Data-Driven Online Decision Making for Autonomous Manipulation

Daniel Kappler*, Peter Pastor†, Mrinal Kalakrishnan†, Manuel Wüthrich* and Stefan Schaal*†
*Autonomous Motion Department, Max Planck Institute for Intelligent Systems, Tübingen, Germany
Email: first.lastname@tuebingen.mpg.de
†Computational Learning and Motor Control Lab, University of Southern California, Los Angeles, USA
Email: {pastorsa,kalakris,sschaal}@usc.edu

***Abstract*—One of the main challenges in autonomous manipulation is to generate appropriate multi-modal reference trajectories that enable feedback controllers to compute control commands that compensate for unmodeled perturbations and therefore to achieve the task at hand. We propose a data-driven approach to incrementally acquire reference signals from experience and decide online *when* and to *which* successive behavior to switch, ensuring successful task execution. We reformulate this online decision making problem as a pair of related classification problems. Both process the current sensor readings, composed from multiple sensor modalities, in real-time (at 30 Hz). Our approach exploits that movement generation can dictate sensor feedback. Thus, enforcing stereotypical behavior will yield stereotypical sensory events which can be accumulated and stored along with the movement plan. Such movement primitives, augmented with sensor experience, are called Associative Skill Memories (ASMs). Sensor experience consists of (real) sensors, including haptic, auditory information and visual information, as well as additional (virtual) features. We show that our approach can be used to teach dexterous tasks, e.g. a bimanual manipulation task on a real platform that requires precise manipulation of relatively small objects. Task execution is robust against perturbation and sensor noise, because our method decides *online* whether or not to switch to alternative ASMs due to unexpected sensory signals.**

I. Introduction

We argue that robust task execution requires to close feedback control loops in novel ways around many more sensory signals than done traditionally. Otherwise robotic systems cannot cope with noise and uncertainty in the sensory-motor system of the robot and the environment. Low-level control systems increase the execution robustness by integrating feedback of high-bandwidth sensors, e.g. force/torque. Yet, task execution often fails due to external perturbations that are hard to model in the low-level control system, using high-bandwidth sensors. We propose to close a high-level feedback loop, leveraging additional low-bandwidth sensor modalities, e.g. vision based object tracking, to increase the task robustness.

One challenge towards autonomous manipulation is to provide feedback controllers with appropriate reference signals and decide online *when* to consider alternative behaviors to counteract perturbations. Model-based approaches, such as [13, 6], have shown great success for motion planning as long as models are accurate enough. However, for complex contact interactions, modeling errors are inevitable and will significantly degrade the performance of the planner [16] , e.g. for grasping and dexterous object manipulation. Uncertainty about the state of the manipulated object increases exponentially fast and therefore planning becomes intractable, unless smart heuristics/biases [1] are applied.

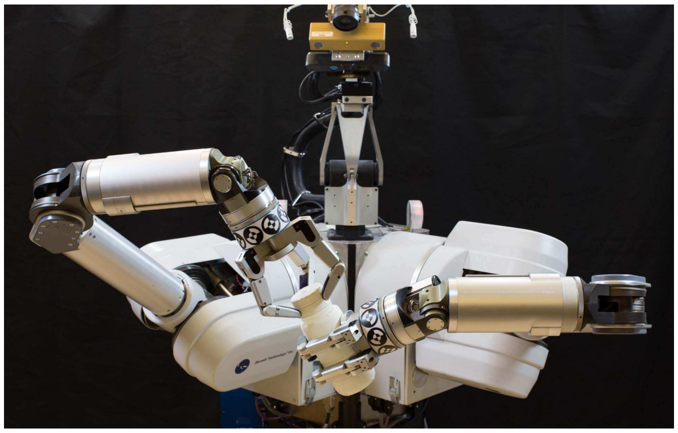
Fig. 1: The ARM-S robot manipulating a bottle.

We propose to formulate manipulation as a data-driven sequential decision making problem. For most manipulation tasks, e.g. unscrewing a bottle (see Fig. 1), the task naturally decomposes into a sequence of skills. A sequence can be encoded in a state machine, termed *manipulation graph*, see Fig. 3. This graph representation imposes constraints onto the possible sequence of manipulation skills, similar to how a grammar constrains the possible sequences of words to form sentences. Such a representation can be either inferred from data [8, 9] or provided by human operators, who usually have good intuition about the necessary high-level information and skill decomposition to accomplish a certain task. However, manually designing rules, based on current sensory information to determine whether or not the current skill execution is valid and where to start an alternative skill, is very hard. This is mainly due to differences in sensor modalities, noise characteristics, and the high dimensionality of the signal space.

In this paper we propose a method which addresses this issue in a data-driven manner. We propose to decompose the underlying online decision making problem into two related classification problems. Both problems have to operate in real-time to close the high-level feedback loop, learning *when* to switch skills and *which* skill to execute next. All required information should be automatically extracted from previous sensory experiences, stored in the *manipulation graph* structure, only requiring user interventions at failure conditions.

The key insight that enables such a data-driven system is that the task-relevant perceived sensor signals are strongly correlated with the executed manipulation action [11]. Thus, given similar task execution, the robot will perceive similar sensory feedback [10]. Therefore, skills used throughout this paper are encoded as Associative Skill Memories (ASMs), sensor information encoded in sync with movement primitives, as introduced in [12]. The associated signals for a manipulation action are used to train our classification-based online decision making system (ODMS).

The main contribution of this paper is an ODMS that is capable of deciding in real-time at any moment of the task execution if the currently executed skill should be replaced with another one (*when*) and which skill to choose as a replacement (*which*). The only required human intervention to bootstrap our system is to stop task execution if the task execution is about to fail. Based on this real robot experimental data, stored in the *manipulation graph* structure, we automatically extract supervised datasets to train our classification based decision making system. Our formulation allows to integrate sensors even if they contain no valuable information about the task. The importance of different sensor modalities at different execution stages is automatically inferred. Thus, manual task specific feature selection is less crucial. We further report qualitative results of our method applied to a dexterous manipulation task performed by a bimanual robotic system, see Fig. 1.

