Learning to Predict Phases of Manipulation Tasks as Hidden States

Oliver Kroemer, Herke van Hoof, Gerhard Neumann, and Jan Peters

***Abstract*— Phase transitions in manipulation tasks often occur when contacts between objects are made or broken. A switch of the phase can result in the robot's actions suddenly influencing different aspects of its environment. Therefore, the boundaries between phases often correspond to constraints or subgoals of the manipulation task.**

In this paper, we investigate how the phases of manipulation tasks can be learned from data. The task is modeled as an autoregressive hidden Markov model, wherein the hidden phase transitions depend on the observed states. The model is learned from data using the expectation-maximization algorithm. We demonstrate the proposed method on both a pushing task and a pepper mill turning task. The proposed approach was compared to a standard autoregressive hidden Markov model. The experiments show that the learned models can accurately predict the transitions in phases during the manipulation tasks.

I. Introduction

Manipulation tasks can generally be decomposed into a series of discrete and distinct *phases* [8]. For example, when lifting a grasped object from a table, the first phase corresponds to increasing the upward force until it matches the object's weight. The second phase begins when the object breaks contact with the table, at which point it can be moved freely in the air. Shifts between phases often correspond to events such as the making and breaking of contacts between objects [6]. As shown in the lifting example, these shifts between phases often correspond to discrete changes in the system's dynamics. Due to these switches in dynamics, the effects of a robot's actions can also change, and the behavior of the robot should adapt accordingly, e.g., in the lifting example, the robot should switch from force control to position control when the phase changes.

Predicting phase switches is an important ability, as these switches often correspond to subgoals or constraints of the task. For example, a phase change occurs when a bottle of water is tilted enough that water starts pouring out. When performing a pouring task, this phase change represents a subgoal of the task. However, when carrying the bottle of water, the contents should not be spilled, and this change in phase represents a constraint that should not be crossed.

In this paper, we present a probabilistic model for representing manipulation tasks with phase changes, and we explain how the parameters of the model can be learned from data. The phases of the task are modeled as discrete hidden variables, each associated to different system dynamics. The structure of the model is similar to that of an auto-regressive hidden Markov model (ARHMM) [10]. In the standard ARHMM the probability distribution over the current latent phase depends on only the previous phase [14]. In the proposed model, the probability of the current phase also depends on the observed state. Thus, the robot can learn to predict when phase changes will occur. We refer to the proposed method as an observed state-based transitions auto-regressive hidden Markov model (STAR). The benefits of the proposed method over the standard ARHMM approach are demonstrated on a robot pushing experiment in Section IV.

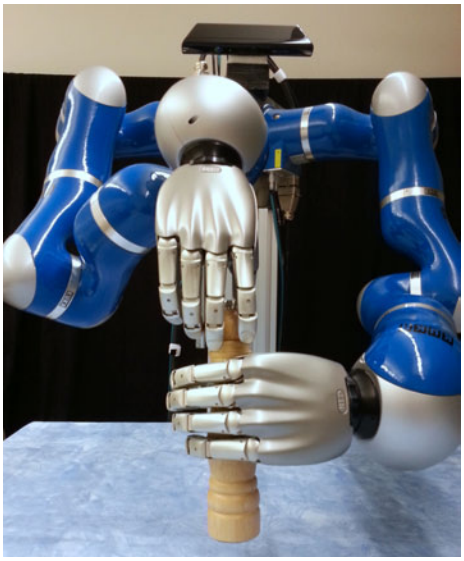

Fig. 1. The Darias robot used in the experiments performing the pepper mill grasping and turning manipulation task.

Previous work on using phases of manipulation tasks has largely assumed that the phases are predefined [7], [15], [1]. Debus et al. estimate the contact state for a peg-in-hole task using an HMM, but they assume the network of contact states and the descriptions of the states are given [7]. Romano et. al [15] propose a human-inspired controller for grasping objects. Their controller consists of several lower level controllers that correspond to grasping phases [9]. The transitions between these phases occur after the robot detects specific tactile events. Andrews and Kry proposed a controller for performing in-hand manipulation based on a three-phase structure [1].

