Planning Surface Cleaning Tasks by Learning Uncertain Drag Actions Outcomes *

David Martínez, Guillem Alenyà and Carme Torras
Institut de Robòtica i Informàtica Industrial (CSIC-UPC)
Llorens i Artigas 4-6, 08028 Barcelona, Spain

Abstract

A method to perform cleaning tasks is presented where a robot manipulator autonomously grasps a textile and uses different dragging actions to clean a surface. Actions are imprecise, and probabilistic planning is used to select the best sequence of actions. The characterization of such actions is complex because the initial autonomous grasp of the textile introduces differences in the initial conditions that change the efficacy of the robot cleaning actions. We demonstrate that the action outcome probabilities can be learned very fast while the task is being executed, so as to progressively improve robot performance. The learner adds only a little overhead to the system compared to the improvements obtained. Experiments with a real robot show that the most effective plan varies depending on the initial grasp, and that plans become better after only a few learning iterations.

1 Introduction

Robotized household environments are a promising field where action planning can be useful. Typically to solve a task, a set of perceptions and a set of actions are defined, and a planner is used to choose the sequence of actions to execute. In this paper we put the attention on repetitive actions that are difficult to model offline because some initial conditions can change their outcomes. We use as an example the task of cleaning surfaces with a robot using an autonomously grasped textile, where the initial grasping of the textile clearly changes the effectiveness of the actions.

The actions used to clean are represented with rules which, given some preconditions on the state, define the expected behaviour of an action as a set of possible outcomes with probabilities associated. To obtain the best results we should use rules that accurately characterize any available action in the current environment.

As initial conditions may introduce large variability, we incorporate a learning system that improves the rules after the execution of each action, leading to better plans in the long term. Common learning methods usually need many executions until good rules are obtained, but we aim to finish the task quite fast after a few action executions. We propose starting with very simple handcrafted rules and update them with a new heuristic based on the m-estimate (Cestnik 1990) to get more accurate rules that will perform quite well after a few executions. This is intended for the initial steps until enough experience is obtained to apply more complex methods that can refine the rules with more details. The criteria of the *number of actions executed* and *time to complete the whole task* will be used to measure the success of the method. A more simple approach, like a reactive action execution, will not provide a proper solution as the combination of different actions cannot be taken into account.

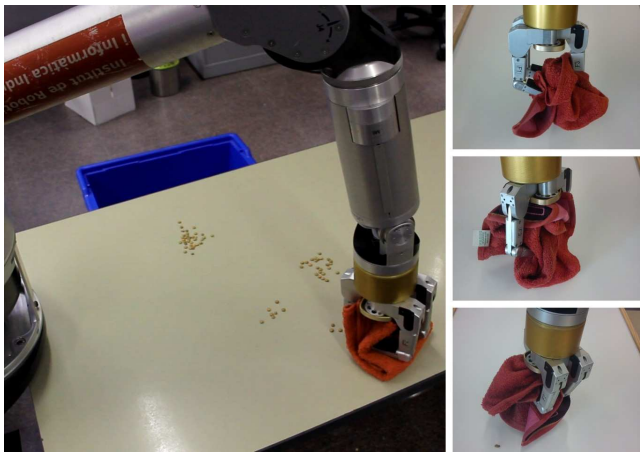


Figure 1: On the left the WAM robot arm is cleaning lentils from a table to a container. On the right three different grasps of the textile used for cleaning, each one leading to different behaviours of the cleaning actions.

The presented approach uses a WAM robot arm to perform the actions, and a Kinect camera to get a representation of the state, which is assumed to be completely observable. Figure 1 shows the environment used for the task, and different grasping configurations. As the actions are stochastic, an online probabilistic planner is used. As mentioned, a learner updates the planning rules to adapt to the grasped textile. The performance of the system depends on the quality of the rules, so learning will be critical to get the best results.

*This work was supported by the Spanish Ministry of Science and Innovation under project PAU+ DPI2011-27510, by EU Project IntellAct FP7-ICT2009-6-269959 and by the Catalan Research Commission through SGR-00155.

