Learning the State Transition Model to Efficiently Clean Surfaces with Mobile Manipulation Robots

Jürgen Hess[1] Jürgen Sturm[2] Wolfram Burgard[1]

***Abstract*— In this paper, we present a novel approach that enables a robot to efficiently learn to clean unknown surfaces. We model the cleaning task as a Markov decision problem (MDP) where the state transition model is unknown and needs to be estimated. Using our method, a robot learns this transition model by observing the outcomes of its actions. At the same time, the robot exploits this learned model to generate paths that favor those parts of the surface that the robot can clean. In experiments carried out with a real mobile manipulation robot, we demonstrate that our approach allows robots to clean surfaces efficiently.**

I. INTRODUCTION

Cleaning services are envisioned to be one of the most relevant applications of mobile service robots in the near future. However, most of the cleaning strategies currently implemented in service robots are not very efficient. For example, the Roomba robot from iRobot cleans a room by random movements. Whereas such an approach guarantees complete coverage as time goes to infinity, it is clearly suboptimal compared to systematic coverage strategies. One typical example of the latter approach is the Mint robot from Evolution Robotics which uses visual markers for global localization and which allow the robot to mostly visit every location exactly once. However, some of these regions may not be dirty and thus do not require cleaning while other spots might even require multiple visits. These limitations can be overcome by methods that enable a robot to observe the outcome of its actions in order to learn a more efficient cleaning strategy that take into account sensor data.

We propose a novel approach that allows a service robot to efficiently clean a surface. It achieves this by learning which features correspond to dirt and which correspond to a clean surface. The robot divides the surface into a grid of cells, and discretizes the color of each cell in the grid into a small set of discrete color classes. The state of all cells together with the position of the tool forms the state space of our approach. In each time step, the robot needs to decide which action to carry out. An action is to move to any location on the surface and to clean it. After each action, the robot observes whether the state of the current cell has changed as a result of cleaning. Given this representation in terms of the states, actions, and observations, we phrase the cleaning task as a Markov decision problem (MDP). To be fully defined, an MDP requires additionally an observation model that maps the input images to (discrete) states and the transition model that describes the effects of the robot's actions on the state variable. In contrast to existing approaches, we do neither require the observation nor the transition model to be specified beforehand, i.e., to know which cells are dirty and which are clean. In this paper, we show how the robot can bootstrap these models autonomously by experimentation.

Fig. 1: Experimental setup. The PR2 cleans a table with a vacuum cleaner. Instead of cleaning the whole table, the robot observes the outcome of its actions, learns which features correspond to dirt, and cleans these regions.

In our concrete scenario, we consider a manipulation robot which is given the task to clean the surface of a table. Our robot detects the table in the point cloud obtained from a tilting laser range finder. After the pose and dimension of the table have been determined, the robot takes a color image of the table and divides the table surface into a grid. Based on the color of each grid cell, the robot assigns a discrete class label to each cell. These class assignments of cells to color classes (together with the position of the robot's end-effector) represent the state of the system. In our experimental setup (see Figure 1), the robot uses a vacuum cleaner that it holds in its end-effector and that it can move across the table. In each time step, the robot can move the end-effector to one of the grid cells. By moving its end-effector over the table and watching for changes in the table state, the robot learns which class labels indicate dirt that can be removed by the robot. By exploiting this knowledge, the robot seeks for the path that maximizes the effects of cleaning. In experiments carried out on a real manipulation robot, we demonstrate

[1] J. Hess and W. Burgard are with the Autonomous Intelligent Systems Lab, Computer Science Department, University of Freiburg, Germany. {hess,burgard}@informatik.uni-freiburg.de

[2] J. Sturm is with the Computer Vision and Pattern Recognition Group, Computer Science Department, Technical University of Munich, Germany. juergen.sturm@in. tum.de

that our approach allows a manipulation robot to clean a table efficiently and, additionally, can verify the success of cleaning.

II. RELATED WORK

For the specific task of fully covering a known surface, the coverage problem can be solved by standard path planning algorithms. In general, the term *coverage path planning* refers to the problem where all nodes in a graph have are connected and where the goal is to find a path that minimizes some cost measure [9].

