Imitation Learning for Bimanual Manipulation

Adriaan Mulder

***Abstract*— Imitation Learning for bi-manual Manipulation represents a cutting-edge area in robotics, aiming to teach robots the intricacies of using two hands through imitation of human or expert demonstrations. Various methods, from traditional behavioral cloning to advanced techniques like deep reinforcement learning, have been developed. We want to review several imitation learning methods and discuss the challenges associated with their application in bimanual robotics. Beginning with dynamic movement primitives, which are the foundation of contact rich manipulation policies using the least squares approach, we will also show some state-of-the-art deep learning methods which are even able to run on low cost robotic hardware and are thus easy to reproduce. We will discuss the advances and achievements of different approaches and possible future research opportunities of combining some of the novel ideas.**

I. Introduction

This work highlights some of the prominent approaches in imitation learning, with a specific focus on methods that have been applied to bimanual manipulation. A review of several key papers in this field is used to emphasize the underlying similarities and key achievements obtained so far. We introduce how these methods are structured from a technical point of view to help the reader gain insight into this emerging field of research.

Furthermore, we discuss how existing works have applied these methods to complex situations like low-tolerance industrial insertion tasks, object transference, and sequential motor skill behaviors.

For this, we spectate papers underlying three major categories of machine learning: motor primitive methods, reinforcement learning and behavioral cloning. We show research that shows the basic concepts of these categories and explain why the application of just the naive methods does not suffice to successfully complete the aforementioned complex tasks having many contacts and thus many interaction forces which are difficult to model.

We start with a basic imitation learning mechanism called Dynamic Movement Primitives which are known to generalize really well and show several improvements in the class of Motor Primitives that enable systems to generalize even in contact rich environments and thus control bimanual manipulators to solve tasks. After this, we review a successful case of learning movement primitives and applying them in reinforcement learning algorithms. For the third category, we explain the basic ideas of behavioral cloning and show it's shortcomings for high precision robotic tasks. Looking at different papers describing approaches which utilize and improve upon the ideas of behavioral cloning, we see very promising results with a bimanual robot in complex contact rich tasks through novel and maybe unintuitive solutions.

Lastly we discuss the methodologies, findings and results of these state of the art techniques to analyze potential future research opportunities.

II. Motor Primitives Methods

The first class of imitation learning algorithms we focus on are Movement Primitives. They originate in a very simple idea of function approximation to copy demonstrated joint trajectories and then calculating needed forces to push a manipulator along those paths. Building on deterministic dynamic movement primitives which allow for very good imitation performance in static environments, improvements based on probability theory deliver outstanding results even in stochastic and changing settings.

A. Dynamic Movement Primitives

The very basis of many successful Imitation learning approaches are DMPs [7, 8, 15]. DMPs originate from the idea, that movement can be modeled through trajectories in space, velocity and acceleration. The objective for our manipulator is to follow predefined trajectories, that we can describe through virtual external forces which are added onto a stable dynamical system as controlling signals [12]. Using a simple damped spring modeled in a second-order differential equation

$$\ddot{y} = \alpha(\beta(g - y) - \dot{y}) \tag{1}$$

g is the goal position, y the current systems observed position, $\dot{y}$ the current systems observed velocity and $\ddot{y}$ is the desired acceleration to reach the goal [3]. Using correct values for α and β, we gain a stable, critically damped system which would - over time - converge to a fixed point $(y, \dot{y}) = (g, 0)$. We can add our trajectory defining forces through a nonlinear function f.

$$\ddot{y} = \alpha(\beta(g - y) - \dot{y}) + f \tag{2}$$

Trajectories are typically modelled by a mapping of timestamps and positions. We can obtain such a trajectory e.g. through a single demonstration of the movement. We can generally encode such modelled trajectories with least squares approximation through the use of weighted radial basis functions Ψ [2]. The function that encodes the demonstration is defined as

$$f(x) = \frac{\sum_{i=1}^{N} \Psi_i(x) w_i}{\sum_{i=1}^{N} \Psi_i(x)} x(g - y_0) \tag{3}$$

with

$$\Psi_i = exp(-h_i(x - \mu_i)^2) \tag{4}$$

being a Gaussian function with the variance h_i centered around c_i. x in equation (3) can be interpreted as a phase-variable replacing a typically used time-variable ($f(t)$ instead of $f(x)$) which eliminates time dependence and thus decouples our trajectories from time and allowing for new coupling modifications such as multiple degree of freedom coupling. x will start at 1 and ultimately converge to 0, which lets the force converge to 0 the closer we get to the goal. This will give the full control of the system back to the pd-controller which we know is stable. We can see an illustration of the practical use of the basis functions for one joint in Figure 1.