This paper is structured as follows. Section 2 presents some related work and where our idea fits in state of the art on automatic learning. The proposed algorithm is introduced in Sec. 3, where perceptions, actions and planning are presented. Section 4 explains the details of the learning procedure. Section 5 shows some experiments of cleaning a surface and the improvements attained when using our approach. Finally, Sec. 6 is devoted to draw some conclusions and future work.

2 Previous work

There are a few recent works in which a robot performs similar tasks to the one presented in this paper. In (Kormushev et al. 2011) and (Sato et al. 2011) robot skills to clean a whiteboard are presented. The robot is trained using imitation learning with hybrid position/force control to learn and execute trajectories trying to maintain the force of the hand against the whiteboard. Force feedback has also been used to learn dynamic motion primitives that ensure that the robot maintains contact and applies the desired force in tasks such as wiping a table (Gams et al. 2010). Moreover, methodologies to sequence motion primitives have been proposed (Nemec and Ude 2012). The surface cleaning strategy is fixed and thus the robot is unable to adapt to different layouts of dirt that could be cleaned more efficiently with simpler trajectories.

For this, a perception system is necessary to acquire the scene state, and actions should be selected accordingly. Then a model-based planner can generate sequences of actions to clean efficiently all kinds of dirt. As we have stochastic actions a probabilistic planner is needed. For large state spaces, a very common planning technique is UCT (Kocsis and Szepesvári 2006), which uses bandit ideas to guide a Monte-Carlo planner. The algorithm finds near-optimal solutions in finite-horizon or discounted MDPs. PRADA (Lang and Toussaint 2010) is a probabilistic planner that handles the uncertainty by converting the rules into a dynamic Bayesian network for state representation, and predicts the effects of action sequences by using an approximate inference method to efficiently propagate beliefs. PRADA has a better performance than classic UCT, so it will be the planner used.

Creating models manually is tedious as it has to be repeated for every task and environment, but learning methods exist to generate models based on experience. Two different approaches can tackle the problem of learning models. The first one is reinforcement learning (RL). Model-based bayesian RL that aims to obtain optimal behaviour as it chooses actions that maximize the expected reward as a function of the belief-state. However, this optimization problem is intractable so near optimal solutions have been proposed (Asmuth and Littman 2011).

The second approach is learning actions models. Using the accumulated knowledge of all executions of an action, a set of rules defining the model is generated. These rules define relational worlds which allow to generalize much better over different states, as the same rule may apply to several similar objects. Methods for learning stochastic actions (Pasula, Zettlemoyer, and Kaelbling 2007) and partially observable domains (Mourao, Petrick, and Steedman 2010) are available. These approaches lack an exploration-exploitation behaviour, and they require several samples for each action before they can get useful models.

(Lang, Toussaint, and Kersting 2010) propose a solution to this exploration-exploitation problem using a strategy based on the Explicit Explore or Exploit E^3 (Kearns and Singh 2002) algorithm and updating the rules with Pasula's algorithm. Although in the end good results are obtained, the algorithm just explores and uses vague rules until enough experience is acquired, getting bad results during the initial executions.

As executing actions in real robots has a large cost, we like our robot to perform as well as possible also during the first executions until enough experience is obtained to learn the action models. We propose using a very simple set of initial rules which can adapt with fast learning heuristics until enough samples are obtained to apply action learning algorithms. The initial rules will begin with very optimistic outcomes, getting the advantages of the optimism under uncertainty bias (Brafman and Tennenholtz 2003).

The rules are provided to the learner which has to update their probabilities online to improve the estimated outcomes of the actions. Several heuristics have been proposed and their performances have been compared (Janssen and Fürnkranz 2010). The family of parametrized heuristics, and in particular the m-estimate (Cestnik 1990), allows to adjust the trade-off between learned estimations and a priori probabilities. Similar to the work of (Agostini, Torras, and Wörgötter 2011), we want to produce confident estimates with a few examples by regulating the influence of the m value, but having stochastic actions we can't know a priori the number of actions needed to cover all possibilities. Therefore we propose a new heuristic that decreases the m value as experiments are carried out so as to adapt quickly the rule outcomes, but the decrease of m is gradually slowed down so that a small influence of the a priori probability is maintained for a long time to take into account unexperienced outcomes.

3 Proposed Method

The method proposed in this paper is aimed at cleaning a surface using a calibrated RGB-D camera and robot arm grasping a cloth.

