Learning and Performing Place-based Mobile Manipulation

Freek Stulp, Andreas Fedrizzi, Michael Beetz

Intelligent Autonomous Systems Group, Technische Universität München, Germany

{stulp|fedrizza|beetz}@cs.tum.edu

Abstract—**What it means for an object to be 'within reach' depends very much on the morphology and skills of a robot. In this paper, we enable a mobile manipulation robot to learn a concept of PLACE from which successful manipulation is possible through trial-and-error interaction with the environment. Due to this developmental approach, PLACE is very much grounded in observed experience, and takes the hardware and skills of the robot into account. During task-execution, this model is used to determine optimal grasp places in a least-commitment approach. This PLACE takes into account uncertainties in both robot and target object positions, and leads to more robust behavior.**

I. INTRODUCTION

Consider the task of approaching a table in order to grasp a cup, as depicted in Fig. 1. A trivial approach to solving this task is simply going to a position such that the target of manipulation is well in reach. However, a more careful look at the question raises some serious issues: What is a good place in the context of an intended manipulation action? Does well-in-reach always imply that the target object can really be reached, given the hardware and control software of the robot? Can we have a least-commitment realization of 'places' such that the robot can refine a 'place' as it learns more about the context (e.g. the clutteredness) of the surroundings? How can such a concept of 'place' take into account uncertainties about the robot's self-localization and estimated target object position?

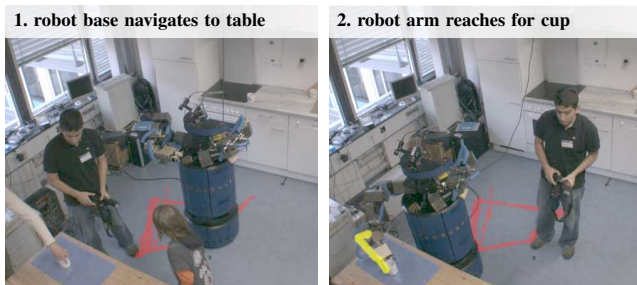**1. robot base navigates to table** **2. robot arm reaches for cup**

Fig. 1. A reach and grasp trajectory performed during a public demonstration. (Note that the operator is holding a camera, not a remote control!)

Manually designing an explicit model that takes all these factors into account is tedious and error-prone. An alternative to explicit modeling is advocated in a recent roadmap paper for manipulation [9]: "it seems almost inevitable that learning will play an important role in robot manipulation". For our mobile manipulation task, we apply learning to the problem of selecting appropriate places to navigate to *in order* to perform a subsequent manipulation task. The main motivation behind this work is that carefully considering and selecting the place from where the manipulation action starts will lead to a simplified manipulation task. This simplification allows the use of a set of standard solutions, such as 'motor primitives', to perform the manipulation. Even if motion planners are able, by conducting a costly search in the state space, to perform manipulation in very complex scenes, we believe it is preferable to avoid and forestall complex tasks if possible. This leads to more robust and natural behavior.

Although much research is done on navigation and manipulation planning as individual topics, the interaction between these areas is not covered as well. The difficulty is, as the many questions in the first paragraph imply, that the coupling between navigation and manipulation depends on many factors, which is why there is no general analytic solution. We believe that the robot should rather develop a concept of place 1) with respect to its own capabilities, which are limited by the hardware and control programs 2) autonomously through learning, from interactions with the environment.
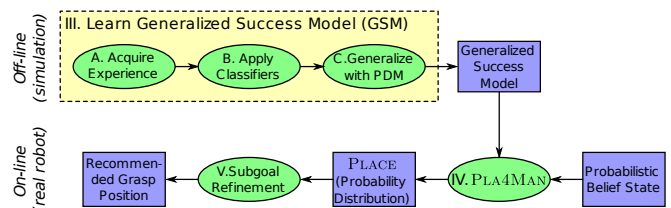

Fig. 2. Computational model (Numerals refer to sections).

As depicted in the computational model of our approach in Fig. 2, the robot learns its concept of place by first gathering experience in simulation, by recording successful and failed attempts at manipulation from different positions. By applying Support Vector Machines, we then acquire classification boundaries between successes and fails for different scenarios, i.e. positions of the target object on the table.

The main novelty of our approach is computing *generalized success models* by generalizing over classifications for specific situations, with a Point Distribution Model (PDM). We relate the internal parameters of the PDM to external parameters that are related to the task, such as the position of the target object on the table. By grounding these generalized success models (GSM) in observed experience, explicit

modeling is no longer required.