The second term $(g-y_0)$ describes a spacial scaling factor that scales the weights based on the total distance/position of the goal [11]. When adding a time-scaling variable τ we are able to speed up or slow down the movement.

$$\tau\ddot{y} = \alpha(\beta(g-y) - \dot{y}) + f \tag{5}$$

The necessary forces needed to recreate a trajectory, especially in imitation learning can be done through observing the trajectories in a demonstration.

$$f_{target} = \ddot{y}_{demo} - \alpha(\beta(g - y_{demo}) - \dot{y}_{demo}) \tag{6}$$

When inserting f_{target} back into our modified damped spring equation, we add the calculated forces back onto the control signal and copy the demonstration. It's also possible to reset the phase variable x to allow for periodic movement or chain multiple DMPs.

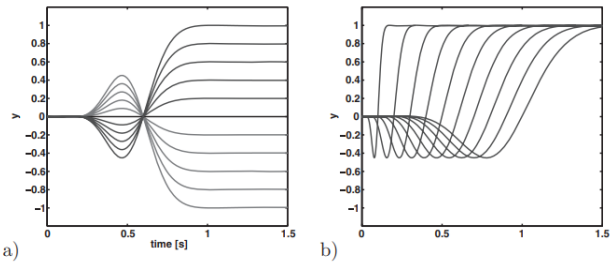

Fig. 1: Illustration of the scaling properties of discrete dynamical systems. (a) the goal position is varied from -1 to 1 in 10 steps. (b) The time constant is changed from 0.15 to 1.7 seconds. Every graph is differently weighted combination of radial basis functions [11].

Basic DMPs can be used for imitation learning as shown and they bring a lot of benefits such as generalization, spacial and temporal scaling and temporal modulation [17]. But they also do bring some problems to the table that is, that the defined trajectory is based on a single demonstration (or on the mean of multiple demonstrations) which is known to not perform optimally in a given environment, but only as good as the demonstrator. We can improve this behaviour and add some new benefits when using a probabilistic movement primitive.

B. Probabilistic Movement Primitives

Unifying several and adding new capabilities of different DMP frameworks, Paraschos et al. introduced the idea of probabilistic Movement Primitives (ProMPs). Building the trajectory not only from a single path from a single demonstration, but as a distribution of multiple demonstrations allows for modeling the variance of the movement at different phases of it. This helps among other things evaluating the importance of precision for each phase of the movement.

Similar to DMPs, ProMPs use a maximum likelihood training procedure to obtain a weighting vector $\mathbf{w}$ for a number of basis functions Φ to encode a demonstrated trajectory $\tau = [y_0, ...y_n]$ with $y_t = [q_t, \dot{q}_t]^T$. To correctly calculate the mean and the variance of the trajectories, we need to synchronise these trajectories by decoupling them from time and comparing them by the phase of the movement they are in. This is necessary, as different trajectories can have different durations. After calculating the weights of all N trajectories, we can obtain the mean and variance of the weight vectors.

$$\mu_w = \frac{1}{N}\sum_{i=1}^{N} w_i, \qquad \Sigma_w = \frac{1}{N}\sum_{i=1}^{N}(w_i - \mu_w)(w_i - \mu_w)^T \tag{7}$$

Finally we can capture the variance of the actual trajectories with $\theta = \{\mu_w, \Sigma_w\}$ and the introduction of the distribution $p(w;\theta)$ over the weight vector.

This formulation allows simple coupling of multiple Degrees of freedom in one trajectory for coordinated movement by extending the trajectory distribution $p(\tau;\theta)$. We just have to maintain the weight distribution $p(\mathbf{w};\theta)$ over a combined weight vector $\mathbf{w} = [w_1^T, ...w_n^T]^T$ for the weight vectors of every DoF. This brings a huge new advantage in correlating the movement of every DoF instead of just synchronising the joints through the phase variable. We are now able to define points in task space which the end effector should reach at a given point in time. This kind of modulation is performed by conditioning the probabilistic model through adding desired points at a given point in time. For this we have to define a point $\mathbf{y}_t^* = [\mathbf{q_t}, \dot{\mathbf{q}}_t]$ with its desired accuracy $\mathbf{x_t}^* = \{\mathbf{y_t^*}, \Sigma_y^*\}$ and applying the Bayes theorem.

$$p(\mathbf{w}|\mathbf{x_t^*}) \propto \mathcal{N}(\mathbf{y_t^*}|\Phi_t\mathbf{w}, \Sigma_y^*)p(\mathbf{w}) \tag{8}$$

To fully exploit the obtained trajectory distribution and stochastic weight distribution Paraschos et al. explain why a simple PD controller is insufficient and a model based controller is needed [17].

