Achieving Robustness by Optimizing Failure Behavior

Manuel Baum Oliver Brock

***Abstract*— The most prominent criterion for learning of manipulation skills is the optimization of task success, modeled as expected reward or probability of success. This is sensible if we only want to optimize a single controller. But if learned manipulation primitives are used as modules in a larger system, then it is also important that their generated sensor traces facilitate recognition of action-outcomes. Optimization solely for expected success of a primitive does not guarantee this. We demonstrate a simple example for optimization of actions towards observability, combined with optimization for expected success. Our experiment is a manipulation task with a soft manipulator, where an action primitive is learned such that its generated sensor trace helps a classifier to distinguish task success and task failure. The experimental results indicate that adding auxiliary forces to the original manipulation primitive can indeed facilitate outcome recognition for manipulation tasks.**

I. INTRODUCTION

To be robust, robotic systems must detect and compensate for errors that occur during action execution. For simple behaviors, motion primitives achieve this based on feedback control, by descending the gradient of a control function [1]. This works well as long as the behavior can be described by a single controller. For more complex behavior, however, controllers are often composed and sequenced into hybrid automata [2]. In this context, it becomes crucial for the controller to fail in a way that indicates to the hybrid automaton how the failure should be reacted to. It therefore would be desirable to design low-level behaviors (i.e. controllers) in such a way that failures requiring different reactions can easily be distinguished.

Error detection during controller execution can be modeled as a classification problem, where sensor traces are mapped to discrete labels: either success or different types of failures. A straightforward approach could look like this: Execute a motion primitive multiple times to gather sensor-traces, then label these as success or failure and train a classifier to predict these labels. If the recorded sensor readings carry enough information, this approach seems adequate. But it can also be the case that the action execution does not generate rich enough sensory data and it becomes very difficult or even impossible to find a good mapping.

We propose a method for optimizing motion primitives for their ability to fail in distinctive ways—*in addition* to optimizing them for expected success. To do this, it is crucial to understand how the parameters of an action affect the quality of error-detection based on the sensor signals the action generates. The method finds action parameters that achieve a high success rate and which generate distinctive sensor signals for distinctive failure modes.

Both authors are with the Robotics and Biology Laboratory, Technische Universität Berlin, Germany. We are grateful for funding by the DFG (grant BR 2248/3-1).

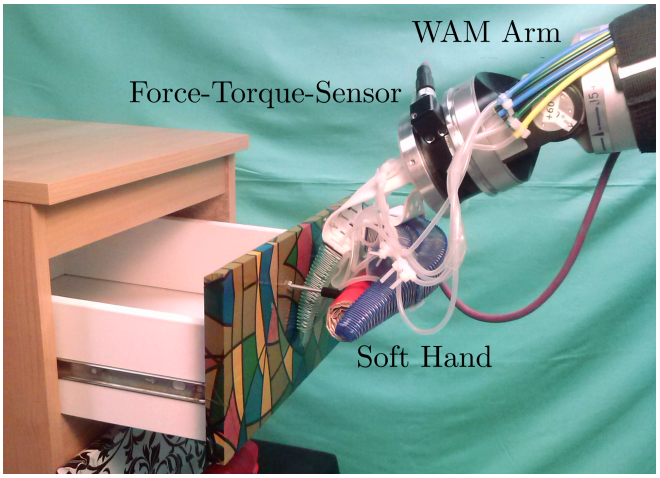

Fig. 1. A WAM robotic manipulator opening a drawer using a soft manipulator and feedback from a force-torque sensor

In manipulation experiments on a real-world robotic platform (Figure 1), we optimize the parameterization of a primitive for success and interpretability of the error. We evaluate this in the context of a drawer-opening task. Our results show that optimizing an action for success *and* distinctive action outcome classification simultaneously contributes to the robustness of the overall robotic system.

II. RELATED WORK

Action outcome estimation for robotic manipulation has been investigated before. We will compare our approach to this problem setting to other works on failure detection in Section II-A. Our approach is an instance of Interactive Perception, which is reviewed in Section II-B. It solves a similar problem as belief space planning, which is briefly covered in Section II-C.