Most of the approaches for coverage assume that the environment is known and seek the shortest path that traverses each location once. For general environments this problem corresponds to the traveling salesman problem (TSP) . However, finding the optimal solution is well-known to be NP-hard. Therefore, practical solutions use efficient approximations to reduce the problem size. Gabriely et. al [5], for example, decompose the surface into a grid and suggest different coverage strategies based on spanning trees. Other approaches use a decomposition into non-overlapping cells of different shapes. Latombe, for example, uses a trapezoidal decomposition [8]. Another approach is the Boustrophedon cellular decomposition [2] which divides the free space into cells which can be covered with vertical back and forth motions that can be connected across the cells. Huang et al. [7] use this decomposition and compute an optimal coverage path by minimizing the number of turns of the robot. Recently, Mannadiar and Rekleitis [10] prosposed a graph structure based on the Boustrophedon cellular decomposition and showed that a complete minimal path through this graph can be computed in polynomial time. In contrast to these approaches, we are not interested in covering the entire surface. Instead, we seek for the path that maximizes the cleaning performance, i.e., that specifically covers those parts of the surface that the robot can actually clean. Additionally, most previous approaches assume that traversing a cell corresponds to cleaning it – and thus in their current form, neither take advantage of sensor readings during path execution. Our approach, however, also considers areas already covered because the robot continuously observes the state of the surface. Thus, robots using our approach can traverse and clean a cell more than once if necessary.

In the area of surface cleaning with a manipulation robot, Urbanek et al. [14] learn low-level motor control of during wiping. They propose learn the parameters of an oscillator from human demonstrations and demonstrate that their system can learn and reproduce different wiping styles, e.g., zigzag or roundish movements from expert demonstrations. Eppner et al. [3] present an approach that enables a robot to learn generalized task descriptions based on imitation learning. Using their approach, a robot can learn to clean various white boards by imitating a human. However, the robot cannot substantially modify the path during reproduction (such as, adding more way points, or changing their ordering while cleaning) and the robot does not observe the state of the board. In contrast to that, our approach observes the state, reacts on it, and allows the robot to recognize that the surface has been successfully cleaned.

Describing effects of actions on the world state has been seen recent interest in the area of affordance learning. Sahai *et. al* [13] for example, use a robot to estimate which tools are suitable for writing on different materials. In their work, the robot learns that a marker writes on nearly all materials while a PVC tube can only be used to write in a bowl filled with beans and rice. Metta et al. [11] propose an approach that allows a robot to learn a categorization for objects as rollable and unrollable. Griffith et al. [6] also categorize objects based on movement pattern resulting from the interaction with the object and learn a model to generalize the learned categories to new objects. For mobile robot navigation Frank et al. [4] learn models of deformable objects by interacting with them and show that navigation task can be performed more efficiently considering the cost of deformation. Ziebart et al. [15] model path planning as an MDP where they assume that the reward function is a linear combination of the features of the environment. Based on the maximum entropy model, they use Inverse Reinforcement Learning to estimate the corresponding weight for each feature from observed trajectories. This enables them to predict, for example, the chosen trajectories of pedestrians or taxi drivers. Our approach describes the effect of the actions of the robot as the transition model of an MDP and updates the transition model while performing the task.

III. APPROACH

To solve the task of cleaning the surface of a table and to estimate which class labels correspond to dirt, we first formulate the problem as a MDP. Second, we show how the current state of the table can be observed and knowledge about the class labels can be obtained. Third, we show how this information can be used to select the next best action and to generate efficient cleaning paths.