C. Interaction Primitives

In contrast to ProMPs the interaction primitive (IP) framework of Amor et al. uses probability theory on DMPs to update future behaviour based on observed partial trajectories. This is used in a setting in which interactions between multiple agents take place. The nature of human-robot interaction, which is presented in the Paper, shows variance in each demonstrated example and thus it is necessary not only to have a modulation of trajectories,

but also to predict the actual trajectory of the interaction partner and adjust the own behaviour.

To achieve this, the DMP parameters $\theta = [w_1^T, g_1, ..., w_N^T, g_N]$ in this approach consist of the basis function weights as shape parameters and new goal attractor parameters to scale the own movement impromptu based on the partners observed movement, where N is the number of DoF. Given the DMP parameters of multiple trajectories, the parameter distribution $p(\theta)$ is estimated to be Gaussian.

We can formulate the likelihood of an observed partial trajectory until a given point in time τ_O of the interaction partner with an Gaussian model.

$$p(\tau_O|\theta) = \mathcal{N}(F^*|\Omega\theta, \sigma^2 I) \tag{9}$$

With $F*$ being the forcing function of the observation, Ω the matrix of basis functions and σ^2 the observation noise variance. Using the likelihood we now can calculate the posterior for the DMP parameters and update the distribution.

$$p(\theta|\tau_O) \propto p(\tau_O|\theta)p(\theta) \tag{10}$$

By learning a new DMP with $\theta = [\theta_O^T, \theta_C^T]^T$, we now can easily correlate the estimated movement of the observed agent and the movement of the controlled agent [1].

D. Bayesian Interaction Primitives

IPs see the phase of the interaction only as the spacial Positions mapped to time $s_t = \delta_t$ using a time-alignment algorithm to align several trajectories. This comes with the weakness of being unable to estimate the correct position of our controlled agent based on the observed agents state, when the observed agents trajectory is partially static. Thus Bayesian Interaction Primitives extend IPs by adding the phase variables velocity $\dot{\delta}$ to the robots state as well as the weights of the basis functions $s_t = [\delta, \dot{\delta}, \mathbf{w}]^T$. Now static trajectories phases will be estimated correctly. But not only that, BIPs are now also able to not only make spacial reasoning but also temporal reasoning, meaning that the phase is not based on an internal clock anymore, but is re-estimated based on current observations [21]. Using this formulation and using the Extended Kalman Filter SLAM, the advantage of being able to couple the phase estimation to the conditioning of the recursive state estimation. With the newly defined state, Bayesian inference avoids high computation cost and aggregation errors of the phase estimations [5].

III. Bi-manual Manipulation via Motor Primitives

Now, having talked about several forms of movement primitives, we can introduce a system for imitation learning of compliant bi-manual manipulation policies that can adapt to a spectrum of disturbances in space and time and also leverage physical contact for themselves in a possibly contact rich environment.

While Methods like Probabilistic Motor Primitives (ProMPs) would deliver a good chance of reproducing the learned demonstrations, they are not able to make an estimation of the tasks progress which is essential for successful manipulation in contact rich environments when using multiple agents. Motor commands may not get executed accurately enough when handling interactions with some force. To be able to generate motor commands that are temporally adequate and are able to overcome obstacles, Interaction Primitives are essential. However compared to IPs Idea to observe an interaction partner, the system uses multi-modal sensing of the force-torque and joint angles on itself. When working on a task, using an ensemble of BIPs (EnBIPs), the system can estimate the progress/the phase and perform spacio-temporal inference based on the observations - the robot determines both what to do and when to do it.

A demonstration of it is shown in figure 2.

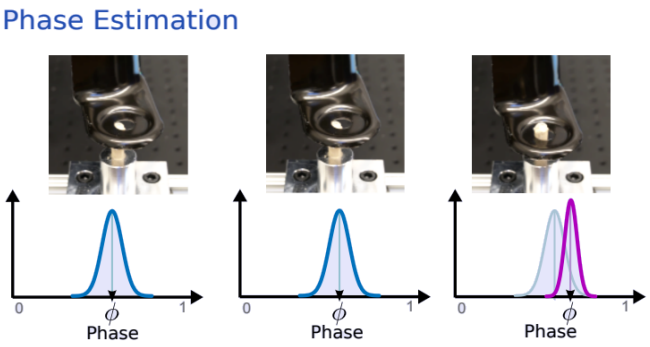

Fig. 2: Phase estimation is a Gaussian distribution. The distribution roughly stays constant as long as the obstacle has not been conquered. In the last picture, after the obstacle has been overcome, the phase estimate moves forward in time [21]

This system uses the principles of admittance control to collect demonstration data without risks, and uses those demonstrations to learn Interaction Primitives. The IPs conduct inference over both robots jointly, but both are controlled separately. The term ”admittance” refers to the system’s willingness or ability to allow external forces or motions to affect it. If running against an obstacle, the use of such an admittance controller prevents the buildup of force.

