Inferring Versatile Behavior from Demonstrations by Matching Geometric Descriptors

Niklas Freymuth[1*] **Nicolas Schreiber**[1] **Philipp Becker**[1] **Aleksander Taranovic**[2]

Gerhard Neumann[1]

[1]**Autonomous Learning Robots**
Karlsruhe Institute of Technology
Karlsruhe, Germany

[2]**Bosch Center for Artificial Intelligence**
University of Tubingen
Tubingen, Germany

Abstract: Humans intuitively solve tasks in versatile ways, varying their behavior in terms of trajectory-based planning and for individual steps. Thus, they can easily generalize and adapt to new and changing environments. Current Imitation Learning algorithms often only consider unimodal expert demonstrations and act in a state-action-based setting, making it difficult for them to imitate human behavior in case of versatile demonstrations. Instead, we combine a mixture of movement primitives with a distribution matching objective to learn versatile behaviors that match the expert's behavior and versatility. To facilitate generalization to novel task configurations, we do not directly match the agent's and expert's trajectory distributions but rather work with concise geometric descriptors which generalize well to unseen task configurations. We empirically validate our method on various robot tasks using versatile human demonstrations and compare to imitation learning algorithms in a state-action setting as well as a trajectory-based setting. We find that the geometric descriptors greatly help in generalizing to new task configurations and that combining them with our distribution-matching objective is crucial for representing and reproducing versatile behavior.

Keywords: Imitation Learning, Versatile Skill Learning, Distribution Matching

1 Introduction

Imitation Learning (IL) [1, 2, 3] from human demonstrations is challenging as humans often solve tasks in versatile ways. Even the same person might solve a task differently when confronted with it multiple times. Behavior can be versatile in terms of task-level decisions, such as planning a route to a target, and individual actions, such as randomly pausing during a movement to think about the next steps. Most recent IL approaches [4, 5, 6, 7] model the behavior in state-action space using Gaussian policies, which assumes the behavior is unimodal and cannot capture this planning versatility. Yet, this assumption is violated by most human expert datasets, often causing poor generalization to human demonstrations [8]. Additionally, IL approaches often need immense amounts of data to achieve generalization [9, 8]. We tackle these challenges by a feature matching approach to IL that is able to generalize to novel contexts from a small amount of expert demonstrations. Such contexts are (typically low-dimensional) vectors that describe a specific task configuration within a family of related tasks, such as e.g., the coordinates of goal locations to reach. The resulting approach, Versatile Imitation from Geometrically Observed Representations (VIGOR), models distributions that match expert trajectories in terms of concise geometric behavioral descriptors. VIGOR represents distributions using Gaussian Mixture Models (GMMs) over Probabilistic Motion Primitives (ProMPs) [10], and utilizes descriptors that are designed to abstract away from concrete contexts and thus facilitate generalization to novel contexts in a sample-efficient way. An example of such descriptors would be geometric features in the form of (only) the distance between a target

*correspondence to `niklas.freymuth@kit.edu`

6th Conference on Robot Learning (CoRL 2022), Auckland, New Zealand.

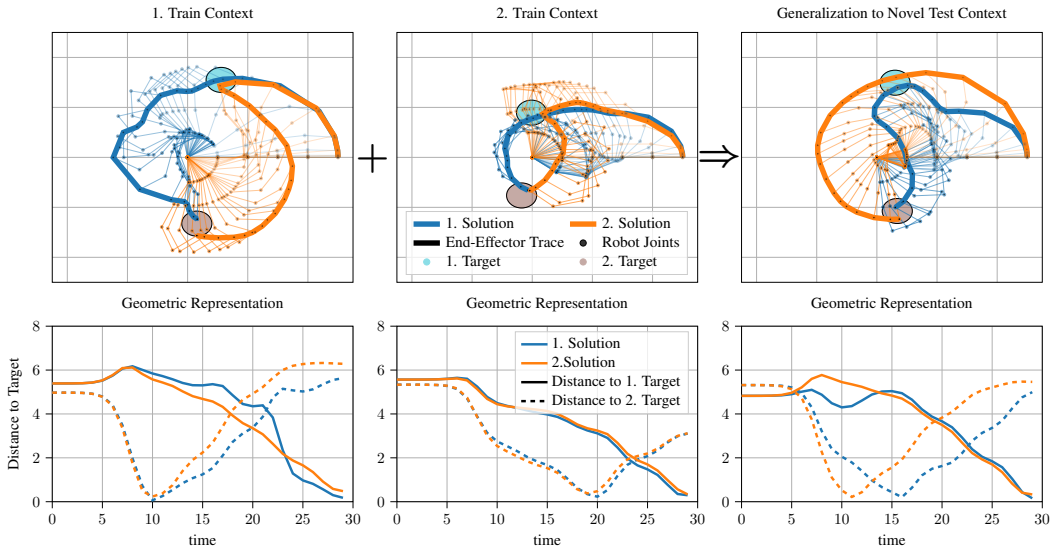

Figure 1: End-effector traces (Top) and distances to target centers (Bottom) for a planar point-reaching task on 3 different contexts. (Left, Middle) Human experts solve the task by reaching the blue target and ending their trajectory in the grey one. (Right) VIGOR matches distributions of behavioral descriptors, such as the depicted target distances, of these demonstrations. This produces versatile behavior in unseen contexts, such as new target positions. Depicted are 2 different component means from a trained GMM Policy. For each context, we show the joint configurations for the first trajectory in blue and the second trajectory in orange.

and the robot's end-effector, plus features that denote the smoothness of the robot. While similar behavioral descriptors have found use in Inverse Optimal Control [11, 12] to recover cost functions from demonstrations, common IL approaches instead use state representations that include both relative and absolute information [5, 13]. The exact form of this set of behavioral descriptors is task-dependent and allows the user to include domain knowledge. Figure 1 shows how VIGOR works on an exemplary reaching task. For training the GMMs, we rely on recent approaches [14, 15, 16] for distribution matching and Variational Inference. For inference, we train individual GMMs for each training context and an arbitrary number of test contexts.

We evaluate VIGOR on a suite of versatile robotic tasks, where demonstrations are collected from human experts through teleoperation. We compare our method to standard Behavioral Cloning (BC) [17], BC with a GMM policy [13, 18], Generative Adversarial Imitation Learning (GAIL) [5] and Inverse Reinforcement Learning (IRL) techniques [7]. We find that these methods either fail to learn useful behavior because they average over multiple solutions, or that they are unable to capture the full versatility of the demonstrations. Contrary to this, VIGOR accurately models highly versatile behavior from already a small number of trajectories demonstrated by human experts. We perform extensive ablation studies to showcase the importance of different parameters and design choices. Datasets and code can be found at `https://www.github.com/NiklasFreymuth/VIGOR`.

To summarize, our list of contributions is as follows:

- We infer *multi-modal distributions* of desired trajectories from highly versatile human expert trajectories by matching distributions over behavioral descriptors between learner and expert.
- Our approach is, to the best of our knowledge, the first Adversarial Imitation Learning method to utilize *concise behavioral descriptors* to facilitate generalization to novel contexts from already a small number of demonstrations.
- We conduct *experiments with human demonstrations* in simulation and on a real robot and find that VIGOR accurately models highly versatile human behavior, outperforming various Imitation Learning baselines.

2 Related Work

Imitation Learning for Skills. A common way to represent skills over trajectories is via Movement Primitives (MPs) [19, 10]. While learning individual MPs from human demonstrations is a simple regression problem, modeling versatile behavior requires a more sophisticated model that can represent such multi-modality, e.g., a mixture model. Using Expectation Maximization (EM) [20], both [21] and [22] fit a single GMM over MP parameters of multiple demonstrations for different contexts and use them to generalize to novel contexts by conditioning. Yet, as they only fit a single GMM for all contexts, they do not explicitly focus on representing versatility. Recent work [23] instead learns a mixture over contextualized MPs using Gaussian Mixture Regression. This has been extended to non-linear relations between MP parameters and context [18] by using Mixture Density Networks (MDNs) [24]. While the above approaches work for tasks with a small number of modes, optimizing MDNs can be challenging in the case of many modes in the demonstrations. Osa et al. [25] use planning algorithms to reproduce human demonstrations in a trajectory-based setting. They tune the parameters of a cost function such that the planned trajectories match the demonstrations and use the extracted cost function to guide the trajectory optimization process. Yet, this approach is limited to uni-modal demonstrations and requires engineered cost functions.

Imitation Learning by Distribution Matching. Classical IL approaches [26, 27] employed feature expectation matching to learn a policy from behavioral descriptors of expert demonstrations. Similarly, a recent body of work [11, 12] in Inverse Optimal Control utilizes geometric behavioral descriptors that are similar to ours to learn cost functions for manipulation tasks from a few demonstrations. Whereas these approaches linearly match the moments/expectations of their features, our method instead matches a versatile distribution over non-linear features using the reverse Kullback-Leibler Divergence (KL) [28]. Directly matching the distribution rather than its moments means that our method can represent complex and multi-modal distributions. More recently, a new class of distribution matching approaches has gained popularity with the advent of Generative Adversarial Nets [29]. Starting from GAIL [5], a whole class of distribution matching based IL approaches was developed [4, 30, 31, 32, 33, 34]. All of these approaches commonly work in a step-based setting and minimize some divergence between the state-action occupancy or state marginals. They generally do not consider versatile behavior, and additionally need to interact with the environment during training in order to generate states to match. We instead work in a trajectory-based setting , which allows us to abstract away the environment's dynamics and learn entirely offline, i.e., without any environment interactions. Common methods in this setting usually employ a maximum likelihood approach, which, as previously discussed, performs poorly on versatile data. As a solution to this problem, Becker et al. [16] propose using the Information-Projection [35] instead, which is able to focus on individual modes of data rather than averaging over all modes of data. This approach has been extended to IRL [36]. Here, we build on these methods, generalizing them to sequential data and geometric behavioral descriptors.

Comparison-Based Approaches. Brown et al. [6] perform IRL by utilizing provided rankings of demonstrations to train a discriminator on a comparison-based loss. As the discriminator learns to favor samples with a high rank, it can be used as a reward function after training. This has been extended by Disturbance-based Reward Extrapolation (D-REX) [7], which automatically generates rankings by fitting a BC policy on the demonstrations and subsequently draws samples with different levels of noise from this policy. We adapt D-REX to our trajectory-based setting as a baseline. Similar to VIGOR, this adapted method trains a model on expert demonstrations on training contexts to optimize GMM policies on novel test contexts. However, while D-REX learns a reward function to do so, VIGOR iteratively (re-)trains a discriminator on the expert demonstrations and policy samples.