II. Related Work

Motion planning approaches [13, 6] are mostly applied to problems neglecting object interactions. One recent approach that creates plans for contact manipulation, proposed in [4], is still using a quasi-static assumption to obtain a feasible search space. Our work builds upon results reported in [10, 11, 12], a skill-based formulation towards autonomous manipulation. In contrast to [12], we propose to close the high-level feedback loop by running a decision making process online at all time and not only selecting the next ASM at the terminal condition of the current ASM. Different to [10], we propose to learn failure case prediction from a supervised signal, obtained from the ASMs organized in the *manipulation graph* structure. Additionally, we reduce the number of open hyper-parameters, compared to previous work [10]. The *manipulation graph* is capable of generating a large number of skill sequences. This structure is assumed to be manually provided by a user. Although out of the scope of this paper, a manipulation graph could be constructed automatically, as proposed in [8, 9]. Kroemer et al. [8] proposed to infer the hidden phases of the manipulation task from several trial executions, using an autoregressive Hidden Markov model. They train logistic classifier, on a small, manually selected set of discriminative features to model the transition probability deciding when to switch between states in their inferred Hidden Markov model. Niekum et al. [9] present another approach to automatically infer a state machine representation which can be incrementally refined. Similar to [12], this approach only allows to switch skills at the end of each skill execution.

III. Problem Formulation

Similar to previous data-driven approaches to manipulation, we assume that tasks can be decomposed into re-usable

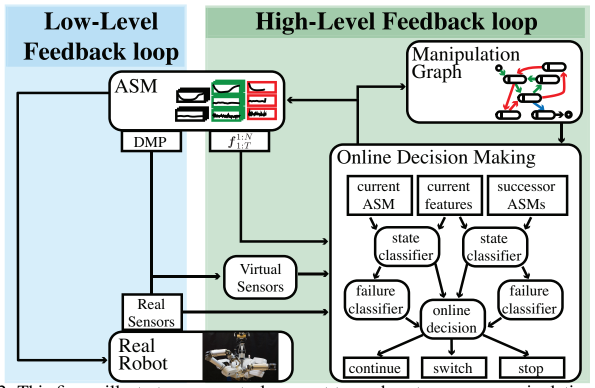

Fig. 2: This figure illustrates our control concept towards autonomous manipulation. The left side shows the high-bandwidth low-level control loop. The DMP of the currently active ASM generates the reference trajectory (black) locally adapting due to (real) sensory feedback. The right side illustrates the main building blocks and their interaction, discussed in this paper. Our system leverages previous sensor experience of successful (green) and erroneous (red) trials to train two sets of classifiers (state and failure). At runtime the manipulation graph restricts the possible successor ASMs, considered for state classification. The box labeled 'Online Decision Making' shows the general data flow and the individual steps to close the high-level feedback loop.

skills. Skills can be interpreted as low-level control systems. Fig. 2 illustrates such an control loop (blue). Although low-level control systems allow to counteract local perturbations, they are not robust enough for manipulation in unstructured environments. This is mainly due to strong unmodeled perturbations (outside of the stability regions), simplistic model assumptions, or missing of high-level sensory feedback. In this paper, we focus on closing a high-level feedback loop (Fig. 2, green) using both low- and high-bandwidth sensors (Sec. III-B), taking the low-level control loop into account. This problem can be interpreted as a real-time online decision making problem, capable of orchestrating skills due to sensor feedback, while improving from past experiences. We argue that the training of such a system should only require *common sense* user feedback and no low-level system knowledge.

A. Manipulation Graph

Inspired by language models, we believe that complex behaviors can be decomposed into skills (words). Similar to grammar models, a task specific directed graph models constraints on possible skill sequences. Each graph node represents a skill, an atomic unit represented as an ASM, see Fig. 3. This graphical representation and the sensory experience contain all the required information for closing the high-level feedback loop.

Despite recent progress in structure learning, we believe that modelling such a directed graph is still a domain where humans outperform state-of-the-art inference methods. Therefore, we propose that a task specific manipulation graph is constructed by a user, using skills from an existing ASM library or additionally taught task specific skills. Initially the manipulation graph consists of a linear sequence (Fig. 3, green) which encodes the expected outcome of each skill. During task execution, the *manipulation graph* builds the connection between the executed ASM and the ASM library, allowing to add annotations such as success, failure, and the path through the graph. This information is initially based on user intervention for new ASMs, stopping as soon as the

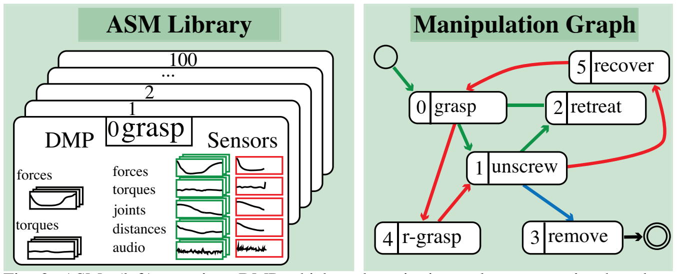

Fig. 3: ASMs (left) contain a DMP which updates its internal representation based on several executions (black). Sensor experience is stored in sync with the DMP progress and split into successful (green) and unsuccessful (red) trials. Sensor signal for unsuccessful trials stop as soon as a failure is indicated. The right hand side shows an exemplary manipulation graph for unscrewing a bottle initially (green) containing only a sequence of three ASMs, connected with their expected successor ASM. As soon as the cap is loose on the bottle we stop the execution and demonstrate a new ASM *remove* (blue), added to *unscrew*. Finally, we add additional recovery ASMs (red) which result in a more robust overall performance.

execution fails. As soon as both successful and erroneous trails are available, interventions can also be due to a trained ODMS. This initially linear graph is extended to achieve the target task by intervention if failures occur. For example, while executing *unscrew*, the cap of a bottle becomes loose and thus can be removed. Therefore, a failure is triggered since the expected skill (*retreat*), opening the gripper, cannot be applied. Then either a new behavior (*remove*) is added as possible successor to the current graph node or a terminal stopping node is added, representing an unrecoverable system condition. This procedure guarantees an efficient teaching process, only considering human labeled failure cases, occurring on the real system [1]. Structuring available ASMs in a graphical representation reduces computational complexity and avoids perceptual aliasing, i.e. when the currently processed features cannot distinguish between two or more ASMs.

B. Features

Data-driven methods strongly depend on the information provided by features, and the representation itself. Robotic systems usually provide a lot of different sensor modalities which allow to observe both the state of the robot and of the environment. For some tasks, it might be intuitively clear which sensors to manually select, to acquire all task relevant information. However, for many tasks, especially those that involve contact interactions and external perturbations from a nominal state, this may not be as clear. We argue that it is beneficial to include many sensors in the ASM representation and automatically infer their relevance from data. This allows to use the same methodology for a wider range of tasks and perturbations. The inherent noise of real sensor readings increases the learning complexity of classification based systems. Therefore, we propose to enrich the ASM representation, compared to [12], with additional (real) as well as (virtual) sensors, adding possible redundancy to the feature representation. Hereafter, we use the term features and sensors for both real as well as virtual sensor information interchangeably. Virtual sensors are features, computed online from (real) sensor signals. Such (manually) uncovered virtual features

[1] Previous related work [12] does not support failure driven incremental teaching, since switching is assumed to take place exclusively during the last 10 percent of the ASM execution.