A. Outcome estimation / failure detection

Learning general behavior for robotic manipulation is a difficult problem. It is commonly decomposed by building a plan out of multiple action primitives, which all solve a specific sub-problem of the complex task. Hybrid automata or manipulation graphs [3] are dynamic plans that switch between action primitives based on task progression. To be robust, such dynamic plans should incorporate means to detect if a controller execution failed. If a failure is detected, it can then switch to a recovery strategy. We will now discuss data-driven and engineering approaches to failure-detection.

1) Engineered failure detection: When engineers combine robotic action primitives into a plan, they use their knowledge about the task dynamics, the robot's sensory capabilities, and the expected sensory signals. They define rules for switching between these primitives and some of these rules may use specific failure detectors to increase the system's robustness. As an example, one autonomous manipulation system [4] used the frequency spectrum of its measured force-torque signal to detect if an electric drill was powered on. It also used that signal to detect if it successfully actuated a paper stapler. The system in [5] also achieved robustness by augmenting the core behavior of the system with engineered failure detection and compensation routines.

2) Data-driven failure detection: Robots can detect failures in a data driven way by monitoring the action execution and detecting anomalies [6]. Learning the structure of manipulation graphs [3] enables switching between action primitives and activating recovery actions in case of failure. Previous data-driven approaches, to the best of our knowledge, all follow the same order of steps: Execute a motion primitive multiple times to gather sensor-traces, then label these as success or failure and train a classifier or an anomaly detector to predict if an action is failing. The quality of this final learning step strongly depends on the quality of the sensor information that is used for training. If the robotic system's behavior does not generate rich sensory training data, then such a separation of first learning actions and afterwards learning action outcome recognition becomes problematic.

B. Interactive Perception / Active Sensing

Interactive Perception emphasizes that some information can only be revealed by interacting with the environment. Instead of dividing behavior into sequenced sensing, planning and acting steps, Interactive Perception approaches actively generate the sensory information that the system requires. Problems tackled with Interactive Perception include object segmentation, pose estimation, articulation model estimation, grasp planning and more. See [7] for a survey on this topic.

As an example from object segmentation, simply being able to push objects apart can facilitate the segmentation problem significantly [8]. We argue that adding a small desired force to the action primitive can also make outcome recognition easier. It can increase the distance between different classes of action outcomes in the sensory space. Now the challenge is to find actions that achieve such separation.

Our approach represents a novel instance of Interactive Perception. We want robots to act in such a way that reveals task-specific information about success or failure of the *interaction* itself. Existing literature on Interactive Perception focuses on interaction to reveal properties of the *environment*.

C. Belief space planning

While we propose to adapt actions based on past experiences, agents can also choose informative actions by planning. In classical planning, an agent needs to plan a sequence of actions in order to transition from a start state to a goal state. Uncertain actions and observations can make it necessary to represent the state as a belief over states. Planning in belief state allows agents to perform actions that achieve the goal state *and* reduce the uncertainty over states [9]. Like our approach, belief space planning can integrate perception and action, so that acquired information supports manipulation tasks [10], but requires an explicit transition model to predict the effects of the robot's actions. This is not the case for our learning based approach.

III. Method

We present our approach to action outcome recognition in this section and want to start by pointing out the high-level structural difference between other approaches and our method with the help of Figure 2. The common solution to the action outcome recognition problem is a two-step process: First execute the action primitive several times to generate sensor traces, then design or learn a classifier that predicts success or failure. When the system adapts to recognize action outcomes, only the final stage is active. The top part of Figure 2 illustrates this. In contrast, the lower part of Figure 2 illustrates which parts of the system actively contribute to action outcome recognition with our approach. We believe that robots should learn outcome recognition and action primitives not one after the other, but at the same time. The robot's actions strongly influence the sensor feedback, and the sensor traces in turn dictate how difficult the outcome recognition problem is. We can use this dependency and adapt the robot's actions to facilitate the outcome recognition problem. Thus we propose to feed back the error from outcome recognition to adapt the action primitive. Such a closed loop learning scheme assures that the learned action supports outcome recognition. Compared to the common two-step approach, the complete pipeline from action to outcome recognition is actively involved in improving outcome recognition.