3 Foundations

In this section, we briefly cover the foundations and previous work that our method builds on.

Probabilistic Movement Primitives. For a trajectory $\tau = (\tau_1, \cdots, \tau_T)$, ProMPs represent the elements as $\tau_t = \boldsymbol{\Phi}(t)^T \boldsymbol{w}$, where τ_t represents a vector of the desired joint angles at time-step t [18] Note that $\boldsymbol{w}$ does not depends on t and that τ can thus be computed at an arbitrary resolution T. Here $\boldsymbol{\Phi}(t)$ are time-dependent features, usually radial basis functions centered around different time points, and $\boldsymbol{w}$ denotes the parameter vector. For a single demonstration, the parameters $\boldsymbol{w}$ are fitted using simple linear regression and compactly represent the entire trajectory.

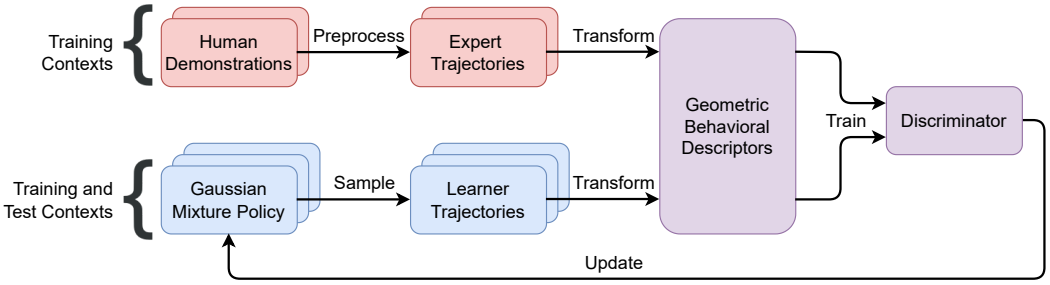

Figure 2: Schematic of VIGOR. We generate a set of expert trajectories from human demonstrations and then transform them into geometric behavioral descriptors. These descriptors are then fed into a discriminator. This process is repeated for samples of a separate GMM policy for each training and test context. We then train the discriminator to distinguish between human and policy trajectories. The geometric descriptors are designed to abstract away from concrete task configurations, causing the discriminator to distinguish how well a trajectory performs rather than which context it acts on. As a result, the discriminator can be used to improve the policies.

Variational Inference for Gaussian Mixture Models. We build on a recent class of efficient Variational Inference approaches that allow modeling versatile behavior with GMMs [16, 14]. These approaches use the Information-Projection [35] to minimize the reverse Kullback-Leibler Divergence $\mathrm{KL}\big(q(\boldsymbol{w})||p(\boldsymbol{w})\big)$ between a model $q(\boldsymbol{w})$ and a target distribution $p(\boldsymbol{w})$. In our case, $p(\boldsymbol{w})$ is the distribution of expert trajectories. Arenz et al. [14] introduced an upper-bound objective based on a variational decomposition. The resulting approach, Variational Inference by Policy Search (VIPS) [14, 15], repeatedly solves the objective

$$q\left(\boldsymbol{w}\right) = \arg\min_{q(\boldsymbol{w})} \mathbb{E}_{q(\boldsymbol{w},\boldsymbol{z})}\left[\log \frac{\hat{q}\left(\boldsymbol{w}\right)}{p\left(\boldsymbol{w}\right)}\right] + \mathrm{KL}\big(q(\boldsymbol{z})||\hat{q}(\boldsymbol{z})\big) + \mathbb{E}_{q(\boldsymbol{z})}\Big[\mathrm{KL}\big(q\left(\boldsymbol{w}|\boldsymbol{z}\right)||\hat{q}\left(\boldsymbol{w}|\boldsymbol{z}\right)\big)\Big], \quad (1)$$

where $\hat{q}$ denotes the model from the previous iteration. This objective decomposes into individual optimizations for the components and categorical distributions, which Arenz et al. [14] solve using trust-region methods from policy search [37, 38].

Distribution Matching for Gaussian Mixture Models. We cannot directly work with VIPS, as we do not know $p(\boldsymbol{w})$ but only have the expert demonstrations. To address this issue, Becker et al. [16] propose using density ratio estimation [39] approaches to approximate the $\log(\hat{q}(\boldsymbol{w})/p(\boldsymbol{w}))$ term in Equation 1, removing the dependency on $p(\boldsymbol{w})$. Specifically, they use logistic regression to estimate the density ratio. The resulting approach, Expected Information Maximization (EIM), resembles Generative Adversarial Nets [29] where the discriminator effectively also estimates a density ratio.

4 Inferring Versatile Behaviors by Matching Geometric Descriptors

Our approach estimates versatile behavior in form of GMM distributions over ProMP weights for all training and test contexts of the given task. We note that while this setup requires the test contexts in advance, it does not need interaction with the real environment since the behavioral descriptors are directly computed from the proposed trajectory and its context. To capture correlations in the demonstrations, we utilize full-covariance GMM components. For each planned trajectory, we compute concise descriptors such as distances to key-points that isolate the performance of the trajectory from its context to allow for sample-efficient generalization to novel contexts. We then build on EIM [16] and use a discriminator to infer GMMs from demonstrations by matching the learner's and expert's distributions in the resulting geometric feature space. Towards this end, we generalize EIM to sequential data and modify the policy parameterization to accommodate for our setting. Figure 2 illustrates an overview of our approach. We provide pseudocode in Appendix A.

Context-Specific Mixture Policies We want to model the expert's behavior for a given small set of training configurations of the task, denoted as training contexts $\boldsymbol{c}_{\text{train}}$, and a set of unseen test contexts $\boldsymbol{c}_{\text{test}}$. We consider the set of all contexts as $\boldsymbol{c} = \boldsymbol{c}_{\text{train}} \cup \boldsymbol{c}_{\text{test}}$. As context-dependent GMMs[2] are hard

[2]Such GMMs could in principle be represented by MDNs. However, attempts with MDN and full-covariance Gaussian components were highly unsuccessful in our experiments.

to learn and we assume a relatively small amount of training contexts, 6 in all our experiments, with multiple demonstrations per training context, we resort to maintaining a single GMM-Policy $q_c(\boldsymbol{w})$ for each context c. This non-amortized formulation simplifies the policy updates as we can optimize the policy for each context individually, given the density ratio estimator, using highly efficient, tailored methods for full-rank GMM approximations [14, 15].

Distribution Matching of Behavioral Descriptors We assume access to behavioral descriptors $\boldsymbol{O} = f_c(\boldsymbol{w})$ that represent features encoded in the parameter vector $\boldsymbol{w}$ with respect to the context c of $\boldsymbol{w}$. For simplicity, we will assume that the mapping f_c between $\boldsymbol{w}$ and $\boldsymbol{O}$ is deterministic, such that we can easily evaluate $\boldsymbol{O}$ for any desired plan $\boldsymbol{w}$. We want to match the distribution of behavioral descriptors of the demonstrator while optimizing for $q(\boldsymbol{w})$, i.e,

$$q^*(\boldsymbol{w}) = \arg\min_{q(\boldsymbol{w})} \mathrm{KL}\big(q\left(\boldsymbol{O}\right) || p(\boldsymbol{O})\big) = \arg\min_{q(\boldsymbol{w})} \mathbb{E}_{q(\boldsymbol{w})}\left[\int_{\boldsymbol{O}} p(\boldsymbol{O}|\boldsymbol{w}) \log \frac{q(\boldsymbol{O})}{p(\boldsymbol{O})} d\boldsymbol{O}\right],$$

where $p(\boldsymbol{O}|\boldsymbol{w})$ is a Dirac delta distribution defined by the mapping f_c and $q(\boldsymbol{O}) = \int p(\boldsymbol{O}|\boldsymbol{w}) q(\boldsymbol{w}) d\boldsymbol{w}$. As the mapping between $\boldsymbol{w}$ and $\boldsymbol{O}$ is deterministic, the EIM algorithm can be directly applied to this setup with the difference that the discriminator is trained using $\boldsymbol{O}$ instead of $\boldsymbol{w}$. Moreover, while we a learn different distributions $q_c(\boldsymbol{w})$ for each context c, we use the same discriminator for all contexts. This way, we can infer distributions $\{q_c(\boldsymbol{w}) | c \in \boldsymbol{c}_{\text{test}}\}$, i.e. infer versatile trajectories for unseen scenarios.

Density Ratio Estimation for Sequential Behavioral Descriptors As in our case, $\boldsymbol{w}$ encodes the desired trajectory τ, the behavioral descriptors are typically computed per time-step, i.e., $\boldsymbol{O} = (\boldsymbol{o}_1, \dots, \boldsymbol{o}_T)^T$, where each $\boldsymbol{o}_t$ is a feature vector. To enable the classifier to deal with sequential data, we consider a sequence-to-sequence neural network $\boldsymbol{\phi}(\boldsymbol{O}) = \boldsymbol{\phi}((\boldsymbol{o}_1, \dots, \boldsymbol{o}_T)^T) = (y_1, \dots, y_T)^T$. The network receives a sequence of inputs $\boldsymbol{O} = (\boldsymbol{o}_1, \dots, \boldsymbol{o}_T)^T$ and outputs a sequence of values $(y_1, \dots, y_T)^T$, where each y_i may depend on multiple $\boldsymbol{o}_j$. In each iteration of our optimization, we (re-)train this network to discriminate between the provided demonstrations $\boldsymbol{O}^{(p)}$ of $\boldsymbol{c}_{\text{train}}$, and an equal number of policy samples $\boldsymbol{O}^{(q)}$ drawn uniformly from $\boldsymbol{c}$. We first consider the case where we want to discriminate full trajectories as in EIM. Denoting the sigmoid function as σ, the discriminator can straightforwardly be trained on a binary cross-entropy loss of the sum of sequence values $\hat{y} = \sum_{t=0}^{T} y_t$, i.e., by minimizing

$$\mathrm{BCE}\big(\boldsymbol{\phi}(\boldsymbol{O}), \boldsymbol{O}^{(p)}, \boldsymbol{O}^{(q)}\big) = -\mathbb{E}_{\boldsymbol{O}^{(q)}}\left[\log\left(\sigma\left(\hat{y}\right)\right)\right] - \mathbb{E}_{\boldsymbol{O}^{(p)}}\left[\log\left(1 - \sigma\left(\hat{y}\right)\right)\right], \tag{2}$$

w.r.t. $\boldsymbol{\phi}(\boldsymbol{O})$. Equation 2 recovers the log density ratio $\boldsymbol{\phi}(\boldsymbol{O}) = \log \boldsymbol{O}^{(p)} - \log \boldsymbol{O}^{(q)}$ at convergence [39]. However, this cost function only provides a single training sample per trajectory and is therefore hard to use for a small number of demonstrations. By utilizing the sequential structure of the trajectories, the classification error can be computed for each time-step, i.e. we minimize

$$\mathrm{BCE}_{\text{step}}\big(\boldsymbol{\phi}(\boldsymbol{O}), \boldsymbol{O}^{(p)}, \boldsymbol{O}^{(q)}\big) = -\mathbb{E}_{\boldsymbol{O}^{(q)}}\left[\sum_{t=0}^{T} \log\left(\sigma\left(y_t\right)\right)\right] - \mathbb{E}_{\boldsymbol{O}^{(p)}}\left[\sum_{t=0}^{T} \log\left(1 - \sigma\left(y_t\right)\right)\right] \tag{3}$$

w.r.t. $\boldsymbol{\phi}(\boldsymbol{O})$. Finally, we train an ensemble of discriminators instead of a single one, using their average logit as the log density ratio estimate for the policy updates. For the network architecture, we find that simple $2-4$ layer $1d$ convolutional neural networks ($1d$-CNNs) work best in our experiments. We compare this choice to other sequential network architectures the ablations in Appendix D.