This framework was tested on a bi-manual setup both in a simulated and in a real-world setting. To demonstrate the adaptability to variance in training data, 30 demonstrations were collected in both settings respectively. In the simulation, the movement was altered through pre-programmed randomization, in the real world setting the demonstrations were collected from eight human subjects using different collection methods. The task was to pick up a bracket with slightly varying starting positions and placing it onto two pins with tolerances of 1 and 5mm in simulation and 6mm in the real world. Through the use of multiple data collection methods such as kinesthetics or backdriving using a 6 DoF Space-Mouse Control Stick, the collection and use of force-

torque sensor data has shown to be critical for performance in a contact-rich environment as it drastically increases its robustness. Overall this system has shown a success rate of 90% using BIPs. It is sample efficient and can adapt to a large range of disturbances but it is limited by the quality of training data, as the phase estimation relies on near optimal demonstration behaviour.

IV. Reward-Based Methods

Some different promising approaches to bi-manual manipulation are based on reward based learning or reinforcement learning. In these methods, the system tries to learn a policy based on a reward function, in which the system tries to maximize it's reward by choosing the best action in a task given the current situation. This enables the robots to gain insights about it's behaviour through performance feedback and adapt it's control signals to optimize the tasks success.

A. Reinforcement Learning

One major challenge in RL methods is defining a reward function that leads to the desired behaviour. While specifying a reward is pretty simple in low dimensional discrete Markov decision processes (MDPs), the complexity of shaping a reward grows fast in higher dimensional, continuous spaces.

One successful approach is learning in multiple stages to solve the task [13]. In a first stage, the robot learns to divide the task into sub-tasks or phases and phase-transitions and represent them in a probabilistic model given the human demonstrations. After having obtained a model, the approach employs a second learning phase, in which motor primitives are created that optimize the transitions between phases.

In the demonstration phase, the system observes the robots time-variant state s_t and taken actions a_t, which lead to the next state s_{t+1}. The actions change in the state depends in the current phase ρ_t (which cannot be directly observed and must be inferred) and is modeled by the probability $p(s_{t+1}|s_t, a_t, \rho_t)$. To learn the conditions of phase transition, we model the transitions by the probability $p(\rho_{t+1}|s_t, \rho_t)$. To learn the specific entry and exit conditions of a phase, a binary termination variable ϵ_t is learned and used that decides, whether a phase transition is possible in the current state and phase. $p(\rho_{t+1}|s_t, \rho_t)$. Terminating a phase is modeled through logistic regression:

$$p(\epsilon_t = 1|s_{t+1}, \rho_t = i) = \frac{1}{1 + e^{-\hat{\omega}_i^T \phi(s_{t+1})}} \tag{11}$$

Initiating a phase is modeled by a distribution:

$$p(\rho_t = j|s_t, \epsilon_{t-1} = 1) = \frac{e^{\hat{\omega}_j^T \phi(s_t)}}{\Sigma_k e^{\hat{\omega}_k^T \phi(s_t)}} \tag{12}$$

where i and j are some phases, $\hat{\omega}_i^T$ and $\hat{\omega}_j^T$ are weight vectors to terminate or initiate a phase and $\phi(s_t)$ is a feature vector of the regression, of which the weight vectors have to be learned.

After having learned these transition probabilities of the multi-phase model, the robot needs to learn how to act in each of the individual phases, which is solved using dynamic movement primitives $\mathcal{M}_t$, which are learned using relative entropy policy search (REPS) [18]. For a given state, the robots action is performed following a chosen DMPs trajectory $p(a_t|\mathcal{M}_t, s_t)$ until the DMP has finished. The value-based learning's reward function now is defined on the phase transition distribution $p(\rho_{t+1}|s_t, \rho_t)$ so that the sequence of DMPs between ρ_0 as the starting phase and ρ_g as the goal phase is the task to optimize.