A. Problem Formulation

We model the cleaning task as an MDP. An MDP is generally defined as the tuple

$$(S, A, P(s'|s,a), R(s,a,s')) \tag{1}$$

where S refers to the set of states and A to the set of actions. $P(s'|s,a)$ refers to the transition function for getting to state s' when taking action a in state s and $R(s,a,s')$ to the reward function that assigns a reward to action a that is carried out in state s and that leads to state s'. Applying this definition to the cleaning task, we define the set of states

$$S = \{s_1, \ldots, s_n, x\} \tag{2}$$

where $s_1, \ldots, s_n \in \{1, \ldots, k\}$ encode the state of the individual grid cells of the workspace. Further, $x \in \{1, \ldots, n\}$ encodes the position of the robot's tool on the workspace. This results in $|S| = k^n \times n$ different states, where k refers to the number of classes in which a cell can be in and n to the number of cells. The set of actions A is defined as moving the tool to one of the n cells of the surface and cleaning it, i.e., we

consider $|A| = n$ different actions from which the robot has choose in each time step. The transition function $P(s' \mid s,a)$ consists of two parts: The movement of the tool and the change of the workspace. We assume the movement of the tool to be deterministic. Thus, we always reach the desired position in the workspace, i.e., $x' \leftarrow a$. However, the outcome of the cleaning action on the state of the current cell is not deterministic and initially not known by the robot.

We define the reward function $R(s,a,s')$ such that the robot gets a reward for every change it accomplishes in the table state, i.e., we assume that change in the table states is an indicator of successful cleaning, i.e.,

$$R_{\text{clean}}(s,a,s') = \begin{cases} 1 \text{ if } s_a \neq s'_a \\ 0 \textit{ otherwise} \end{cases} \tag{3}$$

Further, we penalize the distance into account that the robot has to travel to reach cell a given its current position, i.e.,

$$R_{\text{travel}}(s,a,s') = -\left|x-a\right|, \tag{4}$$

Using these two components, we define the reward function as

$$R(s,a,s') = \alpha R_{\text{clean}}(s,a,s') + (1-\alpha) R_{\text{travel}}(s,a,s'), \tag{5}$$

where α is a weighting factor that trades of cleaning versus travel distance. However, the robot cannot know whether cleaning a particular cell will lead to a change, i.e., the successor state s', or, more specifically, its a-th component s'_a corresponding to the state of cell a, is unknown. Therefore, the robot can only compute the expected reward of an action a using its current estimate of the transition function and the noisy (and thus probabilistic) observation of the cell state s_a, i.e.,

$$\begin{aligned} E[R(s,a,s')] = \alpha \sum_{j=1}^{k} P(s'_a \neq j \mid s_a = j, a) P(s_a = j) \\ - (1-\alpha)\left|x-a\right|, \end{aligned} \tag{6}$$

where $P(s_a = j)$ corresponds to the probability observed from the camera image that cell s_a is assigned to color class j. Further, $P(s'_a \neq j \mid s_a = j, a)$ expresses the probability that action a carried out in a cell assigned to color class j leads to a change of this color class, i.e., that cleaning this cell is successful.

Given the reward function, the robot can now compute the expected future reward (or value) of a state s as

$$V(s) = \max_a \left\{ \sum_{s'} P(s' \mid s,a) \left(E[R(s,a,s')] + \gamma V(s') \right) \right\}. \tag{7}$$

Here, γ is the so-called discount factor and $V(s)$ can efficiently be computed using value iteration. Using $V(s)$, the optimal policy (and thus the best action that the robot can choose in state s) is given by

$$\pi(s) := \operatorname*{argmax}_a \left\{ \sum_{s'} P(s' \mid s,a) \left(E[R(s,a,s')] + \gamma V(s') \right) \right\}. \tag{8}$$

To summarize, in each time step, the robot observes the current state s, computes $V(s)$, and executes action $\pi(s)$ given by the optimal policy.

B. Perception

To determine the workspace for our approach, the robot acquires 3D point clouds as well as calibrated colored camera images. The 3D scans are obtained by integrating several 2D scans from a tilting laser over time. In detail, we carry out the following four steps to determine the states of the table cells $s_1, \ldots, s_n$:

1) To detect the table, we extract the most dominant plane which is located in front of the robot and has an normal vector approximately similar to the ground plane. After determining the inliers of this plane, we compute the minimal rectangle enclosing the convex hull of the inliers which we consider to be the surface to be cleaned.
2) We divide this surface in rectangular cells according to the size of the tool used, in our case, rectangles of a side length of $l = 3\,\text{cm}$.
3) According to this grid, we assign the pixels from the color image to grid cells.
4) We discretize the colors of the grid cells into a small set of distinct color classes.