Then on-line during task execution, the real robot queries these *general* success models with the *specific* current situation, to determine appropriate positions for starting a manipulation task. This is done using a Monte-Carlo simulation, which yields the PLACE, which is modeled by a probability distribution that models manipulation success, as in Fig. 3.

This concept of PLACE 1) takes into account the uncertainty in the pose of the robot and of the target object; 2) models 'place' as an area of locations with different utilities, instead of committing to a specific position; 3) Takes into account all relevant aspects of the robot platform

Fig. 3. Probability of successful manipulation, given the robot's position at the table. This is the PLACE for this particular task and robot.

and the interactions between its skills, as it is grounded in *observed* experience, rather than explicit hand-coded models.

The developmental methods for acquiring and representing PLACE with generalized success models as the main contributions of this paper[1]. Furthermore, we implement and evaluate our approach on a mobile manipulation robot.

The rest of this paper is structured as follows. In the next section, we discuss related work. We present our approach for learning the generalized success model (GSM) in Section III. We then describe how the (GSM) is used to compute and optimize PLACEs in Section IV and V respectively. In Section VI we present an empirical evaluation of the system, and we conclude with Section VII.

II. RELATED WORK

Using a heuristics-driven search in task space has proven to be a very effective approach to plan motion, even in complex cluttered scenes [2]. However, if everyday situations are encountered very frequently, and having a set of standard solutions like skills or motor primitives for these 'standard' situations is more effective than treating each repeated task as a novel task requiring search. As humans use standard motion primitives so frequently, they can be optimized over time, which leads to stereotypical human motion, and improves the predictability of motions. We believe that these are desirable properties of robot behavior as well. If more complex, novel situations do happen to arise, a standard solution will not suffice, and motion planning algorithms can be used to perform a search to find a solution for this novel situation. The two approaches complement each other well.

Kuipers et al. [11] present a bootstrapping approach that enables robots to develop high level ontologies from low

level sensor data including distinctive states, places, objects, and actions. These high level states are used to choose trajectory-following control laws to move from one distinctive state to another. Our approach is exactly the other way around: given the manipulation and navigation skills of the robot (which are far too high-dimensional to learn with trajectory-following control laws), learn places from which these skills (e.g. grasping) can be executed successfully. Our focus is on action and affordance, not recognition and localization. For us, place means 'a cluster of locations from which I can execute my (grasping) skill succesfully', whereas for Kuipers et al. it rather refers to a location that is perceptually distinct from others, and can therefore be well-recognized. Furthermore, their work has not yet considered the physical manipulation of objects, and how this relates to place.

Capability maps are an alternative approach to modelling robot configurations that lead to successful grasping [16]. These maps are generated by separating the workspace into discrete regions and trying to solve multiple inverse kinematics queries for every region. As only kinematic aspects are considered, capability maps do not take the actual skills of a robot into account, and a accurate kinematic model specific to each robot must be designed by hand. Our approach explicitly takes the motion system and robot skills into account and optimizes the initial position so that subsequent manipulation is facilitated.

Learning success models can be considered as a type of pre-condition learning. Most research on learning pre-conditions focusses on learning symbolic predicates from symbolic examples [4]. These approaches have not been applied to robots, as the representations used do not suffice to encapsulate the complex conditions that arise from robot dynamics and action parameterizations. In robotics, the focus in pre-condition learning is therefore rather on grounding pre-conditions in robot experience. For instance, 'Dexter' learns sequences of manipulation skills such as searching and then grasping an object [7]. Declarative knowledge such as the length of its arm is learned from experience. Learning success models has also been done in the context of robotic soccer, for instance learning the success rate of passing [3], or approaching the ball [14]. Our methods extend these approaches by explicitly representing the region in which successful instances were observed, and computing generalized success models from these regions.

III. LEARNING A GENERALIZED SUCCESS MODEL

In this section, we describe the implementation of the off-line phase of the computational model, depicted in Fig. 2.

A. Acquiring Training Data

The robot first gathers training data by repeatedly executing a navigate-reach-grasp action sequence. To acquire sufficient data in little time, we perform the training experiments in the simulator described in the Appendix. The robot is modeled accurately, and the simulator thus provides training data that is also valid for the real robot.
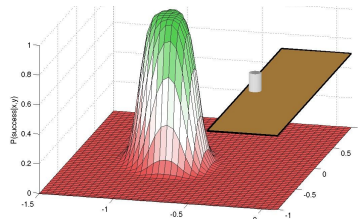
---

[1]The goal of this paper is to apply developmental learning to acquiring a concept of place, as such a concept is very difficult to implement with engineering approaches alone. It is not explicitly our goal to model neurophysiological concepts such as 'place cells' [5].