Learning the phases of a manipulation task has mainly been done in the context of *learning from demonstration*,

All of the authors are members of the Intelligent Autonomous Systems group at the Technische Universitaet Darmstadt, Germany. Jan Peters is also a member of the Max Planck Institute for Intelligent Systems
{kroemer, hoof}@ias.tu-darmstadt.de
{neumann, peters}@ias.tu-darmstadt.de

wherein the task is demonstrated by a human [16], [3], [12], [13], [11], [2]. These approaches make use of the fact that the human's behavior changes according to the phase. Păiş et al. used heuristics to detect when the human's reference frames changed, and the manipulation task was segmented accordingly [12]. In [16], the robot learned to perform manipulation tasks from human demonstrations using a parameterized HMM. Ogawara et al. used HMMs to cluster and recognize different demonstrations of a pouring action [11]. However, rather than determining phase transitions based on the actions, our goal is to detect transitions based on the *effects* of the actions.

The remainder of the paper is structured as follows. We present the probabilistic model in Section II, and we explain how the parameters of the model can be learned using expectation-maximization in Section III. Real robot experiments are presented in Section IV. The experiments include a comparative evaluation of the proposed model against a standard ARHMM on a pushing task, and a demonstration of the proposed method for operating a pepper mill as shown in Fig. 1.

II. Modeling Phases of Manipulation Tasks

In the proposed framework, the phases of a manipulation task are modeled as a discrete hidden variable evolving over time. Each phase is associated with its own forward model describing the system. In this section, we introduce the notation, and we describe the structure of the proposed probabilistic model. We explain how the parameters of the model can be learned in Section III.

The observed state of the robot and its environment at time t are given by the state $s_t \in \mathbb{R}^n$. The robot then performs an action $a_t \in \mathbb{R}^m$, which results in the state transitioning to the next state $s_{t+1} \in \mathbb{R}^n$. This change in state depends on the current phase $\rho_t \in \mathbb{N}$, which is hidden. In this paper, the *phase* corresponds to the hidden state of the HMM, and the *state* refers to the observed state.

The effects of performing an action a_t in state s_t and phase ρ_t are modeled by the transition probability $p(s_{t+1}|s_t, a_t, \rho_t)$. We represent the state transitions $p(s_{t+1}|s_t, a_t, \rho_t)$ according to a linear Gaussian model. Therefore, the distribution of the next state is given by the Gaussian

$$s_{t+1} \sim \mathcal{N}(A_{\rho_t} s_t + B_{\rho_t} a_t, \Sigma_{\rho_t}),$$

where $A_i \in \mathbb{R}^{n\times n}$, $B_i \in \mathbb{R}^{n\times m}$, and $\Sigma_i \in \mathbb{R}^{n\times n}$ are matrices corresponding to phase $\rho = i$.

The probability of the current phase depends on the current state and the previous phase $p(\rho_t|s_t, \rho_{t-1})$. The first phase depends on only the first state $p(\rho_1|s_1)$. The dependency on the previous phase allows the model to represent effects such as hysteresis. It also allows us to incorporate transient state information. For example, the making and breaking of a contact can be detected by dynamic tactile sensing, indicating a transition to the next phase [9]. However, once the contact event is over, the sensor's readings may return to their previous values even though the phase has changed. At that point, the model has already switched to the new phase.

The transitions between phases can be modeled using probabilistic classifiers. We model the phase transition probabilities using logistic regression, i.e.,

$$p(\rho_t = j|s_t, \rho_{t-1} = i) = \frac{\exp(w_{ij}\phi(s_t))}{\sum_k \exp(w_{ik}\phi(s_t))}$$

where $w_{ij} \in \mathbb{R}^d$ is a weight vector for transitioning from $\rho = i$ to $\rho = j$, and $\phi(s_t)$ is a function mapping the state to a d dimensional feature vector. Features may, for example, be a subset of the full state vector or additionally include the positions of objects relative to each other. Similarly, we represent the initial phase distribution as

$$p(\rho_1 = j|s_1) = \frac{\exp(w_{0j}\phi(s_t))}{\sum_k \exp(w_{0k}\phi(s_t))}$$

where w_{0j} is a weight vector for each phase.