The parametrized action primitive we used is described in Section III-A. We further need to measure how well action parameters facilitate task outcome recognition. This classification loss is explained in Section III-C. Our system should only adapt its actions for perception as long as this does not increase probability for failure. We need to formulate this trade-off as a combined task failure and recognition loss. This combined loss is explained in Section III-D.

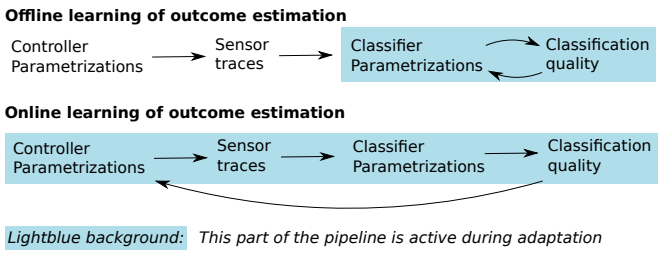

Fig. 2. Two different loops to adapt a primitive for failure detection. The upper diagram shows a loop where classification is learned after interaction. The lower diagram shows a loop where the primitive is adapted to improve classification. Different parts of the pipeline are active during adaptation.

A. Action Primitive

We use an operational space impedance controller [11], with a target pose that is already specified a-priori. It is initialized such that the controller can approximately solve the task already in the beginning of our experiments. This simple, approximate and parametrized task solution is chosen for simplicity and to test our basic hypothesis that outcome recognition can be facilitated by action adaptation. The controller's free parameters are a constant desired force in x direction F_{x_d} and a constant desired torque around the y-axis T_{y_d}. The parameters $w = (F_{x_d}, T_{y_d})$ are being optimized to improve the performance of our action outcome recognition classifier. The system is optimized by minimizing a classification loss $L_C(w)$ on the classifier's predictions by adapting the action parameters w. Before we explain the loss, we will now describe the classifier.

B. Classifier

We use a k-nearest neighbor (k-NN) classifier to predict class-labels y_i for a time-series vector $\mathbf{x_i}$ of variable length T_i. The training phase of this classifier is simple: just store the training data $(\mathbf{X}, \mathbf{Y})$, where $\mathbf{X} = \{\mathbf{x_i} | \mathbf{x_i} \in \mathbb{R}^{T_i}\}$ and $Y = \{y_i | y \in \mathbb{R}\}$. For predicting y' for an unseen $\mathbf{x}'$, the classifier needs a measure $d : \mathbb{R}^{T_i} \times \mathbb{R}^{T_j} \rightarrow \mathbb{R}_{\geq 0}$ to compute the distance between the input sample and each of the training samples. This distance is used to compute the k nearest neighbors of x'. We compute a histogram over the class-labels of the nearest neighbors and predicts the label with the highest count as the unseen data-point's class.

As the distance-metric d needs to handle sensory time-series of different lengths and with possible shifts and scaling in time, common euclidean distance-measures are not applicable. Instead, we use dynamic time warping (DTW) as distance-measure between the samples. Dynamic time warping is a parameter free and very effective measure at comparing time-series [12]. The core idea behind DTW is to find a transformation that locally speeds up or slows down time in order to align two time-series. The alignment should minimize the sum of euclidean distances over all time-steps between the time-series. We use this minimum distance as the distance-measure for the k-NN classifier.

C. Classification Loss

The classification loss $L_C(w)$ should penalize those action parameters w that lead to similar sensor signatures for different outcomes. Optimization of this loss should lead to actions that make it easy to classify. For this reason, we do not use the predicted label y' directly to compute the classification loss, but use the classifiers certainty for predicting the ground truth class-label. We need to make the prediction of k-NN probabilistic. Choosing uniform weights for the voting of the k nearest neighbors, the certainty of the classifier to predict class c is computed as

$$p(y = c|x) = \frac{1}{k} \sum_{i=0}^{k} \delta_{c,c_k}, \quad (1)$$

where c_k is the true class of the kth nearest neighbor and $\delta_{i,j}$ is the Kronecker delta.