$$r(t) = p(\rho_g|S_{t+1}, \rho_t) \tag{13}$$

Finally, learning a policy for sequencing DMPs as separate task makes sense from a standpoint in which not only one task has to be solved, but multiple tasks might be solved using the same DMPs, which can also be called 'skills' and thus allow for building a library of skills. The policy $\pi(\mathcal{M}|s_t, \tilde{\rho}_t)$ to choose the right DMP given the current state and the estimated phase should maximize the reward:

$$\underset{\pi}{max} \sum_{t=1}^{\inf} \gamma^t r(s_t, \rho_t, \mathcal{M}_t) \tag{14}$$

The optimal sequence can be learned through value iteration. In the experimental evaluation of this approach, a box had to be picked up by two hands. Using only two demonstrations, the system could successfully replicate the task when the box was in the same position. When randomly placing the box, the robot had to learn new weights for the DMPs using additional reinforcement learning, in which the phases success was labeled by hand e.g. the first phase was successful when the right hand touched the box. Using this technique, the success of lifting the box could be elevated to over 90% in comparison to 38% using just imitation learning. Using this method, the DMPs also adjusted to using different shapes.

Reward based bi-manual RL methods also have shown to be successful solving different tasks [6] requiring a high degree of dexterity. The described approach by Kroemer et al. and many others achieved high success rates not only using gripper-extended manipulators, but end-effectors mimicking human hands.

B. GAIL

Generative Adversarial Imitation Learning is a distribution matching algorithm that displays a neural technique to enable robots to mimic expert demonstrations more effectively. It can be applied by training in two steps. First a discriminator is trained on the expert demonstrations to provide high probability of correctly differentiating the observations of these demonstrations and the robots actions following the initial policy. In the second step, this discriminator is used as a reward function to train the policy by maximizing over the discriminator [10].

$$\max_{\pi} \min_{D} \mathbb{E}_{\pi}[logD(s,a)] + \mathbb{E}_{\pi\mathbb{E}}[log(1-D(s,a))] - \lambda H(\pi) \tag{15}$$

with $H(\pi) = \mathbb{E}_{\pi}[-log\pi(a|s)]$ being the γ-discounted causal entropy [23]. The goal is to find a saddle point (π, D) to the equation, at which the weights of the generative model are chosen in a way such that the discriminator cannot distinguish data generated from the demonstrations. This is an open research field for bi-manual manipulation in which Drolet et al. are working actively.

V. Supervised Learning / Behavioral Cloning

A. With Feed Forward Networks

Behavioral Cloning is a supervised learning approach in which the learning algorithm is trained through a set of state-action traces of expert demonstrations.

$$\tau = \mathbf{s}_1 \rightarrow \mathbf{a}_1 \rightarrow \mathbf{s}_2 \rightarrow \mathbf{a}_2 \rightarrow \mathbf{s}_3 \rightarrow \mathbf{a}_3 \rightarrow \cdots \tag{16}$$

It learns to directly map sensory observations to actions taken following the Markov property which states that a future state is independent of the past given the present state. Using these traces τ, a policy π_θ with is trained which minimizes the loss function and thus maximizes the likelihood of the learned policy being the one which the expert followed.

$$\mathcal{L} = \mathbb{E}_{(s,a)\sim\tau_E}[-log(\pi(a|s))] \tag{17}$$

This policy, which often is trained in a feed-forward neural network then has the form

$$\begin{aligned} a &= \pi(s) = \phi^T(s)\theta \quad or \\ a &\sim \pi(a|s) = \mathcal{N}(a|\mu(s), \Sigma(s)) \end{aligned} \tag{18}$$

This leads to very good approximated expert behavior given enough training data, but generalizes badly to non-observed states, a phenomenon also known as the covariate shift, as no action is known that leads to recovery. While there are some methods (DAGGER [19], DART [14]) to counteract this and thus able to recover from unseen states, these methods are quite time-consuming.

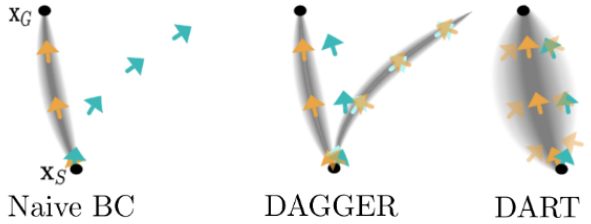

Fig. 3: Visual comparison of generalization between naive BC, DAGGER and DART. Naive BC trains with the given expert demonstrations and has bad behavior in unseen states. DAGGER queries the expert for new knowledge when encountering unseen states and is able to act robustly in the future. DART trains with the original expert data and introduces noise to increase the demonstrated state space.