The action sequence is executed for a variety of task-relevant parameters. In our scenario we tried to grasp a cup and the task-relevant parameters were the x, y position of the cup on a table. The 12 cup positions on the table with which the robot is trained are depicted in Fig. 5. For each cup position, the action sequence was executed 350 times. The initial position for reaching and grasping was randomly sampled, and the result whether the robot was able to grasp the cup or not was stored in a log-file.
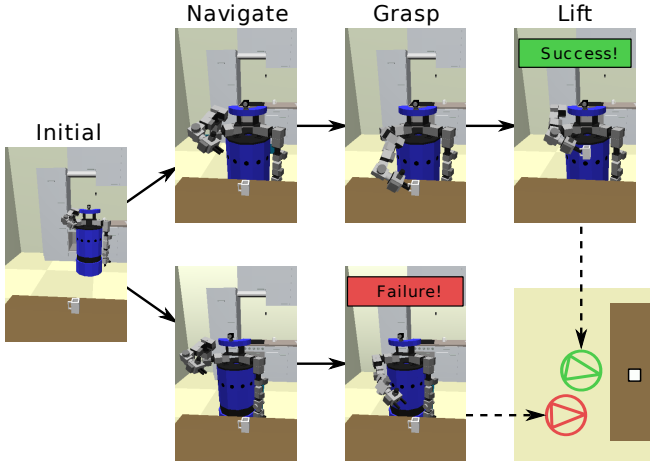

Fig. 4. Two experiment runs with different samples for the robot position. The navigate-reach-grasp sequence in the upper row succeeds. It fails in the lower sequence because the robot is too far away from the cup.

B. Computing Classification Boundaries

To acquire success models, we compute a classification boundary around the successful samples using Support Vector Machines (SVM), using the implementation by [13]. We used a Gaussian kernel with σ=0.03, and cost parameter C=20.0. Fig. 5 depicts the resulting classification boundaries for different configurations of task-relevant parameters[2]. The models on average classify 5% of examples wrongly when using a training/test set that contain 66%/33% of the data respectively, and 3% when using the training data as the test data.

C. Computing the Point Distribution Model

In the next step, we compile all classification boundaries into a generalized compact representation using a Point Distribution Model (PDM), which is a well established method in the field of face recognition [15]. As input a PDM requires n points as input that are distributed over the contour. We distribute 20 points equidistantly over each boundary, and determine the correspondence between points on different

[2]From the robot's point of view the data and the clusters are shifted a bit more to the right than we would have expected when grasping the cup with the right arm. This is due to the hard-ware and kinematics of the robot manipulator, which are not very human-like. This effect supports our experience-based learning over hand-coding, as our intuitions about a good 'place' for robot manipulation apparently do not always correspond to the 'place' that is really the best for a particular robot.
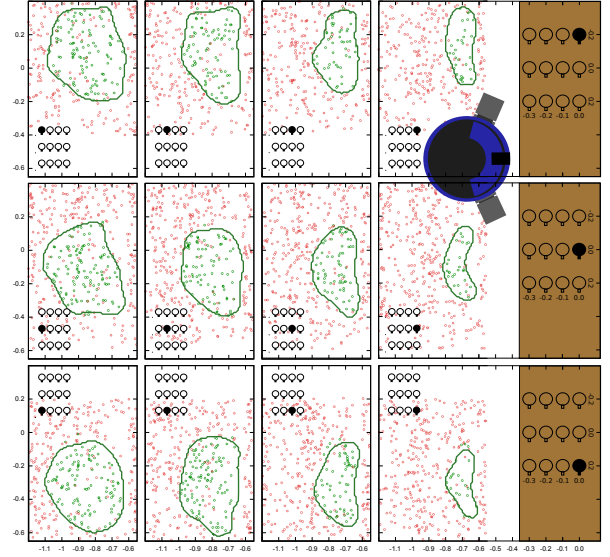

Fig. 5. Successful grasp positions and their classification boundaries. Every sub-image shows the boundary that corresponds to the cup position that is visualized with the black cup. To save space, the table on which the cup is placed is only depicted in the right-most sub-images, and not all failed data points are shown. Data points correspond to the center of the robot base.

boundaries by minimizing the sum of the distances between corresponding points, while maintaining order between the points on the boundary. The result is depicted in Fig. 6, where only 4 of the 12 classification boundaries are depicted for clarity.