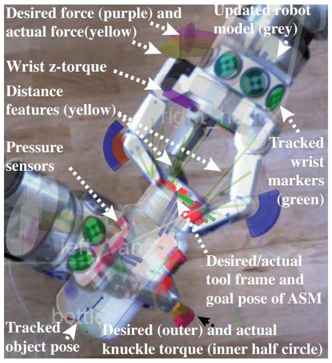
Desired force (purple) and actual force(yellow)
Updated robot model (grey)
Wrist z-torque
Distance features (yellow)
Pressure sensors
Tracked wrist markers (green)
Desired/actual tool frame and goal pose of ASM
Tracked object pose
Desired (outer) and actual knuckle torque (inner half circle)

Fig. 4: The left picture shows an image recorded from the left Bumblebee camera of the ARM-S robot (see Fig. 1) overlaid with sensor and feature information. The green markers at the wrist are visually tracked and used to estimate the pose of the arm, overlaid in gray. The pose of the bottle is tracked using 3D information from the depth camera as introduced in [17]. Predicted and actual forces are also illustrated. The table below lists the feature count and a corresponding description of each feature used during the experiments.

Feature count and descriptions	
4	Joint angles of the right hand
12	Force/torque measurements at both wrists
3	Torques at the knuckles of the right hand
10	Distance between locations on the visually tracked right hand and the corresponding closest point on the tracked bottle
13	Position and orientation (in quaternion notation) of the right palm and tool frame to the tracked pose of the bottle
11	Relative linear and angular velocities of the right palm, tool frame and the tracked bottle
8	Mel-frequency cepstral coefficients (MFCC) computed using the audio recordings of the two microphones located in the sensor head

exploit task knowledge, e.g. the proximity of the robot's end-effector the target object, increasing robustness against sensor noise. These features provide an alternative channel to obtain similar information, e.g. proximity for force/torque measurements. For example, we create virtual vision based features, using existing methods for object tracking. Auditory information is rarely being considered during manipulation, yet we believe that valuable information about the progress of a task can be inferred from this signal type. Especially failure conditions are oftentimes accompanied with sound feedback, as shown for object grasping by Sinapov et al. [15]. Thus, we enrich our ASM representation with audio features. Fig. 4 shows an exemplary feature representation, used for the experimental results. Sec. V-A provides further details of the feature acquisitions process.

IV. Data-Driven Online Decision Making

The principal goal of our work is to close high-level feedback loops (Fig. 2 (green)), integrating low-level control systems while improving from scarce supervised data, using a transparent user process. High-level feedback loops can leverage information from low-bandwidth (object tracker) and high-bandwidth sensors (force/torque), still operating in real-time (30 Hz), adding robustness to autonomous manipulation systems. The change of feature importance throughout the execution of skills is a common issue for the design of data-driven online decision making systems. For example, force/torque sensor readings might have no relevance prior to object grasping and completely different signal characteristics after making contact. Partial skill execution, required to achieve robust recovery behaviors, adds further complexity to the ODMS design. We address these issues by formulating the online decision making problem as two related state-based classification problems (Fig.2). A **state** (see Sec. IV-A) in the context of our method represents a distinct execution stage of an ASM, with a corresponding dataset of feature readings, which is automatically associated from successful and erroneous trials. The first classification process predicts the most likely state of the currently executed skill and successor skills, discretizing the time series into independent **states**. Only possible successor skills, determined by the *manipulation graph*, are considered, reducing the computational complexity. Sec. IV-B describes the optimization procedure for this classification problem and the automatic supervised dataset construction from previous task executions. The second classification task determines whether or not the sensory information is valid or not, meaning if the currently executed skill should be switched or the system should stop (Sec. IV-C). Finally, the high-level feedback loop is closed by fusing the outcome of all predictions, as described in Sec. IV-D.

A. Representation

Our proposed ODMS is optimized based on the following data representation, obtained and updated using information provided by the manipulation graph, sensory feedback and user intervention. Each ASM, stored in the skill-library, can be identified by a unique id and each skill execution during real robot experiment has a corresponding trial id. We store the predecessor trial id and the successor trial id respectively for every execution, allowing us to draw connections between these executions. This information is automatically acquired during experiments, only requiring the user to stop executions at failure cases and select or teach appropriate recovery behaviors. While running an experiment a pre-defined set of features is recorded at the highest possible frequency. The current feature readings are denoted as the feature vector $\boldsymbol{x}$. We assume that unsuccessful trials are interrupted before the skill is fully executed. Skills are represented by ASMs, thus skill execution is based on DMPs [5]. Therefore, unsuccessful trials can be identified by the phase, meaning before the phase variable ρ of the DMP has converged $\rho = 0$. Thus, we store every trial in the ASM library and add success ($\rho = 0$) or failure ($\rho > 0$) labels. All (N) successful trials, associated with a particular ASM are aligned using ρ and discretized in the target frequency (e.g. 30 Hz), resulting in the time series data with T states $\boldsymbol{f}_{1:T}^{1:N}$, where T is determined by the skill execution time and target frequency. Every state $\boldsymbol{s} = \boldsymbol{f}_t^{1:N} \in \boldsymbol{f}_{1:T}^{1:N}$ is treated as a different class with corresponding data samples from the N successful trials. Unsuccessful trials are stored but not aligned.

B. State Classification

State classification predicts the most likely state given the current sensor signals $\boldsymbol{x}$ (Fig. 5). The most likely state $\boldsymbol{s}_c$ of the currently executed ASM is computed independent of the most likely state of all possible successor ASMs $\boldsymbol{s}_s$ (Fig. 2). Directly using the ASM progress ρ to estimate the current state $\boldsymbol{s}_c$ is sub-optimal due to perturbation and noise, e.g. object contact while grasping almost always happens at different stages but has a large effect on the sensor readings. Our formulation decomposes state classification into T isolated

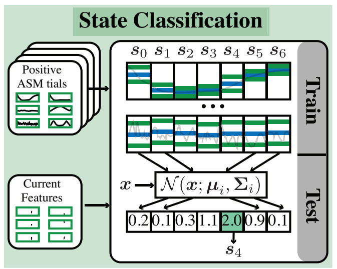

Fig. 5: State classification first discretizes all successful trials of a particular ASM for all sensors, shown in the top. The feature mean (blue) and standard deviation (green) is updated for each feature independently, based on the time aligned signals. At test time the likelihood of the current signal $\boldsymbol{x}$ is computed for all states $\boldsymbol{s}_i$ and the most likely state is returned.

