is the worst case for our cleaning policies, as spatially correlated dirty cells lead to shorter cleaning paths.

Fig. 5 shows the results for the bounded dirt policy $\pi(d_{\max})$. It plots the number of cells needed to be traversed by our cleaning policy for the different types of environments. This includes cells that were not part of the set selected for cleaning but were traversed to reach selected cells. Also included are cells covered on the way from and to the start position. That means that cleaning the entire 1×100 cell environment would yield a traversal cost of 200 cells while cleaning the other environments requires traversing every cell in the environment, but at most the number of cells in the environment plus two (for the 7x15 environment). One can see in the figure that for all values of $d_{\max}$, the number of traversed cells decreases as the environment becomes less stretched. Already starting with the 3×34 scenario, our policy traverses significantly less cells than the full coverage path, which has to visit all cells.

To evaluate the bounded time policy $\pi(\tau_{\max})$, we consider the maximum dirt value in the map is after cleaning for a given period of time. For this experiment, we sampled λ^c values and actual dirt values from the Poisson distributions given λ^c for each cell. We then calculated a cleaning path given the dirt expectation. Fig. 6 shows the maximum dirt value in the map after cleaning as a ratio of the maximum dirt value in the map before cleaning. As expected, the more time the user allows for cleaning, the lower the maximum dirt value is after cleaning. More importantly, one can see that the efficiency does depend on the environment but that the bounded time policy effectively reduces the maximum dirt value even in demandingly structured environments.

These experiments show that the efficiency of our cleaning policies depends on the structure of the environment. Nevertheless, even for environments that are far from beneficial for our approach, a significant reduction in cleaning time and maximal remaining dirt can be achieved.

VI. Conclusion

In this paper, we presented a novel approach using Poisson processes for representing, estimating, and predicting the distribution of dirt in an environment. We described how to acquire the parameters of these models and presented two efficient cleaning policies for robotic vacuum cleaners using these models. In extensive simulation experiments, we showed that compared to the state of the art, which is full and systematic coverage, our informed approach leads to significantly shorter cleaning times while guaranteeing a bound on the maximum dirt level. We furthermore demonstrated the practical applicability of our approach in experiments with a real robotic vacuum cleaner. In the future, we plan to extend our approach to integrate the uncertainty in the path execution as well as the uncertainty of the actuator when calculating the path of the robot.

References

[1] H. Choset. Coverage for robotics–a survey of recent results. *Annals of Mathematics and Artificial Intelligence*, 31(1):113–126, 2001.
[2] L.H. Erickson, J. Knuth, J.M. O'Kane, and S.M. LaValle. Probabilistic localization with a blind robot. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2008.
[3] Y. Gabriely and E. Rimon. Spanning-tree based coverage of continuous areas by a mobile robot. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2001.
[4] J.S. Gutmann, E. Eade, P. Fong, and M. Munich. A constant-time algorithm for vector field slam using an exactly sparse extended information filter. In *Proc. of Robotics: Science and Systems (RSS)*, 2010.
[5] K. Helsgaun. An effective implementation of the lin-kernighan traveling salesman heuristic. *European Journal of Operational Research*, 126:106–130, 2000.
[6] W.Y. Jeong and K.M. Lee. Cv-slam: A new ceiling vision-based slam technique. In *Proc. of Robotics: Science and Systems (RSS)*, 2005.
[7] E. Kruse and F.M. Wahl. Camera-based observation of obstacle motions to derive statistical data for mobile robot motion planning. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 1998.
[8] M. Lagoudakis, E. Markakis, D. Kempe, P. Keskinocak, A. Kleywegt, S. Koenig, C. Tovey, A. Meyerson, and S. Jain. Auction-based multi-robot routing. In *Proc. of Robotics: Science and Systems (RSS)*, 2005.
[9] J.C. Latombe. *Robot Motion Planning*. Springer Verlag, 1990.
[10] G. Lawitzky. A navigation system for cleaning robots. *Autonomous Robots*, 9(3):255–260, 2000.
[11] M. Luber, G.D. Tipaldi, and K. Arras. Place-dependent people tracking. In *Proc. of the Intl. Symposium of Robotics Research (ISRR)*, 2009.
[12] R. Mannadiar and I. Rekleitis. Optimal coverage of a known arbitrary environment. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2010.
[13] H. Moravec and A. Elfes. High resolution maps from wide angle sonar. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 1985.
[14] B. Siciliano and O. Khatib, editors. *Handbook of Robotics*. Springer, 2008.
[15] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, 2006.
[16] Y. Zhang, J. Liu, G. Hoffmann, M. Quilling, K. Payne, P. Bose, and A. Zimdars. Real-time indoor mapping for mobile robots with limited sensing. In *Proc. of the 3rd Intl. Workshop on Mobile Entity Localization and Tracking*, 2010.
[17] R. Zlot, A. Stentz, M.B. Dias, and S. Thayer. Multi-robot exploration controlled by a market economy. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2002.