Given the aligned points on the boundaries, we compute a PDM. Although PDMs are most well-known for their use in computer vision, we use the notation by Roduit et al. [12], who focus on robotic applications. First, the 2D boundaries are merged into one 40x12 matrix $\mathbf{H}$, where the columns are the concatenation of the x and y coordinates of the 20 points along the classification bound-
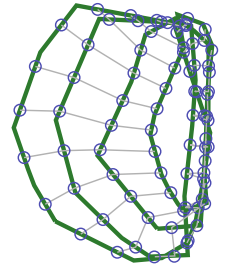

Fig. 6. Point-alignment

ary. Each row represents one boundary. The next step is to compute $\mathbf{P}$, which is the matrix of eigenvectors of the covariance matrix of $\mathbf{H}$. Given $\mathbf{P}$, we can decompose each boundary $\mathbf{h}_k$ in the set into the mean boundary and a linear combination of the columns of $\mathbf{P}$ as follows $\mathbf{h}_k = \overline{\mathbf{H}} + \mathbf{P} \cdot \mathbf{b}_k$. Here, $\mathbf{b}_k$ is the so-called deformation mode of the kth boundary. This is the Point Distribution Model. To get an intuition of what the PDM represents, the first two deformation modes are depicted in Fig. 7(a), where the values of the first and second column of $\mathbf{B}$ are varied between their max. and min. value.

By inspecting the eigenvalues of the covariance matrix of $\mathbf{H}$, we determined that the first 2 components already contain 96% of the deformation energy. Therefore, we use only the first 2 deformation modes, without losing much accuracy.

(a) First and second deformation mode in $\mathbf{B}$.
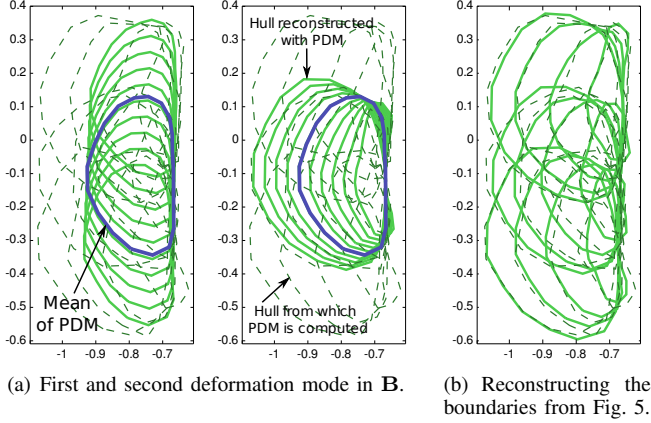
(b) Reconstructing the boundaries from Fig. 5.

Fig. 7. A generalized success model based on a Point Distribution Model.

Fig. 7(b) demonstrates that the original 12 boundaries can be reconstructed well when using combinations of only the first two deformation modes.

The advantage of the PDM is not only that it substantially reduces the high dimensionality of the initial 40D boundaries. It also allows us to interpolate between them in a principled way using only two deformation parameters. The PDM is therefore a compact, general, yet accurate model for the classification boundaries that were determined by the SVM.

D. Relation to Task-relevant Parameters

The final step of model learning is to relate the specific deformation of each boundary (contained in $\mathbf{B}$) to the values of the task-relevant parameters like cup positions that are varied during data collection. Since the correlation coefficients between the first and second deformation modes and the task relevant parameters $\mathbf{T}$ (the x and y coordinates of the cup) are 0.99 and 0.97 respectively, we simply compute the linear relation between them with $\mathbf{W} = [\mathbf{1} \ \mathbf{T}]/\mathbf{B}^T$.

Given a novel position $\mathbf{t}_{new} = \langle x_{new}, \ y_{new} \rangle$ of the cup on the table, the *generalized* success model allows us to quickly compute the area from which a successful grasp can be expected for this *specific* situation. First, we compute the appropriate deformation values from the cup position with $\mathbf{b}_{new} = ([1 \ \mathbf{t}_{new}] \cdot \mathbf{W})^T$. Then the boundary is computed with $\mathbf{h}_{new} = \overline{\mathbf{H}} + \mathbf{P} \cdot \mathbf{b}_{new}$. This boundary estimates the area in which the robot should stand to be able to make a successful grasp. This approach adheres to the proposed strategy of "learning task-relevant features that map to actions, instead of attempting to reconstruct a detailed model of the world with which to plan actions" [9].

IV. COMPUTING MANIPULATION PLACES ON-LINE

In this section, we describe how appropriate PLACEs for manipulation are determined on-line. We denote this module as PLA4MAN, i.e. 'PLACE for manipulation'. As can be seen in the computational model in Fig. 2, this module takes the GSM and the probabilistic belief state as an input, and returns a PLACE such as depicted in Fig. 3.