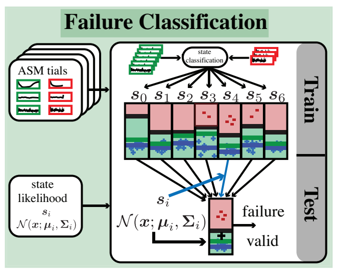
Failure Classification
ASM tials
state classification
s_0 s_1 s_2 s_3 s_4 s_5 s_6
Train
state likelihood
s_i
$\mathcal{N}(x;\mu_i,\Sigma_i)$
s_i
$\mathcal{N}(x;\mu_i,\Sigma_i)$
failure
valid
Test

Fig. 6: Failure classification uses all ASM trials and assigns the likelihood $\mathcal{N}(\boldsymbol{x};\boldsymbol{\mu}_{\boldsymbol{s}_i},\boldsymbol{\Sigma}_{\boldsymbol{s}_i})$ of $\boldsymbol{x}$ to the most likely corresponding state $\boldsymbol{s}_i$. For each state we train a linear SVM using the generated dataset. The decision boundary is propagated to neighboring states in terms of standard deviations of the samples from successful trials. Test time evaluation selects the most likely state and checks if the state likelihood is below the error threshold.

problems. All classifiers optimize their own feature importance model completely independent of each other. This rather strong assumption enables our system to predict at which state a new skill should start, since independent classifiers for each state $\boldsymbol{s}$ do not require any partial information up to the current state.

Our approach is based on two main assumptions. First, we assume that sensor readings $\boldsymbol{x}$ contain enough information to recover the corresponding state $\boldsymbol{s}$. Second, we assume that feature variations during successful executions are solely due to sensor noise or counteracting external perturbation, using low-level feedback control.

Skills have typically a duration in the order of seconds, thus, thousands of classes have to be considered to evaluate new sensor readings $\boldsymbol{x}$ (Fig. 5). To ensure real-time classification, the computational complexity of individual classifiers must be rather low. Fortunately each classifier is only concerned with a short period of time, allowing us to use simple models. We use a probabilistic formulation (Eq. 1), using Gaussian sensor models for the likelihood function, see Eq. 2. The ASM state is denoted by $\boldsymbol{s}$, $\{\boldsymbol{\mu}_{\boldsymbol{s}}, \boldsymbol{\Sigma}_{\boldsymbol{s}}\}$ represent the statistics of a particular ASM state, and $p(\boldsymbol{s}|\rho)$ a progress ρ dependent prior.

$$\arg\max_{\boldsymbol{s} \in ASMs} p(\boldsymbol{s}|\boldsymbol{x}) \propto p(\boldsymbol{s}|\rho)p(\boldsymbol{x}|\boldsymbol{s}) \tag{1}$$

$$\hat{p}(\boldsymbol{s}|\boldsymbol{x}) \propto p(\boldsymbol{s}|\rho)\mathcal{N}(\boldsymbol{x};\boldsymbol{\mu}_{\boldsymbol{s}},\boldsymbol{\Sigma}_{\boldsymbol{s}}) \tag{2}$$

$$\hat{p}(\boldsymbol{s}|\boldsymbol{x}) \propto p(\boldsymbol{s}|\rho)\prod_{i=1}^{n} p(\boldsymbol{x}_i|\boldsymbol{s}_i) \tag{3}$$

Assuming feature independence, $\boldsymbol{\Sigma}$ is a diagonal matrix resulting in a naïve Bayes classifier (Eq. 3). This classifier has been shown to work very well in practice [14], despite its strong assumptions. Crucial for our application is the fast training and testing time. Training is reduced to computing the mean $\boldsymbol{\mu}_{\boldsymbol{s}}$ and variance $\boldsymbol{\Sigma}_{\boldsymbol{s}}$ of the state specific dataset $\boldsymbol{s} = \boldsymbol{f}_t^{1:N}$, obtained automatically from the successful trials of the corresponding ASM (Sec. IV-A). Furthermore, incremental learning is possible by simply re-estimating the feature mean and variance of the likelihood model. This likelihood model automatically trades off the importance of different sensor modalities for each state, allowing to handle completely different sensor modalities.

Our state classifier factorizes completely, therefore training and prediction can be parallelized for all states in the current and possible successor ASMs, which is crucial to achieve real-time performance. Due to recent advances in hardware, especially accelerations in GPU communication, this can be done efficiently, scaling up to hundreds of possible successor ASMs. The progress based prior $p(\boldsymbol{s}|\rho)$ is only meaningful for the currently executed ASM, e.g. imposing a Gaussian prior around the current skill progress. Since it would only reduce perceptual aliasing within the current ASM and the number of classes, we assume a uniform prior for both current and successor skills, resulting in a maximum likelihood classification problem:

$$\hat{p}(\boldsymbol{s}|\boldsymbol{x}) \propto \mathscr{N}(\boldsymbol{x};\boldsymbol{\mu}_{\boldsymbol{s}},\boldsymbol{\Sigma}_{\boldsymbol{s}}) \tag{4}$$

C. Failure Classification

Due to the complexity of manipulation tasks, failure cases are inevitable during task execution. However, we assume that the majority of the task executions succeed. Therefore, less data is available for failure case classification. Failure cases rarely happen several times at the exact same execution stage, adding further complexity to the prediction problem.