Geometric Behavioral Descriptors VIGOR utilizes geometric descriptors that abstract away from a concrete task configuration to generalize to new configurations. To this end, we encode the current state along the desired trajectory with respect to relative geometric features rather than absolute values. The resulting geometric descriptors cause trajectories of similar performance to appear similar in feature space regardless of their context, allowing for sample-efficient generalization to novel contexts. We note that such a descriptor space can often be straightforwardly constructed from the geometry of the task by composing distances of the end-effector to key-points of the objects in a scene. For example, in object manipulation, this space can be composed of distances of the end-effector to the corners of a box to push.

5 Experiments

All experiments use human demonstrations. The tasks were selected due to their highly multi-modal nature and because they allow for an easy collection of human demonstrations. We instructed the

demonstrators to solve the task in varying ways to create versatile solutions. As a preprocessing step, we fit a ProMP for each human demonstration and filter the resulting ProMPs such that only successful demonstrations remain. This step ensures that the human demonstrations can be imitated by the learner and conveniently allows to compress all demonstrations to a fixed length in a principled fashion. We use the resulting preprocessed expert demonstrations for all experiments unless otherwise stated. Similarly, unless noted otherwise, all experiments use 6 training and 6 test contexts, and 5 GMM components for each configuration and method. Experimentally, this number of components is sufficient for matching the distribution of expert descriptors for the considered tasks, whereas more components generally only lead to spurious improvements. We experiment with less components in the ablations in Appendix D. Unless noted otherwise, we do *not* train the categorical distribution of the GMM components, using a uniform distribution instead. We evaluate the GMM policies using a number of samples for each component of each *test* context, and then rank these components according to the average performance of their samples. We then report statistics of the best component of the resulting value over the random seeds. Note that reporting the best component leads to a fair comparison to unimodal approaches. For baselines that do not use GMM policies, we instead draw samples of the trained policies for each test context and average the performance of all of them. For each experiment, we report a measure of how well the task is performed in the main paper, and additional success rates in Appendix C. We repeat each experiment for 10 random seeds. Appendix E shows all hyperparameters and training details.

Baselines. We compare VIGOR to a number of IL baselines, using three state-action baselines, namely GAIL [5], behavioral cloning (BC(S)) and behavioral cloning with a GMM policy (BC-GMM(S)) [13]. For all experiments, the state-action baselines receive geometric descriptors of the current state and a time-step as state information, and output the velocities of the robot joints as actions. Further, we also consider three trajectory-based methods. These are standard behavioral cloning (BC(T)), behavioral cloning with a GMM policy (BC-GMM(T)) (see e.g., Zhou et al. [18]), and a modified version of D-REX (EM+D-REX). The latter is chosen due to its similarities to VIGOR, which are detailed in Appendix B.1. We refer to Appendix B for details on the baselines and to Appendix C for further results on all experiments described below.

Planar Reacher. First, we consider the introductory task shown in Figure 1. In this task, the goal is to reach an intermediate target area with the end-effector of a 5 Degrees of Freedom (DoF) robot with joints of length 1 before ending the trajectory in a final target area. The geometric descriptors for this task are given by the euclidean distance of the end-effector to both targets, paired with the average velocity and acceleration of the joints to encode the smoothness of the motion. This yields a total of 4 features per time-step. The left and middle panels of Figure 1 depict example demonstrations for two training contexts (top) and the corresponding distances between the end-effector and targets over time (bottom). We explore other choices for the geometric descriptors in the ablations in Appendix D. For evaluation, we use the *Distance to Boundaries* of the target areas. For the first target, we consider the minimum distance of the end-effector to the boundary; for the second target, we consider the distance at the last time-step. We train all methods on 5 demonstrations per training context. The left of Figure 3 shows results on test contexts. We find that only VIGOR and EM+D-REX can consistently get close to both targets. Yet, EM+D-REX can only learn a single mode and fails to reproduce the versatility of the demonstrations. We investigate this in more detail in Appendix C.1.

7-D Panda Reacher. We repeat the above point-reaching task using a 7 DoF Franka Emika Panda robot and 3-D intermediate and goal positions. Since we do not care about the end-effector's rotation, the last DoF can be ignored, leading to an effective action dimension of 6. We use 8 basis functions per action dimension for the MPs, leading to a total of MP dimension of 48 and thus a high-dimensional problem space. The expert demonstrations are collected using a teleoperation setup with a virtual twin that records joint values over time. Appendix C.2 explains the setup in more detail. We use the same geometric descriptors as for the planar reacher task. The right of Figure 3 shows target distances on test contexts. Similar to the previous task, only VIGOR and EM+D-REX can consistently get close to both targets while the other methods regularly fail to pursue at least one of the two targets.

Box Pusher. We also evaluate our approach on a contact-rich robot manipulation task. In this task, a simulated Franka Emika Panda robot has a rod attached to its end-effector and needs to use it to push a rectangular box to a given goal position and orientation. The box always starts at the same position, and the task context is given by the desired (x, y) translation of the box, plus its desired rotation in degrees. For simplicity, the task is learned in task space with a fixed height. The demonstrations contain the starting position of the robot as well as its trajectory. Here, the solutions are versatile

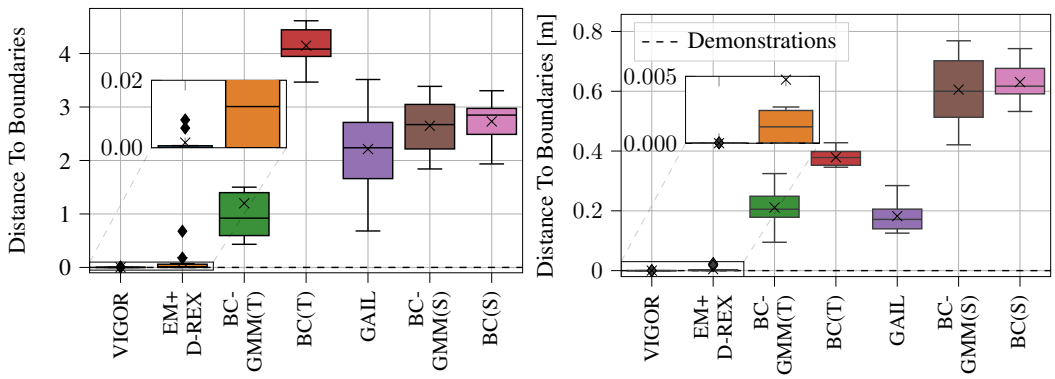

Figure 3: Mean target distance of samples from the best policy component on the Planar (left) and Panda (right) Reaching tasks for test contexts. We find that a trajectory-based setting and the use of mixture policies help performance, as can be seen from VIGOR, EM+D-REX, and BC-GMM (T). Similarly, using a discriminator to iteratively optimize the policies is advantageous on the reaching tasks, as seen from VIGOR and GAIL. Finally, the clear distribution matching objective of VIGOR and EM+D-REX seems crucial for imitation from a low number of versatile demonstrations. VIGOR uniquely combines the above properties and is able to reliably reach both targets for both tasks.

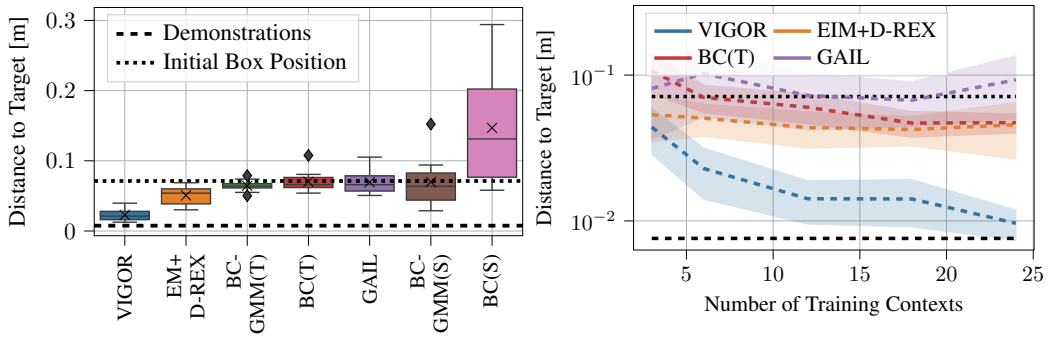

Figure 4: Mean target distance of samples from the best policy component on the Box Pushing task for test contexts. The dotted line denotes the average corner distance between the initial and the desired box position. The dashed line shows the performance of the human demonstrations. (Left) VIGOR is able to consistently push the box to the right configuration. Opposed to this, the other methods only marginally improve over the initial box positions, and in some cases perform worse than it. (Right) We find that VIGOR benefits most from additional training contexts, almost matching the performance of the demonstrator on unseen contexts for 24 training contexts.

because the box can be pushed both from the inside and outside. The geometric descriptors consist of the (x, y) distances of the end-effector to the *initial* and *desired final* position of the corners of the box. To facilitate generalization between contexts, we translate the basis of these distances such that the desired final box aligns with the origin of the coordinate system. We additionally add the end-effector velocity and acceleration as well as the time-step, resulting in 19 features per time-step. This geometric description of the task allows for IL without simulating or explicitly modeling the box, significantly speeding up the training process and facilitating generalization to both new contexts and real-world robots. We use 3 demonstrations per training context and evaluate the average distance of the corners of the final box to that of the *desired final* position. The contact with the box causes a mismatch between a planned trajectory and its execution on the robot, which we illustrate in Appendix C.3. Results are shown in the left of Figure 4. Additionally, the right of Figure 4 shows how the performance varies when changing the number of training contexts. Finally, we illustrate how the planned trajectories of VIGOR perform on a real robot. To this end, we create a real-world replica of the simulated box that the robot manipulates. Rollouts for different policy components of VIGOR trained on 24 training contexts on an exemplary test context are shown in Figure 5.