For a bimanual contact-rich setup, naive behavioral cloning might thus be lacking, as a large set of demonstration data with high variability is necessary to cover sufficiently large areas. The utilization of DAGGER or DART methods could pose challenges or may not be appropriate. DAGGER may face limitations due to the absence of experts, hindering the ability to provide requested demonstrations. On the other hand, employing DART in high-precision scenarios may be impractical, as injecting noise into the demonstrations could result in imminent failure.

B. Implicit Behavioral Cloning

To address some of the limitations of feed-forward BC, implicit behavioral cloning remodels the policy from $a = \pi_\theta(s)$ to $\hat{a} = \underset{a\in\mathcal{A}}{argmin}\, E_\theta(s,a)$ where E is a continuous energy function representing the new policy. To learn the policy's weights θ the least squares Loss is replaced by an InfoNCE-style loss [16]. Now not only the observation is used as input to maximize the likelihood of a policy, but the actions will be taken into account when learning. This results in a higher dimensional function, an energy

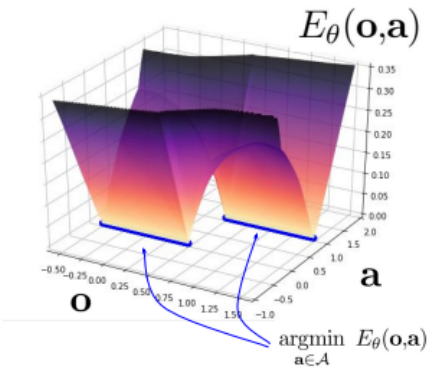

Fig. 4: A visual representation of the continuous energy function [9]

landscape, being used to estimate the policy. Usage of such implicit models is not only advantageous for discontinuous or multi-valued functions, but also brings better extrapolation for states outside of the training data's coverage [9]. They perform better than explicit BC models in many tasks. One of them being a bi-manual sweeping task in which two 6 DoF arms are required to scoop small particles into bowls; a task with vision-based inputs in which the implicit method had a success rate of about 80%, an improvement of 14% compared to explicit MSE BC.

VI. Supervised Learning with ALOHA

Another very novel and promising approach by Zhao et al. [22] addresses compounding errors of BC methods that are unable to be solved with DAGGER or DART due to the complexity of the bimanual setting. They achieve this through a combination of two concepts called action chunking and temporal ensembling implementing explicitly non-Markovian behavior. Action chunking is a concepts of psychology, in which not one action is predicted for each state,

but a sequence of actions is predicted and executed in one go. With temporal ensembling the system doesn't have to wait for one of those chunks to be finished executing. The policy is queried multiple times, so that several chunks overlapping each other are obtained. Those overlapping action chunks are then averaged, leading to smooth behavior. This combination works well even on comparable low-cost hardware. To obtain reliable and well estimated actions beyond the

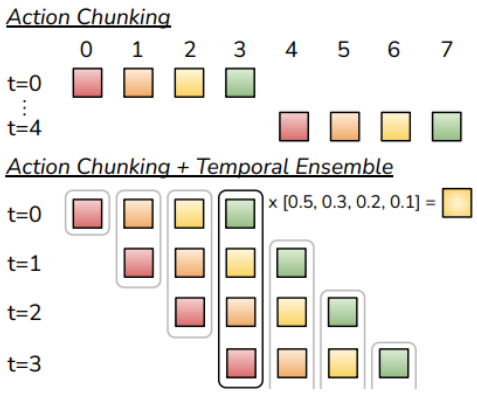

Fig. 5: using action chunking alone would generate a chain of actions every k time steps. To avoid jumping movements and increase smoothness and with the help of temporal ensemble, chunks are queried at every time step and the resulting action for the step will be chosen as weighted average of different chunks [22].

next state ALOHA (a low-cost open-source hardware system for bimanual teleoperation) employs a novel algorithm called 'Action Chunking With Transformers' (ACT). The state of the system is represented by the joint space of the bimanual robot as well as an image feed from four different cameras. Taking human expert demonstrations with a teleoperation setup, the robots next joint positions in time is seen as an actions. Differentiating between the current state and the action (the next joint position) implicitly gives the force needed to move with a PID controller. These state-action traces are then used to train a generative neural network, specifically a conditional variational autoencoder (CVAE) [20], to estimate a series of actions of length k, learning a policy $\pi(a_{t:t+k}|s_t)$.
The encoder of the CVAE is only used to train the policy (decoder) and can be discarded later on. The decoder takes in the robots state (visual camera input processed by convolutional neural networks (CNNs), the joint positions of the robot and the encoders output of the training phase - the style variables distribution) and outputs an action chunk.