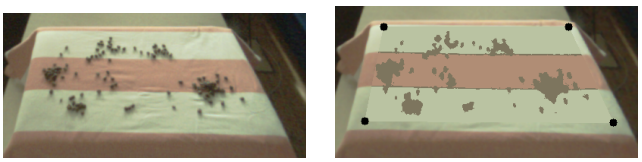
C. Automatic color clustering

We apply unsupervised clustering to learn a suitable mapping from camera images to cell states. This allows the robot to autonomously bootstrap an observation model that is suitable for the current environment of the robot. To achieve this, we use the expectation maximization (EM) algorithm as implemented in OpenCV [1]. In preliminary experiments, we found that the *Lab* color space yielded the best results for our application. Note that k could be selected automatically [12]. However, in our current implementation, we set k manually to a suitable value.

An example of the result of automatic color clustering and table state discretization is shown in Figure 2. The left image shows the original, while the right image shows the result after table detection, segmentation, unsupervised clustering, and discretization. As each cell consists of multiple pixels, we compute the class distribution for each cell, i.e., the probability $P(s_i = j)$ that the i-cell belongs to class j. An example of this class distribution of one specific cell is shown in Figure 3 before and after cleaning. As one can see, the brown components (class 3) were removed by cleaning this cell. Most of these pixels transitioned to the light green class (class 2) and some to the white class (class 1).

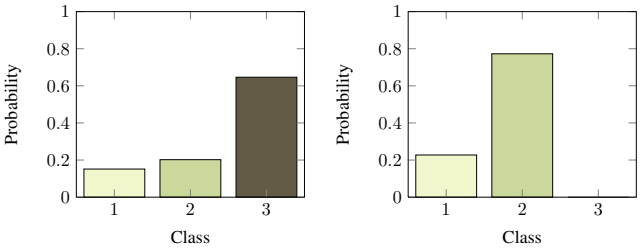
D. Learning the Transition Matrix

For estimating the transition function $P(s'_a \mid s_a, a)$, we use a counting matrix $T \in \mathbb{N}^{k \times k}$ that reflects how often the robot has observed a (pixel-wise) transition between color classes. The elements t_{ij} of T thus correspond to the absolute frequency that a pixel of class i has changed to class j through cleaning. Every time the robot observes a



(a) Original image (b) Discretized image.

Fig. 2: We use unsupervised clustering to discretize the colors on the surface of the table. In this case, we set $k = 3$. The corners of the selected surface is indicated by the four black dots.



(a) Class distribution before cleaning (b) Class distribution after cleaning

Fig. 3: Example of the class distribution of a cell before and after the cleaning action.

cell after moving the tool to this cell, the robot updates T. For each pixel located in the cell that the robot acted upon and observed, we update T by

$$t_{ij} \leftarrow t_{ij} + 1. \tag{9}$$

From this counting matrix, we compute the entries of the probabilistic transition matrix underlying the transition model $P(s'_a \mid s_a, a)$ as follows:

$$P(s'_a = j \mid s_a = i, a) = \frac{t_{ij}}{\sum_{j'=1}^{k} t_{ij'}}. \tag{10}$$

We initialize each element of the counting matrix to a small value to express our prior belief that transitions between all color classes are equally likely.

E. Determining when to Stop Cleaning

Given the current estimate of the transition function, we can compute the expected number of cells that the robot can clean as

$$\lambda = \sum_{a=1}^{n} \sum_{j=1}^{k} P(s_a \neq j \mid s_a = j, a). \tag{11}$$

A possible criterion for stopping to clean the table is to check whether λ drops below some threshold, for example, $\lambda < 1$ would let the robot stop cleaning if it expects that less than one uncleaned cells remain.

IV. EXPERIMENTS

For our experiments, we use the PR2 mobile manipulation robot from Willow Garage. We set up a table and positioned the PR2 such that the entire table could be reached with one arm (see Figure 1). As a tool, we used the top of a vacuum cleaner tube.