Given a model of the system, the robot would select actions according to the current state and phase $p(a_t|s_t, \rho_t)$. However, in this paper, we are focusing on learning the phases model from exploratory actions. Hence, the probability of an action is modeled as $p(a_t)$, and it is assumed to be a uniform distribution. In an imitation learning scenario, where the actions are demonstrated to the robot by a human, the conditional probability $p(a_t|s_t, \rho_t)$ could be used to help infer phases based on changes in the human's behavior. Including the dependency of the action on the state and phase is a straightforward extension of the theory presented here, but it is beyond the scope of this paper.

Given the individual components of the model, the probability of observing a sequence of N samples of states $s_{1:N} = \{s_1, \ldots, s_N\}$, actions $a_{1:T} = \{a_1, \ldots, a_N\}$, phases $\rho_{1:N} = \{\rho_1, \ldots, \rho_N\}$, and next states $s_{2:N+1} = \{s_2, \ldots, s_{N+1}\}$ is given by

$$p(s_{1:N+1}, a_{1:N}, \rho_{1:N}) =$$
$$p(s_1)p(\rho_1|s_1)\prod_{t=1}^{N} p(s_{t+1}|s_t, a_t, \rho_t)p(a_t)\prod_{t=2}^{N} p(\rho_t|s_t, \rho_{t-1}).$$

The graphical model of this probability factorization is shown in Fig. 2. The key difference to an autoregressive HMM is the additional edge from the current state to the current phase. As a result of this edge, the transition between phases depends on the observed state. The graphical model also illustrates the Markov property of the model: given the state s_t and the phase state ρ_t, all future states are independent of the states before $t-1$. This property is important, as it will allow us to learn the model parameters in a computationally efficient manner.

III. Model Learning Using EM

Having defined the structure of the model in the previous section, we now focus on learning the model parameters w, A, B, and Σ, which we will refer to jointly as $\theta = \{w, A, B, \Sigma\}$. Given a set of sampled trajectories of states and actions, we propose using the *expectation-maximization* (EM) algorithm to estimate the parameters. The EM algorithm iterates between an expectation step and a maximization step in

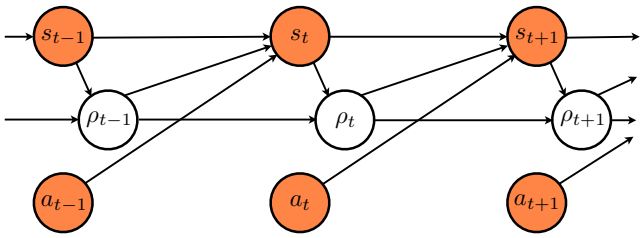

Fig. 2. The graphical model of the proposed task representation with phases. The shaded nodes indicate the observed variables, and the unshaded nodes are latent. The arrows from the states s pointing to the phases ρ indicate that the probability distributions of the phases depend on the observed states.

order to find maximum likelihood estimates of the model parameters given that some of the variables are unobserved, i.e., the phases of the samples are not known. The steps of the algorithm are explained below.

A. Expectation Step

The first step of the EM algorithm is the expectation step. In this step, we will need to compute the distribution over the hidden states, i.e., the phases, given the observed sequence of variables. In particular, we will need to compute the distributions $p(\rho_t = j|s_{1:N}, a_{1:N})$ and $p(\rho_t = i, \rho_{t+1} = j|s_{1:N+1}, a_{1:N})$ for the computations in the maximization step. We can compute these marginal probabilities efficiently by using a forward-backward message passing approach. During the expectation step, we assume that the parameters θ of our model are fixed.

In order to improve the clarity of the methodology below, we will define $z_t = \{s_t, a_t\}$ as the observed state and actions together. For the final sample $z_{N+1} = s_{N+1}$. Thus, we have $p(z_{t+1}|\rho_t, z_t) = p(s_{t+1}|\rho_t, s_t, a_t)p(a_{t+1})$, and $p(\rho_t|\rho_{t-1}, z_t) = p(\rho_t|\rho_{t-1}, s_t)$ as ρ_t is not conditioned on a_t.