We use a supervised discriminative approach to classify failure conditions. Eq. 5 presents a general optimization based formulation to this problem, where l denotes a problem specific loss function (e.g. $-I(c(\boldsymbol{x},\boldsymbol{\theta}) = y)$ where I is the indicator function), c the classifier with open parameters $\boldsymbol{\theta}$, R a regularizer (e.g. l_1-norm $||\boldsymbol{\theta}||_1^2$) to prevent overfitting and encoding additional information to obtain e.g. sparse solutions, and $(\boldsymbol{x},y) \in D$ a supervised dataset of sensor signals $\boldsymbol{x}$ and labels $y \in \{\text{success},\text{failure}\}$.

$$\min_{\boldsymbol{\theta}} \sum_{(\boldsymbol{x},y)\in D} l(c(\boldsymbol{x},\boldsymbol{\theta}),y) + R(\boldsymbol{\theta}) \tag{5}$$

Since our aforementioned state classifiers already determine the most likely state $\boldsymbol{s}$ for both the current and possible successor ASMs, we address failure classification independently for each state. Hence, by associating sensor readings $\boldsymbol{x}$ from successful and unsuccessful trials of a particular ASM with their most likely state, our system automatically generates state dependent supervised datasets, allowing discriminative learning:

$$D' = \{(\boldsymbol{s}_k, d_k, y_k) : \boldsymbol{s}_k = \underset{\boldsymbol{x}\in ASM}{\arg\max}\, \hat{p}(\boldsymbol{s}|\boldsymbol{x}), d_k = \max_{\boldsymbol{x}\in ASM} \hat{p}(\boldsymbol{s}|\boldsymbol{x})\} \tag{6}$$

Success and failure labels y are determined by the label of the trial (see Sec. IV-A). To associate feature vectors $\boldsymbol{x}$ from successful trials $\boldsymbol{f}_{1:T}^{1:N}$ with their corresponding state, leave-one-out cross validation [7] has to be used, otherwise the resulting state classification would be overconfident. This means that state classification is trained on all successful trials except for the one which is currently added to the supervised dataset. Further, the state classification, trained on all successful trials, assigns the last m feature vectors for unsuccessful trials of the corresponding ASM to the most likely states (Eq. 6). Thus, for generating the supervised datasets the user only has to stop an execution in an error case. Since each state classifier returns the predicted likelihood, resembling the similarity of the current feature readings with the state, we can further simplify the given problem into a one dimensional binary classification problem:

$$\min_{\boldsymbol{\theta}} \sum_{(\boldsymbol{s}_k,d_k,y_k)\in D'} l(c_{\boldsymbol{s}_k}(d_k,\boldsymbol{\theta}),y_k) + R(\boldsymbol{\theta}) \tag{7}$$

Hence, our failure classifiers automatically benefit from model improvements of our state classifiers, further reducing the computational complexity at training and test time.

Although the particular classifier is not decisive for the success of the proposed method, using a discriminative classifier allows to easily integrate data from the state classifier without requiring further processing or modelling. We use the automatically generated dataset (Eq. 6) to learn a max-margin linear support vector machine (Eq.7, $R = ||\boldsymbol{\theta}||_2^2$ l_2-norm, $l = \max(0, 1-y(\boldsymbol{\theta}^T d + b))$, hinge loss) [2], for which the global optimum can be found efficiently. In the case of our one dimensional problem this optimization will result in a one dimensional state dependent threshold $\varepsilon_{\boldsymbol{s}}$, which can be used to classify the predicted state likelihood into success or failure. One reason to choose a discriminative over a probabilistic classifier is the dynamical nature of the dataset generation. It is to be expected that the data sample distribution will change throughout experiments, resulting in multimodal distributions, adding additional model complexity.

Successful and unsuccessful trials have different effects on the decision boundary $\varepsilon_{\boldsymbol{s}}$. All data samples are used to determine the decision boundary, yet, data samples from successful trials are also used to update the models of the state classifiers which affects the dataset generation (Eq. 6), such that similar feature readings become more likely. Given the system assumptions, our proposed ODMS does not require any threshold tuning, feature scaling, or additional data annotation. The single user intervention, stopping the system at failure conditions, is sufficient. The decision boundary for failure case detection only exists for a fraction of an ASM due to the state independence assumption. This is theoretically sufficient to learn to predict failure cases, practically this would require to demonstrate unwanted behavior multiple times to cover a reasonable space of an ASM. Therefore, we assume that the dataset (Eq. 6) can be enhanced, by assigning failure examples to neighboring states. Based on the same assumption the learned failure detection decision boundary can be transferred to close by parts of an ASM, with vanishing impact based on the difference in ASM progress, e.g. exponential decay. Since the likelihood statistics of successful trials are different for different states of an ASM, we propose to propagate the decision boundary as scaling factor in terms of one standard deviation of a Gaussian fitted to the likelihood predictions for all successful trials, see Fig. 6. Doing so one must observe an equally unlikely data change at a close by state, for which otherwise no decision boundary would exit, to classify feature readings as failure cases.

D. Decision Making

The box *Online Decision Making* in Fig. 2 visualizes the decision making process. First, the state of the currently executed ASM and independently for all possible successor ASMs is estimated using our probabilistic classifier for every state (Fig. 5). If no failure is classified for the most likely state of the currently executed ASM, the results for the most likely successor are ignored and the execution of the current ASM continues. Otherwise the most likely successor state has to be analyzed by applying the corresponding failure classifier. In case of failure classification for both the most likely current and successor state our system halts, otherwise the ASM is switched to the best successor ASM and state. Although the most likely successor state is only required if a failure for the current ASM is classified, continuous classification of the most likely successor enables further post-processing, e.g. verifying the consistency of the successor, which we want to investigate in future work. Switching could also be done solely based on the likelihood of the current and the best successor state. Perceptual aliasing and system stalling are two reasons to avoid this methodology. For example, a manipulation graph for robust grasping might require several alternative ASMs, but for all ASMs the final configuration might be very similar.

For each decision, our state classification complexity scales linear with the number of considered states. Our proposed failure classification is constant with respect to states and features, because it leverages the information provided by the state classifiers.

V. Experiments

The experimental setup (Fig. 1) consists of the ARM-S robot, a dual armed robot of two Barrett WAMs and a sensor head. For details about the platform we refer to http://thearmrobot.com/aboutRobot.html. The considered manipulation task consists of unscrewing a cap of a bottle as well as to screw the cap back on. This task requires very dexterous manipulation capabilities given the hardware constraints of the robot, the three-fingered Barrett Hands with four degrees of freedoms, and a very small object compared to the size of the hand. All of our experiments were conducted with the same parameters as described in Sec. V-A. The presented results only reflect a subset of our experiments to illustrate different properties of our proposed method. High resolution videos of more experiments are available at https://vimeo.com/117512319 and https://vimeo.com/117515950, also showing how to screw the cap back on the bottle.