The goal of our experiments is to demonstrate that our approach

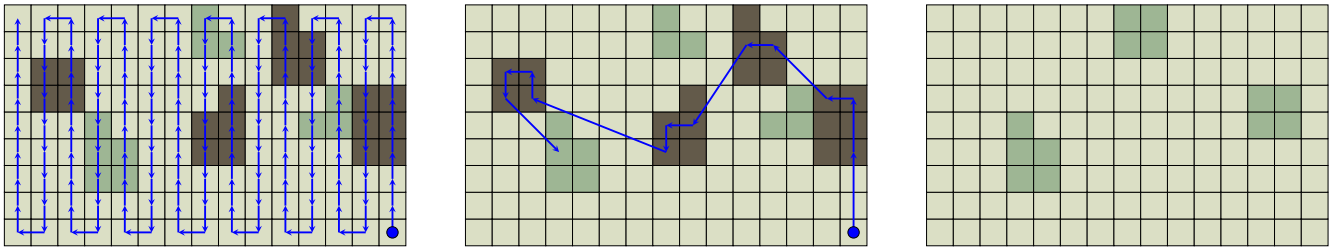
1) applies to different environments,
2) robustly learns the state transition model,
3) significantly increases the speed and efficiency of table cleaning.

We show our first claim by evaluating our approach in two different environments, i.e., on tables with different dirt and cloth colors. In our first environment, the table surface is white but contains a few green patches that cannot be removed by the robot. Furthermore, we added some coffee beans (brown) that the robot could remove using the vacuum cleaner. We set the number of classes $k = 3$ according to the number of distinct features present in this environment. Furthermore, we set the discount factor $\gamma = 0$ which reduces the policy search to a greedy 1-step lookahead problem which can be solved in real-time. In preliminary experiments, we found that this simplification does not signficantly affect the performance of our system. However, if the travel time between different locations on the table played a more significant role, then the order of execution would become more important and thus more steps should be planned ahead.

Figure 4 shows a single run of table cleaning in the first environment. All three figures show the state of the table as it was observed by the robot. To improve readability, we depicted for each cell only the dominating color class per cell.

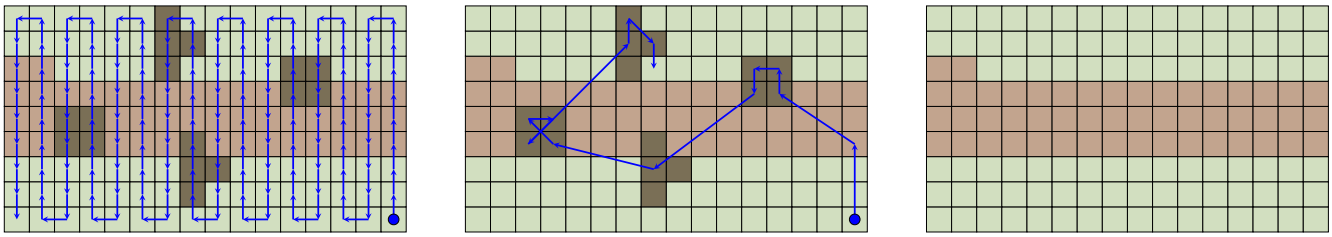
The trajectory planned using the initial transition matrix is shown in Figure 4a. As the expected change in class is initially assumed to be the same for all cells, the difference in reward for cleaning different cells only results from the distance to be traveled. Thus, the path starts at the table cell closest to the initial gripper position which is located in the right bottom corner (indicated by a blue dot). While the robot starts to execute this path, it re-observes the table state and updates the transition matrix. In particular, it detects that it can clean brown cells. As a consequence of this update, it re-plans the cleaning trajectory. Figure 4b shows the actually executed trajectory. It can be seen that this trajectory is significantly shorter than the trajectory that covers the whole table. The final state of table as observed by the robot is depicted in 4c.

The state transition model corresponding to this run is shown in Figure 6a. Note that the values on the diagonal indicate the probability that a cell of this color is not affected by cleaning. In this experiment, the robot estimated the probability of not being able to clean the green (class 1) and white (class 3) cells to 0.89 and 0.77, respectively. The values are not exactly one as a result of our prior assumption and noise in the observations of the robot. In contrast, the robot learned that brown cells (class 2) transition with a high probability (0.71) to white when being cleaned. This experiment shows that the robot correctly recognized that the brown color class (corresponding to the coffee beans) is cleanable and the white and green table cloth is immutable.