A. Uncertainty in Object Position

At the end of the previous section, we demonstrated how a $\mathbf{h}_{new}$ classification boundary is reconstructed, given specific task relevant parameters $\mathbf{t}_{new} = \langle x_{new}, \ y_{new} \rangle$. Due to sensor noise and other factors that influence the state estimation, the task relevant parameters can never be known exactly, and uncertainty must be modeled. The belief state therefore also associates a covariance matrix with each position: $\begin{pmatrix} \sigma_{xx}^2 & \sigma_{yx}^2 \\ \sigma_{xy}^2 & \sigma_{yy}^2 \end{pmatrix}$, computed by our vision-based object localization module [10].

Because of this uncertainty, it does not suffice to compute only one classification boundary given the most probable position of the cup as the PLACE from which to grasp. This might lead to a failure if the cup is not at the position where it was expected. To solve this problem, we use a Monte-Carlo simulation to generate a probabilistic advice on where to navigate to grasp the cup. This is done by taking 100 samples from the Gaussian distribution of the cup position, given its mean and covariance matrix. This yields a matrix of task relevant parameters $\mathbf{t}_s = [\mathbf{x}_s \ \mathbf{y}_s]$. The corresponding classification boundaries $\mathbf{h}_s$ are computed for the samples by using the method described above. In Fig. 8(a), 30 out of the 100 boundaries are depicted. These were generated from the task relevant parameters x=-0.3, y=0.1, $\sigma_{xx}=\sigma_{yy}$=0.05, $\sigma_{xy}=\sigma_{yx}$=0.

(a) Sampled classification boundaries ($\mathbf{h}_s$).
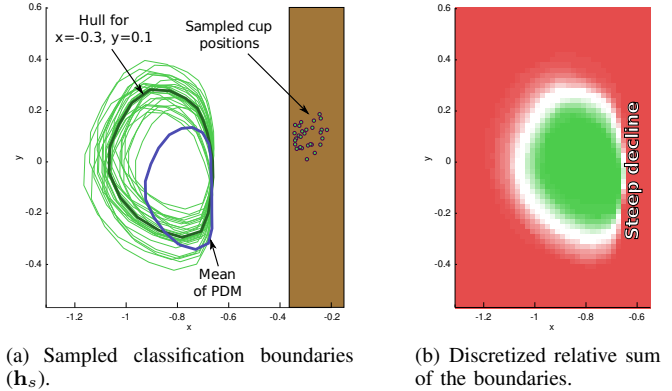
(b) Discretized relative sum of the boundaries.

Fig. 8. Monte-Carlo simulation of class. boundaries to compute PLACE.

We then generate a discrete grid in which each cell measures $2.5 \times 2.5 cm$, and compute the number of classifications boundaries that classify this cell as a success. Dividing the result by the overall number of boundaries yields the probability that grasping the cup will succeed from this position. The corresponding distribution, which takes the uncertainty of the cup position into account, is depicted in Fig. 8(b) (2D), as well as in Fig. 3 (3D).

It is interesting to note the steep decline on the right side of the distribution (in the direction of the table). This is intuitive, as the table is located on the right side, and the robot bumps into the table when moving to the sampled initial position, leading to an unsuccessful navigate-reach-grasp sequence. Therefore, none of the 12 boundaries contain this area, and the variation in $\mathbf{P}$ on the right side of the PDM is low. Therefore, variation in $\mathbf{B}$ does not have a large effect

on this boundary, as can be seen in Fig. 8(b). When summing over the sampled boundaries, this leads to a steep decline in success probability in the direction of the table.

B. Uncertainty in Robot Position

The Adaptive Monte Carlo Localization from the Player project [6] also returns a covariance matrix for the robot's position. This uncertainty must be taken into account in PLACE. For instance, although any position near to the left of the steep incline in Fig. 8(b) is predicted to be successful, they might still fail if the robot is actually more to the right than expected. Therefore, we convolve the PLACE as depicted in Fig. 8(b) with the discretized ($2.5 \times 2.5cm$) probability distribution of the robot's position. The result can be seen in Fig. 9(a). Note that this convolution also works for multi-modal distributions as returned by particle filters.

The distribution in Fig. 9 is the robot's concept of PLACE which takes into account the uncertainty in both the pose of the robot and of the target object. These distributions are generated from a model that is very much grounded in observed experience, as it was learned from observation. Note that this concept is also specific for the task context and the skills of the robot, i.e. using a different robot or controller would lead to different observations, and hence to a different concept of successful PLACEs. It is the developmental process of learning PLACE that allows us to apply it to a wide range of robots and controllers. Fig. 9(b) depicts how the probability distribution is affected by varying task relevant parameters. Please notice how in the first row, it becomes 'more difficult' to grasp the cup (i.e. less likely to succeed) as the cup moves away from the table's edge.