We first send a series of messages forward through the network from $t = 1$ to $t = N$. The forward messages give the probability of observing the sequence of states, actions, and next state up to now and it is defined as

$$\alpha_j(t) = p(z_{1:t+1}, \rho_t = j).$$

The first message, starting at $t = 1$, is initialized according to

$$\alpha_j(1) = p(z_2|\rho_1, z_1)p(\rho_1 = j|z_1)p(z_1).$$

The subsequent messages are computed recursively as

$$\alpha_j(t) = p(z_{t+1}|\rho_t = j, z_t)\sum_i \alpha_i(t-1)p(\rho_t = j|\rho_{t-1} = i, z_t).$$

The second set of messages are sent backwards through the network from $t = N$ to $t = 1$. The backward messages give the probability of observing the remainder of the observed sequence of states, actions, and next states given the current phase and next state

$$\beta_j(t) = p(z_{t+2:N}|\rho_t = j, z_{t+1}).$$

We initialize the backward messages at time $t = N$ as

$$\beta_j(N) = 1,$$

and we recursively compute the messages backwards in time according to the formula

$$\beta_j(t-1) = \sum_i p(\rho_t = i|\rho_{t-1} = j, z_t)p(z_{t+1}|\rho_t = i, z_t)\beta_i(t).$$

Given the forward and backward messages, we can easily compute the required probabilities. The marginal likelihood of the phase at a specific point in time is given by

$$p(\rho_t = j|z_{1:N+1}) = \frac{p(\rho_t = j, z_{1:N+1})}{p(z_{1:N+1})},$$

$$p(\rho_t = j|z_{1:N+1}) = \frac{\alpha_j(t)\beta_j(t)}{\sum_i \alpha_i(t)\beta_i(t)}.$$

Similarly, the joint distribution of a phase and the next phase is given by

$$p(\rho_t = i, \rho_{t+1} = j|z_{1:N+1}) =$$

$$\frac{\alpha_i(t)p(\rho_{t+1} = j|\rho_t = i, z_{t+1})p(z_{t+2}|\rho_{t+1} = j, z_{t+1})\beta_j(t+1)}{p(z_{1:N+1})},$$

where we again make use of the messages computed earlier. Having computed these probabilities, we can now proceed to the maximization step of the algorithm.

B. Maximization Step

In the maximization step of the EM algorithm, we must compute the parameters that maximize the expected log-likelihood of the observed and hidden variables

$$\theta_{new} = \arg\max_\theta \sum_\rho p(\rho_{1:N}|z_{1:N+1}; \theta_{old}) \ln p(\rho_{1:N}, z_{1:N+1}; \theta),$$

where the summation is over all possible sequences of ρ, and the conditional distributions $p(\rho_{1:T}|s_{1:T+1}, \theta_{old})$ are computed using the old model parameters θ_{old} as indicated. By factorizing the joint distribution $p(\rho_{1:T}, s_{1:T+1}|\theta)$ and decomposing the log of a product into a summation of logs, the maximization problem can be rewritten as

$$\begin{aligned}
\theta_{new} &= \arg\max_\theta \sum_\rho p(\rho_{1:N}|z_{1:N+1}; \theta_{old}) \ln p(z_1) \\
&+ \sum_\rho p(\rho_{1:N}|z_{1:N+1}; \theta_{old}) \ln p(\rho_1|z_1) \\
&+ \sum_{t=1}^{N} \sum_\rho p(\rho_{1:N}|z_{1:N+1}; \theta_{old}) \ln p(z_{t+1}|\rho_t, z_t) \\
&+ \sum_{t=2}^{N} \sum_\rho p(\rho_{1:N}|z_{1:N+1}; \theta_{old}) \ln p(\rho_t|\rho_{t-1}, z_t).
\end{aligned}$$