(a) Initial path (b) Executed path (c) Final state

Fig. 4: Samples result of the first experiment with parameter α set to 0.5. Left: The initial path consists of 135 steps, and covers all cells as the robot does not know which cells it can actually clean. Middle: During cleaning, the robot realizes that it can clean the brown cells. The green cells are not affected. Right: The state of the table, as observed after the robot stopped. Under the coffee beans on the right, a green patch appears. Furthermore, the robot has to deal with noisy observations of the cell state (some cells change their state unexpectedly).



(a) Initial path (b) Executed path (c) Final state

Fig. 5: Sample result of the second experiment. Note that we only draw an arrow if the tool position changes. In some cases the robot cleans the same cell several times as sometimes a cell does not become clean entirely. It may also happen that the robot cleans parts of other cells. These effects are due to noise in the robot motion and also depend on the type of feature. Coffee beans for examples are cleaned more easily than the small pieces of paper used in this experiment.

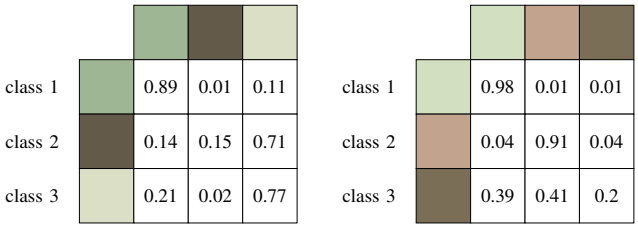


Fig. 6: Learned transition matrix. All rows are normalized to 1. The classes corresponding to each row/column are depicted with their mean color. Rows: class label before cleaning. Column: class label after cleaning.

To demonstrate the robustness and stability of our approach and to show its efficiency in terms of cleaning speed, we repeated this experiment ten times. After every run, we re-set the table back to its initial state. Furthermore, to gather the cleaning statistics, the robot removed its manipulator from the scene to observe the state after each step. Note that this is in general not necessary. In all other experiments, the robot plans the trajectory on our current estimates and delay the observation of the cleaned cells as long as they cannot be observed, e.g., as a result of self-occlusion. Figure 7 shows the results of the dirt versus the number of cleaning actions. After the experiment (and only for the purpose of evaluation), we manually specified that the brown color class corresponded to dirt. The initial plan (corresponding to the full coverage strategy in the absence of the transition matrix) slowly cleans the table, i.e., the amount of dirt slowly reduces. In contrast, the dirt percentage when using our learning approach rapidly reduces and settles at zero within the first 15 time steps. The line corresponds to mean percentage of dirt and the error bars correspond to the doubled standard deviation. This experiment shows that our approach leads to a significant speed up in the number of cleaning steps.

In addition to experiment one, we tried our approach on another environment. For the second experiment, we covered the table with the white and red blanket as shown in Figure 1 and again used coffee beans as dirt. Figure 5 shows the result of one of the runs in this environment. As one can see, the resulting behavior is similar to the first experiment. The robot correctly recognizes that the brown color class is cleanable and thus specifically approaches those cells (see Figure 5c). The learned transition model is depicted in Figure 6b. Note that for the first two classes, there is a very high probability that the class labels are not affected by cleaning. In contrast, class three has a low probability of remaining brown. Compared to the first experiment, the probability of the dirt to not change increased slightly. This

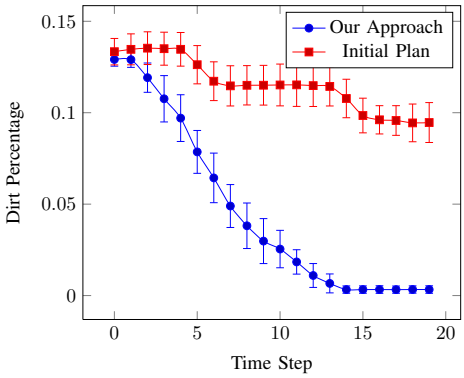


Fig. 7: Experiment: Function of the percentage of dirt still on the table including error bars denoting the double standard deviation. The "initial plan" corresponds to the performance of the initial transition function. The "revised plan" visualizes the path planned using our approach, i.e., by updating the transition function and re-planning at each time step.

result is attributed to the blanket. The coffee beans remained on the blanket a little more often in spite of vacuuming.