The loss over all N roll-outs for a specific choice of action parameters w is defined as

$$L_C(w) = \frac{1}{N^w} \sum_{i=0}^{N^w} |y^w - p(y_i = 1|X^w, y^w_{0:i}, y^w_{i+1:N})|, \quad (2)$$

where N^w is the number of roll-outs, y^w is the ground truth for these samples, X^w are the sensor traces, p is the certainty of our classifier, and the label for success is 1. This can interpreted as leave-one-out cross-validation for the action parametrization w as a hyper-parameter for the classification task.

D. Combining the losses

For each w, we execute the action N times. A human supervisor labels the rollouts as success or failure. This data is used to compute the classification loss L_C. To penalize action parameters that lead to failure we compute a loss

$$L_S(w) = \frac{1}{N^w} \sum_{i=0}^{N^w} 1 - y^w. \quad (3)$$

The losses L_S and L_C are further combined into a single loss L by

$$L(w) = L_S(w) + L_C(w) - L_S(w)L_C(w), \quad (4)$$

which penalizes parametrizations with high probability for failure and miss-classification. We chose this loss function over a simple summation of the individual losses because it provides a balance between L_C and L_S. This is important to avoid parameters w where classification loss is minimized by unbalanced data-sets which contain only failures. The term $-L_S(w)L_C(w)$ gives those parameters w an advantage that have a balanced proportion between miss-classification and task failure. On the other hand it also gives a disadvantage to those parametrizations that succeed often, but if this was a problem we could still add linear coefficients to the individual terms in the sum. Minimization of this loss function should yield action parameters that guarantee success for the action execution itself and success in outcome recognition. As a result, the primitive promises success as an individual unit and as a building block in an autonomous system.

IV. EXPERIMENT

For a specific manipulation task, we tested a complete pipeline of robotic action execution and action outcome recognition under different action parameters w. We compared the classification quality for different w to see if action adaptation can indeed facilitate action outcome recognition.

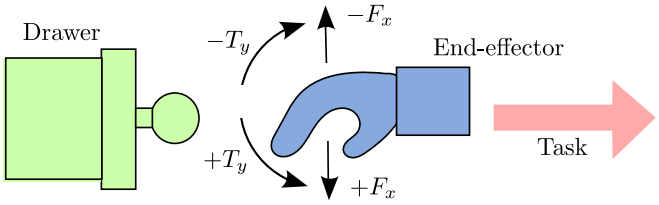

Fig. 3. The end-effector, desired forces and the drawer

A. Setup

For our experiments, we used a compliant, soft manipulator [13], mounted to a Barrett WAM arm. A force-torque sensor is used to measure forces and torques between the soft end-effector and the WAM arm. This setup can be seen in Figure 1 and also in a video at `https://youtu.be/KO3XfspsXJY`. A soft end-effector, as the one we use, has many advantages for robotic manipulation, but can also raise difficulties: As it is highly underactuated, the configuration of the end-effector is not easy to measure. Additionally, the force-torque signal is more difficult to interpret than with a rigid end-effector, because the Soft Hand vibrates when the arm is not in contact and the signal is damped when the manipulator is in contact. This makes it especially important to generate rich feedback.

B. Task

We evaluate our method for a drawer opening task. See Figure 3 for an illustration that accompanies the following explanation of the task. The solution to this task is to pull the end-effector in negative z-direction. We define an action execution as successful, if the robot opened the door for more than 15 cm and remained in contact with the drawer's handle for the complete duration of the interaction. Ground truth values for success are provided by the user. For each execution of the task, the hand is randomly placed on the handle of the drawer, simulating an approximately uniform distribution of good and bad pre-grasps. The inflation of the pneumatic hand was subject to uniform random sampling, to deliberately introduce failures through imperfect grasps. Additionally, for some of the action executions, the end-effector is disturbed by human intervention so that it loses the grasp. We made sure the number of these disturbances is equal for each of the parametrizations.