Observations are continuously acquired with the camera and processed to have an updated representation of the environment. The robot has a set of actions consisting of sequences of movements to clean or displace dirt. Given a representation of the environment and a set of rules defining the available actions, a planner chooses a sequence of actions to clean the surface efficiently. An action is then performed by the robot, and once it is completed, the learner analyzes the changes in the state to update the rules of the executed action accordingly. The system keeps replanning, executing actions and learning if necessary until the task is complete.

Actions

A set of actions is designed to clean a surface containing small objects like lentils. These actions are parametrized

(a) Move to container (b) Join 2 groups (c) Group scattered

Figure 2: Cleaning actions.

with ellipses representing the dirty areas on the scene, and generate a sequence of points defining the cleaning movements. Lentils are detected with a simple RGB segmentation algorithm. The cleaning tool is always oriented perpendicular to the direction of motion to get effective moves.

Actions can be divided in two groups.

- **Cleaning actions** that remove dirt from the surface, moving it towards a container positioned near the edge of the surface (Fig. 2a).
  - `Fast move to container (ellipse)`:
    - Pushes the lentils in a dirty area to the container position using a grasped cloth.
  - `Straight move to container (ellipse)`:
    - It is equivalent to *Fast move to container*, but ensures that the trajectory is straight at the cost of being a little slower.
  - `Short move to container (ellipse)`:
    - It is equivalent to *Fast move to container*, but only does a short movement towards the container, ensuring that the trajectory is straight, although it won't reach the container if the dirt is far from it.
- **Grouping actions** that rearrange the dirty areas on the surface, so that they become easier to clean by means of future actions.
  - `Join 2 Groups(ellipse1, ellipse2)`:
    - The movement pushes the lentils of ellipse1 to ellipse2 joining them (Fig. 2b).
  - `Join 3 Groups(ellipse1, ellipse2, ellipse3)`:
    - Moves ellipse1 and ellipse2 to the position of ellipse3.
  - `Grouping scattered dirt(ellipse)`:
    - Moves scattered groups together to get compact groups that are more manageable (Fig. 2c).

These actions are stochastic due to several factors:

- Actions rely on the accuracy of the depth information provided by RGB-D cameras, which may have some errors. Although using a cloth provides some compliance, sometimes actions may fail to move the dirt as expected.
- The same action may get different outcomes for similar dirty areas. For example, some dirt may spread during the trajectory in some cases, while they may move successfully in other similar cases.
- The cloths used for cleaning produce different results depending on the way they are grasped.

Action:
straightMoveToContainer(X)
Preconditions:
dirt(X), mediumSize(X), ¬scattered(X)
Outcomes (Success probability: predicate changes):
0.4: ¬dirt(X), clean(X)
0.3: ¬mediumSize(X), smallSize(X)
0.2: ¬mediumSize(X), smallSize(X) scattered(X)
0.1: noise

Figure 3: Rule example for removing lentils.

Planning

The planner selects the set of actions to execute based on the perceptions and the probabilistic effects of action sequences. The task is quite complex, and selecting the fastest action sequence is challenging. For example, plans beginning with *grouping* actions may *penalize* in the beginning (remove no dust) compared to *cleaning* actions, but they can provide the best results in the long run.

Using a probabilistic planner is important when actions have several possible outcomes with different probabilities. Deterministic planners (Little and Thibaux 2007) only consider the most probable outcome for each action, while a probabilistic planner takes into account all outcomes.

The planner takes as input the state representing the scene and a set of rules defining the expected results of the actions, and it outputs a plan consisting in a sequence of actions. These actions will be converted into motions that the robot will perform to clean the dirty areas.

State representing the scene: The state s is defined as

$$s = (d_1, ..., d_n, near(d_i, d_j), ..., near(d_k, d_l)) \quad (1)$$

where d_i are the dirty areas represented by ellipses and $near(d_i, d_j)$ indicates dirty areas whose positions are close to each other.

Each dirty area is defined as $d_n = (Id, s, \sigma)$

- Id: Identifier.
- s: Size, where $s \in \{big, medium, small\}$.
- σ: Scattered, where $\sigma \in \{true, false\}$ accounts for compact or scattered distributions.