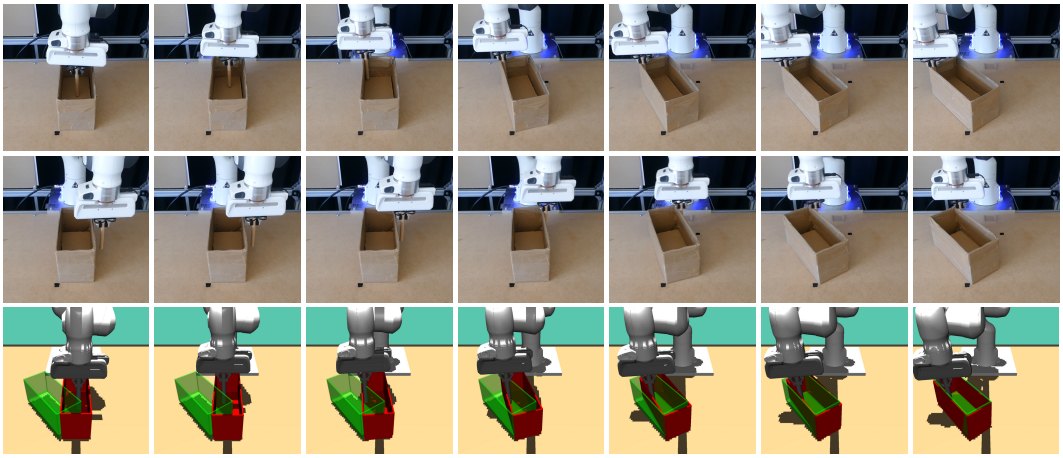

Figure 5: Trajectory executions of three component means of a VIGOR policy trained on 24 training contexts for the same test context. The first two rows show a real Franka Panda executing the planned end-effector trajectories. The last row shows the same setup in the simulation, where the target box is overlaid in green. All three executions push the box in different ways, capturing the versatility of the human demonstrations; while the robot in the first row pushes the box from the inside, the robot in the second row pushes it from the outside. The third trajectory also pushes from the inside but is much farther away from the corner of the box.

Ablations. We conduct extensive ablation studies to investigate how our design choices affect our approach. We find that a concise choice of behavioral descriptors, a preprocessing of the expert demonstrations to ProMPs, and a $1d$-CNN discriminator are crucial for performance. We also notice that the policy benefits from additional components and that we see modest improvements for an ensemble of discriminators and the stepwise loss proposed in Equation 3. We refer to Appendix D for the full results and more thorough ablations of both VIGOR and EM+D-REX.

6 Limitations

Scale. We currently train and maintain a separate GMM over ProMP parameters for each context, leading to linear space and time complexity w.r.t. the total number of contexts. One way to address this challenge would be to instead train a joint Mixture Density Network $q(\boldsymbol{w}|c)$ over all contexts, which is left for future work. Additionally, representing trajectories with a single ProMP limits them to single smooth movements. To alleviate this, MP *chaining* [40, 41, 42] can be straightforwardly integrated into our approach to allow for the representation of more complex movements

Geometric Descriptors. VIGOR facilitates generalization to novel task configurations by using geometric descriptors. While this allows the user to include domain knowledge, it also requires a clear idea about which aspects of the task are important. Instead, geometric descriptors could be automatically extracted from the task for example by using NNs processing point clouds [43, 44].

Re-training. Finally, our method assumes that the test configurations $\boldsymbol{c}_{\text{test}}$ are known in advance. Since the test configurations are optimized jointly with the training configurations, the only way to integrate new test configurations is to re-train the algorithm. In the future, we want to alleviate this issue by recovering a reward representation from ranked intermediate policies of the given training configurations, essentially combining our approach with some of the ideas of D-REX.

7 Conclusion

We proposed VIGOR, a novel feature-matching approach to Imitation Learning in environments with versatile solutions. Utilizing a combination of movement primitives, mixture policies and matching geometric behavioral descriptors, our method can closely imitate the behavior distribution of human experts from a few demonstrations. We show the effectiveness of our approach on a suite of challenging robot coordination tasks. The results show that VIGOR is able to closely match the distribution of the demonstrator, outperforming the chosen baselines in all considered settings.

Acknowledgments

NS and GN were supported by the Carl Zeiss Foundation under the project JuBot (Jung Bleiben mit Robotern). The authors acknowledge support by the state of Baden-Württemberg through bwHPC.

References

[1] S. Schaal. Learning from demonstration. *Advances in Neural Information Processing Systems (NeurIPS)*, 9, 1996.

[2] B. D. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469–483, 2009.

[3] A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne. Imitation learning: A survey of learning methods. *ACM Computing Surveys (CSUR)*, 50(2):1–35, 2017.

[4] J. Fu, K. Luo, and S. Levine. Learning robust rewards with adverserial inverse reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2018.

[5] J. Ho and S. Ermon. Generative adversarial imitation learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 29, 2016.

[6] D. Brown, W. Goo, P. Nagarajan, and S. Niekum. Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations. In *International Conference on Machine Learning (ICML)*, pages 783–792. PMLR, 2019.

[7] D. S. Brown, W. Goo, and S. Niekum. Better-than-demonstrator imitation learning via automatically-ranked demonstrations. In *Conference on Robot Learning (CoRL)*, pages 330–359. PMLR, 2020.

[8] M. Orsini, A. Raichuk, L. Hussenot, D. Vincent, R. Dadashi, S. Girgin, M. Geist, O. Bachem, O. Pietquin, and M. Andrychowicz. What matters for adversarial imitation learning? *Advances in Neural Information Processing Systems (NeurIPS)*, 34, 2021.

[9] T. Osa, J. Pajarinen, G. Neumann, J. Bagnell, P. Abbeel, and J. Peters. An algorithmic perspective on imitation learning. *Foundations and Trends in Robotics*, 7(1-2):1–179, 2018.

[10] A. Paraschos, C. Daniel, J. R. Peters, and G. Neumann. Probabilistic movement primitives. *Advances in Neural Information Processing Systems (NeurIPS)*, 26, 2013.

[11] P. Englert, N. A. Vien, and M. Toussaint. Inverse kkt: Learning cost functions of manipulation tasks from demonstrations. *The International Journal of Robotics Research*, 36(13-14):1474–1488, 2017.

[12] P. Englert and M. Toussaint. Learning manipulation skills from a single demonstration. *The International Journal of Robotics Research*, 37(1):137–154, 2018.

[13] A. Mandlekar, D. Xu, J. Wong, S. Nasiriany, C. Wang, R. Kulkarni, L. Fei-Fei, S. Savarese, Y. Zhu, and R. Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *5th Annual Conference on Robot Learning*, 2021.

[14] O. Arenz, G. Neumann, M. Zhong, et al. Efficient gradient-free variational inference using policy search. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, volume 80, pages 234–243. Proceedings of Machine Learning Research, 2018.

[15] O. Arenz, M. Zhong, and G. Neumann. Trust-region variational inference with gaussian mixture models. *J. Mach. Learn. Res.*, 21:163–1, 2020.

[16] P. Becker, O. Arenz, and G. Neumann. Expected information maximization: Using the i-projection for mixture density estimation. In *International Conference on Learning Representations (ICLR)*, 2020.

[17] M. Bain and C. Sammut. A framework for behavioural cloning. In *Machine Intelligence 15*, pages 103–129, 1995.

[18] Y. Zhou, J. Gao, and T. Asfour. Movement primitive learning and generalization: Using mixture density networks. *IEEE Robotics & Automation Magazine*, 27(2):22–32, 2020.

[19] S. Schaal. Dynamic movement primitives-a framework for motor control in humans and humanoid robotics. In *Adaptive Motion of Animals and Machines*, pages 261–280. Springer, 2006.

[20] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1): 1–22, 1977.

[21] K. Mülling, J. Kober, O. Kroemer, and J. Peters. Learning to select and generalize striking movements in robot table tennis. *The International Journal of Robotics Research*, 32(3): 263–279, 2013.

[22] M. Ewerton, G. Neumann, R. Lioutikov, H. B. Amor, J. Peters, and G. Maeda. Learning multiple collaborative tasks with a mixture of interaction primitives. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1535–1542. IEEE, 2015.

[23] A. Pervez and D. Lee. Learning task-parameterized dynamic movement primitives using mixture of gmms. *Intelligent Service Robotics*, 11(1):61–78, 2018.

[24] C. M. Bishop. *Mixture density networks*. Aston University, 1994.

[25] T. Osa, A. M. G. Esfahani, R. Stolkin, R. Lioutikov, J. Peters, and G. Neumann. Guiding trajectory optimization by demonstrated distributions. *IEEE Robotics and Automation Letters*, 2(2):819–826, 2017.

[26] P. Abbeel and A. Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first International Conference on Machine Learning (ICML)*, page 1, 2004.

[27] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey. Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pages 1433–1438. Chicago, IL, USA, 2008.

[28] S. Kullback and R. A. Leibler. On information and sufficiency. *The Annals Of Mathematical Statistics*, 22(1):79–86, 1951.

[29] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. *Advances in Neural Information Processing Systems (NeurIPS)*, 27, 2014.

[30] I. Kostrikov, K. K. Agrawal, D. Dwibedi, S. Levine, and J. Tompson. Discriminator-actor-critic: Addressing sample inefficiency and reward bias in adversarial imitation learning. In *International Conference on Learning Representations (ICLR)*, 2018.

[31] F. Torabi, G. Warnell, and P. Stone. Adversarial imitation learning from state-only demonstrations. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pages 2229–2231, 2019.

[32] I. Kostrikov, O. Nachum, and J. Tompson. Imitation learning via off-policy distribution matching. *arXiv preprint arXiv:1912.05032*, 2019.