The impedance controller we chose as parametrized action primitive has a high stiffness in z-direction, so that it can pull open the drawer in the direction of the user defined target pose. The stiffness for translation along x-axis and rotation around y-axis is also relatively high, while it is relatively low for all other dimensions. The free parameters T_{y_d} and F_{x_d} are a desired sensed torque around y and desired sensed force in x direction. Depending on the values of these parameters, the robot will move its end-effector in the respective direction until the sensed values reach these desired values. We expect that some action parametrizations will lead to more robust action executions than others. The robot can solve the task more robustly if it pushes down on the handle and applies a desired torque that rotates the end-effector further into the gap behind the handle. In contrast, if it has a desired force upwards, away from the handle or a desired torque that rotates the end-effector out of the handle gap, it will lose the grasp more often.

In order to increase statistical support and also to make the classification problem more difficult, we regard the six available force-torque signals independently. For each of the individual learning and recognition problems, the robot can only use one of the signals. So it has to optimize its action such that the single signal it can observe helps it to discriminate success and failure. We expect that for each of these artificially isolated force or torque sensorizations, different action parametrizations will maximize outcome recognition success. The two-dimensional parametrization of the action is discretized in a 5x5 grid, where the desired force is uniformly subdivided in the interval $[-5.0N, 5.0N]$ and the desired torque is uniformly subdivided in the interval $[-0.75Nm, 0.75Nm]$. For each of the 25 parameter sets in the parameter-grid, the action was executed 16 times, summing up to 400 executions in total. We chose the parameter k for the nearest neighbor classifier to be $k = 3$. We also tested $k = 5, 7, 9$, which either led to similar or slightly worse results.

V. RESULTS

Figure 4 shows the success loss L_S for the parameter grid. This plot shows that a positive desired force in x-direction and a positive desired torque around the y-axis both increase the probability of success. In contrast, the tested action executions fail often for negative F_{x_d} and T_{y_d}. Unsurprisingly, action parameters influence success ratio.

The classification loss L_C indicates how easy it is to predict action outcomes for action parameters w. Figure 5 shows this loss for each of the six force-torque sensor signals. A loss of $L_C = 0$ would indicates that there are no miss-classifications. For example, this is the case for 5 different parameter sets, when force signal F_x is used. Those parameter sets with desired force $F_{x_d} = 5.0$ can create sensor traces for sensed F_x that are easy to classify. But the sensor traces for $F_{x_d} = -5.0$ are trivial to classify, as the highly unbalanced data for these action parameters only contains failures. For other action parameters, the classification loss can be as high as 0.5, which is basically random

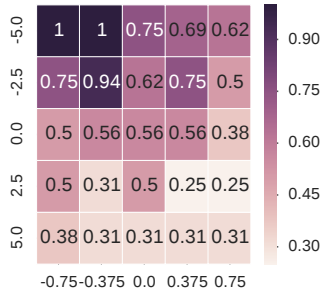

Fig. 4. The loss L_S, defined as the ratio of success, plotted for different action parameters. Brighter colors indicate higher rate of success and correspondingly lower loss.