V. SUBGOAL REFINEMENT WITH PLACE

Instead of committing to a specific position in advance, a PLACE enables least-commitment planning, as a whole range of positions is predicted to be successful, or at least probable. For instance, the robot could choose any of the positions for which $P(succ|\mathbf{t}) > 0.95 * max(P(succ|\mathbf{t}))$, and optimize secondary criteria such as execution duration or energy consumption to determine the actual position the robot will navigate to. Selecting subgoal parameters such that they optimize secondary criteria is known as *subgoal refinement* [14]. In this paper, we do not take secondary criteria into account, but rather optimize the probability of success, as depicted in Fig. 9(a). As depicted in Fig. 2, this is the position returned by Subgoal Refinement.

Finally, action-related PLACEs for multiple actions can be composed by intersecting them. Assuming that their success probability is independent of each other, the poses in the intersection are determined as the product of the probabilities of each single action. Fig. 10 illustrates this for the task of concurrently grasping two cups. This composition would be impossible if the robot commits to specific positions in advance.
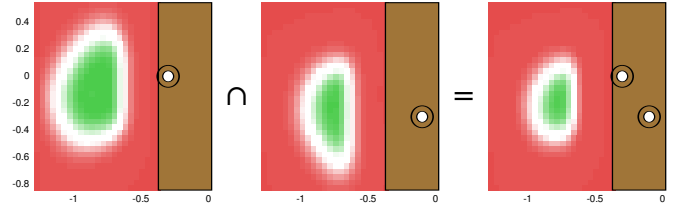

Fig. 10. PLACE as a composition of the probability distributions of successful individual task executions. Left distribution: grasp cup with left gripper. Center distribution: grasp cup with right gripper. Right distribution: Grab both cups with left/right gripper respectively. It is the product of the other two individual distributions.

VI. EMPIRICAL EVALUATION

At a open house day at our university, the mobile manipulation platform described in the Appendix continually performed an application scenario, where it locates, grasps and lifts a cup from the table, and then transports it to a work-surface next to the kitchen oven. Fig. 1 depicts two pictures taken during the demonstration. The robot performed this scenario 50 times in approximately 6 hours, which has convinced us that the robot hardware and software are robust enough to be deployed amongst the general public.

After the open day, we ran the same experiment, but this time with the PLA4MAN module included. The focus of this experiment was on our error-recovery system described in [1], and the improved performance of the robot cannot quantatively be attributed to the addition of the error-recovery system, or the addition of the PLA4MAN module. However, a major qualitative improvement was that the cup could now be grasped from a much larger area on the table. Without PLA4MAN, the cup always had to be placed on approximately the same position on the table to enable successful grasping.

Analytically, we compared the PLA4MAN module with three other strategies: 1) Fixed: the B21 always goes to a fixed position at the table 2) Relative: the B21 always goes to a position relative to the cup position 3) Mixed: which is basically Relative, until a certain minimum distance to the table is reached. Then it becomes Fixed to avoid bumping into the table. In one experimental episode, we first determine the real position of the cup, and sample an observed position given the real position and the covariance matrix of the cup. The same is done for the robot position. Given the 'estimated' cup and robot position, the robot then uses the PLA4MAN module to compute a PLACE to perform manipulation. Afterwards it is determined if the grasp from this position is successful, given the 'real' position of the robot and the cup. Determining a successful grasp is done by determining if the 'real' position of the robot is classified as a success, given the classification boundary corresponding to the 'real' position of the cup.

The results of this simulation are depicted in Fig. 11, for varying values of x_{target} (i.e. the cup), σ_{target}, and σ_{robot}, where the covariance matrices of the robot and the cup are computed with $\sigma^2 \cdot \mathbf{I_2}$. The y coordinate of the cup is fixed, as it hardly influences the success rate. We draw the following

(a) Distribution for parameters: $x=-0.3$, $y=0.1$, $\sigma_{xx}=\sigma_{yy}=0.05$.