A. Task Demonstration and System Parameters

For the exemplary bimanual manipulation task we teach the required ASMs in the sequence, *grasp*, *unscrew*, and *retreat*. Every ASM is executed at least 5 times to gather the required statistics. As soon as the cap of the bottle comes off the first time during the execution of *unscrew*, we stop the execution and teach a new *remove* ASM. Therefore, we obtain a manipulation graph similar to Fig. 3. For each trial execution the ASM, contains the previous and next manipulation graph

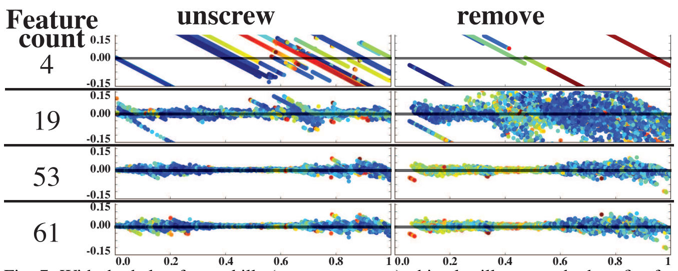

Fig. 7: With the help of two skills (*unscrew*,*remove*), this plot illustrates the benefit of a rich set of features for state classification. The position of each data point indicates the estimated DMP progress (y-axis) vs. the deviation from the ground truth DMP progress (x-axis). Therefore, the closer the points are to the zero-line, the better the prediction. The colors indicates the negative log-likelihood, where blue means small therefore high confidence and red vice versa. With just 4 features, (right finger joints) it is impossible to classify the right state. Using 19 features, most similar to [12], the state classification is possible for *unscrew* but still impossible for *remove*. Integrating virtual vision based features (53) we achieve a good correlation. Adding audio features (61), not providing any further task information, does not degrade our system performance.

node. We teach the DMPs for the corresponding ASMs similar to the methodology described in [11, 12]. First, movements are learned from kinesthetic demonstrations and encoded into DMPs. Second, learned DMPs are executed multiple times on the robot and experienced sensor information are accumulated. Third, the mean position and force trajectories are computed and re-encoded into DMPs. During execution, the end-effector and finger positions as well as forces are controlled and adapted as described in [11], i.e. only positional and force trajectories are encoded into DMPs. Given the re-encoded DMPs, the experienced 61 features (see Fig. 4) from at least 5 trials are linearly re-sampled (30 Hz) and stored time aligned with each DMP to form ASMs.

In contrast to [12], end-effector trajectories are encoded into the start frame of the movement. This encoding scheme generates movements that are locally similar to the demonstration. It thus enforces a strong correlation among subsequent movements despite changing absolute coordinates. End-effector force/torque trajectories are encoded in the goal frame of the movement (see [11]). This ensures that the desired forces/torques are independent of the (global) end-effector pose. We track the target object in real-time using the algorithm proposed in [17], matching 3D information from the depth camera against the geometric model of the bottle. Based on the object pose we enrich our ASM representation with 10 additional distance based features between the manipulator and the surface of the tracked target object, e.g. the fingertip to bottle, palm to bottle, finger segment to bottle distance (Fig. 4 distances). Parallel to using vision based virtual features, our system integrates vision sensors to improve the estimation of the robot end-effector pose, directly improving low-level feedback control. Therefore, we use fiducials, round markers, to visually track the hands during manipulation, arranged around the wrist (Fig. 1). The fiducial detector [2] is used in combination with triangulation to estimate the 3D position of each marker to obtain a least-squares fit of the hand pose. Together with the known control input, these measurements are used in a Kalman filter to smooth the position estimates over time. In addition, we use the lower 8 Mel-frequency cepstral

[2]The fiducial detector has kindly been provided by Paul Hebert, Jeremy Ma, and Nicolas Hudson from the Jet Propulsion Laboratory, Pasadena.

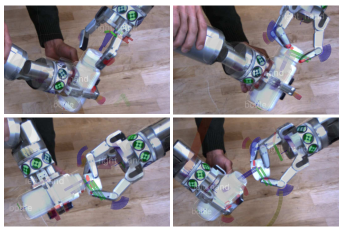
Fig. 8: Snapshots of the robot performing the bottle opening task despite perturbation introduced by moving the left hand of the robot. The compliance of our control framework including DMP adaptation as well as closed loop visual tracking of the bottle and both hands enable the robot to give in without loosing necessary accuracy to perform the task. The corresponding video is available at https://vimeo.com/117512319#t=3m50s.

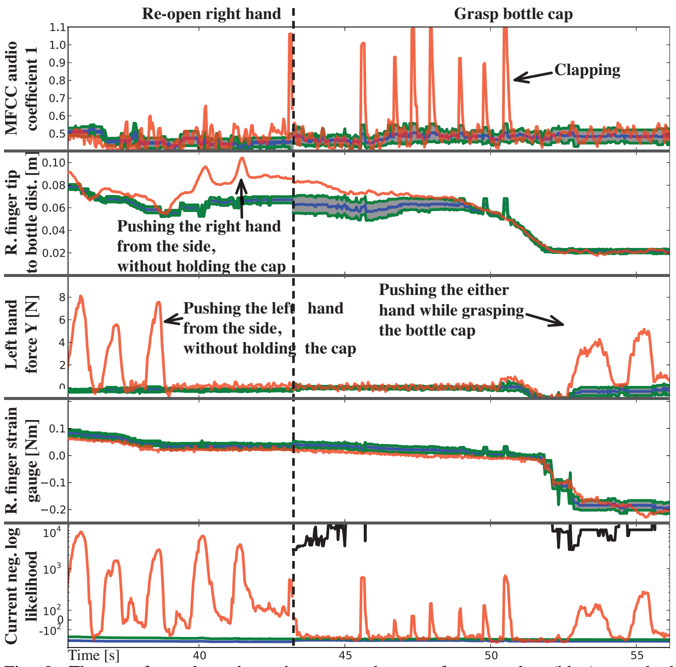

Fig. 9: The top four plots show the expected mean feature value (blue), standard deviation (green), and the currently processed feature value (red) of a subset of all 61 processed features. Transitions between subsequent ASMs are indicated by horizontal dashed lines. The selected features are the first MFCC coefficient, the distance of the right fingertip of the right hand to the visually tracked bottle, the Y component of the experienced force at the wrist of the left hand, and the torque experienced at the right finger knuckle of the right hand (top to bottom). Note, due to space constraints the plot only shows four out of 61 features, nevertheless, all features (auditory,visual, and haptic) are used for classification. The negative log-likelihood (bottom) shows that each individual perturbation of each sensor modality (top three plots) has been scaled appropriately since each perturbation became noticeable. The corresponding video is available at https://vimeo.com/117512319#t=3m25s.

coefficients (MFCC) [3] as audio features. They sufficiently cover the frequency spectrum of interest. To obtain the state classifiers for each ASM, the mean and variance are computed as described in Sec. IV-B, using only successful trials. We obtain the failure classification thresholds ε_s for all ASMs for which erroneous trials exist, using the automatically generated datasets (Eq. 6) with the last $m = 5$ samples of erroneous executions and all corresponding positive trials, as described in Sec. IV-C. Thresholds ε_s are propagated to neighboring states using exponential growth ($\varepsilon_{s_i} = \min(\varepsilon_{s_i}, 1.05^{|i-k|}\varepsilon_{s_k})$), since we are optimizing the negative log likelihood. Thresholds are always encoded in terms of the state variance of the negative log likelihood based on the generated datasets (Fig. 6).