Training is done by again minimizing the mean-squared loss function e.g. maximizing the log likelihood of the policy. By employing action chunks, the overall magnitude of compounding errors is minimized through the reduction of the effective horizon in lengthy trajectories, achieved by lowering the sampling rate of the policy. This also helps with spatio-temporal correlation, such as pauses in the movement. The benefits of the ACT algorithm can be observed in experimental results. After merely ten minutes of demonstrations, the robots are able to slot batteries into a remote control, slide a ziploc, and put a shoe onto human feet among others with success rates of about 90%-100% each.

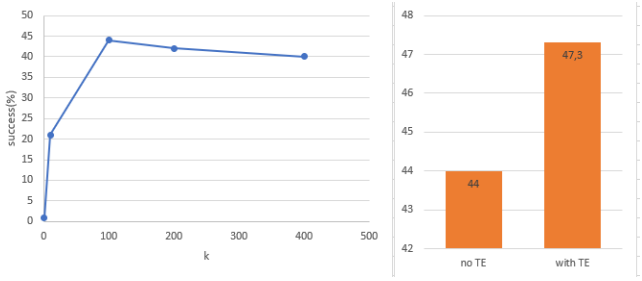

Fig. 6: Comparison of (a) The success rate of action chunking without temporal ensembling based on the size of the chunk. chunks of size k=1 had a 1% success rate across several settings of scripted or human-provided data, while chunks of size 100 improved to 44%. (b) shows the importance of temporal ensembling.

VII. Discussion

Faced with similar challenges, all of the mentioned approaches use different architectures, sensors and models and achieved comparable results. For the control, each approach employed different types of controllers. Stepputis et al. [21] and Kroemer et al. [13] implemented closely related admittance and impedance controllers, which are designed to regulate the interaction between a robot and its environment and thus making it safe to use in dynamic environments with robot-human interaction. The paper about the ACT algorithm by Zhao et al. [22] uses a traditional PID controller.
The state space is also defined differently in those methods. For the task of picking up a box with two high dexterity hands [13] included the position and orientation of the box, as well as the positions of the hands and the torque sensors of the fingers into the state space representation, the complete input dimensionality was reduced from about 60 inputs to a total of 8 using principal component analysis, such that reinforcement learning algorithms could gain insights in reasonable time. Learning how to pick up the box using only imitation learning to learn DMPs or motor skills gave a 38% success rate, applying RL (REPS) to learn the skills increased it more than 90%. [21]'s EnBIPs used a similar definition and included the force-torque sensor data, the joint angles and the translational-rotational objective positions as the state space, however without dimensionality reduction. Through the use of different teleoperation mechanics, Stepputis et al. found, that especially the force-torque data was critical to achieve success and was able to achieve 90-100% success at inserting an bracket onto pegs depending on the tolerance and disturbance in the task.
The ACT algorithm, which can be placed into the family of BC algorithms, defined the state space using the joint positions as well as visual data of multiple cameras and a style variable obtained by the encoder for a total of 1202x512

dimensions, which is a huge increase compared to the other methods. Using this representation to predict action chunks allowed solving tasks like insertion tasks, knot tying, and object transference from one hand to another. To compare the different approaches reasonably fairly, the cube transfer, which can be somewhat compared to the box pickup as it required stable contact of an object from two opposing sides had an success rate of 50-86% based on the quality of the demonstration data. The insertion task, which consisted of one arm holding the bracket and one holding the pin with very low tolerance, resulted in at least 20% success, which needs to be interpreted cautiously as the arms also had to pick up the bracket or pin respectively.
The ALOHA paper, inspired by psychological research of natural human behavior, showed, that using a method based on the Markov property might not be the only and best strategy nowadays for solving such complex tasks. It increases the success of BC algorithms in contact rich settings vastly, coming at the cost of losing ad-hoc flexibility, as action suggestions of old action chunks are still valued for new situations.

VIII. Conclusion

Our discussion of the prominent approaches to bimanual manipulation (via imitation learning) is important for gleaning meaningful insights into how practitioners can approach open robotics problems that require human-like coordination. By extension, this discussion helps to uncover areas of research that have not been explored yet. We summarized some immportant findings of success-critical factors in different areas of imitation learning which would be interesting to be looked into to integrate into existing or new methods. We hope this survey can contribute to not only advances in controllers for manipulation but also to controllers for locomotion, where both settings require high-dimensional and continuous control (as discussed in the Humanoid Robotics Seminar at TU Darmstadt).