(b) These images show how varying certain task-relevant parameters affects the shape of the distribution. The table and the cup are drawn to scale in the xy-plane.
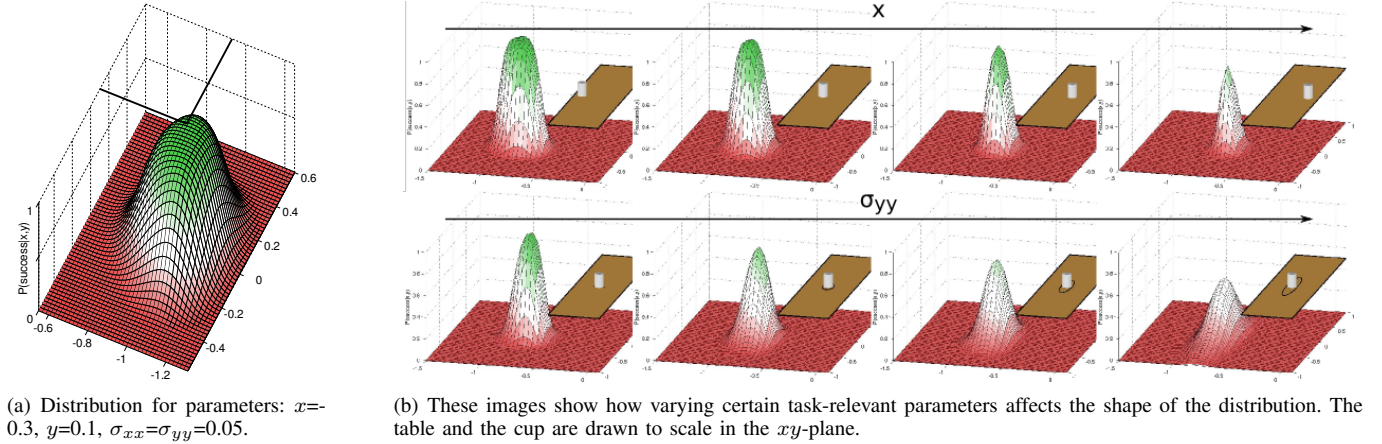
Fig. 9. Final distributions, after convoluting the uncertainty in the robot pose with a distribution as depicted in Fig. 8(b). These distributions represent the robot's probabilistic least-commitment PLACE, which is task-related, skill-specific, and grounded in experience.
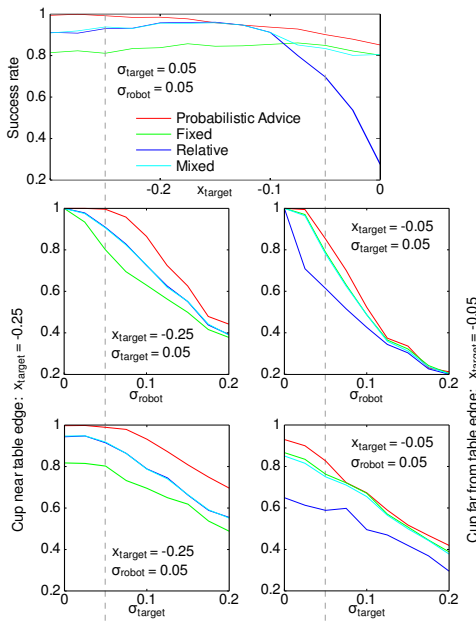

Fig. 11. Results of the empirical evaluation.

conclusions from these graphs: 1) overall, the PLA4MAN module achieves on average a 7% improvement over the second best strategy, which is 'Mixed' 2) It is more difficult to grasp the cup when it is further away from the edge of the table. The robot essentially has less freedom in choosing where to go, and incorrect estimation of the position of the cup or the robot is more likely to lead to a failed grasp 3) with very low or high certainty, the methods often perform equally well. It is in the middle range (i.e. the values most frequently observed during actual robot deployment), that PLA4MAN has the greatest advantage of taking uncertainty into account.

Finally, in simulation the robot conducted 750 approaches with PLA4MAN and the Fixed strategy with σ_{target} ranging between 0.0 and 0.2. The success rates were 0.76 and 0.56 respectively, a significant increase ($p < 0.01$ with a χ^2-test).

VII. CONCLUSION

In this article, we have presented a system that enables robots to learn a concept of PLACE that is compact, grounded in observed experience, and tailored to the robot's hardware and controller. The main advantage is that our system integrates the often disconnected research areas of navigation and manipulation planning. By studying the coupling and interactions between both skills, it is often possible to find PLACEs where the objects can be grasped by using standard behavior like motion primitives. PLACE is modelled as a probability distribution, which enables the robot to perform least-commitment planning, instead of prematurely committing itself to specific positions that could be suboptimal. Optimizing the probability of successful grasping leads to more robust behavior on our mobile manipulation platform.

We are currently extending our approach in several directions. We are applying our approach to more complex scenarios, and different domains. For instance, we are learning higher-dimensional PLACE concepts, which take more aspects of the scenario into account, i.e. different object sizes. We are also investigating extensions and other machine learning algorithms that will enable our methods to generalize over this larger space. Objects which require very different grasps, such as using two hands to manipulate them, will require more sophisticated methods for acquiring and reasoning about place. Generalization of our place concept with respect to situations and task contexts is a research challenge which we have on our mid-term research agenda.