[33] S. K. S. Ghasemipour, R. Zemel, and S. Gu. A divergence minimization perspective on imitation learning methods. In *Conference on Robot Learning (CoRL)*, pages 1259–1277. PMLR, 2020.

[34] X. Zhang, Y. Li, Z. Zhang, and Z.-L. Zhang. f-gail: Learning f-divergence for generative adversarial imitation learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 33:12805–12815, 2020.

[35] K. P. Murphy. *Machine Learning: A Probabilistic Perspective*. MIT press, 2012.

[36] N. Freymuth, P. Becker, and G. Neumann. Versatile inverse reinforcement learning via cumulative rewards. *4th Robot Learning Workshop at Neural Information Processing Systems (NeurIPS)*, 2021.

[37] J. Peters, K. Mulling, and Y. Altun. Relative entropy policy search. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

[38] A. Abdolmaleki, R. Lioutikov, J. R. Peters, N. Lau, L. P. Reis, and G. Neumann. Model-based relative entropy stochastic search. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 3537–3545, 2015.

[39] M. Sugiyama, T. Suzuki, and T. Kanamori. *Density Ratio Estimation in Machine Learning*. Cambridge University Press, 2012.

[40] C. Daniel, G. Neumann, O. Kroemer, J. Peters, et al. Hierarchical relative entropy policy search. *Journal of Machine Learning Research*, 17:1–50, 2016.

[41] G. Neumann, W. Maass, and J. Peters. Learning complex motions by sequencing simpler motion templates. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 753–760, 2009.

[42] S. Manschitz, J. Kober, M. Gienger, and J. Peters. Learning to sequence movement primitives from demonstrations. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4414–4421. IEEE, 2014.

[43] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, pages 652–660, 2017.

[44] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in Neural Information Processing Systems (NeurIPS)*, 30, 2017.

[45] S. Wang, S. Toyer, A. Gleave, and S. Emmons. The `imitation` library for imitation learning and inverse reinforcement learning. https://github.com/HumanCompatibleAI/imitation, 2020.

[46] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[47] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research (JMLR)*, 22(268):1–8, 2021. URL http://jmlr.org/papers/v22/20-1364.html.

[48] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8): 1735–1780, 1997.

[49] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems (NeurIPS) 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[50] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In Y. Bengio and Y. LeCun, editors, *3rd International Conference on Learning Representations (ICLR) 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[51] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research (JMLR)*, 15(1):1929–1958, Jan. 2014. ISSN 1532-4435.

[52] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning (ICML) - Volume 37*, ICML'15, page 448–456. JMLR.org, 2015.

Algorithm 1: VIGOR

Input: Contexts $\boldsymbol{c} = \boldsymbol{c}_{\text{train}} \cup \boldsymbol{c}_{\text{test}}$
Input: Mappings f_c with $\boldsymbol{O} = f_c(\boldsymbol{w})$ for each $c \in \boldsymbol{c}$
Input: Expert Descriptors $\boldsymbol{O}^{(p)} = \{\boldsymbol{O}_c^{(p)} | c \in \boldsymbol{c}_{\text{train}}\}$
Input: Initial Policies $\{q_c(\boldsymbol{w}) | c \in \boldsymbol{c}\}$
Output: Converged Policies $\{q_c(\boldsymbol{w}) | c \in \boldsymbol{c}\}$

$n_{\text{samples}} \leftarrow |\boldsymbol{O}^{(p)}|/|\boldsymbol{c}|$
$\hat{q}_c(\boldsymbol{w}) \leftarrow q_c(\boldsymbol{w}) \quad \forall c \in \boldsymbol{c}$
while *not converged* **do**
  Gather new policy samples
  $\boldsymbol{O}^{(q)} \leftarrow \{\}$
  for $c \in \boldsymbol{c}$ **do**
    $\hat{\boldsymbol{O}}_c^{(q)} \leftarrow \{\boldsymbol{w}_c^{(j)}\}_{j=1\ldots n_{\text{samples}}} \sim q_c(\boldsymbol{w})$
    $\boldsymbol{O}^{(q)} \leftarrow \boldsymbol{O}^{(q)} \cup \{f_c(\boldsymbol{w}) | \boldsymbol{w} \in \hat{\boldsymbol{O}}_c^{(q)}\}$
  end

  Train discriminator $\phi(\boldsymbol{O})$ (c.f. Eq. 3)
  $\phi(\boldsymbol{O}) \leftarrow \arg\min_{\phi(\boldsymbol{O})} \text{BCE}(\phi(\boldsymbol{O}), \boldsymbol{O}^{(p)}, \boldsymbol{O}^{(q)})$

  Update policies for each context with discriminator $\phi(\boldsymbol{O})$ (c.f. Eq. 1,)
  for $c \in \boldsymbol{c}$ **do**

$$q_c(\boldsymbol{w}) \leftarrow \arg\min_{q(\boldsymbol{w})} \mathbb{E}_{q(\boldsymbol{w},\boldsymbol{z})}[\phi(\boldsymbol{O}(\boldsymbol{w}))] + \text{KL}\big(q(\boldsymbol{z}) || \hat{q}_c(\boldsymbol{z})\big) + \mathbb{E}_{q(\boldsymbol{z})}\Big[\text{KL}\big(q(\boldsymbol{w}|\boldsymbol{z}) || \hat{q}_c(\boldsymbol{w}|\boldsymbol{z})\big)\Big]$$

    $\hat{q}_c(\boldsymbol{w}|\boldsymbol{z}) \leftarrow q(\boldsymbol{w}|\boldsymbol{z}) \quad \forall \boldsymbol{z}$
  end
end
return $\{q_c(\boldsymbol{w}) | c \in \boldsymbol{c}\}$

A Pseudocode

Pseudocode for VIGOR is provided in Algorithm 1.

B Baselines

We compare VIGOR to a number of strong IL baselines in both a trajectory-based and state-action setting.

B.1 EM+D-REX

We modify Disturbance-based Reward Extrapolation (D-REX) [7], a state-of-the-art IL algorithm to work in a trajectory-based setting and with a GMMs policy. For this, we first use EM to fit a GMM with a small number of components on each configuration in $\boldsymbol{c}_{\text{train}}$, and use the resulting distributions to draw samples for the D-REX reward. To generate the rankings required by D-REX, we multiply the covariance of each component of the resulting GMM with different scalars, resulting in GMMs with different noise levels. More precisely, we fit the EM-GMM on the samples, and multiply its component-wise covariance with a *Base Noise* σ_{base} to get an initial distribution. We repeat this process for a number of *Noise Levels*, linearly increasing the noise up to $\sigma_{\text{base}} \cdot \lambda_{\text{max}}$, where we call λ_{max} the *Noise Multiplier*. We then draw samples from each noise level of the GMMs corresponding to each context in $\boldsymbol{c}_{\text{train}}$ and rank them against each other using the comparison-based loss of D-REX, where samples drawn from a lower noise level are ranked higher. The resulting ranked demonstrations

are transformed into geometric descriptors $\boldsymbol{O}$ and used to train a reward function $R(\boldsymbol{O})$, which in turn is used by a policy optimization algorithm to optimize policies on unknown contexts $\boldsymbol{c}_{\text{test}}$. For the policy optimization algorithm, we use VIPS [14, 15]. To make results more stable, we optimize on a scaled reward $\hat{R}(\boldsymbol{O}) = \alpha_{\text{scale}} R(\boldsymbol{O})$ instead of the regular reward, where α_{scale} is a scalar *Reward Scale*. We call the resulting algorithm EM+D-REX. Comparing EM+D-REX and VIGOR, both approaches train a discriminator that uses expert demonstrations on training contexts to fit novel test contexts. Similarly, both make use of VIPS to optimize marginal policies to match multi-modal distributions over geometric behavioral descriptors. However, EM+D-REX uses the discriminator to represent a reward function, while VIGOR iteratively re-trains the discriminator until convergence of the test policies.

B.2 State-Action Baselines

We also compare our approach to common state-action based imitation learning algorithms to see how well these approaches work on versatile human demonstrations. More precisely, we compare to BC [17] and Generative Adversarial Imitation Learning (GAIL) [5] as implemented in the *Imitation* [45] repository. The state-action baselines use velocities as actions, and receive the geometric descriptors of the current robot state plus the current time-step as state information. For the Point Reaching tasks, we also experiment with encoding which of the points has been reached to make the tasks Markovian. We use the Proximal Policy Optimization (PPO) [46] implementation of Stable Baselines3 [47] as the policy for GAIL. To accommodate for versatility on an action level, we also compare to BC-GMM [13], which trains a policy whose head outputs the parameters of a GMM over actions per state. We note that we do not make use of the Low Noise Evaluation Trick proposed in Mandlekar et al. [13], which leads to minor improvements in their experiments and does not affect our results. During evaluation, we also generate each trajectory from a fixed component rather than sampling a new component mean per step for consistency with the other approaches. Experimentally, we found no significant difference in performance between these two evaluation methods. In our experiments, both BC and BC-GMM also make use of an auxiliary entropy regularization term that is similar to Zhou et al. [18]. This prevents the covariances from collapsing. All state-action baselines use regular MLPs for their policy.

B.3 Trajectory-based Baselines

Finally, we employ BC and BC-GMM on a trajectory-based level, i.e., we imitate full trajectories instead of individual state-action pairs. These baselines are added to see how simple (multi-modal) behavioral cloning works on full expert trajectories. In this setup, the methods see as input the context of the task, e.g., the position of the target points for the point reaching tasks, and output a contextual distribution $q(\boldsymbol{w}|c)$ over ProMP parameters. This distribution is implemented as a simple MLP. To discriminate between state-action and trajectory-based baselines, we append suffixes '(S)' and '(T)' respectively. For example, we denote state-action BC as 'BC(S)' and trajectory-based Mixture-Density BC as 'BC-GMM(T)'.

C Additional Task Information and Results

C.1 Planar Reacher

Setup. Each context is specified by its target positions, which are drawn from independent isotropic Gaussians with means $(0.5, 2.5)$ and $(0.5, -2.5)$ and standard deviation 0.5. Both targets have a radius of $r = 0.5$ to be considered *reached*. The robot always starts with all joints at a resting position of $0\,\text{deg}$. We collect demonstrations via a joystick-based setup to control the end-effector's (x, y)-position and rotation. This is paired with an inverse kinematics controller to control the joints.