References

[1] Amor, Neumann, Kamthe, Kroemer, Peters. "Interaction Primitives for Human-Robot Cooperation Tasks". In: (2014).

[2] Bishop. *Pattern-Recognition-and-Machine-Learning*. Springer, 2006.

[3] Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, Giuseppe Oriolo. "Robotics; Modelling, Planning and Control". In: (2000).

[4] Rui Camacho and Donald Michie. "Behavioral Cloning: A Correction". In: (1995).

[5] Campbell, Amor. "Bayesian Interaction Primitives: A SLAM Approach to Human-Robot Interaction". In: (2017).

[6] Yuanpei Chen et al. "Bi-DexHands: Towards Human-Level Bimanual Dexterous Manipulation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2023), pp. 1–15. DOI: 10.1109/TPAMI.2023.3339515.

[7] d'Avella, Bizzi. "Shared and specific muscle synergies in natural motor behaviors". In: (2005). URL: https://www.pnas.org/doi/10.1073/pnas.0500199102.

[8] Dominici, Ivanenko, Cappellini, d'Avella, Mondì, Cicchese, Fabiano, Silei, Paolo, Giannini. "Locomotor primitives in newborn babies and their development". In: (2011).

[9] Pete Florence et al. "Implicit Behavioral Cloning". In: *CoRR* abs/2109.00137 (2021). arXiv: 2109.00137. URL: https://arxiv.org/abs/2109.00137.

[10] Jonathan Ho and Stefano Ermon. "Generative Adversarial Imitation Learning". In: *CoRR* abs/1606.03476 (2016). arXiv: 1606.03476. URL: http://arxiv.org/abs/1606.03476.

[11] Ijspeert, Nakanishi, Hoffmann, Pastor, Schaal. *Dynamical Movement Primitives: Learning Attractor Models for Motor Behaviors*. 2013. URL: https://homes.cs.washington.edu/~todorov/courses/amath579/reading/DynamicPrimitives.pdf.

[12] Ijspeert, Nakanishi, Schaal, Stefan. "Learning control policies for movement imitation and movement recognition". In: (2003).

[13] Oliver Kroemer et al. "Towards learning hierarchical skills for multi-phase manipulation tasks". In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. 2015, pp. 1503–1510. DOI: 10.1109/ICRA.2015.7139389.

[14] Michael Laskey et al. "DART: Noise Injection for Robust Imitation Learning". In: *1st Annual Conference on Robot Learning, CoRL 2017, Mountain View, California, USA, November 13-15, 2017, Proceedings*. Vol. 78. Proceedings of Machine Learning Research. PMLR, 2017, pp. 143–156. URL: http://proceedings.mlr.press/v78/laskey17a.html.

[15] Moro, Tsagarakis, Caldwell. "On the kinematic motion primitives (kMPs) - theory and application". In: (2012).

[16] Aäron van den Oord, Yazhe Li, and Oriol Vinyals. "Representation Learning with Contrastive Predictive Coding". In: *CoRR* abs/1807.03748 (2018). arXiv: 1807.03748. URL: http://arxiv.org/abs/1807.03748.

[17] Paraschos , Daniel, Peters, Neumann. "Using Probabilistic Movement Primitives in Robotics". In: (2017).

[18] J. Peters, K. Mülling, and Y. Altun. "Relative Entropy Policy Search". In: Max-Planck-Gesellschaft. Menlo Park, CA, USA: AAAI Press, July 2010, pp. 1607–1612.

[19] Stéphane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. "No-Regret Reductions for Imitation Learning and Structured Prediction". In: *CoRR* abs/1011.0686 (2010). arXiv: 1011.0686. URL: http://arxiv.org/abs/1011.0686.

[20] Kihyuk Sohn, Xinchen Yan, and Honglak Lee. “Learning structured output representation using deep conditional generative models”. In: *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*. NIPS’15. Montreal, Canada: MIT Press, 2015, pp. 3483–3491.

[21] Stepputtis, Bandari, Schaal, Amor. “A System for Imitation Learning of Contact-Rich Bimanual Manipulation Policies”. In: (2022).

[22] Tony Z. Zhao et al. *Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware*. 2023. arXiv: 2304.13705 [cs.RO].

[23] Zhengyuan Zhou, Michael Bloem, and Nicholas Bambos. “Infinite Time Horizon Maximum Causal Entropy Inverse Reinforcement Learning”. In: *IEEE Transactions on Automatic Control* PP (Nov. 2017), pp. 1–1. DOI: 10.1109/TAC.2017.2775960.