B. State Classification and Feature Importance

Our proposed method is based on the assumption that every state can be modeled individually and identified given the current sensor readings. Fig. 7 illustrates that our method is capable of state classification, despite the strong independence

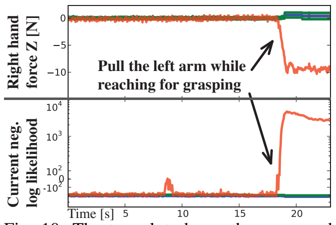

Fig. 10: The top plot shows the measured right wrist force in Z direction (red) as well as the expected mean (blue) and standard deviation (green). The intentional perturbation is immediately detected as shown in the lower plot indicated by the rise of the negative log-likelihood (red). Without demonstrated failure cases our system assumes that the introduced perturbation, although highly unlikely, does not constitute a problem.

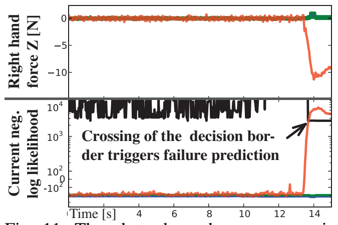

Fig. 11: The plots show the same experiment as in Fig. 10 *after* the data obtained in Fig. 10 has been processed as an unwanted (erroneous) execution. Given this failure case example data, our approach updated the decision border (black) which now enables our system to correctly detect the unwanted failure condition indicated by the arrow. Now, our system immediately stops the execution, as shown in the video available at https://vimeo.com/117512319#t=4m25s, if a similar perturbation is detected.

assumption. Results are obtained by leave-one-out cross validation, training the state classifiers for *unscrew* and *remove* on at least 5 successful trials. The correlation between the classified state and the DMP progress is less good at the beginning, the end of an ASM execution, and when object contact happens. Such events rarely happen at exactly the same time which is why the DMP progress is not a good estimate for the current state. One key difference to other approaches [12, 9, 8] is that we propose to always use a high dimensional feature representation and automatically infer the feature importance. Fig. 7 supports the hypothesis that more features do improve the system performance, although not required for all skills. The lower two plots show that our method does not degrade in performance even if we add features which contain no further information for the task. This supports the hypothesis that our method is capable of learning the changing feature importance for an ASM and thus, manual task specific feature selection is less crucial.

C. Robustness through Perception-Action Loop

Robustness of our system is achieved through closing two control loops. The low-level loop implements a compliant control framework, using the DMP adaptation mechanism as proposed in [11] and integrating information from visual based tracking of both hands and the bottle. To demonstrate the low level-adaptability, we introduce a large perturbation by moving the left arm during the manipulation task as depicted in Fig. 8. Despite this perturbation the right hand is able to maintain contact and continue the DMP execution due to vision in the loop, adapting the goal of the DMP online. The high-level control loop is capable of detecting perturbations of different sensor modalities as shown in Fig. 9. Our method is able to automatically determine the importance of all available 61 features and fuse the information into a one dimensional signal, for which failure detection is easily achieved.

D. Failure Demonstration and Detection

Adding recovery behaviors incrementally in the case of failure, e.g. due to external perturbation, is the key ingredient to add robustness to task execution. To demonstrate a typical use case, we start our manipulation graph and pull the left

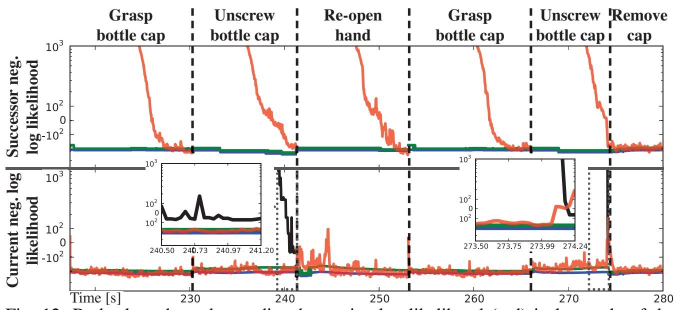

Fig. 12: Both plots show the predicted negative log-likelihood (red) in log scale of the most likely state within the currently active ASMs. The bottom plot shows that the system is certain about the task progress since the predicted negative log-likelihood remains close to the expected mean. The top plot shows that the successive ASM becomes more likely towards the end right before the transition to this ASM. Note that the duration of the second unscrew movement is shorter than the initial unscrew movement since our approach correctly detected that the cap came off the bottle. The two overlaid plots show a closeup of these two events. The subtle event of lifting the cap has been detected as the negative log-likelihood (red) passed the decision border (black).

arm during the first *grasping* attempt. Although our system classifies the event as highly unlikely, it will not detect a failure condition initially, see Fig. 10. This design choice, to assume that even unlikely events will not result in failure conditions if not known otherwise, allows easy system bootstrapping, but is not required for our system. After retraining the classifiers to corresponding ASM (Sec. IV-B and IV-C), taking this erroneous execution into account, our system detects perturbations of similar magnitude and will stop (Fig.11). The only user information required to teach this failure detection is stopping the current ASM execution before it has finished. If a possible successor/recovery already exists, the user can add the connection in the manipulation graph to enable automatic recovery.

E. Failure Detection and Recovery

Fig. 12 shows a sequence of automatically chosen ASMs, successfully unscrewing the cap of a bottle without external perturbation. This experiment was part of of a longer experiment to unscrew the bottle. During the course of which we repeatedly screwed the cap back on the bottle (during ASM *retreat* execution), as shown in the corresponding video. This demonstrates that our method automatically detected the subtle event, when the cap becomes completely loose from the bottle, prior to finishing the ASM execution and successfully switched to ASM *remove*. Fig. 13 shows the prediction of *unscrew* at different stages of an successful attempt to unscrew a cap of a bottle with manually introduced unknown perturbations. For this experiment an additional ASM has been taught named *recover*. This ASM has been added to the manipulation graph as additional outcome of *unscrew* after stopping the execution of *unscrew* at the beginning of an external perturbation similar to the one in this experiment. Our method successfully determines if the sensor information indicates a failure condition and selects the right successor ASM, either *recover*, *retrieve*, or *remove*. Again, the skill execution time is different due to the online detection of failure conditions. Some failure conditions are detected while not even reaching half of the expected skill execution time (Fig. 13, left), not possible without closing the high-level feedback loop [12, 9].