The terms of the log functions are however dependent on only a small set of phase variables. The first term does not depend on the parameters and it can be removed from

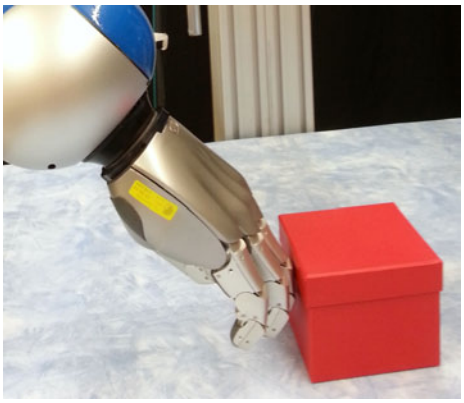

Fig. 3. The robot performing the pushing task. The robot is entering the second of three phases, wherein it needs to increase the pushing force until it overcomes the box's stiction.

the arg max. The majority of factors in each term can be marginalized out, and the maximization can be rewritten as

$$\begin{aligned}\theta_{new} &= \arg\max_{\theta} \sum_{\rho_1} p(\rho_1|z_{1:N+1},\theta_{old}) \ln p(\rho_1|z_1) \\ &+ \sum_{t=1}^{N} \sum_{\rho_t} p(\rho_t|z_{1:N+1};\theta_{old}) \ln p(z_{t+1}|\rho_t,z_t) \\ &+ \sum_{t=2}^{N} \sum_{\rho_{t-1:t}} p(\rho_{t-1},\rho_t|z_{1:N+1};\theta_{old}) \ln p(\rho_t|\rho_{t-1},z_t).\end{aligned}$$

The marginal distributions $p(\rho_t|z_{1:N+1},\theta_{old})$ and $p(\rho_{t-1},\rho_t|z_{1:N+1},\theta_{old})$ were already computed during the expectation step, as explained in Section III-A. The new parameters can now be computed in a straightforward manner using weighted linear regression and weighted logistic regression.

Estimates of the parameter matrices A and B can be computed using weighted linear regression. We begin by defining a matrix X that has one column per sample and each column consists of the concatenated state s and action a of the sample

$$X = \begin{bmatrix} s_1 & \dots & s_N \\ a_1 & \dots & a_N \end{bmatrix}.$$

Similarly, we define a matrix Y, which also has N columns, each corresponding to a sampled next state

$$Y = \begin{bmatrix} s_2 & \dots & s_{N+1} \end{bmatrix}.$$

The new estimates of the parameter matrices are then given by

$$\begin{bmatrix} A_j \\ B_j \end{bmatrix} = YW_jX^T(XW_jX^T)^{-1}$$

where the T indicate the transposes of the matrices, and W_j is a diagonal matrix, wherein the t^{th} diagonal element is given by $[W]_{tt} = p(\rho_t = j|s_{1:N+1},\theta_{old})$. Often, the A matrix may be trivial to define and one wishes to learn only the effects of the actions B. In this case, the columns of Y are defined as the changes in states for each sample $s_{t+1} - As_t$, and X contains only the actions a_t. The computation of B_j is then the same as before. The new estimates of the covariance matrices are given by

$$\Sigma_j = \frac{\sum_{i=1}^{N} p(\rho_i = j|s_{1:N+1},\theta_{old})(s_{i+1}-\mu_{ji})^T(s_{i+1}-\mu_{ji})}{\sum_{k=1}^{N} p(\rho_k = j|s_{1:N+1},\theta_{old})},$$

where $\mu_{ji} = A_j s_i + B_j a_i$ is the expected next state given the updated A and B matrices.