Action rules: We are using noisy indeterministic deictic (NID) rules (Pasula, Zettlemoyer, and Kaelbling 2007), where each outcome has associated the list of predicates that change when the rule is applied. There may be several rules with different preconditions for every action, and several outcomes for every rule. An example is shown in Fig 3.

4 Learning

The planner needs a set of rules defining the actions that may be performed, and the quality of the plans will depend on the precision of these rules. As our environment is stochastic, it is difficult to define accurate rules for every surface, type of dust, robot and grasping of cleaning tools. To solve this problem, we will learn the actual outcome probabilities for each configuration while performing the cleaning task with

initially inaccurate rules. The learner updates the expected rule outcomes for every action that is executed to reflect the result of the execution. Also, it saves a record of previously executed actions and their results to get better estimates of the outcomes.

Having stochastic actions means that the rule defining the action may have several outcomes for just one set of preconditions. After executing an action that we want to learn, the robot will have to take a new perception and look carefully for differences in the state to know which outcome was obtained. When no outcome matches the result of the action, the noise outcome probability will be increased.

Learning heuristics

We want the system to rapidly refine the outcomes to adapt to the new environment, but we also want to avoid wrong premature estimations to degrade the performance of the system, as it is learning at the same time that it is solving the task.

A learning heuristic to prevent these premature wrong estimations is the m-estimate (Cestnik 1990), that includes a parameter m to implement a trade-off between learned outcomes and a priori probabilities in rule outcomes

$$P = \frac{p + mP_0}{p + n + m}, \tag{2}$$

where P is the estimated probability, P_0 the a priori probability, p the number of positive examples and n the number of negative examples.

The problem with this heuristic is that small values of m may yield wrong estimates of rule outcomes, while a high value of m would entail the system taking too much time to converge to the learned estimates. We propose to use a different heuristic:

$$P = \frac{p + (m/\sqrt{p+n})P_0}{p + n + (m/\sqrt{p+n})}. \tag{3}$$

This decreasing-m-estimate is similar to the m-estimate when there are only a few examples, favouring a priori probabilities. But as the number of examples increases, its influence decreases, leading to better estimates that have little influence from a priori probabilities. The value of m should depend on the stochasticity of the task. In our experiments a value of 10 provided good estimates and was low enough to converge fast to the learned estimates.

Stop learning: Learning adds some overhead to the system. After executing an action to be learned, the arm has to leave the visual field of the camera to get a good perception of the surface and estimate correctly the outcome obtained. Otherwise, planning would rely on partial perceptions that may have occlusions.

Therefore, we only learn actions until we have enough examples to consider that the learned estimate is quite accurate. Using the Hoeffding inequality, we can have a bound for having a high probability $(1 - \delta)$ that our estimate $\widehat{p}$ is accurate enough $|\widehat{p} - p| \leq \epsilon$. The number of trials required is

$$T \geq \frac{1}{2\epsilon^2} \ln \frac{2}{\delta}. \tag{4}$$

Initial rules

The initial rules were written manually, defining only the basic expected behaviours of the actions. As the proposed learner is intended only for the first executions, little effort should be spent on these initial rules, as they will be changed later to more precise ones when enough experience is obtained. Based on the optimism under uncertainty bias (Brafman and Tennenholtz 2003), all rules are defined with a probability 1.0 of getting the best outcome, and as actions get executed, this probability tends to the actual one.

Another possible set of optimistic rules is learning the probabilities of outcomes for the best scenario, which in our case is a very good cloth grasp, and use them as the initial rules. These rules will converge faster to the actual ones as they already have learned some of the dynamics of the system and their outcome probabilities will be closer to the actual current ones.