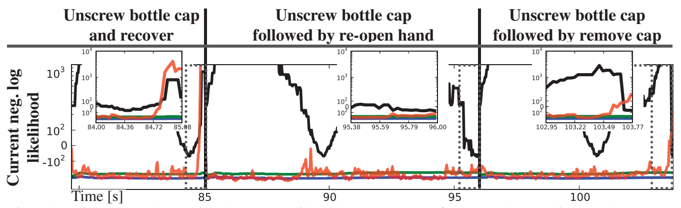

Fig. 13: The plots show the three considered outcomes of one real experimental attempt of unscrewing the bottle, triggering the learned recovery behavior after pulling the left arm (left), triggering the retreat behavior because the cap of the bottle is not yet detached (middle), and triggering the remove behavior immediately after cap has correctly been detected to have detached from the bottle (right). Three small plots at the bottom show a close up of the negative log-likelihood at the point of switching to the corresponding successor ASM (dotted line). The solid horizontal lines indicate transitions. Note, the intermediate ASMs have been left out due to space constraints. The complete experiment along with corresponding plots is available at https://vimeo.com/117512319#t=4m52s.

VI. Discussion and Conclusion

In this paper we proposed an incremental data-driven approach to autonomous manipulation. By leveraging stereotypical movements and an object-relative encoding, we can learn a classification-based real-time online decision making process. Our experiments demonstrate qualitatively that the proposed method is capable of detecting otherwise catastrophic perturbations and failure conditions in real-time, using 61 features from different sensor modalities. Furthermore, our system automatically triggers recovery behaviors if needed, closing a high-level feedback loop. These results were achieved without the need for any task-specific programming. Therefore, we believe that our method can enable non-expert users to demonstrate robust strategies for executing a wide range of robotic manipulation tasks.

Our system assumes that consecutive readings are independent. This assumption allows our method to run all *state classifications* in parallel and address changing feature importance throughout the execution of an ASM. However, relaxing the independence assumption could be used to reduce the computational complexity, e.g. clustering similar consecutive states and using a progress based prior.

Failure classification is another important building block for our ODMS. In this paper, we assume that perturbations only represent failure cases if previously demonstrated to the system. This assumption enables easy teaching of the system but it might be necessary to introduce a global failure threshold at system deployment. In addition, failure classification is not considering the features but the unnormalized posterior. Therefore perturbations, affecting this quantity, trigger failure cases even if completely different features are affected. This is a very conservative assumption and it might be beneficial to add failure cases as additional classes. Yet, this would add more complexity and open hyper-parameters, since the model parameters for a probabilistic classifier cannot be obtained in the same manner as for successful trials.

We think an interesting avenue for future work would be to learn a lower dimensional sensory representation based on unlabeled data, which can be gathered in larger quantities. Such compression of the time series representation could enable automatic graph inference, at least locally, and the prediction complexity would be rendered sub-linear with respect to the number of discrete time-steps in an Associative Skill Memory.

References

[1] O. Brock. Berlin Summit on Robotics. In *Conference Report*, Berlin Summit on Robotics, December 2011.

[2] Corinna Cortes and Vladimir Vapnik. Support-vector networks. In *Machine Learning*, 1995.

[3] S. Davis and P. Mermelstein. Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 28, 1980.

[4] M. Dogar and S. Srinivasa. Push-Grasping with Dexterous Hands: Mechanics and a Method. In *IEEE/RSJ Intl Conf. on Intelligent Robots and Systems*, 2010.

[5] A. J. Ijspeert, J. Nakanishi, P. Pastor, H. Hoffmann, and S. Schaal. Dynamical Movement Primitives: Learning Attractor Models for Motor Behaviors. *Neural Computation*, 25(2), 2013.

[6] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal. STOMP: Stochastic Trajectory Optimization for Motion Planning. In *IEEE Intl Conf. on Robotics and Automation*, 2011.

[7] Ron Kohavi et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, volume 14, pages 1137–1145, 1995.

[8] O. Kroemer, H. van Hoof, G. Neumann, and J. Peters. Learning to predict phases of manipulation tasks as hidden states. In *IEEE Intl Conf. on Robotics and Automation*, 2014.

[9] S. Niekum, S. Chitta, B. Marthi, S. Osentoski, and A. G. Barto. Incremental Semantically Grounded Learning from Demonstration. In *Robotics: Science and Systems (R:SS)*, 2013.

[10] P. Pastor, M. Kalakrishnan, S. Chitta, E. Theodorou, and S. Schaal. Skill Learning and Task Outcome Prediction for Manipulation. In *IEEE Intl Conf. on Robotics and Automation*, pages 3828–3834, 2011.

[11] P. Pastor, L. Righetti, M. Kalakrishnan, and S. Schaal. Online movement adaptation based on previous sensor experiences. In *IEEE/RSJ Intl Conf. on Intelligent Robots and Systems*, 2011.

[12] P. Pastor, M. Kalakrishnan, L. Righetti, and S. Schaal. Towards Associative Skill Memories. In *IEEE-RAS Intl Conf. on Humanoid Robots*, 2012.

[13] Nathan Ratliff, Matthew Zucker, J. Andrew (Drew) Bagnell, and Siddhartha Srinivasa. CHOMP: Gradient Optimization Techniques for Efficient Motion Planning. In *IEEE Intl Conf. on Robotics and Automation*, 2009.

[14] Irina Rish. An empirical study of the naive Bayes classifier. In *IJCAI workshop on empirical methods in artificial intelligence*, volume 3, 2001.

[15] Jivko Sinapov, Connor Schenck, Kerrick Staley, Vladimir Sukhoy, and Alexander Stoytchev. Grounding semantic categories in behavioral interactions: Experiments with 100 objects. *Robotics and Autonomous Systems*.

[16] J. Weisz and P.K. Allen. Pose Error Robust Grasping from Contact Wrench Space Metrics. In *IEEE Intl Conf. on Robotics and Automation*, 2012.

[17] M. Wüthrich, P. Pastor, M. Kalakrishnan, J. Bohg, and S. Schaal. Probabilistic Object Tracking using a Depth Camera. In *IEEE/RSJ Intl Conf. on Intelligent Robots and Systems*, 2013.