The phase transition parameters w are computed using weighted logistic regression. Logistic regression does not have a closed-form solution and the weight vectors need to be computed iteratively using gradient decent. However, the optimization is convex and, therefore, a global optimal solution can be easily found. To compute the gradient for transition from $\rho_{t-1} = i$ we define three matrices: 1) S is a matrix with the columns containing the features of the sampled states $\phi(s_t)$. 2) The matrix L contains the weights from the E step and it has elements given by $[L]_{tj} = p(\rho_{t-1} = i, \rho_t = j|z_{1:N+1};\theta_{old})$. 3) The third matrix P has the same form as the L matrix, but it contains the predictions $p(\rho_t = j|s_t,\rho_{t-1} = i)$ given the current weights w. The gradient of the weighted log-likelihood with respect to w are given by the elements of the matrix

$$G = S(P-L).$$

Regularization is added to both the linear regression and the logistic regression in order to avoid overfitting and to improve the model's ability to generalize to new situations.

After the maximization step has been completed, the algorithm computes the expectation step again with the new parameters. The process iterates between the two steps until the model has converged to a solution.

IV. Experiments

The proposed method was implemented and evaluated on a real robot, as shown in Fig. 1. The robot consists of two Kuka lightweight robot arms, and two five-fingered DLR-HIT II hands. The arms were controlled using task-space impedance control, while the fingers were controlled using joint-space impedance control [4]. The robot's vision system is based on a Microsoft Kinect sensor.

A. Pushing an Object

The first experiment was designed as a benchmark task to compare the performance of the proposed STAR method to a standard ARHMM. The robot was given the task of pushing a filled box across the table in a straight line, as shown in Fig. 3. The hand was initially not in contact with the box, and the box was initially located at various distances relative to the robot's hand. The state space consisted of the position of the robot's hand, the desired position of the hand, and the position of the box. The position of the box was tracked using the Kinect sensor. The features ϕ included the position of the hand relative to the box and the position of the hand relative to the desired position. The pushing action was defined as a change in the end-effector's desired position by 1 cm and the same action was applied at every time step. Ten trials of

the task were recorded for learning the model, and each trial contained 30 pairs of states and actions.

For this experiment, the A matrices are trivial to define, and hence only the B matrix was learned. The B matrices were initially set randomly. For comparison, a standard autoregressive hidden Markov model (ARHMM) was also learned. This model is similar to the one shown in Fig. 2, except that the arrows from the states s to the phases ρ are removed. Hence, the phase transition probabilities of the ARHMM only depend on the previous phase $p(\rho_t|\rho_{t-1})$, and they are therefore modeled as a stochastic matrix. The initial distribution $p(\rho_1)$ was modeled as a vector summing to one. Both methods were initialized with the same parameters in order to obtain a fair comparison.

Both ARHMM and STAR were able to successfully segment the training data into three phases: 1) the hand is in free space and the box is stationary. 2) the hand is in contact with the box and both are stationary while the force (proportional to the distance between the hand and the desired position) ramps up to overcome friction. 3) the hand is pushing the box and both are moving together. Different initial parameters were tested, but the results were consistent. The methods therefore performed equally well on the training data.

We also attempted learning the STAR model with different numbers of phases. When only two phases were used, the model still found one of the phases, but the other two phases were merged together resulting in an underfitted model. Using four phases resulted in redundant models, where one of the actual phases would be modeled by two model phases. Often these two phases would switch between each other at every time step, and they would model similar system dynamics. As the number of phases increases, the number of samples allocated to each becomes fewer, and the quality of the individual models decreases. Incorporating a method for merging similar models would therefore be beneficial. Another approach to selecting the appropriate number of parameters is to use cross-validation.

In order to evaluate and compare the two 3-phase models' abilities to predict trajectories, we recorded another ten pushing trajectories using the same procedure. For each of these test trials, we initialized the learned models with the initial states and sampled 50 trajectories from each model. Thus, 500 trajectories were sampled from each model. These sampled trajectories were compared to the actual recorded trajectories, and the absolute error in the predicted positions of the box and the hand were computed. The errors are shown in Fig. 4. At the end of the trajectory, the RMS errors in position were 3.85 cm for STAR and 11.19 cm for the standard HMM.

As one would expect, the errors increase for both models as the prediction horizon increases due to the accumulation of errors. However, the proposed method ultimately performs better due to its more accurate predictions of when phase transitions will occur based on the observed state. The results indicate that the proposed STAR representation is a better model of the manipulation task than a standard ARHMM.