5 Experimental results

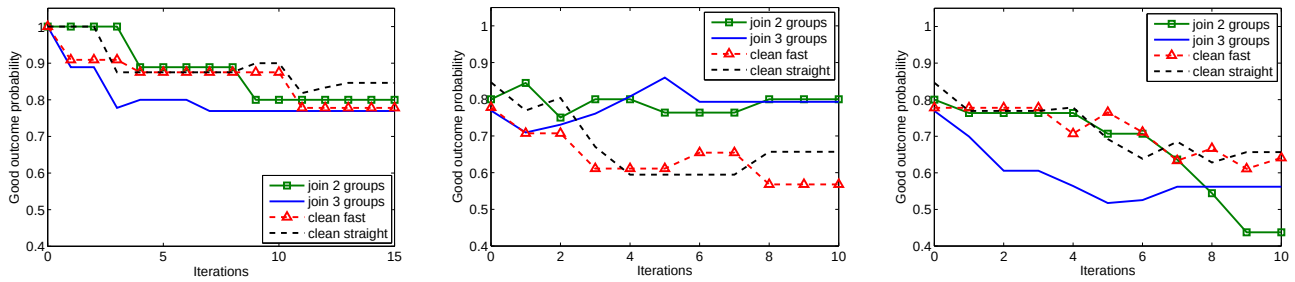
We have carried out two experiments to analyze the learner performance in the task of surface cleaning. Both experiments involved cleaning a surface with 30 lentils spread over it. The robot had to move the lentils to a container positioned near an edge of the surface. The task was repeated a number of iterations in each experiment to analyze the learning process. The value of m was set to 10 and T to 12 in both experiments.

First experiment: rules evolution over time. We analyzed the estimated probabilities of the rules over time (Fig 4). For clarity, only 4 rules are shown. Figure 4a shows the learning process starting with optimistic rules with a 1.0 probability for the best outcome, and along 15 iterations. Then the resulting rules were used in new situations with two different graspings (Figs. 4b and 4c).

As can be observed, these two grasps produce different rules. The grasp of Fig. 4b yields very good results with *join* actions, while the grasp in Fig. 4c gets very bad results with them, particularly the one joining two groups. Also the *clean fast* action has significant differences between both grasps.

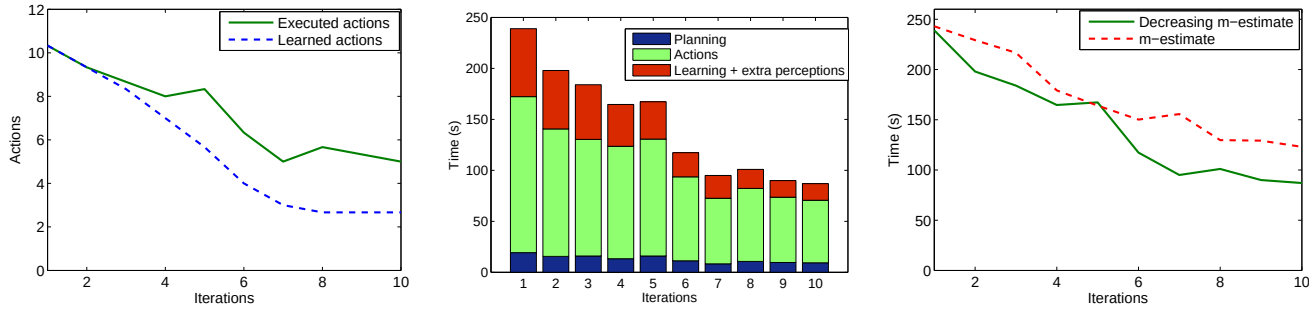
Second experiment: improvement using learning. Using the rules obtained during the previous experiment, we measured the number of actions and time taken to clean with new grasps. The experiment was repeated 4 times and the average of the results is shown in Fig. 5. As can be seen, the number of actions required to complete the tasks decrease as the rules improve. Also, the learner stops refining actions once it gets enough examples of them, reducing the learning time after a few iterations.

Moreover, the same experiment was repeated using the original m-estimate to compare its performance with our proposal. Although the m-estimate also improves the rules, the decreasing m-estimate obtains better results with fewer iterations as shown in Fig. 5c.



(a) Initial rule learning. (b) Refining rules from (a) with a new grasp. (c) Refining rules from (a) with another grasp.

Figure 4: Rule learning over time with m=10. In (a) initial rules are obtained with a common grasp. In (b) and (c) rules from (a) are refined with new grasps. Observe that each particular grasp changes the outcome probabilities of the different rules, e.g. join 2 groups performs well in (b) but bad in (c).