APPENDIX: MOBILE MANIPULATION PLATFORM

The hardware platform we use for our experiments is a B21r mobile robot from Real World Interface. Its wheels allow this robot to move forward and turn around its center axis. Two 6-DOF lightweight arms from Amtec with slide grippers are mounted on this base, allowing for the manipulation of objects at table height. For localization and navigation, we use several modules from the Player project [6], being Adaptive Monte Carlo Localization, pmap for map building,

and the AMCL Wavefront Planner for global path planning. These modules use the SickLMS400 laser range scanner and odometry provided by the base. A combination of Dynamic Movement Primitives [8] and Vector Fields is used for reaching and grasping. The inverse kinematics computations are performed using the Kinematics and Dynamics Library (KDL) from Orocos. Detection and localization of the objects to be grasped is done using the method described in [10]. Player and YARP provide the middleware that enable these software modules to communicate effectively with the hardware, and other software modules. The Gazebo simulator [6] is used for efficient data collection and debugging purposes.

REFERENCES

[1] Michael Beetz, Freek Stulp, Piotr Esden-Tempski, Andreas Fedrizzi, Ulrich Klank, Ingo Kresse, Alexis Maldonado, and Federico Ruiz. Generality and legibility in mobile manipulation. *Autonomous Robots Journal (Special Issue on Mobile Manipulation) (submitted)*, 2009.

[2] Dmitry Berenson, Rosen Diankov, Koichi Nishiwaki, Satoshi Kagami, and James Kuffner. Grasp planning in complex scenes. In *IEEE-RAS International Conference on Humanoid Robots*, 2007.

[3] Sebastian Buck and Martin Riedmiller. Learning situation dependent success rates of actions in a RoboCup scenario. In *Pacific Rim International Conference on Artificial Intelligence*, page 809, 2000.

[4] B. J. Clement, E. H. Durfee, and A. C. Barrett. Abstract reasoning for planning and coordination. *Journal of Artificial Intelligence Research*, 28:453–515, 2007.

[5] A. Ekstrom, M. Kahana, J. Caplan, T. Fields, E. Isham, E. Newman, and I. Fried. Cellular networks underlying human spatial navigation. *Nature*, 425(184–188), 2003.

[6] Brian Gerkey, Richard T. Vaughan, and Andrew Howard. The Player/Stage Project: Tools for multi-robot and distributed sensor systems. In *Proceedings of the 11th International Conference on Advanced Robotics (ICAR)*, pages 317–323, 2003.

[7] S. Hart, S. Ou, J. Sweeney, and R. Grupen. A framework for learning declarative structure. In *RSS-06 Workshop: Manipulation for Human Environments*, 2006.

[8] A. J. Ijspeert, J. Nakanishi, and S. Schaal. Movement imitation with nonlinear dynamical systems in humanoid robots. In *International Conference on Robotics and Automation (ICRA2002)*, 2002.

[9] C. Kemp, A. Edsinger, and E. Torres-Jara. Challenges for robot manipulation in human environments. *IEEE Robotics and Automation Magazine*, 14(1):20–29, 2007.

[10] Ulrich Klank, Muhammad Zeeshan Zia, and Michael Beetz. 3D Model Selection from an Internet Database for Robotic Vision. In *International Conference on Robotics and Automation (ICRA)*, 2009.

[11] Benjamin Kuipers, Patrick Beeson, Joseph Modayil, and Jefferson Provost. Bootstrap learning of foundational representations. *Connection Science*, 18:145–158, 2006.

[12] Pierre Roduit, Alcherio Martinoli, and Jacques Jacot. A quantitative method for comparing trajectories of mobile robots using point distribution models. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2441–2448, 2007.

[13] S. Sonnenburg, G. Raetsch, C. Schaefer, and B. Schoelkopf. Large scale multiple kernel learning. *Journal of Machine Learning Research*, 7:1531–1565, 2006.

[14] Freek Stulp and Michael Beetz. Refining the execution of abstract actions with learned action models. *Journal of Artificial Intelligence Research (JAIR)*, 32, June 2008.

[15] Matthias Wimmer, Freek Stulp, Sylvia Pietzsch, and Bernd Radig. Learning local objective functions for robust face model fitting. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 30(8):1357–1370, 2008.

[16] F. Zacharias, Ch. Borst, and G. Hirzinger. Capturing robot workspace structure: representing robot capabilities. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3229–3236, 2007.