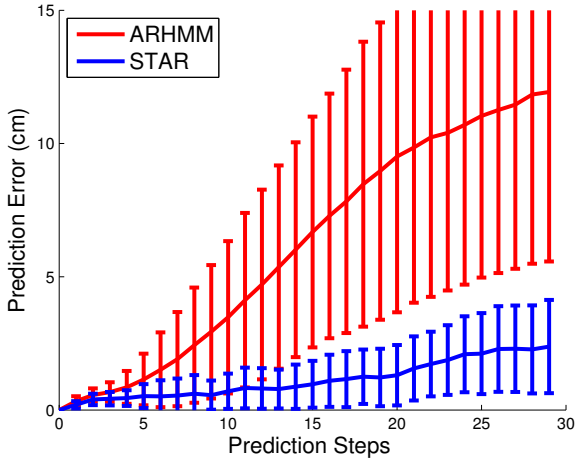

Fig. 4. The plots show the absolute errors in predicting the position of the box and the robot hand during the pushing task. The predictions are initialized with only the starting state of the trajectory. The red line represents the performance of the HMM model, and the blue line shows the performance of the proposed method. The error bars indicate one standard deviation.

B. Operating a Pepper Mill

In the second experiment, we investigated using the proposed method to detect the phases when grasping and turning a pepper mill. In order to perform the task, the robot held the pepper mill with its left hand, and it used the right hand to turn the mill, as shown in Fig. 1. Using the compliance of the fingers, the robot was shown a suitable grasp of the top of the pepper mill (Fig. 5), which was then used to define an eigengrasp, i.e., a grasp synergy [5]. The first action dimension corresponds to changing the desired hand configuration according to the eigengrasp. The second action dimension corresponds to rotations of the hand around the axis of the pepper mill. The state space contains the desired hand orientation and finger configuration, as well as the recorded values. The measured joint torques of the fingers were also projected into the lower-dimensional space of the eigengrasp. These projected torques were included in the state. The final element of the state vector contained the angle of the pepper mill's rotation. The rotation was measured using a webcam and an augmented reality marker mounted on the bottom side of the pepper mill. The features ϕ corresponded to all of the recorded data, except for the angle of the pepper mill. Data was recorded of the robot performing 15 grasp-twist-release trajectories of varying lengths and with different amounts of grip. A model with three phases was learned.

The three learned phases seem to correspond to: 1) the hand is open, 2) the hand has made contact with the peppermill, and 3) the hand has applied a firm grasp and it can turn the pepper mill. The B matrices indicate that rotating the hand has almost no effect on the pepper mill in the first two phases. In the third phase, the pepper mill rotates together with the hand. This phase was also active for 91% of the samples in which the pepper mill was turned. The

Fig. 5. The grasp used during the pepper mill grasping and turning experiment.

proposed model was therefore able to learn when the pepper mill can be turned, without relying on the angle of the pepper mill.

The learned model also describes how phase transitions change the effects of attempting to close the hand. In the first phase, changing the desired finger configuration resulted in the fingers moving. In the third phase, the fingers' joint torques increased instead and the fingers moved only a small amount. The learned model therefore captures the restricted movement of the fingers when grasping the pepper mill.

The weight vectors w indicate when the phase transitions occur. The phase transitions $1 \rightarrow 2$ and $2 \rightarrow 3$ have large positive weights corresponding to the joint torques. In contrast, the largest weights for the transitions $3 \rightarrow 2$ and $2 \rightarrow 1$ correspond to the joint angles. Therefore, the phase transitions seem to be controlled by the measured torques during grasping and the measured joint angles when releasing the object.

The quality of the model depends on the initial parameter setting, and the EM algorithm is known to find only local optima rather than the global optimum. The learned model sometimes transitions from phase two to phase one when the hand was rotated, even though one would expect it to remain in phase two. A more accurate model of the task could potentially be learned by incorporating more phases, but this approach would also require more data to avoid overfitting.