Success Rates. We define a demonstration as successful if it reaches both target circles and ends its trajectory in the second circle, i.e., if its *target distance* is 0. As such, a demonstration is considered successful if it has the same target distance as the expert demonstrations. The left of Figure 6 shows success rates on test contexts for different approaches trained on 6 training contexts.

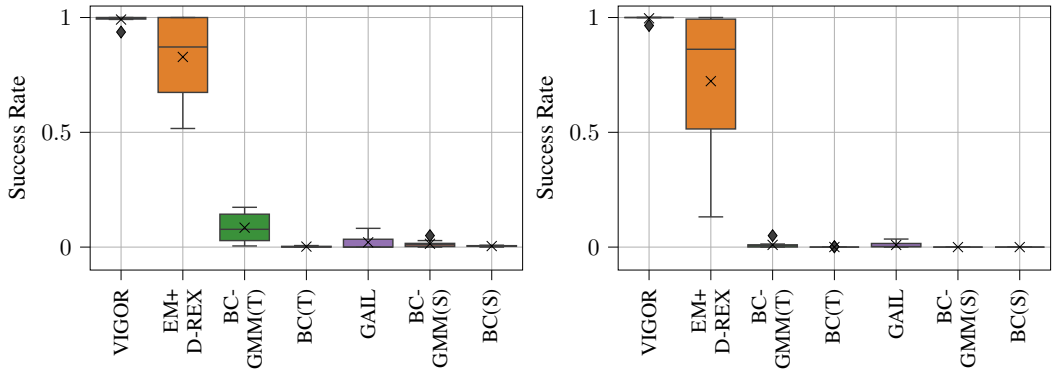

Figure 6: (Left) Mean success rates on test contexts for the Planar Reacher task. (Right) Mean success rates on test contexts for the Panda Reacher task. For both tasks, VIGOR consistently reaches both target circles, while all other methods except EM+D-REX struggle to do so. In general, the baselines with a GMM policy perform better than those without. GAIL performs the best of any state-action based approach.

Qualitative results. To explore the versatility of solutions learned by VIGOR and EM+D-REX, we visualize exemplary policies learned by both methods in Figures 7. We find that VIGOR produces versatile policies that closely match the demonstrations of the expert, with most components finding a different solution to the given task and samples of these components generally hitting both targets. At the same time, the policies of EM+D-REX often collapse to only 1 or 2 different solutions, likely due to a lack of a clear distribution matching objective.

C.2 Panda Reacher

Setup. In the teleoperation setup, the human demonstrator moves a physical robot. The robot then sends its movement to a virtual twin. The joint values of the twin are recorded over time to generate expert demonstrations. An overview of the virtual part of this setup is given in Figure 8. The environment ranges from $(0, -0.5, 0)^T$ to $(1, 0.5, 1)^T$ meters. The targets are drawn randomly from uniform distributions with range 0.1 meters and means at $(0.5, 0.2, 0.6)^T$ and $(0.3, -0.1, 0.3)^T$ meters respectively. A target is considered reached within a radius of 0.05 meters. The robot starts all trajectories at its default resting position.

We preprocess the human demonstrations by removing initial steps until the robot starts moving, and repeating the last recorded time-step before fitting the ProMPs to ensure that the fit trajectories end near the goal position. This is done to counteract a potential over-smoothing effect of the ProMPs fitting the human trajectories.

Success Rates. The right of Figure 6 shows success rates on test contexts for different approaches trained on 6 training contexts. A demonstration is considered successful if it reaches both target circles and ends its trajectory in the second circle, i.e., if its *target distance* is 0cm. This corresponds to the target distance of the expert demonstrations.

C.3 Box Pusher

Setup. Demonstrations are collected using a virtual setup similar to that of the Planar Reacher task. The demonstrator controls a mouse in the (x, y)-plane that the end-effector of the robot follows using inverse kinematics. The rotation of the end-effector is fixed to have it point straight down. The height of the end-effector is fixed such that the rod is slightly above the table. The trajectories, including the initial position of the end-effector, are fit in the (x, y) plane of the task-space for simplicity. To roll out a given trajectory on the robot, the (x, y) coordinates over time are concatenated with a fixed z-value and a downward rotation are given to the inverse kinematic controller. We sample the contexts from box translations and rotations that are feasible to demonstrate within a single smooth movement. As there tends to be a high correlation between the (x, y)-displacement and the rotation angle in the resulting contexts, we make sure that the training contexts are balanced w.r.t. this correlation.

In order for the state-action baselines to be able to choose an initial position for the trajectory, we use a separate copy of the behavioral descriptors as input for the state-action at time-step 0. In other words, we have 2 inputs in the neural network for every feature. One of these is always 0 except at step 0, where it is set to the geometric descriptors. The other is always set to the geometric descriptors, except at step 0, where it is set to 0. The output of the policy for this first time-step is then interpreted as the starting position relative to the center of the box. Outputs in later steps correspond to velocities in the (x, y)-plane.

Task visualization. Figure 9 shows final states of the Box Pusher task for VIGOR for 6 test contexts on test contexts for policies trained with demonstrations from 24 training contexts. Figure 10 shows the difference between desired and executed trajectories for 3 learned GMM component means on the same test context. These differences result from the contact with the (comparably heavy) box and are compensated by our approach.

Success Rates. The left of Figure 11 shows success rates on test contexts for different approaches trained on 6 training contexts. A demonstration is considered successful if the average distance of the corners of the final box to that of the desired final position is less than 1.5cm, which roughly corresponds to the maximum error made by any expert demonstration.

Results on training contexts. Results of VIGOR trained on 6 training contexts on the Box Pusher task for *training* contexts can be seen on the right of Figure 11.

Online task variants. To further validate the effectiveness of our geometric descriptors, we experimented with variants of BC and BC-GMM, where the distances are computed w.r.t. the current box position. We find that this choice of descriptors leads to diverging behavior on test contexts.

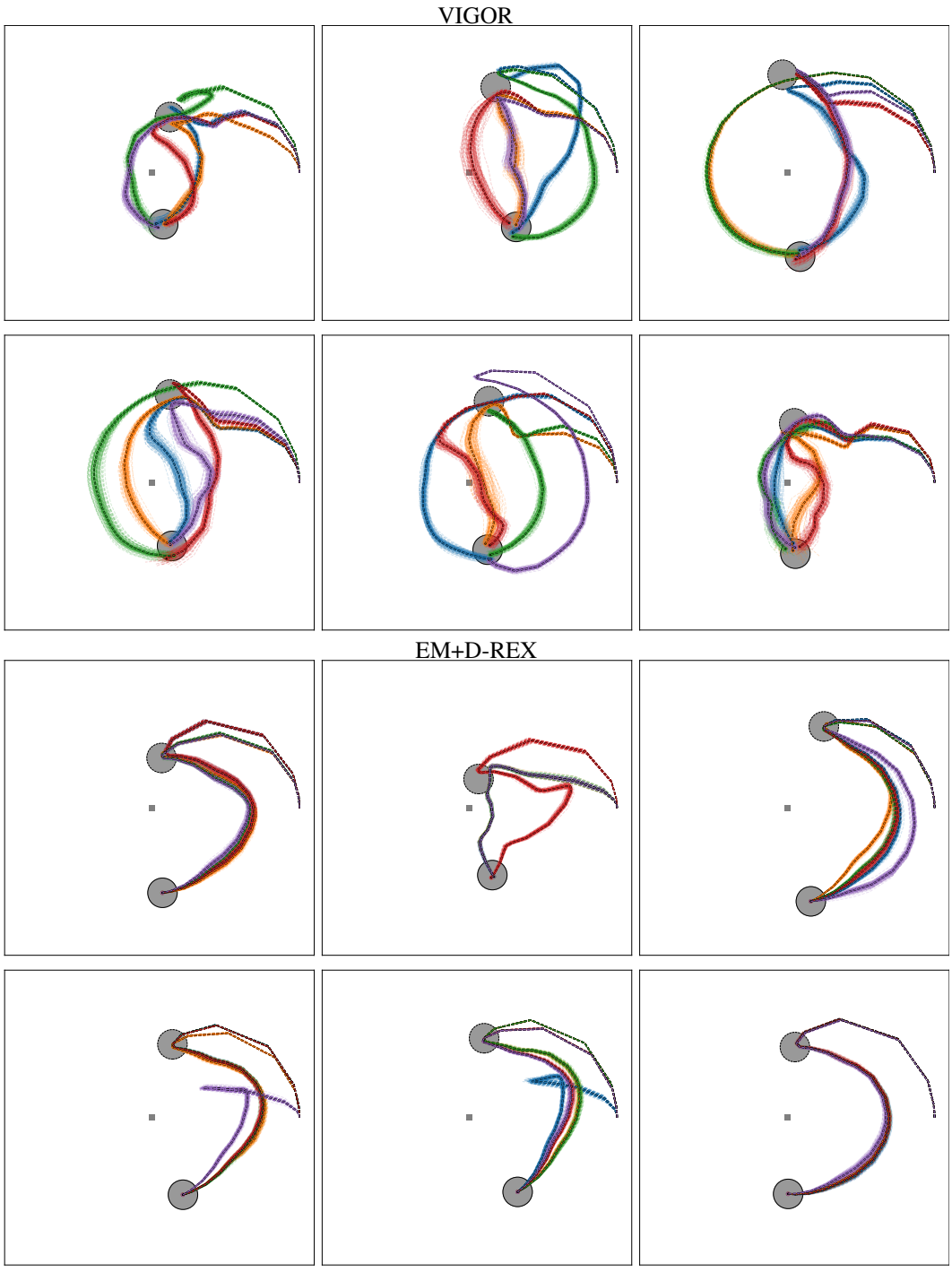

Figure 7: Visualization of end-effector traces of rollouts sampled from policies trained with VIGOR and EM+D-REX. Component means are marked with a black border, and for each component 100 samples are drawn using the same color. Note how for most contexts, VIGOR uses the components to specialize on different solutions, and how samples from this component are feasible variations of this solution. For EM+D-REX the components generally reach the targets but do not capture the versatility in the behavior.