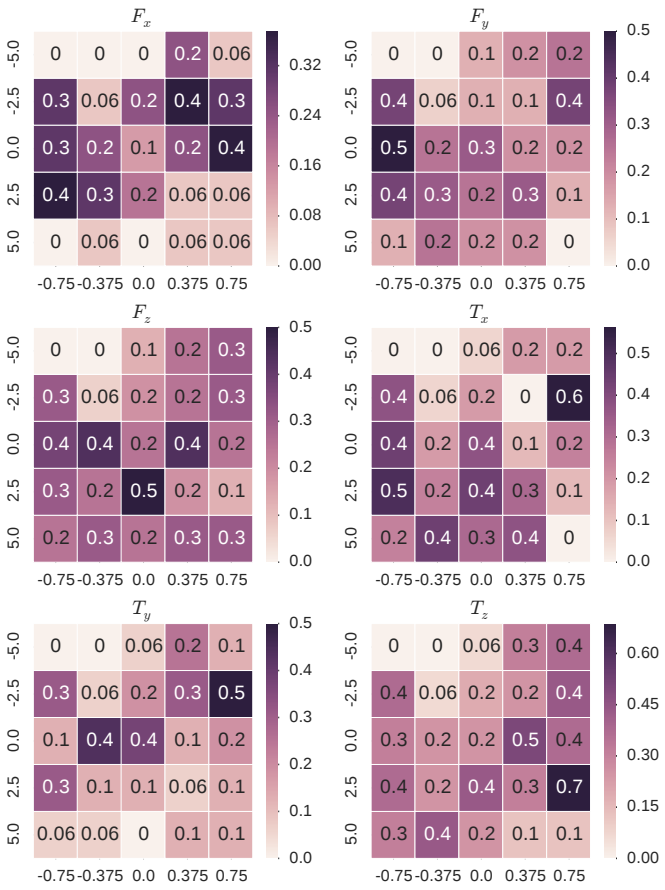

Fig. 5. The classification losses for different sensor signals. Matrices F_x,F_y and F_z show classification losses for the force signals in x-, y- and z-direction. Matrices T_x,T_y and T_z show classification losses for the torque signals around x-, y- and z-axis.

guessing. This shows that, different action parameters can indeed facilitate the subsequent classification problem as we hypothesized. But the fact that action parameters which always lead to failure and make the classification problem trivial also shows, that we need to have a combined loss that relates classification and task success.

Figure 5 also reveals that different sensor capabilities lead to different optimal action parameters. The parameter sets where the classification loss is minimal is different for each of the isolated sensor signals $F_x, ..., T_z$. This shows that information gathering by interaction should adapt to the characteristics of the available sensors. As an example, Figure 6 visualizes sensor traces for T_y, the measured torque around the y-axis. The classes are difficult to separate with the unchanged action parameters ($F_{x_d} = 0N$,$T_{y_d} = 0Nm$), which leads to a classification loss of $L_C = 0.4$ (see Figure 5). Adding a small desired torque $T_y = 0.375Nm$ facilitates discrimination of the classes, which is reflected in a better classification loss of $L_C = 0.1$.

The combined loss L can be seen in Figure 7. Notice the difference between L and the success loss L_S from Figure 4. Several parametrizations with high desired force in x-direction were equally likely to be optimal, if solely optimized for success. Now if we add another term to the

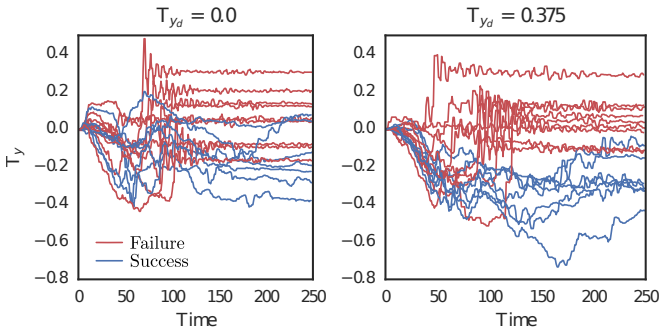

Fig. 6. Two different plots of measured torque around y-axis T_y over time. The two subplots show the signal generated for action parameters $F_{x_d} = 0$ and $T_{y_d} = 0, 0.375$.

loss, namely the classification loss which penalizes actions where it is difficult to recognize failures, there are clearly some parametrizations that are better suited for each of the artificially reduced sensorizations.

To examine if the minimization of this loss leads to a more robust skill, we also have to test the action as part of a larger system. Exemplarily we will show an analysis for sensor modality F_x, but analyses of the other modalities lead to similar results. If we minimize the combined loss L for sensor modality F_x, optimal parameters are in the region of $F_{x_d} = 5.0Nm$ and also $F_{x_d} = 2.5N$ when $T_{y_d} = 0.375Nm$ or $T_{y_d} = 0.75Nm$. To see if this region of minimum loss L corresponds to a more robust overall system, we simulated an experiment. In our Monte-Carlo simulation loop, a system virtually performs the manipulation action, classifies the gathered sensor information as success or failure and only stops the loop if it predicted success. For each parameter set, the action succeeds with the probability we obtained in our real-world experiment. The conditional probabilities for predicting success or failure given the sampled ground truth success or failure were also according to the classification likelihoods we obtained on experimental data. By repeating this process until convergence, we gather statistics about the mean number of action executions the robot performs for the action parameters and can compute the probability of actually finishing with a successful outcome.