In the future, we plan to have the robot use the STAR models in order to efficiently learn to perform manipulation tasks. Given the promising results of Romano et al. [15], we also plan to incorporate dynamic tactile signals into the proposed framework to detect certain phase transitions more accurately.

V. Conclusions

In this paper, we presented a probabilistic model for representing manipulation tasks with multiple phases. The phases were represented as hidden variables corresponding to different system dynamics. We explained how the model parameters could be learned from a set of trajectories using the expectation-maximization algorithm. Unlike a standard autoregressive hidden Markov model, the proposed model incorporates the observed state variables when predicting the transitions between the hidden phases.

The proposed method was implemented and used to learn the phases of a pushing task and a pepper mill turning task. The results showed that the state-based transitioning allows the robot to predict the phase changes more accurately, resulting in better predictions overall.

Acknowledgment

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7-ICT-2013-10) under grant agreements 610878 (3rdHand) and 610967 (TACMAN)

References

[1] S. Andrews and P.G. Kry. Goal directed multi-finger manipulation: Control policies and analysis. *Computers and Graphics*, 37(7):830 – 839, 2013.

[2] K. Bernardin, K. Ogawara, K. Ikeuchi, and R. Dillmann. A sensor fusion approach for recognizing continuous human grasping sequences using hidden markov models. *IEEE Transactions on Robotics*, 21(1):47–57, 2005.

[3] J. Butterfield, S. Osentoski, G. Jay, and O.C. Jenkins. Learning from demonstration using a multi-valued function regressor for time-series data. In *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 328–333, Dec 2010.

[4] Zhaopeng Chen, Neal Y. Lii, Thomas Wimboeck, Shaowei Fan, Minghe Jin, Christoph Borst, and Hong Liu. Experimental study on impedance control for the five-finger dexterous robot hand dlr-hit ii. In *IROS*, pages 5867–5874. IEEE, 2010.

[5] Matei Ciocarlie and Peter Allen. Hand posture subspaces for dexterous robotic grasping. *The International Journal of Robotics Research*, 28:851–867, 07/2009 2009.

[6] M. Cutkosky and J.M. Hyde. Manipulation control with dynamic tactile sensing. In *proceedings of International Symposium on Robotics Research*, 1993.

[7] Thomas Debus, Pierre E. Dupont, and Robert D. Howe. Contact state estimation using multiple model estimation and hidden markov models. In Bruno Siciliano and Paolo Dario, editors, *ISER*, volume 5 of *Springer Tracts in Advanced Robotics*, pages 517–526. Springer, 2002.

[8] J. R. Flanagan, M. C. Bowman, and R. S. Johansson. Control strategies in object manipulation tasks. *Curr Opin Neurobiol*, 16(6):650–659, December 2006.

[9] Roland S. Johansson and Randall J. Flanagan. Coding and use of tactile signals from the fingertips in object manipulation tasks. *Nature reviews. Neuroscience*, 10(5):345–359, April 2009.

[10] K. Murphy. Switching Kalman filters, 1998.

[11] K. Ogawara, J. Takamatsu, H. Kimura, and K. Ikeuchi. Modeling manipulation interactions by hidden markov models. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 2, pages 1096–1101 vol.2, 2002.

[12] A. L. Pais, Keisuke Umezawa, Yoshihiko Nakamura, and A. Billard. Learning robot skills through motion segmentation and constraints extraction. HRI Workshop on Collaborative Manipulation, 2013.

[13] P. Pastor, M. Kalakrishnan, L. Righetti, and S. Schaal. Towards associative skill memories. In *IEEE-RAS International Conference on Humanoid Robots*, 2012.

[14] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, February 1989.

[15] Joseph Romano, Kaijen Hsiao, Günter Niemeyer, Sachin Chitta, and Katharine J. Kuchenbecker. Human-inspired robotic grasp control with tactile sensing. *IEEE Transactions on Robotics*, 27:1067–1079, 2011.

[16] Leonel Rozo, Pablo Jiménez, and Carme Torras. A robot learning from demonstration framework to perform force-based manipulation tasks. *Intell. Serv. Robot.*, 6(1):33–51, January 2013.