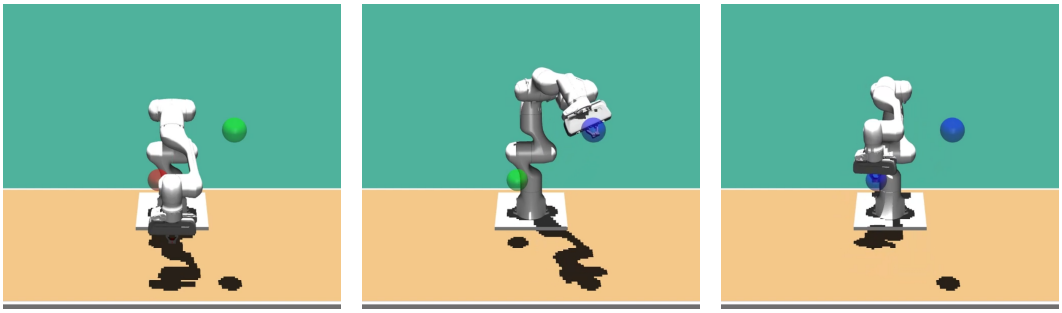

Figure 8: Virtual part of the setup for collecting human demonstrations for the Panda Reacher task. Starting from a resting position (left), the human demonstrator is tasked to reach the intermediate target (green). Once reached, both targets change color (middle). The task is successful if both targets have been reached (right).

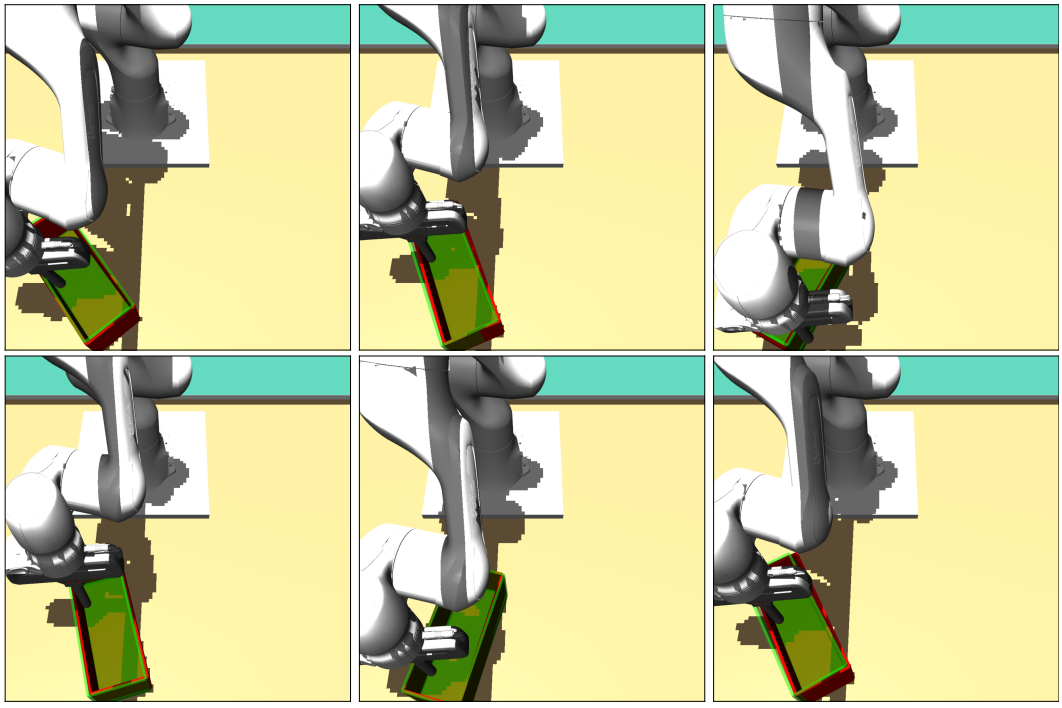

Figure 9: Visualization of the final state of the Box Pusher task on 6 test contexts for VIGOR policies trained on 24 training contexts. The pushed boxes (red) need to align as closely as possible with the target box positions (green).

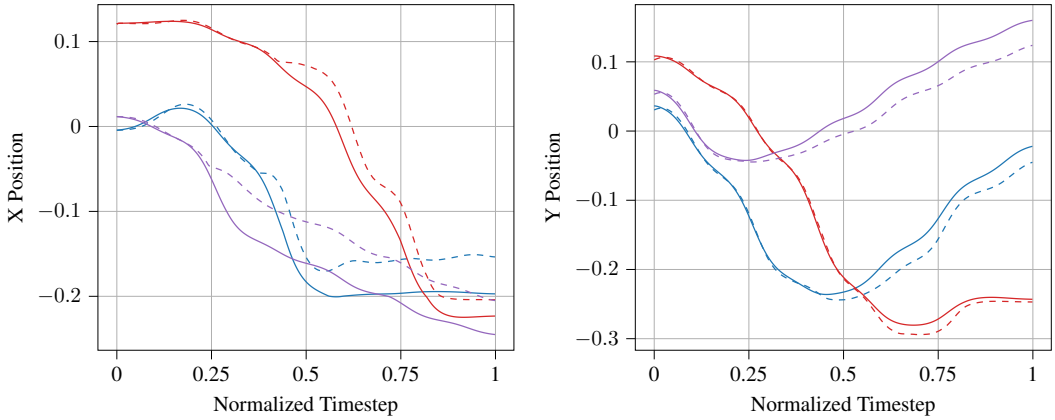

Figure 10: Difference between planned (straight line) and executed (dotted line) trajectories for 3 learned GMM component means on the same test context. The executed trajectories closely match the desired ones up to the context with the box, where the force required to push the box changes the trajectory. Note that VIGOR learns from geometric descriptors of planned expert demonstrations and is thus able to compensate for the contact with the box.

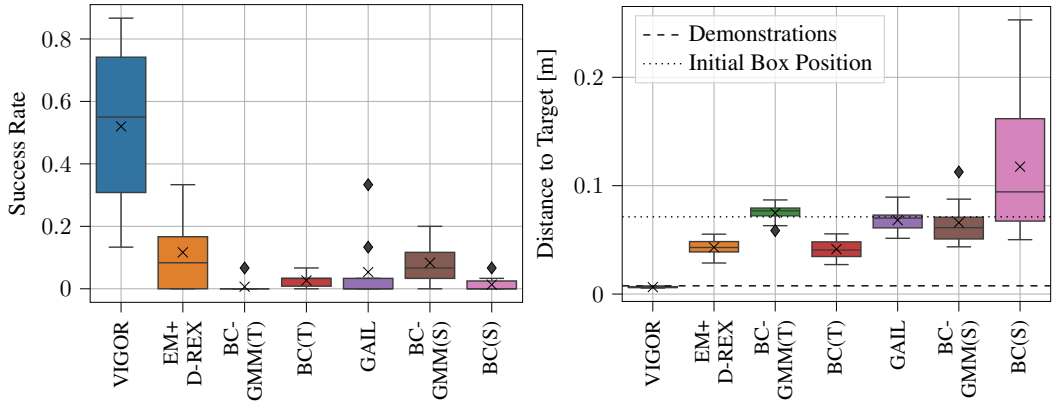

Figure 11: (Left) Mean success rates on test contexts for the Box Pusher task. We find that VIGOR produces the most successful demonstrations, while EM+D-REX and BC-GMM (S) are able to sometimes solve the task. Interestingly, BC-GMM (T) does not produce successful solutions even though it improves the distance to target, as seen on the left of Figure 4. We find that this corresponds to trajectories that push the box in the right general direction, but do not account for a precise combination of target positions and angle. (Right) Mean distance to target for training contexts of the box pushing task. Most methods reliably improve over the initial box position on training contexts. Interestingly, EM+D-REX benefits very little from evaluating on the training set, presumably because the direct correspondence to the training data is lost by the intermediate reward representation. VIGOR reaches human performance on this setup, suggesting that it can match the distribution of behavioral descriptors of the human demonstrations.

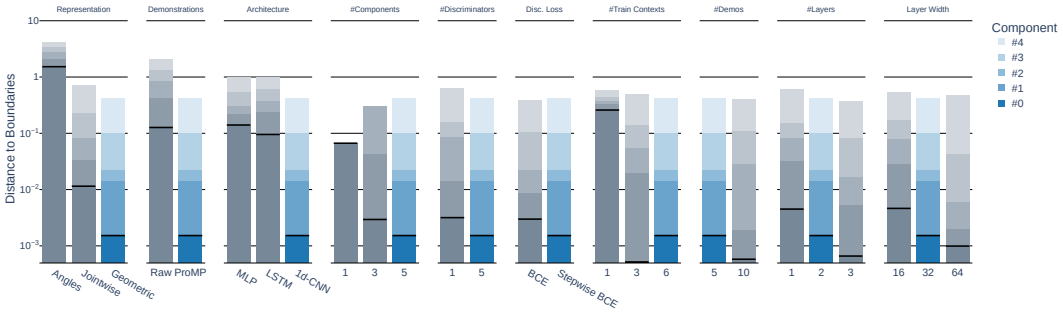

Figure 12: Full ablation study of VIGOR on the Planar Reacher task. VIGOR benefits from concise behavioral descriptors and is better suited to fit ProMP fits of the human demonstrations rather than the demonstrations themselves. We benefit from additional data in form of more demonstrations and more training contexts, as well as from additional components and discriminators.

D Ablations

We perform additional ablations for VIGOR and EM+D-REX on both the Planar Reacher and the Panda Reacher task. The ablations for VIGOR and EM+D-REX on the Planar Reacher task can be seen in Figures 12 and 13 respectively. The ablations for the Panda Reacher tasks for both methods are found in Figures 14 and 15. All figures show the performance of all mixture components of VIGOR and EM+D-REX compared to various ablations. In all figures, each bar is split into segments showing the ranked components of the learned mixture policies. In the main experiments, we evaluate the performance of the best component. This is highlighted by a black line. The blue/orange bars show the performance of VIGOR/EM+D-REX respectively, the grey bars those of the ablations.

Overall, we ablate the choice of geometric descriptors, trying both a concatenation of joint angles and context (*Angles*) as well as distances and velocity and acceleration for each robot joint (*Jointwise*). Additionally, we look at the performance of VIGOR when trained directly on human demonstrations that are sup-sampled to T time-steps (*Raw*) without first fitting a ProMP on the said demonstrations. We also evaluate the discriminator's architecture, trying out both a shared MLP per step (*MLP*) with an additional encoding of the time-step, and a Long Short-Term Memory (*LSTM*) [48]. Both architectures are configured to be of similar size to the $1d$-CNN. We also evaluate the effects of different numbers of mixture components (*#Components*), using an ensemble of discriminators (*#Discriminators*), the stepwise BCE loss (*Disc. Loss*, c.f. Eq. 3), and the amount of training contexts (*#Contexts*). Finally, we inspect the number of used demonstrations per context (*#Demos)*, and the number of $1d$-CNN layers (*#Layers*) as well as the number of convolutional channels per layer (*Layer Width*).