Figure 8 shows the outcome of the simulation for a classifier based on sensed F_x. Three main regions can be identified in these matrices: The top-left region, where the loop would end after a successful action, but with a relatively large mean number of required actions. A stripe in the middle where the loop can often end with a failed action, and the bottom region, where the loop probably ends after a successful action and with a low number of actions on average. Notably, the initial interaction with parameters $F_{x_d} = 0, T_{y_d} = 0$ (no additional forces) leads to a significantly higher probability to end in failure than most actions with additional force or torque. This is because of high probability of miss-classification due to relatively poor sensory input. This shows that even a simple system that uses the action primitive can benefit from adapting the primitive for informative feedback.

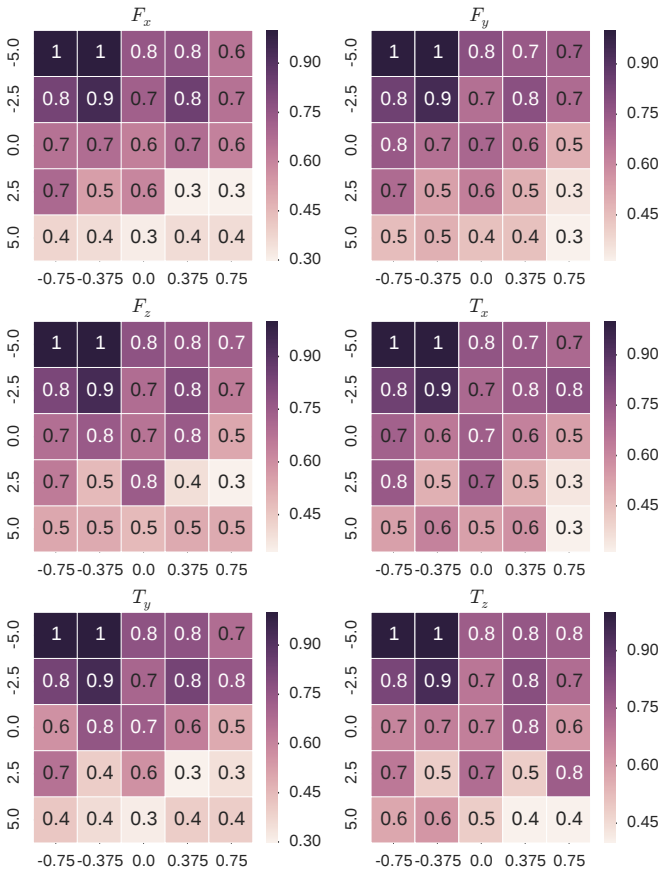

Fig. 7. The combined success and classification loss L for different sensor signals. Matrices F_x, F_y and F_z show L for force signals in x-, y- and z-direction as sensor modality. Matrices T_x,T_y and T_z show L for the torque signals around x-, y- and z-axis as sensor modality.

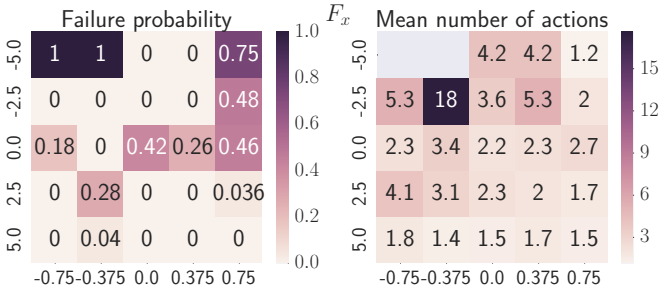

Fig. 8. The outcome of simulating repeated execution of the action, estimating outcome with a classifier on F_x and only ending this loop if success is estimated. The left plot shows probability to end with a failed action, the right plot shows the mean number of action executions.