(a) (b) (c)

Figure 5: Improvements using the learner. **(a)** Number of actions executed and the number of them that required learning as they were considered unknown. **(b)** Distribution of time between planning, action execution and learning. **(c)** Time taken to clean the board using the proposed decreasing m-estimate and the original m-estimate.

6 Conclusions

In this work we have shown the use of a learner integrated in a surface cleaning system where a planner is used to choose good sequences of actions to clean efficiently with very little experience. Different grasps of the cloth vary significantly the rule outcomes probabilities, which makes the learner a very important piece to get accurate rules for the planner. As seen in the experiments, good rules improve the plans obtained, which allow the system to clean faster.

The learner produces quite accurate rules after a few executions using the decreasing m-estimate heuristic. It also adds a little overhead to the system, which almost disappears after a few executions when most used rules get already learned. Overall we can conclude that the inclusion of a learner for rule refinement is highly recommendable in dynamic environments where accurate rules are not available. Once enough experience is obtained, more complex methods for refining the rules preconditions and outcomes (Pasula, Zettlemoyer, and Kaelbling 2007) can be used to get more accurate rules.

A future improvement to the system would be integrating the initial learning heuristic with the more complex action model learner to incrementally update the rules preconditions and outcomes, thus getting the best of using both learning methods simultaneously.

References

Agostini, A.; Torras, C.; and Wörgötter, F. 2011. Integrating task planning and interactive learning for robots to work in human environments. In *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2386–2391.

Asmuth, J., and Littman, M. L. 2011. Learning is planning: near bayes-optimal reinforcement learning via monte-carlo tree search. In *Proc. of Conference on Uncertainty in Artificial Intelligence (UAI)*, 19–26.

Brafman, R. I., and Tennenholtz, M. 2003. R-max-a general polynomial time algorithm for near-optimal reinforcement learning. *The Journal of Machine Learning Research* 3:213–231.

Cestnik, B. 1990. Estimating probabilities: A crucial task in machine learning. In *Proc. of European Conference on Artificial Intelligence*, 147–149.

Gams, A.; Do, M.; Ude, A.; Asfour, T.; and Dillmann, R. 2010. On-line periodic movement and force-profile learning

for adaptation to new surfaces. In *Proc. of the IEEE-RAS Int. Conf. on Humanoid Robots*, 560–565.

Janssen, F., and Fürnkranz, J. 2010. On the quest for optimal rule learning heuristics. *Machine Learning* 78(3):343–379.

Kearns, M., and Singh, S. 2002. Near-optimal reinforcement learning in polynomial time. *Machine Learning* 49(2):209–232.

Kocsis, L., and Szepesvári, C. 2006. Bandit based Monte-Carlo planning. In *Proc. of the European Conference on Machine Learning*, 282–293.

Kormushev, P.; Nenchev, D. N.; Calinon, S.; and Caldwell, D. G. 2011. Upper-body kinesthetic teaching of a free-standing humanoid robot. In *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, 3970–3975.

Lang, T., and Toussaint, M. 2010. Planning with noisy probabilistic relational rules. *Journal of Artificial Intelligence Research* 39:1–49.

Lang, T.; Toussaint, M.; and Kersting, K. 2010. Exploration in relational worlds. *Machine Learning and Knowledge Discovery in Databases* 178–194.

Little, I., and Thibaux, S. 2007. Probabilistic planning vs replanning. In *Proceedings of the ICAPS07 Workshop on the International Planning Competition: Past, Present and Future*.

Mourao, K.; Petrick, R. P.; and Steedman, M. 2010. Learning action effects in partially observable domains. In *Proc. of European Conference on Artificial Intelligence (ECAI)*, 973–974.

Nemec, B., and Ude, A. 2012. Action sequencing using dynamic movement primitives. *Robotica* 30(5):837.

Pasula, H. M.; Zettlemoyer, L. S.; and Kaelbling, L. P. 2007. Learning symbolic models of stochastic domains. *Journal of Artificial Intelligence Research* 29(1):309–352.

Sato, F.; Nishii, T.; Takahashi, J.; Yoshida, Y.; Mitsuhashi, M.; and Nenchev, D. 2011. Experimental evaluation of a trajectory/force tracking controller for a humanoid robot cleaning a vertical surface. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 3179–3184.