For EM+D-REX, we also look at how the number of fit EM components influences the performance, and how the algorithm-specific hyperparameters (see Appendix B.1) need to be tuned for good performance. The ablations show that EM+D-REX performs well if tuned right, but that it is very sensitive to the choice of hyperparameters. The most important choice of hyperparameter seems to be the number of EM components compared to the number of demonstrations per training context. Depending on the task and the remaining parameters, EM+D-REX either prefers a number of components similar to the number of demonstrations, or only a single component. We assume that this sensitivity to the choice of hyperparameters comes from the setup of EM+D-REX. While VIGOR iteratively improves its policies to match the distribution of the expert demonstrations under the geometric behavioral descriptors, EM+D-REX trains a reward function once and then uses this recovered reward function for training policies on the test contexts. As a result, any mistake in the recovered reward will directly influence the way that the newly trained policies are optimized.

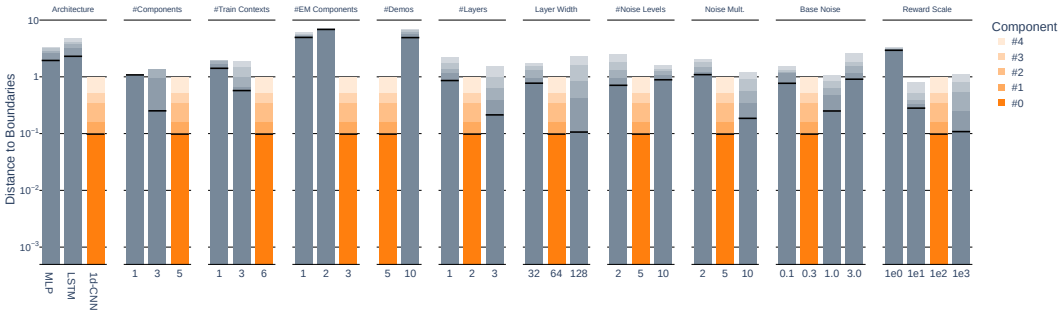

Figure 13: Ablation study of EM+D-REX on the Planar Reacher task. EM+D-REX performs well if tuned correctly, but is very sensitive to its choice of hyperparameters.

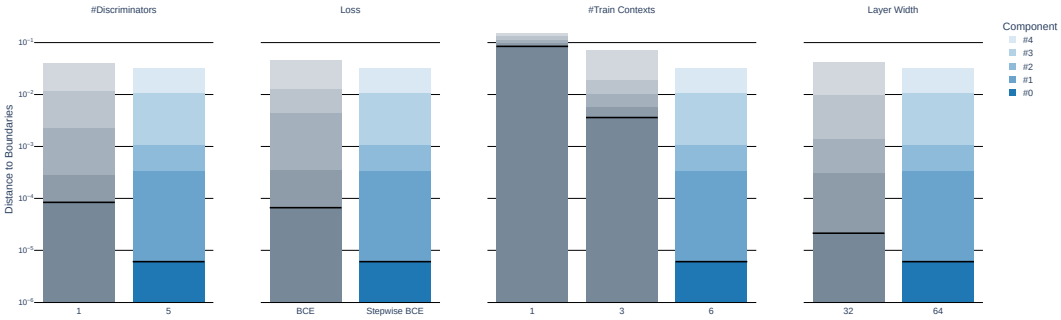

Figure 14: Ablation study of VIGOR on the Panda Reacher task.

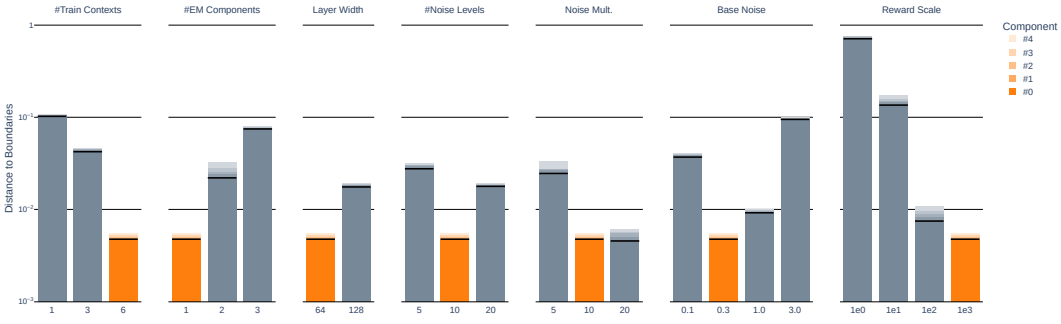

Figure 15: Ablation study of EM+D-REX on the Panda Reacher task. On this task, EM+D-REX often collapses to a single solution for the task, which can be seen by the proximity of all 5 components in each bar. The method is again very susceptible to the choice of hyperparameters, which need to be tuned on a by-task basis.

E Hyperparameters and training details

E.1 Environment Parameters

Table 1 gives an overview of default environment parameters. Note that parameters with a '*' are varied in the ablations. For each environment, we randomly draw both training and test contexts at runtime from a fixed set of contexts, making sure that training and test contexts are disjoint for any given seed.

Table 1: Default parameters of the different environments.

Parameter	Planar Reacher	Panda Reacher	Box Pusher
Trajectory length (T)	30	50	50
ProMP basis functions	5	8	7
Action dimension	5	6	2
Context dimension	4	6	3
Dimension of geometric descriptors	4	4	19
Evaluation samples per component	100	100	5
Number of training contexts ($\lvert\boldsymbol{c}_{\text{train}}\rvert$)	6	6	6
Training Demonstrations per context	5	5	3
Evaluation samples per component	100	100	5

E.2 Algorithm Hyperparameters

All neural networks are trained in PyTorch [49] using the Adam [50] optimizer. We employ Dropout [51], early stopping and Batch Normalization [52] to avoid overfitting. All methods are trained until convergence. All mixture policies use 5 components for all experiments.

All methods use the geometric descriptors of the respective environments as input. We use a learning rate of $3.0e-4$ for all methods except EM+D-REX, for which we found a learning rate of $3.0e-5$ to be more stable. VIGOR and EM+D-REX both use the equations introduced in [16] for updating their policies. We found that the method performed very similarly for a range of tested KL-Bound hyperparameters, and thus set it to 0.2 for all experiments for simplicity. We also always draw a sufficient amount of samples to estimate a full-rank quadratic surrogate to update the parameters of their mixture models in each iteration. Both VIGOR and EM+D-REX also make use of an ensemble with 5 networks, Dropout of 0.2 and use early stopping with a validation split of 0.1. We do not train the categorical distribution for either VIGOR or EM+D-REX. We find that Batch Normalization improves performance for EM+D-REX, but not for VIGOR, and as such only use it for EM+D-REX. Table 2 lists the remaining hyperparameters by task for VIGOR, Table 3 those of EM+D-REX.

For both BC and BC-GMM we find that training for 3000 and 30000 epochs on the state-action and trajectory-based settings works best respectively. Both use an entropy regularization term with a weight of $1.0e-3$. For BC-GMM, we additionally add an option to either train or not train the categorical distribution of the mixture to our hyperparameter search. If not trained, the distribution defaults to a uniform distribution over all components, similar to that of VIGOR and EM+D-REX. Remaining hyperparameters by task for BC(S) and BC-GMM(S) are given in Tables 4 and 5 respectively. Those of BC(T) and BC-GMM(T) are given in Tables 6 and 7. Hyperparameters for GAIL are listed in Table 8.

Table 2: Hyperparameters for VIGOR. Parameters with a '*' were optimized using grid searches.

Parameter	Planar Reacher	Panda Reacher	Box Pusher
$1d$-CNN layers *	2	3	4
Neurons per layer *	32	64	128
CNN kernel size	5	7	7
Batch size	64	64	64

Table 3: Hyperparameters for EM+D-REX. Parameters with a '*' were optimized using grid searches.

Parameter	Planar Reacher	Panda Reacher	Box Pusher
$1d$-CNN layers *	2	2	4
Neurons per layer *	64	64	64
CNN kernel size	5	7	7
Batch size	128	128	128
Fit EM Components *	3	1	1
Total EM Samples	8192	16384	16384
Base Noise *	0.3	0.3	0.3
Noise Levels *	5	10	5
Maximum Noise Multiplier *	5	10	2
Reward Scale *	100	1000	1000

Table 4: Hyperparameters for state-action-based Behavioral Cloning (BC(S)). Parameters with a '*' were optimized using grid searches.

Parameter	Planar Reacher	Panda Reacher	Box Pusher
MLP layers *	2	2	3
Neurons per layer *	64	32	64
Batch size *	128	64	64
Include target encoding	False	True	–
State-action framestacks *	1	1	1

Table 5: Hyperparameters for state-action-based Behavioral Cloning with a Gaussian Mixture Model policy (BC-GMM(S)). Parameters with a '*' were optimized using grid searches.

Parameter	Planar Reacher	Panda Reacher	Box Pusher
MLP layers *	2	2	3
Neurons per layer *	64	64	64
Batch size *	64	64	64
Include target encoding	False	True	–
State-action framestacks *	5	1	1
Train categorical distribution *	True	True	False

Table 6: Hyperparameters for trajectory-based Behavioral Cloning (BC(T)). Parameters with a '*' were optimized using grid searches.

Parameter	Planar Reacher	Panda Reacher	Box Pusher
MLP layers *	4	4	3
Neurons per layer *	256	64	128
Batch size *	4	4	4

Table 7: Hyperparameters for trajectory-based Behavioral Cloning with a Gaussian Mixture Model policy (BC-GMM(T)). Parameters with a '*' were optimized using grid searches.

Parameter	Planar Reacher	Panda Reacher	Box Pusher
MLP layers *	4	3	3
Neurons per layer *	64	64	128
Batch size	4	4	4
Train categorical distribution *	False	False	False

Table 8: Hyperparameters for GAIL. Parameters with a '*' were optimized using grid searches.

Parameter	Planar Reacher	Panda Reacher	Box Pusher
Discriminator MLP layers *	2	2	3
Discriminator neurons per layer *	32	64	64
Policy MLP layers	2	2	2
Policy MLP neurons per layer	32	32	32
Batch size (Discriminator) *	128	64	64
Batch size (Policy) *	128	64	256
Total time-steps *	$1e6$	$3e6$	$1e6$
Include target encoding	True	False	–
State-action framestacks *	5	5	1
Share networks	False	False	False
Policy steps	2048	2048	2048