V. CONCLUSIONS

In this paper, we presented a novel approach to enable a manipulation robot to learn how to clean surfaces. Our approach phrases the cleaning task as an MDP. The state transition model for this MDP is previously unknown. Our approach learns this transition model by observing the outcomes of the actions carried out by the robot. After each action the robot adapts its path using the updated transition model. In practical experiments, we demonstrated that our approach enables a service robot to improve its performance in a cleaning task by learning how to recognize dirt and to specifically clean only these dirty regions.

Despite these encouraging results, there is still room for improvements. For example, the knowledge about the transition function also enabled us to derive a stopping condition, i.e., to determine that the table has been cleaned sufficiently. In future experiments, it remains to be experimentally validated that the proposed measure is a suitable stopping criteria. Furthermore, the robot could learn different transition matrices for different tools, or/and different environments. In particular, we would like to consider a wet sponge that would enable the robot to clean coffee stains, but not help to clean fluffs of dust. If the number of tools and states in the environment grows, own experimentation might require too much time to find the right tool for the right type of dirt. In this case, the robot could learn (or initialize) the transition function more efficiently by observing a human demonstrator.

REFERENCES

[1] G. Bradski and A. Kaehler. *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly Media, 2008.

[2] H. Choset and P. Pignon. Coverage path planning: The boustrophedon cellular decomposition. In *Int. Conf. on Field and Service Robotics*, 1997.

[3] C. Eppner, J. Sturm, M. Bennewitz, C. Stachniss, and W. Burgard. Imitation learning with generalized task descriptions. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, Kobe, Japan, 2009.

[4] B. Frank, R. Schmedding, C. Stachniss, M. Teschner, and W. Burgard. Learning Deformable Object Models for Mobile Robot Navigation using Depth Cameras and a Manipulation Robot. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2010.

[5] Y. Gabriely and E. Rimon. Spanning-tree based coverage of contin uous areas by a mobile robot. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, volume 2, pages 1927–1933, 2001.

[6] S. Griffith, J. Sinapov, M. Miller, and A. Stoytchev. Toward interactive learning of object categories by a robot: A case study with container and non-container objects. In *Proc. of the Int. Conf. on Development and Learning (ICDL)*, 2009.

[7] W.H. Huang. Optimal line-sweep-based decompositions for coverage algorithms. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, volume 1, pages 27–32, 2006.

[8] J.C. Latombe. *Robot motion planning*. Springer Verlag, 1990.

[9] S.M. LaValle. *Planning Algorithms*. Cambridge University Press, 2006.

[10] R. Mannadiar and I. Rekleitis. Optimal coverage of a known arbitrary environment. In *Proc. of IEEE International Conference on Robotics and Automation (ICRA)*, pages 5525–5530, Anchorage, AK, 2010.

[11] G. Metta and P. Fitzpatrick. Early integration of vision and manipulation. *Adaptive Behavior*, 11(2):109–128, 2003.

[12] D. Pelleg and A.W. Moore. X-means: Extending K-means with efficient estimation of the number of clusters. In *Proc. of the Intl. Conf. on Machine Learning (ICML)*, San Francisco, CA, USA, 2000.

[13] R. Sahai, S. Griffith, and A. Stoytchev. Interactive identification of writing instruments and writable surfaces by a robot. In *Proc. of the Workshop on Mobile Manipulation in Human Environments at the Robotics Science and Systems Conf. (RSS)*, Seattle, WA, 2009.

[14] H. Urbanek, A. Albu-Schäffer, and P. van der Smagt. Learning from demonstration: repetitive movements for autonomous service robotics. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 424–431, 2009.

[15] Brian D. Ziebart, Nathan Ratliff, Garratt Gallagher, Christoph Mertz, Kevin Peterson, J. Andrew Bagnell, Martial Hebert, Anind K. Dey, and Siddhartha Srinivasa. Planning-based prediction for pedestrians. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2009.