VI. CONCLUSION

We showed that naive action executions can yield poor sensory feedback which makes failure detection difficult. To resolve this, we proposed to use error feedback from action outcome recognition to adapt the parameters of action primitives. By this, we introduced a novel instance of Interactive Perception, where robots should adapt their actions not only for expected success or to reveal information about the environment, but also to facilitate recognition of their actions' outcomes. As a first step to test this idea, we performed an experiment where a robot opens a drawer under the effect of severe disturbances which introduce failure. We used a classifier to recognize success or failure as action outcomes, and optimized an action primitive for classification success and task success at the same time. We proved that a system that uses such an optimized primitive becomes more robust. This supports our claim that if robot actions can fail, then they should fail in distinctive ways.

As a next step, it would be interesting to try this approach with more powerful action parametrizations. Compliant Movement Primitives as a combination of Dynamic Movement Primitives as task-specific solution for the kinematic part, augmented by Torque Primitives [14] could be a promising replacement for the impedance controller.

REFERENCES

[1] G. F. Franklin, J. D. Powell, A. Emami-Naeini, and J. D. Powell, *Feedback control of dynamic systems*. Addison-Wesley Reading, 1994, vol. 2.

[2] M. Egerstedt, "Behavior based robotics using hybrid automata," in *International Workshop on Hybrid Systems: Computation and Control*. Springer, 2000, pp. 103–116.

[3] D. Kappler, P. Pastor, M. Kalakrishnan, M. Wuthrich, and S. Schaal, "Data-driven online decision making for autonomous manipulation," *Proceedings of Robotics: Science and Systems, Rome, Italy*, 2015.

[4] L. Righetti, M. Kalakrishnan, P. Pastor, J. Binney, J. Kelly, R. C. Voorhies, G. S. Sukhatme, and S. Schaal, "An autonomous manipulation system based on force control and optimization," *Autonomous Robots*, vol. 36, no. 1-2, pp. 11–30, 2014.

[5] C. Eppner, S. Höfer, R. Jonschkowski, R. Martín-Martín, A. Sieverling, V. Wall, and O. Brock, "Lessons from the amazon picking challenge: Four aspects of building robotic systems," in *Proceedings of Robotics: Science and Systems*, AnnArbor, Michigan, June 2016.

[6] D. Park, Z. Erickson, T. Bhattacharjee, and C. C. Kemp, "Multimodal execution monitoring for anomaly detection during robot manipulation," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 407–414.

[7] J. Bohg, K. Hausman, B. Sankaran, O. Brock, D. Kragic, S. Schaal, and G. Sukhatme, "Interactive perception: Leveraging action in perception and perception in action," *arXiv preprint arXiv:1604.03670*, 2016.

[8] K. Xu, H. Huang, Y. Shi, H. Li, P. Long, J. Caichen, W. Sun, and B. Chen, "Autoscanning for coupled scene reconstruction and proactive object analysis," *ACM Transactions on Graphics (TOG)*, vol. 34, no. 6, p. 177, 2015.

[9] B. Bonet and H. Geffner, "Planning with incomplete information as heuristic search in belief space," in *AIPS*, 2000.

[10] L. P. Kaelbling and T. Lozano-Pérez, "Unifying perception, estimation and action for mobile manipulation via belief space planning," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 2952–2959.

[11] N. Hogan, "Impedance control: An approach to manipulation: Part iiimplementation," *Journal of dynamic systems, measurement, and control*, vol. 107, no. 1, pp. 8–16, 1985.

[12] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, and E. Keogh, "Querying and mining of time series data: experimental comparison of representations and distance measures," *Proceedings of the VLDB Endowment*, vol. 1, no. 2, pp. 1542–1552, 2008.

[13] R. Deimel and O. Brock, "A novel type of compliant and underactuated robotic hand for dexterous grasping," *The International Journal of Robotics Research*, vol. 35, no. 1-3, pp. 161–185, 2016.

[14] T. Petrič, L. Colasanto, A. Gams, A. Ude, and A. J. Ijspeert, "Bio-inspired learning and database expansion of compliant movement primitives," in *Humanoid Robots (Humanoids), 2015 IEEE-RAS 15th International Conference on*. IEEE, 2015, pp. 346–351.